COGNITIVE CONNECTIONIST MODELS FOR RECOGNITION OF STRUCTURED PATTERNS

JAMES WONG JIA JUN

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

Acknowledgements

I would like to thank my thesis advisor, Dr. David Cho Siu-Yeung, for his invaluable guidance and support during my work on this research. His knowledge, vision, inspiration and encouragement have guided me through the various stage of research. His involvement and dedication have been a great source of motivation. He has been a fantastic mentor and excellent role model to me during the course of my PhD candidature.

I would like to thank my family for their contribution of time and help in looking after my two children, Sarah and Joel, while I am away in Forensic and Security Lab, NTU writing my thesis. I am forever indebted to my lovely wife, Chen Lin, for her understanding and supportive for all these years. Without her support, I would not have been able to have enough time and energy to complete this thesis. I would like to dedicate this thesis to my two lovely children, Sarah and Joel, whom I might have spent lesser time than I wish.

I would like to thank Forensics and Security Lab, for providing space and equipment to perform my research. I would like to thank Nanyang Technological University for the opportunity and scholarship that they have provided.

Abstract

Traditional pattern recognition by computers focuses on the problem of identifying simple two-dimensional templates, such theories are too simplistic to account for the human's abilities to recognize varied and novel patterns. Feature theories ignore evidence that processing of global form often takes priority over processing of local features and are sensitive to context in which the stimulus appears. Pattern recognition systems usually consist of three steps of data acquisition, feature extraction and classification. Feature extraction process in pattern recognition, produces errors, more than often, is due to the operating environment that the feature extractor is used. Typically, a recognized object can be subjected to various degrees of changes. This motivates us to develop another kind of feature representations for pattern recognition.

Many natural or artificial systems are more appropriately modelled using "Data Structures". By incorporating structures in extracted features, it would facilitate the data processing process and later pattern recognition process by making it more efficient and noise tolerant. This thesis is presented to investigate the use of connectionist models to generalize structural information, which perform like a human cognition for recognizing erratic patterns. Erratic patterns here mean that incomplete features are extracted by feature extractor in a pattern recognition system, caused by occlusions in the data or un-filterable noise in the pattern.

A computational framework for learning a flavour of structural connectionist models is of paramount importance for both pattern recognition and development of

brain-inspired systems, since it allows the treatment of structured information very naturally and, in several cases, very efficiently. The details of this framework will be investigated in this thesis. Several research issues are addressed and examined for this framework in which they are included: (1) Slow convergence speed causing learning process of complicated structured patterns to be unreasonably long; (2) Long-term dependency problem presented in conventional recursive neuron model in learning of deep tree structures; (3) Discriminative capability in the connectionist model would be an issue when input features with unknown output classes are required to be discriminated; (4) Randomly chosen learning parameters in the initialization stage posed to be a problem in the generalization rate of the model.

In this thesis, a connectionist model namely Probabilistic Recursive Neural Network (PRNN) model is presented in which a hybrid structure of Gaussian Mixture Models (GMMs) and weighted sum of sigmoid function is formed. This model is a hybrid in which learning is unsupervised locally for feature discrimination, but remains supervised globally for pattern classification. A special learning algorithm is proposed to overcome the problems of slow convergence speed and long-term dependency brought about by large and complicated structured patterns.

Moreover a brain-inspired connectionist model namely Local Experts Organization (LEO) model is presented in the next to mainly address the issue of high initialization sensitivity when determining the model parameters randomly in the PRNN model. The architecture of LEO model employs support vector machine (SVM) as a local expert and reduced multivariate polynomial (RM) classifier as a fusion classifier. In addition, it is demonstrated that the LEO model is capable of generalizing structure patterns to organize the feature extracted from various regions of images.

Several studies in relating to the application of the potential of connectionist models for pattern recognition were carried out to provide evidence that the connectionist model is able to improve the accuracy of recognition rates of variations presented in the extracted features. Particularly, facial emotion recognition is applied to demonstrate the robustness of the connectionist models for recognizing faces with occlusions, which forms erratic patterns of the features and caused problems in recognition.

In essence, the works reported in this thesis conclude that using structured feature representation and adaptive processed using cognitive connectionist models such as the proposed PRNN and LEO models, the successful recognition rates achieved by face recognition systems are higher than those obtained using unstructured feature representation. Structured feature representation is somewhat similar to that of the human's cognitive functions where various areas of the brain is trained and specialize in performing a specific function. Both PRNN and LEO models will potentially be applied to other applications where erratic patterns are presented.

Table of Contents

ACKNOWLEDGEMENTS II

ABSTRACT III

CHAPTER 1 INTRODUCTION 1

1.1 BACKGROUND 1

1.2 MOTIVATION 4

1.3 PROBLEM STATEMENT 7

1.3.1. Convergence speed 9

1.3.2 Long term dependency 9

1.3.3 Discriminative capability 10

1.3.4 Initialization sensitivity 10

1.3.5 Pattern recognition capability 11

1.4 CONTRIBUTIONS 11

1.5 PUBLICATIONS 13

1.6 ORGANIZATION OF THIS THESIS 15

CHAPTER 2 STRUCTURAL FEATURES REPRESENTATION 17

2.1 UNSTRUCTURED FEATURE REPRESENTATION 19

2.2 STRUCTURED FEATURE REPRESENTATION 22

2.2.1 Quadtree representation 23

2.2.2 Binary Partition Tree representation 24

2.2.3 Computation of Binary Partition Tree 25

2.2.4 Region-based Tree representation 27

2.2.5 Region Merging Strategy for tree representation of images 28

2.2.6 Notation of Tree Representation 30

CHAPTER 3 ADAPTIVE PROCESSING OF TREE STRUCTURES 32

3.1 BACKGROUND 32
3.2 NEURAL NETWORKS FOR PROCESSING TREE STRUCTURES 35
3.3 BACKPROPAGATION THROUGH STRUCTURE (BPTS) ALGORITHM 38
3.4 LONG-TERM DEPENDENCY PROBLEM 40
3.5 AN IMPROVED ALGORITHM FOR BPTS 41
3.6 UNSUPERVISED MODEL FOR ADAPTIVE PROCESSING OF STRUCTURED PATTERNS 49
3.7 GENETIC EVOLUTION PROCESSING OF STRUCTURED PATTERNS 51
3.8 CONCLUDING REMARKS 55
CHAPTER 4 PROBABILISTIC RECURSIVE NEURAL NETWORK 56
4.1 MOTIVATION 59
4.2 MODEL DESIGN 60
4.2.1 Architecture 60
4.2.2 Universal Approximation 63
4.3 LEARNING FRAMEWORK 64
4.3.1 Locally Unsupervised Learning for GMMs 65
4.3.2 Globally Structural Supervised Learning for Recursive Network 67
4.3.3 Summary of the proposed learning algorithms 70
4.4 SOME ANALYTICAL STUDIES 72
4.4.1 Decision Boundary Analysis 72
4.4.2 Computational Complexity 75
4.4.3 Convergence Analysis 77
4.5 EXPERIMENTAL RESULTS AND DISCUSSION 83
4.5.1 Simulation of the Traffic policeman signaling problem 83
4.6.2 Natural Scene Image Classification Simulation 86
4.6 CONCLUSIONS 94

CHAPTER 5 LOCAL EXPERTS ORGANIZATION MODEL 96

5.1 MOTIVATION 98

5.2 LEO MODEL DESIGN 101

5.3 PARAMETERS OPTIMIZATION 102

5.4 LOCAL EXPERTS 103

5.5 FUSION CLASSIFIER 105

5.6 EXPERIMENTAL RESULTS AND DISCUSSION 110

5.6.1 Simulation of the Traffic Policeman signaling problem 110

5.6.2 Natural Scenery Images Classification 112

5.7 CONCLUSION 116

CHAPTER 6 FACIAL IMAGE UNDERSTANDING AND INTERPRETATION: A COGNITIVE APPROACH 118

6.1 BACKGROUND OF TRADITIONAL RECOGNITION METHODS 120

6.1.1 Facial Recognition for Biometrics 120

6.1.2 Emotion Recognition 123

6.3 FRAMEWORK DESIGN 126

6.3.1 System Architecture 126

6.3.2 Feature Extraction 129

6.3.3 Localized Gabor Feature Extraction 132

6.3.4 Gabor to Tree Structure Representation 134

6.4 DATABASE PREPARATION 137

6.4.1 Biometric Facial Recognition 137

6.4.2 Emotion Recognition 139

6.5 EXPERIMENTAL RESULTS AND DISCUSSION 143

6.5.1 Biometric Facial Recognition 144

6.5.2 Emotion Recognition 149

6.5.3 Comparison with other recognition approaches 163

6.6 CONCLUSION 164

CHAPTER 7 CONCLUSIONS AND FUTURE RESEARCH 167

7.1 SUMMARY 167

7.1.1. The Cognitive Connectionist Models 168

7.1.2. Facial Processing Applications 169

7.1.3. RECOGNITION OF ERRATIC PATTERNS 170

7.2 FUTURE RESEARCH 171

REFERENCES 173

APPENDIX 185

A1 ASIAN EMOTION DATABASE 185

A1.1 CREATION OF ASIAN EMOTION DATABASE 186

A1.2 STATISTICS OF PARTICIPANTS 188

A2 38 FACIAL FEATURE COMPONENTS OF HFTS 190

A3 GABOR FILTER RESPONSE ON 6 BASIC EMOTIONS 191

A4 FACS ACTION UNITS 192

List of Figures

Figure 1.1 – Example of eyes related features that might be occluded by feature extractor when the subjects are wearing sunglasses. 2

Figure 1.2 - Anatomy of the human brain (a) Human Brain and the location of the visual cortex (Blue), dorsal stream (Green) and the ventral stream (Purple) (Source: Wikipedia), (b) The lobes of the human brain – Temporal lobe (Green), Occipital Lobe (Pink), Parietal Lobe (Yellow), Frontal Lobe (Blue) (Source: Wikipedia). 3

Figure 1.3 – A general scheme of typical pattern recognition system 5

Figure 2.1 – A Scene showing a house with some details, adapted from (Tsoi, 1998) 19

Figure 2.2 – Eigenfaces are extracted out of an image data using Principal Component Analysis (PCA) (image source – AT&T Labs). 20

Figure 2.3 – A tree representation of a flower image 22

Figure 2.4 – A quadtree representation of an image region. The four children of each node correspond to the upper left, upper right, lower left, and lower right quadrants. M=mixed; F=foreground; and B=background. 23

Figure 2.5 – Example of binary partition tree: (a) original image, (b) partitioned image with 6 regions, (c) a form of binary tree to represent the image 25

Figure 2.6 - Example of region merging to create a binary tree. (a) Five regions created by the segmentation method, (b) Four-levels binary tree 29

Figure 3.1 - An illustration of a data structure with its nodes encoded by a single-hidden-layer neural network. (a) a Directed Acyclic Graph (DAG); (b) The encoded DAG. 36

Figure 3.2 – The genetic evolution framework for processing of tree structures (adopted from Cho and Chi, 2005) 51

Figure 4.1 – (a) Architecture of the probabilistic based recursive neural network using a Gaussian Mixture Model. (b) Structure of a Gaussian Mixture Model, where Σ and Ω denotes summation and Gaussian basis function operators, respectively. 61

Figure 4.2 - Simulation of Traffic Policeman Signaling Situation (a) An image created by the combination of primitives. (b) A tree representation of the traffic policeman with the shaded blocks representing the primitives. 73

Figure 4.3 – Scatter plot of various inputs. (a) plot of input at the root node. (b) plot of the likelihood function obtained after the EM learning phase. (c) plot of the likelihood function obtained after the fine-tuning decision boundary phase... 74

Figure 4.4 – (a) Convergence performances of the different algorithms with 4 hidden nodes in the traffic policeman signalling simulation. (b) Convergence performances of the different algorithms with 6 hidden nodes in the traffic policeman signaling simulation. (BPTS : Back-Propagation Through Structures, Prob-GD: Probabilistic based gradient descent algorithm, Prob-LS: probabilistic based least squares algorithm, Prob-PO: probabilistic based Penalized Optimization algorithm). 84

Figure 4.5 – (a) Training performances , (b) Classification performances of the different algorithms with varying number of hidden nodes. (Prob-GD: Probabilistic based gradient descent algorithm, Prob-LS: probabilistic based least squares algorithm, Prob-PO: probabilistic based Penalized Optimization). 86

Figure 4.6 - An example of region merging to create a binary tree, five regions created by the segmentation method. 88

Figure 4.7 - An example of region merging to create a BSP tree, five regions created by the segmentation method. 89

Figure 4.8 - Samples images of ten categories of Natural Scene Images. 89

Figure 4.9 - Image segmentation. Top: Segmented image, Bottom: Five segmented regions. 90

Figure 4.10 – ROC curves related to the classification results of the various models. 94

Figure 5.1 – Neuroanatomical account of face processing (a) Normal face processing. The green route shows the covert dorsal route via the IPL (inferior parietal lobule) and the STS (superior temporal sulcus). The red route is the overt ventral route to recognition. (b) In prosopagnosia the overt ventral route is damage, hence face recognition is compromised. (c) This account can also be applied to explain Capgras delusion, where damage is postulated to be in the covert dorsal route. Adopted from (Haxby *et al.*, 2000). 99

Figure 5.2 – Demonstration of Thatcher illusion for the three orientations (a), (b) and (c) shows the original picture (d), (e) and (f) shows the Thatcher version of the picture in upright, ninety degree, inverted orientation respectively. 99

Figure 5.3 – Organization chart of a typical IT company. 100

Figure 5.4 – (a) Directed Acyclic Graph (DAG) representation of features extracted from the House Scene in (Tsoi, 1998) for LEO network, (b) The architecture of a LEO node. 101

Figure 5.5 - Classification Accuracy for Traffic Police Officer using Local Expert Organization Model (LEO). LEO – Local Expert Organization, PRNN - (probabilistic recursive neural network), BPTS – (back propagation through structures), SVM – support vector machine, C4.5 decision tree. 110

Figure 5.6 – Time taken for training Traffic Police Officer by various models (LEO – Local Experts Organization, SVM- Support Vector Machine, C45 – Decision Tree, BPTS – Back Propagation Through Structures, PRNN – Probabilistic Recursive Neural Network). 111

Figure 5.7 - Receiver-Operating-Characteristic (ROC) curves of the natural scene image classification obtained from different classifiers (LEO – Local Experts Organization, SVM – Support Vector Machine, KNN – K-Nearest Neighbour, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial) 114

Figure 5.8 - Receiver-Operating-Characteristic (ROC) curves of the natural scene image classification obtained from different classifiers (LEO – Local Experts Organization, SVM – Support Vector Machine, KNN – K-Nearest Neighbour, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial) under different scenarios of (a) noise variation, (b) blur variations, (c) brightness variations, (d) darkness variations. 115

Figure 5.9 – Time taken for training for Natural Image classification database by the various models (LEO – Local Experts Organization, SVM – Support Vector Machine, C45 – Decision Tree, BPTS – Back Propagation Through Structures, PRNN – Probabilistic Recursive Neural Network). 116

Figure 6.1 – Example on how a feature locator would detect the location of the primary features under different scenario. (a) perfect unobstructed frontal image (b) subject wearing a pair of sunglasses (c) subject wearing a scarf.. 118

Figure 6.2 - Six basic emotions (from left to right): anger, joy, sadness, surprise, fear and disgust (images taken from NTU Asian Emotion database). 124

Figure 6.3 – Various muscles used in making facial expressions in facial emotion. 124

Figure 6.4 – Brain inspired model for Human Face Interpretation. (a) Signal path in the brain for face processing. The green route shows the covert dorsal route via the IPL (inferior parietal lobule) and the STS (superior temporal sulcus). The orange route is the overt ventral route to recognition. (b) Proposed recognition system for face processing. The yellow block mimics the visual cortex functions. The green block mimics the Ventral Route function of face recognition. The orange block mimics the Dorsal Route function of emotion recognition. 127

Figure 6.5 – Gabor wavelet response of the 6 basic emotions..... 130

Figure 6.6 - Four primary feature locations and entire face region. Crosses denote the centre of fiducial points. Rectangle box denotes of region of interest..... 131

Figure 6.7 - Sixty feature regions denoted by rectangle boxes at various detail levels (from left to right). (a) level 2: upper, lower, left, right and centre region of face. (b) level 3: forehead, left and right eye, eyes, nose, mouth, left and right cheek and nostril. (c) level 4: forehead, left and right eye, eyes, nose, mouth, left and right cheek, left and right nose. (d) detail features of various regions of interests. 131

Figure 6.8 – A Typical FEETS representation of a Human Face. Blue nodes are nodes in the used for HFTS representation. 136

Figure 6.9 - Normal lighting conditions. 138

Figure 6.10 – Extreme lighting conditions. 138

Figure 6.11 – Pose 0 to 8 in the YALE Face Database..... 138

Figure 6.12 – Original Images of 92 x 112 pixels of various persons in the ORL Database. 138

Figure 6.13 – Pose 1 to 10 of the ORL Database, they are cropped and resized..... 139

Figure 6.14 – Original image of 256 x 256 pixels of various persons in JAFFE database. 139

Figure 6.15 – Original Images of 640 x 480 pixels in Cohn-Kanade AU-Code Face Expression Database (a) happy (AU 6+12+25) (b) anger (AU 4+L14+17) (c) disgust (AU 4+7+17+23+24) (d) fear (AU 1+2+5d+25+27) (e) happy (AU 6+12+16+25) (f) disgust (AU 15d +17e+B22) (g) anger (AU 4+6+7+9d+17b) (h) anger (AU 4+17+23+24) (i) fear (AU 1+2+5+25+27) (j) happy (AU 6+12+25) (k) sad (AU 25) (l) fear (AU 1+2+5+16+20+25) (m) disgust (AU 4+6+7+9d+17d+25). 140

Figure 6.16 – Overall performance for HFTS model against other methods for missing fiducial points..... 146

Figure 6.17 – Recognition, verification, and false accept rate for each person in the ORL database obtained by different classifier..... 148

Figure 6.18 – Performances of FEETS vs QuadTree using different number of tree levels on JAFFE database. 150

Figure 6.19 - Six Basic Emotions a) Anger, b) Disgust, c) Fear, d) Happy, e) Sad and f) Surprise. 153

Figure 6.20 - Output response of PRNN for the Six Basic Emotions a) Anger, b) Disgust, c) Fear, d) Happy, e) Sad and f) Surprise. 153

Figure 6.21 – (a) Subject without any artifacts; (b) Histogram showing the output response of the PRNN model..... 154

Figure 6.22 – (a) Subject wearing sunglasses, eye detection is unable to detect locations of eyes; (b) Histogram showing the output response of the PRNN model..... 154

Figure 6.23 – (a) Subject wearing veil, nose and mouth detection is unable to detect locations of nose and mouth; (b) Histogram showing the output response of the PRNN. 155

Figure 6.24 - (a) Subject without any artifact, (b) subject wearing a veil, (c) subject wearing sunglasses. 157

Figure 6.25 - Likelihood histogram showing LEO output response for fig. 13 scenarios, (a) subject without any artifacts, (b) subject wearing a veil, (c) subject wearing sunglasses. 157

Figure 6.26 - Performance results of missing features evaluation by the LEO model against other models for dataset A (subject dependent). (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial)..... 158

Figure 6.27 - Performance results of missing features evaluation by the LEO model against other models for dataset B (subject independent). (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial)..... 159

Figure 6.28 - Scalability performance of model using different numbers of persons in training and testing set. (LEO – Local Experts Organization, PRNN – Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial). 160

Figure 6.29 - Examples of features detection errors. (a) feature location is 5 pixel or less off from the ideal center of features, (b) features are 6 to 10 pixels off

from the ideal center of features, (c) 11 to 15 pixels off from the ideal center of features, (d) more than 16 pixels off the ideal center of features. 161

Figure 6.30 - Chart showing the performance of the classification models for detecting error in the fiduical point locations. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, KNN – K-Nearest Neighbor, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial).....162

Figure 6.31 - Chart showing the amount of time taken during training for the two models. Note that the codes in these models are not optimized, and are running based on both Matlab platform and from WEKA package. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SMO – Sequential Minimum Optimization, C45 – decision tree).....162

List of Tables

Table 4.1 – Computation time taken for training by different learning algorithms. *n* - number of hidden nodes, *t* - Computation time for training. 85

Table 4.2 – A confusion matrix of image classification by the PRNN model..... 93

Table 4.3 – Comparative results of the image categorization..... 93

Table 5.1 - Image Classification Confusion Matrix of the LEO model..... 113

Table 5.2 - Classification rates averaged by 5-fold cross validations obtained from different classifiers (LEO – Local Experts Organization, PRNN – Probabilistic Recursive Neural Network, BPTS – Back Propagation Through Structures, SVM – Support Vector Machine, KNN – K-Nearest Neighbor, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial) . 113

Table 6.1 – Distribution of Training and Test images. 142

Table 6.2 – Performance of HFTS model against other methods for different lighting conditions. 145

Table 6.3 – Performance of HFTS model against other methods for various poses. 145

Table 6.4 – Benchmarking with the other well-known models for face authentication. 147

Table 6.5 – Verification Performance of the proposed model against the tested classifiers..... 147

Table 6.6 – Recall and generalization rates of FEETS vs 4-level QuadTree using PRNN model on JAFFE database. Dataset A – Subject Dependent, Dataset B – Subject Independent. 149

Table 6.7 – QuadTree vs FEETS on CMU database benchmarking with other classifiers. Dataset A – Subject Dependent, Dataset B – Subject Independent. 151

Table 6.8 – Performances of the proposed model benchmarking against other models on JAFFE database. 152

Table 6.9 – Confusion Matrices showing the interclass recognition errors using PRNN model in JAFFE. 152

Table 6.10 – Performances of the proposed model against other models on CMU database. 152

Table 6.11 – Confusion Matrices showing the interclass recognition errors using PRNN model in CMU. 153

Table 6.12 - Performance of face emotions recognition for missing fiducial points in subject-dependent and subjects-independent conditions. Set A – Subject Dependent, Set B – Subject Independent. 155

Table 6.13 - Performance of FEETS/LEO model against other classifiers. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial). Dataset A – Subject Dependent, Dataset B – Subject Independent. 156

Table 6.14 - Comparison with other recognition approaches. 163

Chapter 1

Introduction

1.1 Background

In 1967, Ulric Neisser published a book with the title "*Cognitive Psychology*". Although scientists had been researching human thought from a cognitive perspective for a couple of decades before this, Neisser's book helped to define *Cognitive Psychology* or *Cognition* as a discipline. Neisser defined cognition as "all the processes by which the sensory input is transformed, reduced, elaborated, stored, recovered, and used." Cognition is the study of the mental processes underlying our ability to perceive the world, remember, talk about and learn from our experiences, and modify our behaviour accordingly. It comprises topics such as perception, attention, memory, knowledge, language, problem solving, reasoning and decision making, and aspects of intelligence, emotion and consciousness. Cognition can be viewed as an information processing system through which people interact with the external world. It is the product of top-down and bottom-up processes. Top-down processing refers to the influence of knowledge and expectations on functions such as language, perception and memory. Bottom-up processing is processing driven by an external stimulus. Cognitive functions are often assumed to be modular that is to operate independently of each other.

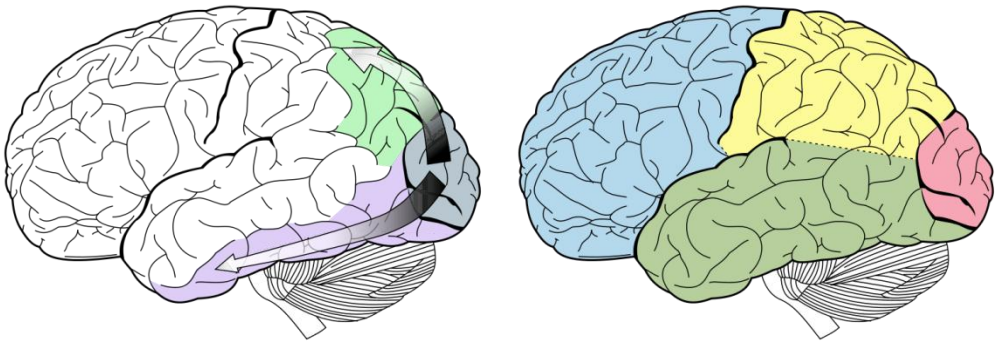
Think about recognizing a person from his/her face, a human is able to recognize with another person even if such a person is wearing sunglasses (for example, see a photo in Figure 1.1), which blocks out the eyes where are one of the key feature for recognizing a person. What makes the human brain so special that it is capable of recovering from such a loss of information and yet be able to make a fairly accurate assessment? Are the modern classifiers capable of making such an evaluation when presented with a loss of critical information? All these are the open questions in which researchers are still working hard to solve and explain for them.

Figure 1.1 – Example of eyes related features that might be occluded by feature extractor when the subjects are wearing sunglasses.

Many psychologists assume that cognitive processes are modular. Modules are clusters of processes that function independently from other clusters of processes. Each module processes particular type of information, for example visual objects or faces. Modularity also underpins the assumption of localization of function that mental processes map onto specific regions of the brain. It can also emerge from the activity of distributed, rather than localized, networks of neurons. This means that modular cognitive functions can be mimicked by connectionist networks.

Taking a look into our human brain as shown in Figure 1.2, whenever human eyes see an image, the retinal is linked to the visual cortex, which is linked to the

ventral stream. The ventral stream is one of the two primary pathways of the visual cortex, is associated with form recognition and object representation. The pathway begins with the V1 section of the visual cortex and goes through V2 and V4 and to the inferior temporal lobe as well as the limbic system. V1, V2 and V4 are various part of the visual cortex and each is tuned to be sensitive to interpret different information in the visual image from the retina. This is much like the feature extraction portion of a computer vision system. The temporal lobes are part of the cerebrum, and it contains the hippocampus, which is involved in memory formation. The limbic system includes many different cortical and sub-cortical brain structures. The limbic system influences the formation of memory by integrating emotional states with stored memories of physical sensations (visual is part of this sensation). The human brain is capable of recognizing objects with incomplete data through unconscious inference. Much of the inner working of the human brain is still an active research area by neurologist and psychologist.



(a) (b)

Figure 1.2 - Anatomy of the human brain (a) Human Brain and the location of the visual cortex (Blue), dorsal stream (Green) and the ventral stream (Purple) (Source: Wikipedia), (b) The lobes of the human brain – Temporal lobe (Green), Occipital Lobe (Pink), Parietal Lobe (Yellow), Frontal Lobe (Blue) (Source: Wikipedia).

The image falling on each retina is two-dimensional and each encounter with a particular object gives a different image that depends on our viewpoint. Object

recognition is an interesting perceptual problem because we must use these unique, two-dimensional images to work out the three-dimensional shape of the object. Neuropsychological case studies provide support for claims that object recognition occurs after early visual processes such as feature detection. People with visual agnosia have normal vision, in that they can perceive colors and movement, but cannot identify objects or even simple shapes such as letters or circles, which might suggest that they cannot organize incoming visual information to perceive complete forms.

On the other hand, theories of pattern recognition focus on the problem of identifying simple two-dimensional shapes such as letters and numbers from sets of features. Template theories propose that two-dimensional patterns, such as the letter "A", are recognized by matching the visual stimulus to a template stored in memory. Feature theories, for example Selfridge's pandemonium model, propose that we compare sets of features (rather than complete forms) against the features of prototypes in memory. Template theories are too simplistic to account for our ability to recognize varied and novel patterns. Feature theories ignore evidence that processing of global form often takes priority over processing of local features and are sensitive to the context in which the stimulus appears. Therefore, neither feature nor template theories are well-equipped to explain recognition of complex, three-dimensional objects or our ability to recognize the likely function of novel objects.

1.2 Motivation

In a typical pattern recognition system as shown in Figure 1.3, pattern recognition usually consists of three steps: (1) data acquisition, (2) feature extraction, and (3) classification. First, the data for classification are gathered from the environment via a set of sensors. They can be numerical, linguistic, or both. Then the feature

extraction is performed to search for internal structure in the data. It is desirable that the dimensions of the feature space be much smaller than those of the data space so that classification techniques can be efficiently applied. Finally, classification is performed via a classifier.

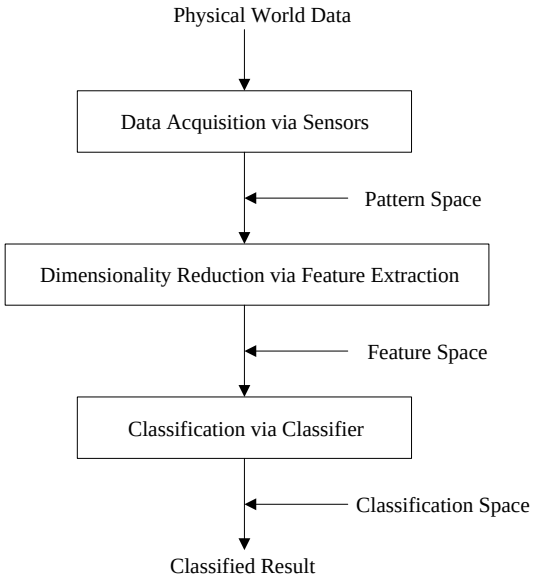


Figure 1.3 – A general scheme of typical pattern recognition system

An ideal feature extractor would yield a feature representation that makes the classifier's job trivial. The goal of a feature extractor is to characterize an object to be recognized by extracting distinct features for objects whose values are very similar for objects that belong to the same category and very distinct for objects belonging to a different category (Duda *et al.*, 2000). However, in the real applications, there is no perfect feature extractor, which would not introduce errors into the extracted features.

Specifically, face recognition can be considered as a special type of object recognition in which faces often share the same basic features (eyes, nose, mouth etc.) arranged in the same way. Yet we are exceptionally well able to discriminate different faces and to identify different emotions expressed in a single face. Face recognition is particularly sensitive to orientation. It is much easier to recognize an upright face than an inverted face, and the difference in ease is greater than for other types of object. Faces are more than the sum of their parts; the precise configuration

of feature is important for face recognition. The so-called Thatcher illusion (Thompson, 1980) illustrates the importance of configurational information for perception of upright but not inverted faces. Young *et al*. (1987) provided another demonstration of this. They cut pictures of famous people in half horizontally and asked participants to name the top half of each face. When the faces were upright, naming was much harder when the top half was aligned with the wrong bottom half than when it was viewed alone. When the faces were presented upside down, the naming of the mismatched faces actually improved. This highlights that feature extraction or feature representation is a vital part of the pattern recognition system. Feature extractor used in pattern recognition applications produces errors in feature extraction, not because the feature extractors are incapability, but due to the operating environment for the feature extractor being used. Typically, the object that needs to be recognized can be subjected to various degrees of changes. For example, in face recognition, the image captured can be subject to lighting variation (due to varying amount of sunlight, if taken outdoors), pose variation, expression variation. This motivates us to develop another kind of representations for pattern recognition.

Many natural or artificial systems are more appropriately modelled using "*Data Structures*". By incorporating structures in the extracted features, where such structures exist, it would facilitate the data processing process and later the pattern recognition process. The advantages of this processing are that to make the process more efficient and may be more noise tolerant. Often feature extraction is domain-specific, i.e., the methods used to extract features in one domain may not be useful for extracting features from another domain. Fundamentally, the human visual system uses different techniques for feature extraction of visual information, which require quite different types of processing techniques. Feature extraction is domain

dependent. In this research, the starting point is that the features of the entity have already been extracted, and that pre-processing steps have been performed. For example, in pattern recognition, we assume that relevant features from the image have already been extracted. Two possible avenues can be preceded which is whether one assumes that there exist structures among the extracted features, or not. If we do not assume that there exists an underlying structure to the extracted feature, then we can proceed in some manner to extract relevant information from the data concerning the entity. On the other hand, if we assume that there are underlying structures in the extracted features then we can proceed in a different manner. However, no matter what assumptions we made, a problem arises that how can the machine generalize the structure and later on to proceed to the pattern recognition task in the system. Along with this research issue, this thesis is presented to investigate the use of connectionist models to generalize structural information, which perform like a human cognition for recognizing erratic patterns. The erratic patterns here mean that incomplete features are extracted by the feature extractor in a pattern recognition system, for example in face recognition, where occlusions such as sunglasses or scarf is presented onto a subject's face, in which the information would be lost from the features extracted incompletely.

1.3 Problem Statement

Modern connectionist models have been successfully employed for solving learning tasks characterized by relatively poor data types. For example, feed-forward neural networks and probabilistic mixture models can only deal with static data types such as records or fixed-size numerical arrays. Sequences are the first significant improvement over static data in which a serial order relation is defined to provide us with some information that is not encoded into the variables themselves. Serial order

makes sequences naturally suited for modelling data in temporal domains, but serial order is only a very special relation. More complex relations amongst entities may exist in other learning domains. These relations can be conveniently represented using "*Graphs*" or "*Trees*".

In image processing for object or pattern recognition, a central issue is how to understand a particular given image scene (Jain, 1989). If we present the image by its pixels, then there could be significantly memory storage. For example, if the image consists of 1,000 x 1,000 pixels, then the memory required to stored it is 1Mbits, assuming that it is a black and white image, i.e., it has 2 levels of gray only. In addition, it does not take into account any possible relationships among the objects in the image, assuming that the image is about a number of objects. On the other hand, if we pre-process the image so as to extract some primitives (objects), the image could be represented using a much smaller amount of memory. In addition, the relationships among the objects would be more transparent that is more appropriately represented using a tree model. A question that is often asked is: given a number of scenes or objects, can we recognize the differences among them. If we can represent these scenes using tree models, an equivalent question to ask is: can we distinguish one class of trees from another. If we are given a number of trees, can we separate them into categories?

In order to answer such questions, it is essential to introduce a computational framework for learning tree structures model or so-called Adaptive Processing of Data Structures (Cho *et al.*, 2003; Frasconi *et al.*, 1998; Goller & Kuchler, 1996; Tsoi, 1998). This framework is of paramount importance for both pattern recognition and the development of brain-inspired systems, since it allows the treatment of structured information very naturally and, in several cases, very efficiently. The details of this

framework will be discussed in the later chapters of this thesis. The purpose of this research is to investigate the potential of this connectionist model for recognition of erratic patterns. Some research issues are addressed and examined in this thesis as below:

1.3.1. Convergence speed

Essentially, supervised neural networks are able to perform the classification of data structures (Sperduti & Starita, 1997). This approach is based on using generalized recursive neurons and a back-propagation through structure (BPTS) algorithm (Goller & Kuchler, 1996). The basic idea of this BPTS algorithm is to extend a back-propagation through time (BPTT) algorithm (Rumelhart & McClelland, 1986) to encode data structures by recursive neurons. In the BPTT algorithm, the gradients of the weights to be updated can be computed by back-propagating the error through the time sequence. Similarly, if learning is performed on a data structure such as a directed acyclic graph, the gradients can be computed by back-propagating the error through the data structures, which is known as BPTS algorithm. However, the rate of convergence is slow so that the learning process cannot be guaranteed to be completed within a reasonable time for most complicated data structures. Although the algorithm can be accelerated simply by using a larger or an adaptive learning rate, this would probably introduce oscillation and might result in a failure in finding optimal solution.

1.3.2 Long term dependency

An interesting research issue for this connectionist model is about the tree model may exhibit some kind of long term dependency problem. In conventional feed-forward neural network learning, it is known that if there are too many hidden layers, then

because of the fact that the back-propagating error is multiplied by the derivative of the sigmoidal function which is between 0 and 1, it is plausible that the product of the derivatives and the gradient for very deep layers could become very small, thus the parameters are not updated. Similarly, in the case of learning tree structure, the gradient contribution disappears at a certain tree level when the error back-propagates through the deep tree structures such that the learning information is latched. This is because the decreasing gradient terms tend to zero since the back-propagating error is recursively multiplied by the derivative of the sigmoidal function, which is between 0 and 1, in each neural node. This results in convergence stalling and yields a poor generalization.

1.3.3 Discriminative capability

One issue would be concerned about whether the connectionist model is able to discriminate a given set of input features with unknown output classes. A method is worth to develop that it will automatically organize the patterns so that the patterns are grouped together without a priori knowledge of the patterns. The original connectionist model for learning tree structures is able to provide the linear discriminative capability. However, there is no guarantee that the linear discriminate will separate the groups properly if they are linearly separable or not. Therefore, nonlinear discrimination is essential to give smoothing discriminant boundaries to facilitate the models in generalizing the non-linearly separable problems.

1.3.4 Initialization sensitivity

In the introduced tree models, we have assumed that the learning parameters are randomly chosen in the initialization stage. It is quite obvious that the learning session can be longer if the parameters are initialized improperly. It affects the

generalization rate if the model is sensitive to the initialization. One way to solve this problem is to introduce an approach to estimate optimal values for the initial parameters. This can reduce the learning time if the starting point of the optimization is very close to the true minimum. However, it would not be able to solve the generalization problem especially if there a number of local minima exist in the problem. In this case, it is possible and necessary to adapt an idea of developing a model, which is relatively insensitive to the initialized parameters, while at the same time obtaining the optimal solution.

1.3.5 Pattern recognition capability

One of the major focuses of this research is to investigate the potential of connectionist models for pattern recognition. Face recognition is a hard problem in which the challenges of variation in lighting conditions, pose variations, and obstructions of features always existed. The investigations would be included whether the connectionist models are able to discriminate different faces and to identify different emotions expressed in a single face. The processing tasks include the feature extraction for facial components, feature representations with tree models, and the connectionist models for recognition. In addition, the investigation of recognizing faces with the presence of erratic patterns is essential in this research.

1.4 Contributions

The major original contributions of this research are presented as follows:

1. In Chapter 4 of this thesis, a novel connectionist model is presented for classification of structured patterns, namely Probabilistic Recursive Neural Network (PRNN) model. This model has a hybrid structure of Gaussian Mixture Models (GMMs) and weighted sum of sigmoid function models. The

GMMs make use mainly of semi-parametric techniques for approximating probability density functions (pdf) and can assume both feature independence and Gaussian distribution. Discriminative information is acquired during learning and used for classifying structured patterns. The major contribution of this model is as a hybrid in which learning is unsupervised locally for feature discrimination, but remains supervised globally for pattern classification. However, in learning by means of the gradient based algorithm for the global supervised portion, this probabilistic recursive model may still suffer from the problem of convergence speed and long-term dependency brought about by large and complicated structured patterns. A special learning algorithm is presented that overcome those problems.

2. In Chapter 5 of this thesis, a novel brain-inspired connectionist model, namely Local Experts Organization (LEO) model is presented. The motivation of this LEO model is to overcome the limitations of the PRNN model in which the computational cost is high when learning the structured patterns using iterative optimization and the initialization sensitivity is high when determining model parameters randomly. The LEO architecture is a hybrid structure that uses support vector machine (SVM) and reduced multivariate polynomial (RMP) classifier as local expert as well as fusion classifier respectively. The learning time required for processing the tree structures is able to be reduced in certain degree comparing to the PRNN model. Additionally, such LEO model is able to generalize structural patterns to organize the features extracted from various regions of images.
3. In Chapter 6 of this thesis, tree structures models are presented for representing facial components to use for the application of face recognition

by the cognitive connectionist models. Feature extraction used by Gabor wavelet extraction is employed and the extracted features are presented in hierarchical manner from holistic to local representations. Using such hierarchical information, the tree models are more robust that the recognition system is able to tackle the problems of environmental lighting, pose variations and obstructions. It is also demonstrated that the cognitive connectionist models are more robust when missing to extract proper facial features. A new Asian face database is created to test the robustness of the models.

1.5 Publications

This research has in many parts been shaped by reviewers' comments and suggestion regarding many of the publications (both journal articles and conference presentations) listed below:

Journal Articles:

1. Jia-Jun Wong and Siu-Yeung Cho, *FEETS: A Face Emotion Tree Structure Representation with Probabilistic Recursive Neural Network Modeling*, IEEE Transactions on System, Man, and Cybernetic Part B, 2008 (under review).
2. Jia-Jun Wong and Siu-Yeung Cho, *A Local Experts Organization Model with Application to Face Emotion Recognition*, Expert Systems and Applications, (in press), 2008.
3. Siu-Yeung Cho and Jia-Jun Wong, *Human Face Recognition by Adaptive Processing of Tree Structures Representation.* Neural Computing and Applications, 17(3), pp.201-215, 2008.

4. Jia-Jun Wong and Siu-Yeung Cho, *Local Experts Organization Model for Natural Scene Image Classification*, Neural Processing Letters, 26(2), pp.83-99, 2007.
5. Siu-Yeung Cho and Jia-Jun Wong, *Structural Image/Object Representation by Probabilistic Recursive Model*, Pattern Analysis and Applications, 2006. (under review);
6. Jia-Jun Wong and Siu-Yeung Cho, *A Brain-Inspired Framework for Emotion Recognition*, Neural Information Processing Letters, 10(7), pp. 169-179, 2006.

Book Chapters:

1. Jia-Jun Wong and Siu-Yeung Cho, *A Brain-inspired model for recognizing human emotional states from facial expression*, Neurodynamics of Cognition and Consciousness (Understanding Complex Systems), Edited by: Leonid I. Perlovsky and Robert Kozma, Springer-Verlag, 2007;
2. Siu-Yeung Cho and Jia-Jun Wong, *Probabilistic Based Recursive Model for Face Recognition*, in *Lecture Notes in Artificial Intelligence*, Lipo Wang and Yaochu Jin, Editors, Springer-Verlag GmbH, Vol. 3614, pp. 1245 - 1254, 2005.

Conference Papers:

1. Jia-Jun Wong and Siu-Yeung Cho. *Adaptive Processing of Face Emotion Tree Structures*, in proc. of *International Conference on Pattern Recognition*, Hong Kong, 2006;
2. Jia-Jun Wong and Siu-Yeung Cho, *Recognizing Human Emotion From Partial Facial Features*, in proc. of *IEEE World Congress on Computational Intelligence (IJCNN)*, Vancouver, Canada, 2006;

3. Jia-Jun Wong and Siu-Yeung Cho. *Facial Emotion Recognition by Adaptive Processing of Tree Structures*, in proc. of *ACM - Symposium on Applied Computing*, Dijon, France, pp. 23-30, 2006;
4. Siu-Yeung Cho and Jia-Jun Wong. *Robust Facial Recognition by Localised Gabor Features*, in proc. of *International Workshop for Advanced Image Technology*, Cheju National University, Jeju Island, Korea, 2005.

1.6 Organization of this thesis

This thesis comprises of seven chapters. They are organised as follows:

Chapter 2 presents the basic idea of tree structures model to represent structural patterns for pattern recognition. A brief review is first given to describe the unstructured features representation and then presented how the tree structures models are used to represent the structural patterns. Several tree structures are demonstrated in feature representation for the structural pattern recognition.

Chapter 3 presents how a connectionist model can be used to generalize the tree structures model for pattern recognition. The chapter begins with an introduction of the tree structure model and its inherent problems. A review of new models which overcome the initial difficulties, poor generalization rate and difficult learning due to large number of parameters to be learnt, present in connectionist models. A learning algorithm called backpropagation through structure (BPTS) algorithm is reviewed and analyzed on how it tackles these initial problems and new challenges faced by the connectionist model after tackling the initial problems.

Chapter 4 concentrates on solving the problems presented in the BPTS algorithm. A novel connectionist model for learning structured patterns, namely Probabilistic Recursive Neural Network (PRNN) model is proposed. Both analytical

and empirical studies are discussed to demonstrate the capabilities of the PRNN model.

Chapter 5 offers a brain-inspired model called Local Experts Organization (LEO) for solving the problems of high computational complexity for learning structure patterns and high sensitivity of initialization of the model's parameters to the generalization results consistency. Empirical studies are conducted to evaluate the performance of the LEO model in solving those problems.

Chapter 6 discusses on the application of the proposed cognitive connectionist models presented in Chapter 4 and 5 to the domain of face recognition. Tree structures models are presented to represent the features extracted from the human face for representation. Two kinds of recognitions are exhibited. One presents the classification and verification performances of face recognition to demonstrate and benchmark against other recognition models. Another presents the capability of the proposed models to recognize human emotions from the face expressions and their performances are benchmarked against other models. The robustness of the models is also presented to demonstrate that the cognitive connectionist models are able to recognize erratic patterns.

Chapter 7 draws the conclusion and summarizes the contributions of this thesis. Further research issues along with this work are also addressed in this chapter.

Chapter 2

Structural Features Representation

Features can be defined in terms of local neighbourhood operations applied to any kind of images. Thus, the procedures of these operations are referred to as feature extraction in a typical pattern recognition system. Feature extraction is worked without making a local decision, thereby the result is often referred to as a feature image. Consequently, a feature image can be seen as an image in the sense that it is a function of the same spatial (or temporal) variables as the original image, but where the pixel values hold information about image features instead of intensity or color. This means that a feature image can be processed in a similar way as an ordinary image generated by an image sensor. Feature images are also often computed as integrated step in algorithms for feature detection.

The choice of feature representation is a critical issue for developing a pattern recognition system. In many applications, it is not sufficient to extract only one type of feature to obtain the relevant information from the image data. Instead two or more different features are extracted, resulting in two or more feature descriptors at each image point. A common practice is to organize the information provided by all these descriptors as the elements of one single vector, commonly referred to as a feature vector. The set of all possible feature vectors constitute a feature space. A common example of feature vectors appears when each image point is to be classified as

belonging to a specific class. Assuming that each image point has a corresponding feature vector based on a suitable set of features, meaning that each class is well separated in the corresponding feature space, the classification of each image point can be done using standard classification method. Another example occurs when neural network based processing is applied to images. The input data fed to the neural network is often given in terms of a feature vector from each image point, where the vector is constructed from several different features extracted from the image data. During a learning phase, the networks can find which combinations of different features that are useful for solving the problem.

Nevertheless, the abovementioned feature vector is an unstructured feature representation, hence it is unable to achieve any information about the relationships among the objects in the image. Moreover, since feature extraction and its associated pre-processing are domain-specific, which means that the methods used to extract features in one domain may not be useful for extracting features from another domain. For example, in speech recognition, it is known that some of the underlying information is extracted in an utterance using some common linear predication models, but this kind of models is not useful in image processing. In image processing, if we assume that relevant features from the image have already been extracted and some underlying structures are existed to the extracted features, we can proceed to incorporate structures in the extracted features, which would facilitate the feature representation for pattern recognition.

In this chapter, the first part will introduce the basic idea of an image using unstructured feature representation and address the associated problems of this representation. The second part will introduce how the structured patterns can be

used to represent the image contents. Several different types of tree structures models are presented in this part.

2.1 Unstructured Feature Representation

In pattern recognition, a major issue is how to generalize and understand a particular given image. For example, if an image as shown in Figure 2.1 consists of 1024 x768 pixels, then the memory would require 2.35Mbytes, assuming that it is a 24bit RGB image stored as bitmap format. The memory storage requirements would be significant, thus some pre-processing tasks are necessary to perform for extracting dominant features for both storage and recognition. For instance, edge detection is used so that most of the non-essential details, like shading, textures, could be eliminated. Each part can be labelled such that an image can be represented by a flat vector as in the following form:

$$\mathbf{y} = \{A, B, C, D, a, b, c, d, e, f\} \tag{2.1}$$

where **y** is supposed to be the extracted image feature vector the scene which uses a much smaller amount of memory. The features extracted for each part are depended on the extraction algorithm used.

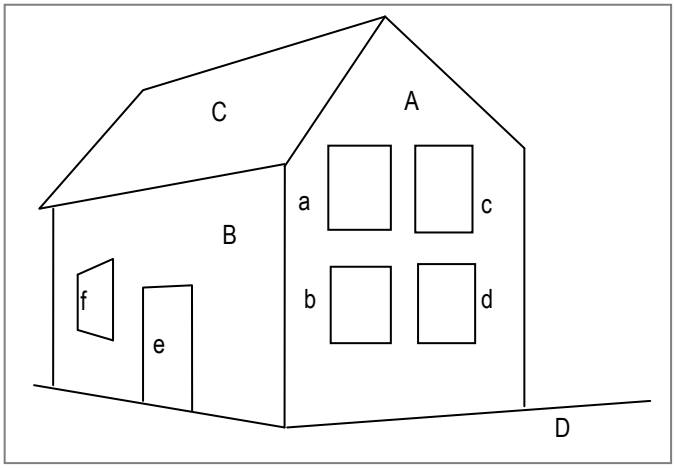


Figure 2.1 – A Scene showing a house with some details, adapted from (Tsoi, 1998)

For developing a face recognition system, Principle Component Analysis (PCA) is commonly used to perform feature representation or feature extraction, which preserves information about light and shade patterns rather than edges. A major advantage of PCA is that once patterns have been found in data, such data can be compressed. PCA reduces number of features needed for effective feature representation by discarding those linear combinations that have small variances and retain only those terms that have large variances. PCA is applied on re-sampled (uniform size) and normalized face images to form a set of eigenfaces (Kirby & Sirovich, 1990; Sirovich & Kirby, 1987; Turk & Pentland, 1991), see examples as shown in Figure 2.2.

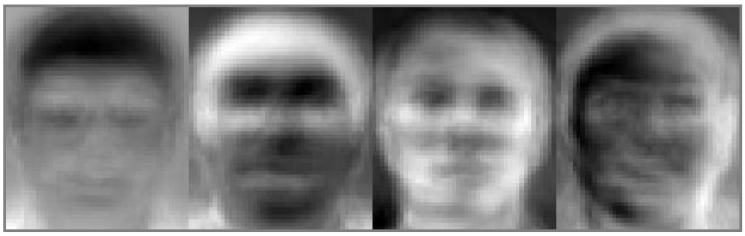


Figure 2.2 – Eigenfaces are extracted out of an image data using Principal Component Analysis (PCA) (image source – AT&T Labs).

PCA algorithm consists of two phases: encoding and decoding. The linear projection from the data space to the feature space represents an encoder for the approximate representation of the data vector. On the other hand, the mapping from feature space back to the data space represents a decoder for the approximate reconstruction of the original data vector.

PCA in its original form treats the entire face as an array or a vector, which causes two major problems. Firstly, due to the linear nature of PCA in image space, variations in geometry would cause problems. Craw *et al.* demonstrated by warping face images to an average geometry face, thus aligning key features such as eyes and mouth appeared to resolve this issue (Craw *et al.*, 1995). Secondly, occlusions or

other localized perturbations, such as variations in hairstyle or facial hair, causes problem to PCA recognition, due to its sensitivity nature. In the holistic representation of the face images, the perturbations have a negative effort on all expansion coefficients and cannot be easily disregarded. One way to deal with this is to perform localized PCA on fiducial points (eyes, nose, and mouth) as additional pixel vectors from which to extract more features by PCA (Moghaddam & Pentland, 1997).

Apart from PCA, Independent Component Analysis (ICA) expands on PCA as it considers higher order statistics. ICA is a computational method for separating a multivariate signal into additive subcomponents supposing the mutual statistical independence of the non-Gaussian source signal. ICA has been widely used to transform interdependent coordinates into significant and independent ones without much loss of information (Bingham & Hyvarinen, 2000). Since ICA is an extension from PCA, it also suffers from the same two problems of PCA, variation of geometry as well as unreliable components from occlusions or other localized perturbations.

To sum up, both global and local visual features such as localized PCA or ICA cannot represent image contents at a semantic level. Moreover, it is very difficult to integrate visual features from different domains and at different levels to measure the similarity between images. A simple solution is to use a weighted distance, where different weights are assigned to individual components in the feature vectors according to their importance in a specific application. Euclidean distance is one of the most commonly used metrics, but it is unable to measure the image contents at a semantic level.

2.2 Structured Feature Representation

Normally, images are stored in pixel map format. However, individual pixel intensities have little meaning to human perception. For human beings, image contents are interpreted at a semantic level. In general, an image consists of different objects or regions, which are arranged according to their spatial relationships. We believe that a structural representation is more suitable than the unstructured one. For example, as shown in Figure 2.3, a tree representation of a flower image can be used for content-based flower image retrieval and flower classification. Obviously, the image can be segmented into two major regions (i.e., the background and foreground regions) and flower regions can then be extracted from the foreground region. A tree-structure representation (to some extent of a semantic representation) can then be established and the image content can be better described. The leaf nodes of the tree actually represent individual flower regions and the root node represents the whole image. The intermediate tree nodes denote combined flower regions. For flower classification, such a representation will take into account both flower regions and the background. All the flower regions and the background in the tree representation will contribute to the flower classification to different extents partially decided by the tree structure.

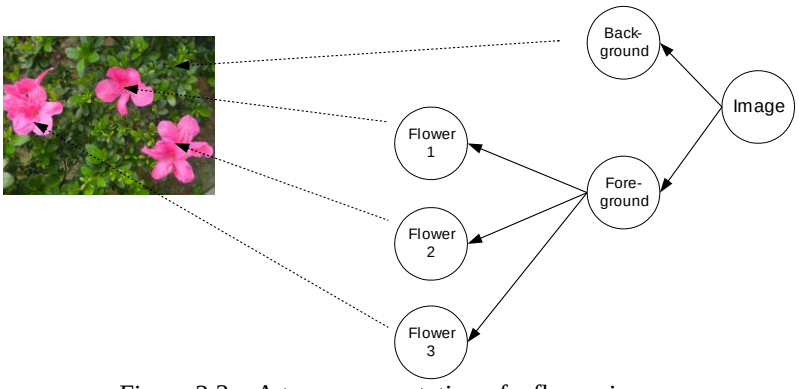


Figure 2.3 – A tree representation of a flower image

There are various types of tree structures representation are commonly used for image processing, segmentation, information retrieval and visual browsing. The following sub-sections describe these representations and their associated principles working in both image processing and pattern recognition.

2.2.1 Quadtree representation

Quadtree is a space-saving region representation that encodes the whole region. In general, each Region Of Interest (ROI) would be represented by a quadtree structure. Each node of a quadtree represents a square region in the image and can have one of three labels, i.e. namely foreground, background, or mixed. If the node is labelled foreground, then every pixel of the square region it represents is a pixel of the ROI. If the node is labelled background, then there is no intersection between the square region it represents and the ROI. If the node is labelled mixed, then some of the pixels of the square region are pixels of the ROI and some are not. Only the mixed nodes in a quadtree have children. The full nodes and empty nodes are leaf nodes. Figure 2.4 illustrates a quadtree representation of an image region. The region looks blocky, because the resolution of the image is only 8x8, which leads to a four-level quadtree. Many more levels would be required to produce a reasonably smoothly curved boundary.

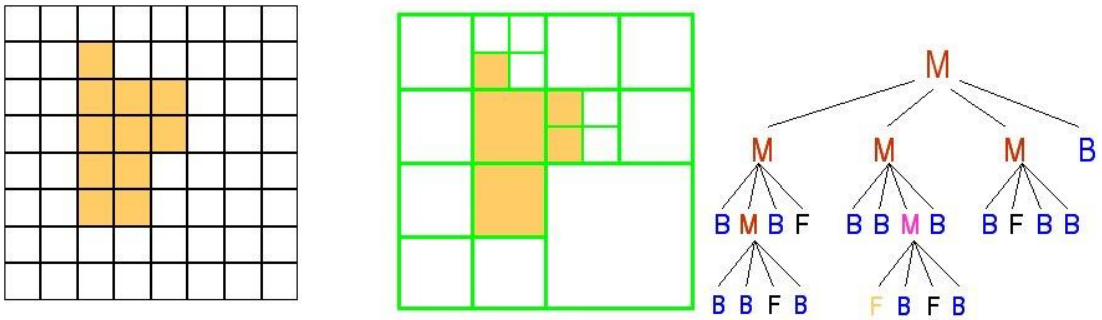


Figure 2.4 – A quadtree representation of an image region. The four children of each node correspond to the upper left, upper right, lower left, and lower right quadrants. M=mixed; F=foreground; and B=background.

For this quadtree representation, it uses the segmentation by partitioning an image by blocks. The disadvantage of this partitioning method is that the image is represented with only one level and it is difficult to select an appropriate block size. If the block is too small, the number of the blocks to be processed becomes too many. On the other hand, some detailed information may be missing if the block size is too large.

2.2.2 Binary Partition Tree representation

Binary partition trees concentrate in a compact and structured representation to a set of meaningful regions that can be extracted from an image (Radha *et al.*, 1996; Salembier & Garrido, 2000). They offer a multiscale representation of the image and define a translation invariant 2-connectivity rule among regions. A binary partition tree is a structured representation of the regions that can be obtained from an initial partition. An example is shown in Figure 2.5. The leaves of the tree represent regions that belong to the initial partition shown in Figure 2.5(b). The remaining nodes of the tree represent regions that are obtained by merging the regions represented by the two children of the node. The root node represents the entire image support. As can be seen, the tree represents a set of regions at different scales. Large regions appear close to the root whereas small details can be found at lower levels. In Figure 2.5(c), the leaf nodes at the second level correspond to the pink and brown areas of in Figure 2.5(b) and the lowest level correspond to the base level. This representation should be considered as a compromise between representation accuracy and processing efficiency. The connectivity encoded in the tree structure is binary in the sense that a region is explicitly connected to its sibling, but the remaining connections between regions of the original partition are not represented in

the tree. Therefore, the tree encodes only part of the neighbourhood relationship between the regions of the initial partition.

(a) (b) (c)

Figure 2.5 – Example of binary partition tree: (a) original image, (b) partitioned image with 6 regions, (c) a form of binary tree to represent the image

2.2.3 Computation of Binary Partition Tree

The most critical aspect of the binary tree construction process is to select the partitioning lines. At each stage of the recursive partitioning, the binary partition line selection consists of two major steps. First, since the number of lines that can partition the image region under consideration is large, one has to quantize the space of all possible lines that partition the region. Two parameters, i.e., the slope and the intercept are used for the representation of a line by $y = \alpha x + \beta$ where α and β denote the slope and y-intercept respectively. Assume that an image region has *N* number of points at the region boundary. All the partition lines can be formed by point-to-point connection at the boundaries of each image region. The maximum number of the line quantized in the whole image should be equal to or less than $\frac{1}{2}\left(\frac{N}{d}\right)^2$, where *d* denotes the quantization step-size. The number of line orientations (i.e. discrete values of slope) should be dependent on the size of the region of interest and the quantization step-size should be a function of the line orientation. This line-quantization process generates a finite set of lines so that each line needs to be considered. Second, after forming a finite set of quantized lines, it has to select a

partitioning line for a given region from the set of quantized lines based on a criterion. Note that different criteria will affect line selection and therefore the final binary tree constructed. If Λ_R represents the set of quantized lines considered for partitioning a region *R*, each line l_{ij} in set Λ_R should be tested for selecting the partitioning line to separate the region into two sub-regions. In this work, an entropy measure is used for the line selection. In this approach, a partitioning line is selected if this line minimizes the entropy of two sub-regions partitioned by the line. Let $H\left(R;l_{ij}\right)$ denote the entropy measure resulting from partitioning the image region *R* by line l_{ij}. The selected line $\hat{l}_{ij}$ satisfies the following condition:

$$\hat{l}_{ij} = \min_{l_{ij}\in\Lambda_R}\left\{H\left(R;l_{ij}\right)\right\}. \tag{2.2}$$

The entropy function $H\left(R;l_{ij}\right)$ of the image region is based on the color homogeneity of two sub-regions partitioned by the tested line, which can be defined as

$$H = -\left(\frac{N_A}{N_T}\sum_{k=1}^{M} p_k^A \log_2 p_k^A + \frac{N_B}{N_T}\sum_{k=1}^{M} p_k^B \log_2 p_k^B\right), \tag{2.3}$$

where N_A, N_B, and N_T are the number of pixel for the two partitioned sub-regions (A and B) and the total number of this testing region respectively. p_k represents the percentages of the pixels at the *k*-th colour in the region which can be defined by probability as

$$p_k(u) \triangleq \text{Prob}\{k=u\} \approx \frac{\text{number of pixels at } k\text{-th color}}{\text{total number of pixels in the region}}, \tag{2.4}$$

where $k = 1,\ldots,M$.

The colour spaces of the image are quantized to *M* colours. In most common cases, 64 colours are sufficient. As the aforementioned approach, one of the main advantages of this partitioning method is that the image signal within the unpartitioned regions can be characterized using simple features.

The above procedure is repeated recursively until the terminating criterion is reached. The binary tree construction is terminated if either the entropy of region is smaller than a preset tolerance or the number of nodes for the binary tree reaches a preset maximum number. For the binary tree representation, the characteristics of the image content within the region associated with the node are referred to the node attributes. The attributes of the root node are the global features of the whole image, and the attributes of the leaf nodes characterize the local features of the meaningful regions of the image. The attributes of an intermediate node reflect the characteristics of an intermediate region that contains two or more leaf regions. Various features such as color, texture and shape can be extracted from the region to characterize each node of the binary tree.

2.2.4 Region-based Tree representation

Similar to binary partition tree, region-based tree representation can offer an accurate representation which involves a number of regions that is much lower than the number of original pixels. The construction of image representation is based on the extraction of the relevant regions in the image. The extraction is typically obtained by a region-based segmentation in which the algorithm can extract the interesting regions of images. Once the regions of interest have been extracted, a node is added to the graph for each of these regions. Relevant regions to describe the objects can be merged together based on the merging strategy. A binary tree structure can be formed

as a semantic representation whose nodes correspond to the regions of the image and arcs represent the relationships among regions.

2.2.5 Region Merging Strategy for tree representation of images

The idea of creating and processing tree structure image representation is an attempt to take benefit from the attractive features of the segmentation results. Most of the studies are started from the terminated nodes and merge two similar neighbouring regions associated with the child nodes based on their contents. This merging is iteratively operated by a recursive algorithm until the children nodes of the root node (i.e., most likely representing the background and foreground regions). The following is explaining a simple merging strategy to create a binary tree from a color image. Assume that the merged regions are denoted as $O\left(R_i, R_j\right)\Big|_{i\neq j} \in \Omega_{i,j}$, where R_i, R_j for $i, j = 1 \ldots P$ denote the P regions and the entropy function is $M_{R_i \cup R_j}$ for a pair of regions (i.e. R_i and R_j) for the merging criterion. The merging criterion is based on examining the maximum entropy of all pairs of regions and the merging is terminated until the last pair of regions merged to become the entire image. At each step, the algorithm searches for the pair of most similar regions' contents, which should be the pair of child nodes linked with their parent node. The most similar regions pair is determined by maximizing the entropy:

$$O\left(R_i, R_j\right)\Big|_{i\neq j} = \arg \max_{O\left(R_i, R_j\right) \in \Omega_{i,j}} \left\{ M_{R_i \cup R_j}\Big|_{i\neq j} \right\}. \tag{2.5}$$

The entropy function $M_{R_i \cup R_j}$ of regions R_i and R_j is computed based on the color homogeneity of two sub-regions, which is defined as:

$$M_{R_i \cup R_j}\Big|_{i\neq j} = -\left(\frac{N_{R_i}}{N_T} \sum_{k=1}^{K} p_k^{R_i} \log_2 p_k^{R_i} + \frac{N_{R_j}}{N_T} \sum_{k=1}^{K} p_k^{R_j} \log_2 p_k^{R_j} \right), \tag{2.6}$$

where N_{R_i}, N_{R_j} are the number of pixels for two regions R_i and R_j, N_T is the total number of pixels for the parent region, K is the number of quantized colors and p_k represents the percentages of the pixels at the k[th] color in the region. The above computation is done recursively until no more regions can be merged. For a natural scene image as shown in Figure 2.6(a), the image is segmented into five regions, so the algorithm merges them in four steps. In the first step, suppose that the pair of most similar regions is regions '1' and '2', which can be merged to create 'a'. In the second step, node '3' is merged with region '5' to create 'b' corresponding to the foreground. Then, the created region 'a' is merged with region '4' to form region 'c' corresponding to the background region. Finally, node 'c' is merged with region 'b' to create region 'd' which is the root node corresponding to the whole image. The merging sequence is:

$$a = O(1,2) \rightarrow b = O(3,5) \rightarrow c = O(a,4) \rightarrow d = O(b,c), \tag{2.7}$$

and the tree constructed is shown in Figure 2.6(b).

(a) (b)

Figure 2.6 - Example of region merging to create a binary tree. (a) Five regions created by the segmentation method, (b) Four-levels binary tree

2.2.6 Notation of Tree Representation

In the example of Figure 2.6, suppose that a tree used for representing structural image information is a pair $T=(V,E)$, where V is a set of vertices ($[1,2,3,4,5],[a,b,c,d]$) and E is a binary relation on V. An edge $(v,w)\in E$ is directed if and only if (v,w) is in E but (w,v) is not in E. Therefore, the definition of a directed tree is: If an edge (v,w) is present in E, then v is a parent of w and w is a child of v. Vertex w is a descendant of vertex v if a path exists from v to w. In such a case, v is an ancestor of w. $pa[v]$ is denoted as the set of parent of v, and $ch[v]$ is denoted as the set of children of v. The outer-degree of vertex v is the number of edges (v,w) leaving from v and the outer-degree of vertex v is the number of edges incident on v. A vertex having zero in-degree is a root (node "d"). A vertex with zero inner-degree is a leaf. Vertex s is said to be a super-source for T if each vertex in $V\setminus\{s\}$ there exist a path from s to v. An image content structure developed in the previous section is a tree whose nodes and edges are marked by a set of domain variables, called labels. We assume that all the labels in a graph are disjoint sets. The domain variables contained into labels are called attributes, which may be numerical in continuous form or discrete form. These variables express features attached to a given node (usually features are a set of low level features, such as color, texture or shape). The presence of an edge (v,w) indicates that the variables contained in v and w is related. For the sake of simplicity, we assume that the edges are unlabeled. The attribute attached to node is assumed to be a real-valued representation with $\chi\subset\Re^m$ when connectionist models are used. The skeleton of a tree structure is obtained by ignoring all the labels, but retraining the topology of the tree, in which can be referred

to as the skeleton of *T*, denoted $skel(T)$. Clearly, any two tree structures can be distinguished because they have different skeleton, or if they have the same skeleton, but they have different node attributes (i.e., different image features).

Chapter 3

Adaptive Processing of Tree Structures

3.1 Background

As discussed in Chapter 2, tree structures models can be used to represent image content in a semantic manner that is just like human perception. In general, the tree-structure processing by some specified models can carry out on the sequential representation based upon the construction of trees. However, this approach has two major drawbacks. First, the sequential mapping of data structures, which are necessary to break some regularities inherently associated with the data structures, hence they will yield poor generalization. Second, since the number of nodes grows exponentially with the depth of the trees, a large number of parameters need to be learnt, which makes learning difficult and inefficient.

To overcome these difficulties, new recursive models have become increasingly important and one such example is the Labeling RAAM model (Pollack, 1990). New models such as this facilitate the adaptive processing of data structures (Tsoi, 1998). The benefit of adaptive processing is that it allows neural networks to classify static information, temporal sequences (Hammer, 2000) or structured patterns and to perform automatic inferring or learning. Incorporating the underlying structures that exist in most data processing efforts into the extracted features in the manner used for neural networks would facilitate the structural pattern recognition or

classification process. Sperduti and Startita proposed the use of supervised neural networks for such a classification of data structures (Sperduti & Starita, 1997). This approach is based on the use of generalized recursive neurons, with recursive here implying that copies of the same neural network are used to encode every node of the structured patterns. Each generalized recursive neuron receives two kinds of inputs that it then uses to generate its output. The first is the output of its node's children, while the second is the input attributes of related vertices that are provided by the structure of the underlying pattern. Such architecture has been shown to be useful in performing classification tasks involving structured patterns.

Significant advances have been made in this area (Frasconi *et al.*, 2001; Giles & Gori, 1998). Most significantly, a learning algorithm called a backpropagation through structure (BPTS) algorithm (Goller & Kuchler, 1996; Tsoi, 1998) was proposed. This algorithm extends a backpropagation through time (BPTT) algorithm (Rumelhart & McClelland, 1986) in order to encode data structures by using recursive neurons. In this BPTS algorithm, learning is performed on a structured pattern, such as on a directed acyclic graph (DAG). The gradients are calculated by backpropagating error through the data structures.

As promising as it appears however, this method has several drawbacks. Firstly, the convergence speed is so slow that it may not be completed within a reasonable time. Secondly, the algorithm is prone to local minima (Gori & Tesi, 1992) and it is extremely difficult for the recursive model to learn a very deep tree structure. This is known commonly as the long short-term memory (Hochreiter & Schmidhuber, 1997) or the long-term dependency problem (Bengio *et al.*, 1994). It is due mainly to the fact that when the error backpropagates through the deep tree structure, the gradient contribution disappears at a certain tree level. It latches on to the terminal

nodes as the gradient disappears. The activation function is usually a sigmoidal or hyper-tangent of derivatives between 0 and 1. Because the backpropagating error is recursively multiplied by the derivative of the activation function in each neuron, the decreasing gradient term tends to zero. The decreasing gradient may then result in the stalling of the convergence and yield a poor generalization of the structured patterns.

Recently, an improved algorithm was proposed to solve the above-mentioned problems (Cho *et al.*, 2003). The algorithm was created to optimize the free parameters in the generalized recursive neuron by conducting least-squares optimization on one layer after another. Cho *et al.* (2003) reported not only achieving a fast learning speed, but also tackling the long-term dependency problem common in learning efforts involving very deep tree structures.

While supervised recursive neural network architecture could indeed be successfully developed for structural pattern classification, the use of unsupervised learning models is a viable alternative paradigm for the adaptive processing of tree structures. Given that self-organizing maps (SOM) (Kohonen, 1995) including visualization-induced (Yin, 2002) and probabilistic regularized SOMs (Wu & Chow, 2005) can preserve topological features and perform data clustering in any data domain, an extension of the standard SOM would be valuable to deal with complex tree structures in an unsupervised fashion. An unsupervised neural network approach to structured pattern was described in (Goller *et al.*, 1999). A maximum entropy method was used to extract features in the form of vector representations of the DAG in an unsupervised manner. Taking things a little further, Hagenbuchner and Tsoi (2003) proposed a fully unsupervised SOM model as this would allow structured objects to be mapped directly onto a topological map. Their approach was adopted in the contextual SOM (Voegtlin, 2000), such that recurrent connections were used

between the units in the map to create a specific data structure model. Most recently, Hammer *et al*. (2004) defined a general framework for the unsupervised processing of structured data. They showed how the SOM can be generalized with recursive dynamics and how unsupervised learning can be derived for this SOM. Although these unsupervised SOM models are able to deal with structural pattern classifications without requiring supervised information, data clustering might not always be available and discriminative information was found to be very difficult to obtain during the learning process. Without acquiring discriminative information from the learning patterns, the absence of *a priori* class characterization will degrade the ability of the system to discriminate between classes.

This chapter describes the basic idea of the adaptive processing of tree structures approach to represent and recognize structured patterns. The model, learning algorithm and its problems are presented. In addition, some other advanced approaches are briefly discussed at the last part of this chapter.

3.2 Neural Networks for Processing Tree Structures

In recent years, neural networks for the representation and processing of tree structures have been developed. This kind of networks is of paramount importance both in structural pattern recognition and for the development of hybrid systems, since they allow the treatment of structured information. The problem of devising neural network architectures and learning algorithms for the adaptive processing of data structure is addressed in the content of classification of structured patterns. The encoding method by recursive neural networks is based on and modified by the research works of (Sperduti & Starita, 1997; Tsoi, 1998). We consider that a structured domain D and all graphs (the tree is a special case of the graph). In the following discussion, we will use either graph or tree when it is appropriate. G is a

learning set representing the task of the adaptive processing of data structures. This representation by the recursive neural network is shown in Figure 3.1.

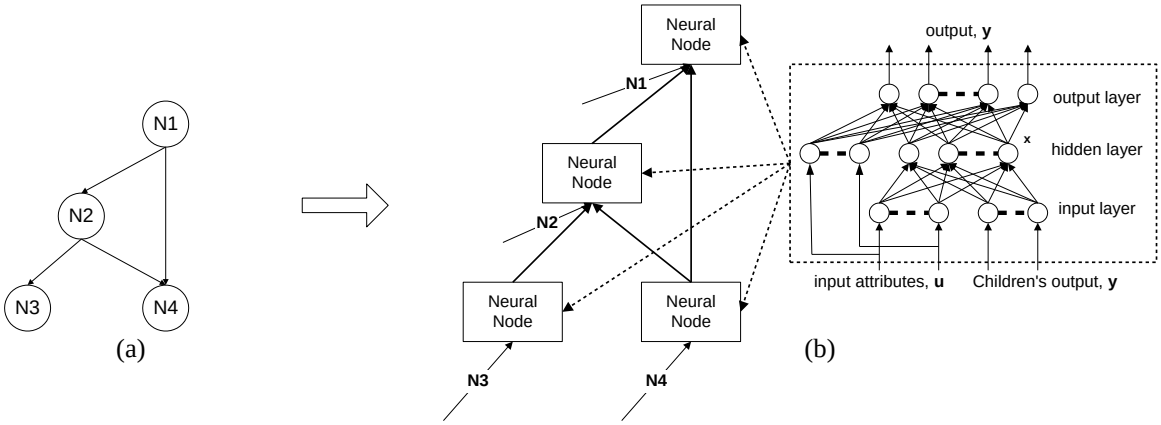


Figure 3.1 - An illustration of a data structure with its nodes encoded by a single-hidden-layer neural network. (a) a Directed Acyclic Graph (DAG); (b) The encoded DAG.

As shown in Figure 3.1, a copy of the same neural network (shown on the right-side of Figure 3.1(b)) is used to encode every node in the graph *G*. Such an encoding scheme is flexible enough to allow the model to deal with DAG's of different internal structures and with a different number of nodes. Moreover, the model can also naturally integrate structural information into its processing. In the Directed Acyclic Graph (DAG) shown in Figure 3.1(a), the operation is run forward for each graph, i.e., from terminals nodes (N3 and N4) to the root node (N1). The maximum number of children for a node (i.e., the maximum branch factor *c*) is predefined for a task domain. For instance, a binary tree (each node has two children only) has a maximum branch factor *c* equal to two. At the terminal nodes, there will be no inputs from children. Therefore, the terminal nodes are known as frontier nodes. The forward recall is in the direction from the frontier nodes to the root in a bottom up fashion. The bottom up processing from a child node to its parent node can be denoted by an operator q^{-1}. Suppose that a maximum branch factor of *c* has been pre-defined, each of the form q_i^{-1}, $i = 1, 2, \ldots c$, denotes the input from the *i*th child node

into the current node. This operator is similar to the shift operator used in the time series representation. Thus, the recursive network for the structural processing is formed as

$$\mathbf{x} = F_n\left(\mathbf{A}q^{-1}\mathbf{y} + \mathbf{B}\mathbf{u}\right), \tag{3.1}$$

$$\mathbf{y} = F_p\left(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}\right), \tag{3.2}$$

where **x**, **u**, and **y** are the *n* dimensional output vector of the *n* hidden layer neurons, the *m* dimensional inputs to the neurons, and the *p* dimensional outputs of the neurons respectively. q^{-1} is a notation indicating that the input to the node is taken from its child so that,

$$q^{-1}\mathbf{y} = \begin{pmatrix} q_1^{-1}\mathbf{y} \\ q_2^{-1}\mathbf{y} \\ \vdots \\ q_c^{-1}\mathbf{y} \end{pmatrix}. \tag{3.3}$$

The parametric matrix **A** is defined as follows:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}^1 & \mathbf{A}^2 & \cdots & \mathbf{A}^c \end{pmatrix}, \tag{3.4}$$

where *c* denotes the maximum number of children in the graph. **A** is a $n \times (c \times p)$ matrix such that each $\mathbf{A}^k$, $k = 1, 2, \ldots, c$ is a $n \times p$ matrix, which is formed by the vectors $\mathbf{a}_j^i$, $j = 1, 2, \ldots, n$. **B**, **C**, and **D** are respectively $(n \times m)$, $(p \times n)$, and $(p \times m)$ dimensional matrices. $F_n(\cdot)$ and $F_p(\cdot)$ are *n* and *p* dimensional parametric vectors respectively given as follows:

$$F_n(\alpha) = \begin{pmatrix} f(\alpha_1) \\ f(\alpha_2) \\ \vdots \\ f(\alpha_n) \end{pmatrix}, \tag{3.5}$$

where $f(\alpha)$ is the nonlinear function defined as $f(\alpha) = 1/\left(1 + e^{-\alpha}\right)$.

3.3 BackPropagation Through Structure (BPTS) algorithm

In accordance with the research work by Hammer and Sperschnedier (1997), based on the theory of the universal approximation of the recursive neural network, a single hidden layer is sufficient to approximate any complicated mapping problems. The input-output learning task can be defined by estimating the parameters **A**, **B**, **C**, and **D** in the parameterization from a set of training (input-output) examples. Each input-output example can be formed in a tree data structure consisting of a number of nodes with their inputs and target outputs. Each node's inputs are described by a set of attributes **u**. The target output is denoted by **t**, where **t** is a *p* dimensional vector. So, the cost function is defined as a total sum-squared-error function:

$$J = \frac{1}{2}\sum_{i=1}^{N_T}\left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T\left(\mathbf{t}_i - \mathbf{y}_i^R\right), \tag{3.6}$$

where N_T is the total number of the learning data structures. $\mathbf{y}^R$ denotes the output at the root node. Note that in the case of structural learning processing, it is often assumed that the attributes, **u**, are available at each node of the tree. The main step in the learning algorithm involves the following gradient learning step:

$$\theta(k+1) = \theta(k) - \eta \left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta(k)}, \tag{3.7}$$

where $\theta(k)$ denotes the free learning parameters $\theta:\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ at the *k*th iteration and η is a learning rate. $\left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta(k)}$ is the partial derivative of the cost function with respect to θ evaluated at $\theta = \theta(k)$. The derivation of the learning algorithm involves the evaluation of the partial derivative of the cost function with respect to the parameters in each node. Thus, the general form of the derivatives of the cost function with respect to the parameters is given by:

$$\frac{\partial J}{\partial \theta} = -\sum_{i=1}^{N_T} \left(\mathbf{t} - \mathbf{y}_i^R\right)^T \Lambda\left(\mathbf{y}_i^R\right) \delta\left(\nabla_\theta \mathbf{x}_i\right), \tag{3.8}$$

where $\Lambda(\mathbf{y})$ is a $p \times p$ diagonal matrix defined by the first derivative of the nonlinear activation function. δ is defined as *n*-dimensional vector which is the function of the derivative of **x** with respect to the parameters. It can be evaluated as:

$$\nabla_\theta \mathbf{x} = \Lambda(\mathbf{x}) \mathbf{A} q^{-1} \frac{\partial \mathbf{y}}{\partial \theta}, \tag{3.9}$$

where $\Lambda(\mathbf{x})$ is a $n \times n$ diagonal matrix defined in a similar manner as $\Lambda(\mathbf{y})$. It is noted that $q^{-1} \frac{\partial \mathbf{y}}{\partial \theta}$ essentially repeats the same computation such that the evaluation depends on the structure of the tree. This is so called either the folding architecture algorithm or backpropagation through structure algorithm.

In the formulation of the learning structural processing task, it is not required to assume *a priori* knowledge of any data structures or any *a priori* information concerning the internal structures. However, we need to assume the maximum number of children for each node in the tree is pre-defined. The parameterization of the structural processing problem is said to be an over-parameterization if the pre-defined maximum number of children is so much greater than that of real trees, i.e., there are many redundancy parameters in the recursive network than required to describe the behavior of the tree. The over-parameterization may give rise to the problem of local minima in the BPTS learning algorithm. Moreover, the long-term dependency problem may also affect the learning performance of BPTS approach due to the vanishing gradient information in learning deep trees. The learning information may disappear at a certain level of the tree before it reaches at the frontier nodes so that the convergence of the BPTS stalls and a poor generalization results. A detailed analysis of this problem will be given in the next section.

3.4 Long-Term Dependency Problem

For backpropagation learning of multi-layer perceptron (MLP) networks, it is well known that if there are too many hidden layers, the parameters at very deep layers are not updated. This is because backpropagating errors are multiplied by the derivative of the sigmoidal function, which is between 0 and 1 and hence the gradient for very deep layers could become very small. Bengio et al. (1994) have analytically explained why backpropagation learning problems with the long-term dependency are difficult. They stated that the recurrent MLP network is able to robustly store information for an application of long temporal sequences when the states of the network stay within the vicinity of a hyperbolic attractor, i.e., the eigenvalues of the Jacobian are within the unit circle. However, Bengio et al. have shown that if its eigenvalues are inside the unit circle, then the Jacobian at each time step is an exponentially decreasing function. This implies that the portion of gradients becomes insignificant. This behavior is called the effect of vanishing gradient or forgetting behavior (Bengio *et al.*, 1994). In this section, we briefly describe some of the key aspects of the long-term dependency problem learning in the processing of data structures. The gradient based learning algorithm updates a set of parameters $\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ in the recursive neural network for node representation defined in eqns. (3.1) and (3.2) such that the updated parameter can be denoted as

$$\Delta\theta = \eta \nabla_\theta J \,, \tag{3.10}$$

where η is a learning rate and ∇_θ is the matrix defined as

$$\nabla_\theta = \left[\frac{\partial}{\partial \theta_1} \quad \frac{\partial}{\partial \theta_2} \quad \cdots \quad \frac{\partial}{\partial \theta_n} \right]. \tag{3.11}$$

By using the chain rule, the gradient can be expressed as:

$$\nabla_\theta J = -\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \nabla_{\mathbf{x}^R} \mathbf{y}_i^R \nabla_\theta \mathbf{x}^R \,. \tag{3.12}$$

If we assume that computing the partial gradient with respect to the parameters of the node representation at different levels of a tree is independent, the total gradient is then equal to the sum of these partial gradients as:

$$\nabla_\theta J = -\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \nabla_{\mathbf{x}^R} \mathbf{y}_i^R \cdot \left(\sum_{l=1}^{R} J_{\mathbf{x}}^{R,R-l} \nabla_{\theta^l} \mathbf{x}^l \right), \tag{3.13}$$

where $l = 1 \ldots R$ represents the levels of a tree and $J_{\mathbf{x}}^{R,R-l} = \nabla_{\mathbf{x}^l} \mathbf{x}^R$ denotes the Jacobian of (3.6) expanded over a tree from level R (root node) to l backwardly. Based on the idea of Bengio et al. (1994), the Jacobian $J_{\mathbf{x}}^{R,n}$ is an exponentially decreasing function of n since the backpropagating error is multiplied by the derivative of the sigmoidal function which is between 0 and 1, so that $\lim_{n\to\infty} J_{\mathbf{x}}^{R,n} = 0$. This implies that the portion of $\nabla_\theta J$ at the bottom levels of trees is insignificant compared to the portion at the upper levels of trees. The effect of vanishing gradients is the main reason why the BPTS algorithm is not sufficiently reliable for discovering the relationships between desired outputs and inputs, which we term the problem of long-term dependency. Therefore, we are now proposing a new method to avoid this effect of vanishing gradients by the BPTS algorithm so that the evaluation for updating the parameters becomes more robust in the problem of deep tree structures.

3.5 An improved algorithm for BPTS

Cho *et al.* (2003) have proposed an improved algorithm to tackle the long term dependency problem by BPTS algorithm. This algorithm is modified by using linear least squares method in a layer-by-layer fashion so that the convergence of the algorithm can be accelerated. The updated parameters do not need to be evaluated by

the total gradients through data structures so that the problem of long-term dependency can be eliminated. In their work, suppose that a multi-layer preceptron network with a single hidden layer network is adopted to encode each node in a tree shown by equations (3.1) to (3.5). The free learning parameters of $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ are optimized to generalize the data structures. In our study, the cost function of equation (3.6) can be rewritten as:

$$J=\frac{1}{2}\sum_{i=1}^{N_T}\left(\mathbf{d}_i^R-\begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{x}_i^R\\ \mathbf{u}_i^R\end{bmatrix}\right)^T\left(\mathbf{d}_i^R-\begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{x}_i^R\\ \mathbf{u}_i^R\end{bmatrix}\right), \tag{3.14}$$

where $\mathbf{d}_i^R=F_p^{-1}(\mathbf{t}_i)$ which is the inverse function of the target output at the root node. The task of the learning algorithm is to optimize the parameters of the network by minimizing the cost function as shown below:

$$\min\ J=\min\ \frac{1}{2}\left\|\mathbf{d}^R-\mathbf{V}\cdot\mathbf{X}^R\right\|^2, \tag{3.15}$$

where $\|\cdot\|$ denotes the Euclidean norm. $\mathbf{d}^R=\left(\mathbf{d}_1^R,\ldots,\mathbf{d}_{N_T}^R\right)$ and $\mathbf{X}^R=\begin{pmatrix}\mathbf{x}_1^R,\ldots,\mathbf{x}_{N_T}^R\\ \mathbf{u}_1^R,\ldots,\mathbf{u}_{N_T}^R\end{pmatrix}$ represent the matrices of inversed function of the target output and the input patterns of the output layer, respectively. The matrix $\mathbf{V}=\begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}$ denotes the parameters of the output layer of the network. All the derivatives of J with respect to $\mathbf{V}$ are,

$$\frac{\partial J}{\partial \mathbf{V}}=-\left(\mathbf{d}^R\mathbf{X}^{RT}-\mathbf{V}\cdot\mathbf{X}^R\mathbf{X}^{RT}\right). \tag{3.16}$$

The optimal values of the parameters of the output layer of the network can be obtained by solving $\frac{\partial J}{\partial \mathbf{V}}=\mathbf{0}$:

$$\mathbf{V}=\begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}=\left(\mathbf{X}^R\mathbf{X}^{RT}\right)^{-1}\left(\mathbf{d}^R\mathbf{X}^{RT}\right), \tag{3.17}$$

where "-1" represents the pseudo-inverse operation.

As the cost function in (3.14) is convex with respect to all parameters, the optimum values of these parameters can be determined by the linear least squares method so that the convergence of the algorithm is accelerated. Similarly, the parameters matrix of $\mathbf{W} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$ is also updated by the layer-by-layer least squares method. The optimum values are again evaluated by taking the derivatives of the cost function J with respect to the parameters matrix $\mathbf{W}$ as:

$$\frac{\partial J}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{A}} & \frac{\partial J}{\partial \mathbf{B}} \end{bmatrix} = \mathbf{0}. \tag{3.18}$$

The derivatives of J with respect to $\mathbf{A}$ and B are respectively as,

$$\frac{\partial J}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{a}_n} \end{bmatrix}, \text{ and } \frac{\partial J}{\partial \mathbf{B}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{b}_n} \end{bmatrix}. \tag{3.19}$$

By the chain rule,

$$\frac{\partial J}{\partial \mathbf{a}_j} = \left(\sum_{i=1}^{N_T} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}_j} \cdot \frac{\partial J}{\partial \mathbf{x}_i} \right)^T, \tag{3.20}$$

and

$$\frac{\partial J}{\partial \mathbf{b}_j} = \left(\sum_{i=1}^{N_T} \frac{\partial \mathbf{x}_i}{\partial \mathbf{b}_j} \cdot \frac{\partial J}{\partial \mathbf{x}_i} \right)^T \quad j = 1 \ldots n. \tag{3.21}$$

We can obtain the following equations:

$$\frac{\partial J}{\partial \mathbf{x}_i} = -\mathbf{C}^T \left(\mathbf{d}_i^R - \mathbf{C}\mathbf{x}_i \right), \tag{3.22}$$

$$\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}_j} = \left[\Lambda(\mathbf{x}) \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i \right) + \mathbf{A} q^{-1} \frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j} \right) \right]^T, \tag{3.23}$$

and

$$\frac{\partial \mathbf{x}_i}{\partial \mathbf{b}_j} = \left[\Lambda(\mathbf{x}) \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A} q^{-1} \frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j} \right) \right]^T, \tag{3.24}$$

where $\Lambda(\mathbf{x})$ is a $n\times n$ diagonal matrix defined by the first derivative of the sigmoid activation function. $\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)=\mathbf{Q}_y\cdot diag\left(q^{-1}\mathbf{y}_i\right)$ and $\mathbf{H}(\mathbf{u}_i)=\mathbf{Q}_u\cdot diag(\mathbf{u}_i)$, where $\mathbf{Q}_y$ and $\mathbf{Q}_u$ denote $n\times(c\times p)$ and $n\times m$ matrices respectively with all elements being 1's. $diag\left(q^{-1}\mathbf{y}_i\right)$ is a $(c\times p)\times(c\times p)$ diagonal matrix of $q^{-1}\mathbf{y}_i$ and $diag(\mathbf{u}_i)$ is a $m\times m$ diagonal matrix of $\mathbf{u}_i$.

Let $\mathbf{e}_i^R=\left(\mathbf{d}_i^R-\mathbf{C}\mathbf{x}_i\right)$ which defines the linear error signal between the outputs of desired state and the root node, eqns. (3.20) and (3.21) become, respectively,

$$\frac{\partial J}{\partial \mathbf{a}_j}=-\sum_{i=1}^{N_T}\mathbf{e}_i^{RT}\mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j}\right), \tag{3.25}$$

and

$$\frac{\partial J}{\partial \mathbf{b}_j}=-\sum_{i=1}^{N_T}\mathbf{e}_i^{RT}\mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}(\mathbf{u}_i)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j}\right). \tag{3.26}$$

If

$$\mathbf{e}_i^{hidT}=\mathbf{e}_i^{RT}\mathbf{C}\Lambda(\mathbf{x}), \tag{3.27}$$

denotes the error signal between the desired and the output states at the hidden layer, then the desired state at the hidden layer can be defined as $\mathbf{z}_i=\left(\mathbf{x}_i+\mathbf{e}_i^{hid}\right)$. In our study, $\mathbf{z}_i$ can be approximated by a first-order Taylor series as,

$$\begin{aligned}\mathbf{z}_i(\mathbf{A},\mathbf{B})&\approx \mathbf{z}_i(\mathbf{A}_0,\mathbf{B}_0)+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\ldots\\ &\approx\left(\mathbf{A}_0q^{-1}\mathbf{y}_i+\mathbf{B}_0\mathbf{u}_i+\mathbf{e}_i^{hid}\right)+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\ldots\\ &\approx\left(\mathbf{A}_0q^{-1}\mathbf{y}_i+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j+\ldots\right)+\left(\mathbf{B}_0\mathbf{u}_i+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\ldots\right)+\mathbf{e}_i^{hid}\\ &\approx\mathbf{z}_i(\mathbf{A})+\mathbf{z}_i(\mathbf{B})+\mathbf{e}_i^{hid}\end{aligned}. \tag{3.28}$$

Therefore, the following two approximated functions can be obtained,

$$\mathbf{z}_i(\mathbf{A})\approx\hat{\mathbf{A}}q^{-1}\mathbf{y}_i+\gamma\mathbf{e}_i^{hid}, \tag{3.29}$$

$$\mathbf{z}_i(\mathbf{B}) \approx \hat{\mathbf{B}}\mathbf{u}_i + (1-\gamma)\mathbf{e}_i^{hid}, \tag{3.30}$$

where γ denotes the portion of error contributions. In our experimental study, γ is set to 0.5. Therefore, from equations (3.25), (3.28) and (3.29), the derivatives of J with respect to $\mathbf{A}$ can be written as,

$$\frac{\partial J}{\partial \mathbf{A}} = -2 \cdot \begin{bmatrix} \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{A}) - \hat{\mathbf{A}} q^{-1}\mathbf{y}_i\right)^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{A}) - \hat{\mathbf{A}} q^{-1}\mathbf{y}_i\right)^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right) \end{bmatrix}. \tag{3.31}$$

$$\text{If} \quad \begin{bmatrix} \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{A}) - \hat{\mathbf{A}} q^{-1}\mathbf{y}_i\right)^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{A}) - \hat{\mathbf{A}} q^{-1}\mathbf{y}_i\right)^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{3.32}$$

$$\text{then } \hat{\mathbf{A}} = \begin{bmatrix} \sum_{i=1}^{N_T} q^{-1}\mathbf{y}_i{}^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right)^T \\ \vdots \\ \sum_{i=1}^{N_T} q^{-1}\mathbf{y}_i{}^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right)^T \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{i=1}^{N_T} \mathbf{z}_i(\mathbf{A})^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \mathbf{z}_i(\mathbf{A})^T \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right) \end{bmatrix}, \tag{3.33}$$

where the desired state at the hidden layer is approximated as $\mathbf{z}_i(\mathbf{A}) = \mathbf{A}q^{-1}\mathbf{y}_i + 0.5\mathbf{e}_i^R \mathbf{C}\Lambda(\mathbf{x})$. The derivative $q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j} = \begin{bmatrix} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^1} \\ \vdots \\ \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^c} \end{bmatrix}$ is a $(c \times p) \times (c \times p)$ matrix with the output gradients of the child nodes with respect to the $\mathbf{A}$ parameters. Generally, if the number of tree level, $l = 1, \ldots, L-1$, where L is the total tree level, the derivative,

$$\frac{\partial q^{-l}\mathbf{y}_i}{\partial \mathbf{a}_j^c} = \mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}\left(q^{-(l+1)}\mathbf{y}_i\right) + \mathbf{A}q^{-(l+1)}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j^c}\right). \tag{3.34}$$

Similarly, the derivatives of J w.r.t the parameters B from eqns (3.20), (3.26) and (3.30):

$$\frac{\partial J}{\partial \mathbf{B}} = -2\cdot\begin{bmatrix} \sum_{i=1}^{N_T}\left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T}\left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix}. \tag{3.35}$$

If $$\begin{bmatrix} \sum_{i=1}^{N_T}\left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T}\left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{3.36}$$

Then $$\hat{\mathbf{B}} = \begin{bmatrix} \sum_{i=1}^{N_T}\mathbf{u}_i^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right)^T \\ \vdots \\ \sum_{i=1}^{N_T}\mathbf{u}_i^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right)^T \end{bmatrix}^{-1}\cdot\begin{bmatrix} \sum_{i=1}^{N_T}\mathbf{z}_i(\mathbf{B})^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T}\mathbf{z}_i(\mathbf{B})^T\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix}, \tag{3.37}$$

where the desired state at the hidden layer are approximated as $\mathbf{z}_i(\mathbf{B}) = \mathbf{B}\mathbf{u}_i + 0.5\mathbf{e}_i^R\mathbf{C}\Lambda(\mathbf{x})$. The derivative $q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j} = \begin{bmatrix} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{b}_j^1} \\ \vdots \\ \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{b}_j^c} \end{bmatrix}$ is a $(c\times p)\times m$ matrix with the output gradients of the child nodes with respect to the **B** parameters. Again, if the number of tree level, $l = 1,\ldots,L-1$, where L is the total tree level, the derivative,

$$\frac{\partial q^{-l}\mathbf{y}_i}{\partial \mathbf{b}_j^c} = \mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-(l+1)}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j^c}\right), \tag{3.38}$$

Computationally, the evaluations of these two derivatives (in equations. (3.34) and (3.38)) depend on the structure of the tree and essentially repeats the computations propagating through the structure backwardly. For instance, the total derivatives of the output nodes with respect to the **A** parameters for the whole structure are, generally, defined as:

$$\frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^c} = \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}\left(q^{-2}\mathbf{y}_i\right) + \mathbf{C}\Lambda(\mathbf{x})\mathbf{A}\cdot\left(\mathbf{C}\Lambda(\mathbf{x})\begin{pmatrix}\mathbf{H}\left(q^{-3}\mathbf{y}_i\right)+\cdots\cdots \\ \mathbf{A}\cdot\left(\mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}\left(q^{-(L-1)}\mathbf{y}_i\right)+\mathbf{A}\right)\right)\end{pmatrix}\right)\cdot \tag{3.39}$$

From the above discussion, suppose we are dealing with an extremely deep tree structure by this proposed algorithm, let $L \to \infty$, so the total derivatives can be simplified by taking:

$$\lim_{L\to\infty} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^c} \approx \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}\left(q^{-2}\mathbf{y}_i\right), \tag{3.40}$$

as the expression at right-hand-side tends to be zero because of infinite multiples of $\Lambda(\mathbf{x})$ (it is noted that all elements of $\Lambda(\mathbf{x})$ are between 0 to 1). Similarly, the total derivatives of the output nodes with respect to the **B** parameters for the whole structure are obtained by the same manner as:

$$\lim_{L\to\infty} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{b}_j^c} \approx \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}\left(\mathbf{u}_i\right). \tag{3.41}$$

Based on the above evaluations, the effect of vanishing gradients can be avoided so that the problem of long-term dependency can be eliminated for the processing in a deep tree structure.

In accordance with the above updated equations (3.17), (3.33) and (3.37), the parameters (**A**, **B**, **C**. and **D**) of the node representation can be evaluated by the least squares method and a sub-optimal solution can be found in a few iterations. This is

fairly efficient when operating under simple convex error surfaces with only a unique minimum. But under highly convoluted non-convex surfaces encountered in practical problems of data structure representation, the least squares methods often lead to non-optimal or unacceptable solutions. Once the algorithm is trapped into a non-optimal state, the learning performance cannot be improved by increasing the number of iterations; hence the convergence stalls. Thus, in this paper, a heuristic mechanism is proposed to provide a force trajectory that will yield increasing opportunities of obtaining an optimal solution. Our proposed mechanism introduces a penalty term into the sub-optimal solutions when the convergence stalls. The element of penalty term is generated by a Gaussian random generator to provide a random search in the corresponding parameters space. The search is performed repeatedly until the cost function converges again outside the sub-optimal state. Afterwards, the learning procedure continues to search locally by using the least squares method. According to the above statements, the updated set of parameters $\theta:\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ formulation by the heuristic can be written as:

$$\theta^{**} = \hat{\theta}(k-1) + \alpha\left(\theta^{*} - \hat{\theta}(k-1)\right) + \beta\Phi, \tag{3.42}$$

where θ^{**} represents a new solution of the estimated parameters of the node representations. $\hat{\theta}(k-1)$ is the solution at the previous iteration and θ^{*} represents the sub-optimal state. α is a positive constant which is less than but close to unity. $\Phi = \mathbf{U}\cdot\left\|\frac{\partial J}{\partial \theta}\right\|$, where $\mathbf{U}$ denotes a matrix with the same size as the estimated parameters, which consists of a set of unit vectors generated by the Gaussian random generator to provide a random search direction to force trajectory out of the sub-optimal state when the convergence stalls. $\left\|\frac{\partial J}{\partial \theta}\right\|$ is a norm of the total gradient

matrices through the whole structure which is evaluated by the components of the derivatives shown in (3.40) and (3.41) approximately. The step-size of the search is denoted by β of which its value increases gradually by a small fraction to enhance the searching ability. The increasing quantity of the step-size is a critical consideration according to the following conditions:

1. If the increase in β is too large, a significant change in the estimated parameters is possible because of the strong effect of the penalty term. In this case, the algorithm is unstable since it may result in an invalid solution.
2. If the increase in β is too small, only a small change in the parameters is allowed for the trajectory to escape because the penalty term is virtually redundancy.

As a result, in their experimental study, the quantity of step-size β is increased by 0.5% of the previous values until an escape from the sub-optimal state is achieved.

3.6 Unsupervised model for adaptive processing of structured patterns

Hagenbuchner *et al.* (2003) proposed the first fully unsupervised model, which is an extension of traditional self-organization maps (SOMs), for the processing of labelled directed acyclic graphs (DAGs). This extension was obtained by using the unfolding procedure adopted in recurrent and recursive neural networks, with the replicated neurons in the unfolded network comprising of a full SOM. This approach enables the discovery of similarities among objects including vectors consisting of numerical data. In the basis of their works, assume the SOM learning algorithm is to learn a feature map as:

$$\Re : I \to F , \tag{3.43}$$

which given a vector in the spatially continuous input space I returns a point in the spatially discrete output display space F. This is obtained in the SOM by associating each point in F with a different neuron. Moreover, the output space F topology is typically endowed by arranging this set of neurons as the computation nodes of a one- and two-dimensional lattice. However, in the case where the input space is in a structured domain, the realized function:

$$\Re^{\#} : I^{\#(c)} \to F, \tag{3.44}$$

is specified. Given a training set $T = \{(D_i, t_i)\}_{i=1,\ldots,N}$, where for each structured pattern D_i a desired target value t_i is associated. Then we can define:

$$\Re^{\#}(D) = \begin{cases} \text{nil}_F, & \text{if } D = \Im \\ \Re_{\text{node}}\left(y_s, \Re^{\#}\left(D^{(1)}\right), \ldots, \Re^{\#}\left(D^{(c)}\right)\right), & \text{otherwise} \end{cases}, \tag{3.45}$$

where nil_F is a special coordinate vector into the discrete output space F, and

$$\Re_{\text{node}} : Y \times \underbrace{F \times \cdots \times F}_{\text{c times}} \to F, \tag{3.46}$$

is a SOM, defined on a generic node, which takes as input the label of the node and the "encoding" of the subgraphs $D^{(1)}, \ldots, D^{(c)}$ according to the $\Re^{\#}$ map. By "unfolding" the recursive definition in (3.45), it turns out that $\Re^{\#}(D)$ can be computed by starting to apply $\Re_{\text{node}}$ to leaf node and proceeding with the application of bottom-up from the frontier nodes to the root node of the tree. $\Re_{\text{node}}$ is playing the same role of the function as shown in (3.1) and (3.2) with the difference that the equation (3.1) returns a real-valued vector representing a reduced descriptor of the node, while $\Re_{\text{node}}$ returns the coordinates of the winning neuron, which is due to the data reduction capability of the SOM, still constitutes a reduced descriptor of the node.

3.7 Genetic evolution processing of structured patterns

Cho and Chi (2005) have introduced an adaptive and global approach to learning by genetic evolution neural network for processing of tree structures. In their study, the major objective is to determine the parameters $\theta:\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ of the recursive neural network over the whole data structures. Their proposed genetic approach consists of two major considerations. The first one is to consider the string representation of the parameters. Based on two different string structures, the objection function for fitness criterion is the other main consideration. Different string representations and object functions can lead to quite different learning performance. Their proposed genetic evolution framework for processing of tree structures is shown in Figure 3.2.

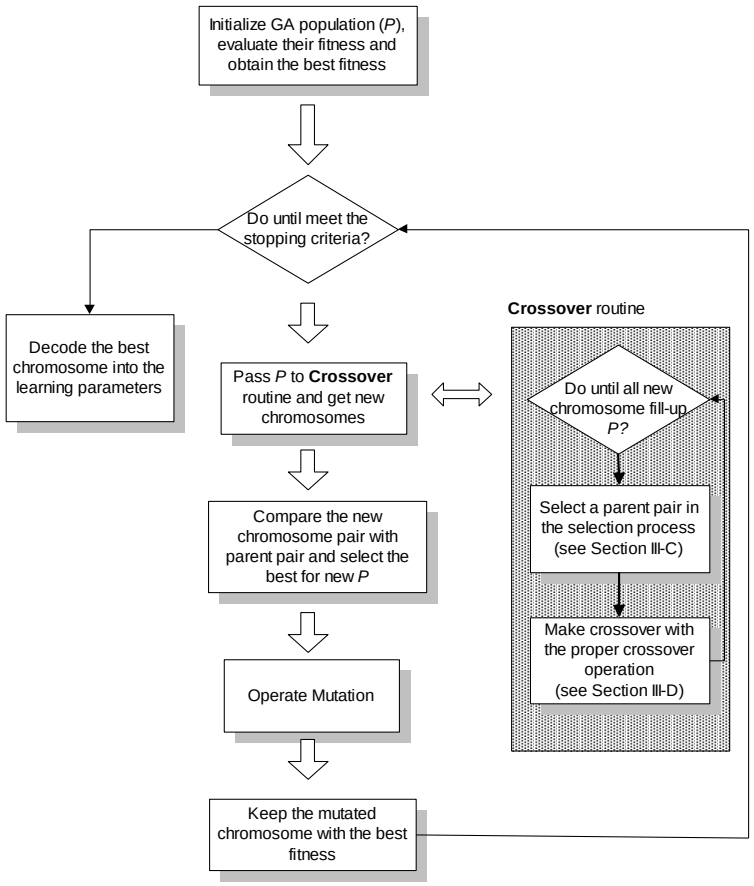


Figure 3.2 – The genetic evolution framework for processing of tree structures (adopted from Cho and Chi, 2005)

In their presented string structure representation, a proper string structure for GA operations is selected depending on fitness evaluation. One of a simple way is a

'whole-in-one' structure in which all parameters are encoded into one long string. The encoding for the 'whole-in-one' structure is simple and the objective function is simply evaluated by the error between the target and the root output values of data structures. But the dimension may be very high so that the GA operations may be inefficient. Moreover, this 'whole-in-one' structure representation has the permutation problem. It is caused by the many-to-one mapping from the chromosome representation to the recursive neural network since two different networks have equivalent function but they have different chromosomes. This permutation problem makes the crossover operator very inefficient and ineffective in producing good offspring. Thus, another string structure representation called '*4-parallel*' structure is used to overcome the above problem. The GA process becomes efficient when we apply it over each group of parameters individually. It is likely to perform a separate GA process on each group of parameters in parallel, but the limitation lies on its inability of performing the correlation constrains among the learning parameters of each node. The objective function is essentially designed for this '*4-parallel*' string structure so as to evaluate the fitness criteria for GA operations of structural processing.

The recursive network for this genetic evolution processing can be rewritten in matrices form as

$$\begin{pmatrix} \mathbf{x} & \mathbf{h}_1 \\ \mathbf{h}_2 & \mathbf{y} \end{pmatrix} = F\left\{ \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \cdot \begin{pmatrix} q^{-1}\mathbf{y} & \mathbf{x} \\ \mathbf{u} & \mathbf{u} \end{pmatrix} \right\}. \tag{3.47}$$

Note that $\mathbf{h}_1$ and $\mathbf{h}_2$ are used as two dummy vectors. The matrix $\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}$ can be encoded into one binary string for the '*whole-in-one*' structure. A very long chromosome is formed as:

$$chromosome(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}) := \{00100\ldots0000110\}\big|_{d=n\times(c\times p)+n\times m+p\times n+p\times m}. \quad (3.48)$$

On the other hand, for the '*4-parallel*' structure representation, four binary strings in the dimensions of $n\times(c\times p)$, $n\times m$, $p\times n$ and $p\times m$ respectively for the parametric matrices **A**, **B**, **C**, and **D** are formed as

$$chromosome(\mathbf{A}) := \{00100\ldots0000110\}\big|_{d=n\times(c\times p)}, \quad (3.49)$$

$$chromosome(\mathbf{B}) := \{00100\ldots0000110\}\big|_{d=n\times m}, \quad (3.50)$$

$$chromosome(\mathbf{C}) := \{00100\ldots0000110\}\big|_{d=p\times n}, \quad (3.51)$$

$$chromosome(\mathbf{D}) := \{00100\ldots0000110\}\big|_{d=p\times m}. \quad (3.52)$$

Note that *d* represents the number of parameters to be learned so that the total size of this chromosome is *d* × number of encoding bits.

The genetic algorithm with the arithmetic crossover and non-uniform mutation is employed to optimize the parameters in the neural processing of data structures. The objective function is defined as a mean-squared-error between the desired output and the network output at the root node:

$$E_a = \frac{\sum_{i=1}^{N_T}\left(\mathbf{t}_i-\mathbf{y}_i^R\right)^T\left(\mathbf{t}_i-\mathbf{y}_i^R\right)}{N_T\cdot p}, \quad (3.53)$$

where N_T is the total number of the data structures in the learning set. $\mathbf{t}$ and $\mathbf{y}^R$ denote *p* dimensional vectors of the desired output and the real output at the root node. For GA operations, the objective is to maximize the fitness value by setting the chromosome to find the optimal solution. In order to perform operations in the '*whole-in-one*' structure representation, the fitness evaluation can be simply defined based on E_a

$$fitness_a = \frac{1}{1+\sqrt{E_a}}. \tag{3.54}$$

Basically, the above fitness is applied to the 'whole-in-one' structure but cannot be applied directly to the '*4-parallel*' string structure. The objective function for the '*4-parallel*' string representation is evaluated as follows. Let an error function, $e_i(\theta) = |t_i - y_i|$ be approximated by a first-order Taylor series as,

$$e_i(\theta) \approx e_i(\theta_0) + \nabla_\theta e_i \cdot \Delta\theta, \tag{3.55}$$

where $\theta = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ represents the parameters of our proposed processing, and so,

$$\nabla_\theta = -\left\{\frac{\partial}{\partial \mathbf{A}} \quad \frac{\partial}{\partial \mathbf{B}} \quad \frac{\partial}{\partial \mathbf{C}} \quad \frac{\partial}{\partial \mathbf{D}}\right\}. \tag{3.56}$$

Therefore, equation (3.55) becomes:

$$e_i(\theta) \approx e_i(\theta_0) + \left(-\frac{\partial y_i}{\partial \mathbf{A}} \cdot \Delta\mathbf{A} - \frac{\partial y_i}{\partial \mathbf{B}} \cdot \Delta\mathbf{B} - \frac{\partial y_i}{\partial \mathbf{C}} \cdot \Delta\mathbf{C} - \frac{\partial y_i}{\partial \mathbf{D}} \cdot \Delta\mathbf{D}\right). \tag{3.57}$$

In equation (3.57), the first term is the initial error term while the second term can be denoted as a smoothness constraint that is given by the output derivatives of learning parameters. Thus, the objective function of this constraint becomes,

$$E_b = \frac{\sum_{i=1}^{N_T}\left(-\frac{\partial y_i^R}{\partial \theta} \cdot \Delta\theta\right)}{N_T}. \tag{3.58}$$

So, the fitness evaluation for the '*4-parallel*' string structure representation is thus determined:

$$fitness_b = \frac{1}{1+\alpha\sqrt{E_a} + (1-\alpha)E_b}, \; 0 \le \alpha \le 1, \tag{3.59}$$

where α is a constant and $(1 - \alpha)$ weighs the smoothness constraint. It is noted that the range of the above fitness evaluation is within [0,1]. This smoothness constraint is

a trade-off between the ability of the GA convergence and the correlation among four groups of parameters. In their study, they empirically set α=0.9.

3.8 Concluding Remarks

This chapter presents the basic idea of how a connectionist model, which is a neural network model, is used to generalize the tree structures model for structural pattern recognition and classification. Some research issues like the model design, learning algorithm and its associated problem are addressed. In addition, some recent advanced techniques, for example, an improved BPTS learning algorithm, unsupervised model as well as genetic evolution processing of tree structures, are discussed in this chapter. However, other important research issues addressed in the problem statements of Chapter 1 are essential to highlight in the later chapters. Firstly, the discriminative capability of the connectionist model will be addressed in this thesis. Since the current connectionist model is only able to provide the linear discriminative capability. However, there is no guarantee that the linear discriminate will separate the groups properly if they are linearly separable or not. Secondly, the initialization sensitivity is a problem where is affected the generalization rate if the parameters of the model are initialized improperly. Therefore, it is possible and necessary to adapt an idea of developing a model, which is relatively insensitive to the initialized parameters, while at the same time obtaining the optimal solution. Thirdly, investigating the potential of connectionist models for pattern recognition, particularly whether the connectionist models are able to discriminate different faces and to identify different emotions expressed in a single face, will be the major focus of this research. The later chapters will address all the above issues and present the novelty works solving those problems.

Chapter 4

Probabilistic Recursive Neural Network

In Chapter 2, the basic idea of an image using unstructured feature representation was presented and the associated problems of this representation were also addressed. In Chapter 3, one of the most popular frameworks for adaptive processing of tree structures was proposed by (Frasconi *et al.*, 1998) which uses a Back-Propagation Through Structures (BPTS) algorithm (Goller & Kuchler, 1996) to carry out learning in a directed acyclic graph (DAG) structure. The gradients are calculated by back-propagation error through the data structure. However this BPTS based method has two drawbacks: slow convergence speed and long-term dependency problems. The reasons why the BPTS algorithm suffered from these two drawbacks is that: when the error back-propagates through the deep tree structure, the gradient contribution disappears at a certain tree level. The back-propagated error latches on to the terminal nodes as the gradient disappears, since the activation function is usually a sigmoidal or hyper-tangent of derivatives between 0 and 1. The decreasing gradient term tends to zero as the back-propagation error is recursively multiplied by the derivative of the activation function in each neuron. In addition, this connectionist model for learning tree structures is able to provide the linear discriminative capability, however, there is no guarantee that the linear discriminate will separate the groups properly if they are linearly separable or not. Although some unsupervised SOM models (Hammer *et al.*,

2004) are able to deal with structured pattern classifications without requiring supervised information, data clustering might not always be available and discriminative information was found to be very difficult to obtain during the learning process. There are some SOM models that can be used for structured pattern classification (described in Chapter 3), but they are still unable to guarantee that proper discriminative information can be acquired from an unavailable data cluster. Without acquiring discriminative information from the learning patterns, the absence of *a priori* class characterization will degrade the ability of the system to discriminate between classes.

In this chapter, a probabilistic based recursive neural network (PRNN) is presented for classification of structured patterns. This probabilistic network is an extension and modification of the generalized recursive neuron as shown in the previous chapter (Chapter 3). The architecture of each tree node at the hidden layer is represented by a hybrid of Gaussian Mixture Models (GMMs) and the architecture of each tree node at the output layer is represented by weighted sum of sigmoid function models. The GMMs make use mainly of semi-parametric techniques for approximating probability density functions (pdf) and can assume both feature independence and Gaussian distribution. Discriminative information is acquired during learning and used for classifying structured patterns. In this case, the learning of the discriminative information is performed in an unsupervised manner by estimating the parameters of the GMMs. This unsupervised learning of the GMMs is formulated as a maximum likelihood problem where the mean vectors (i.e., the cluster centers), covariance matrices (i.e., cluster widths), and mixture coefficients are estimated typically by the expectation maximization (EM) algorithm (Streit & Luginhuhl, 1994). This algorithm enables general nonlinear discrimination and gives

smooth discriminant boundaries. It is very effective compared to other unsupervised clustering algorithms, such as the *k*-mean algorithm or the Hebbian learning algorithm.

After obtaining and fine-tuning the optimum parameters of the GMMs and determining the discriminative information from the observed input attributes of each node, the weighting parameters of the sigmoid function model at the output layer are trained by a supervised learning algorithm. These weighting parameters are used to characterize results from the discriminative information. A gradient descent method may have originally been used to learn the weighting parameters but there is still the risk of encountering the problems of slow convergence, and long-term dependency. More advanced algorithms such as the layer-by-layer least squares optimization method (Cho & Chow, 1998) and the penalized optimization (Cho & Chow, 1999), are necessary in order to improve performance. These methods have been validated by different experimental studies of structured patterns representation. They involve the synthetic recognition of the traffic signals created by policemen, and the correlation of images from natural environments. Experimental results show not only that these newer methods outperform the traditional machine learning models, but also that they improved the performance of the original recursive models for structural pattern recognition.

The major contributions in this chapter are the use of GMM architecture at the hidden layer and the use of recursive neurons at the output layer for adaptive processing of data structures. The architecture proposed could be described as a hybrid in which learning is unsupervised locally, but remains supervised globally. However, in learning by the means of the gradient-based algorithm, this probabilistic recursive model may still encounter the problem of local minima and long-term dependency brought about by large and complicated structured patterns, such as the

patterns extracted from natural scene images. A special learning algorithm is presented in this chapter that overcome those problems. Both conceptual and empirical analyzes are also discussed in this chapter.

This chapter is organized as follows: Section 4.1 describes the background idea of the motivation to use probabilistic approach. Section 4.2 describes the probabilistic approach we take towards structural processing and discusses the problems associated with neural network architecture and universal approximation capabilities. Section 4.3 describes the details of deriving the proposed learning algorithm for the probabilistic based model. In Section 4.4, analytical studies in terms of decision boundary, computational complexity and convergence analysis are carried out to demonstrate the effectiveness of this model. Empirical studies benchmarking the probabilistic recursive model against the BPTS algorithm are shown in Section 4.5. Finally, conclusions are drawn in Section 4.6.

4.1 Motivation

Probabilistic Neural Networks (PNN) is one of several techniques that can embed discriminative information in the classification model and are successfully used for providing clustering analysis from the input attributes. This section begins with describing the probabilistic neural network models that have been recently proposed before detailing the process that will be for the adaptive processing of data structured patterns in the next section.

Streit and Luginbuhl (Streit & Luginhuhl, 1994) had demonstrated that by means of the parameters of a Gaussian Mixture distribution, a probabilistic neural network model can estimate the probability density functions. They proposed a four layer feed-forward PNN that uses a general Gaussian kernel, or Parzen window that accurately implements the general homoscedastic Gaussian mixtures in order to

approximate the optimum classifier. Roberts and Tarassenko (Roberts & Tarassenko, 1994) proposed a robust method for Gaussian Mixture Models (GMMs), using a GMM together with a decision threshold to reject unknown data during classification task. Mak and Kung proposed using Elliptical Basis Function Networks (EBFNs) (Mak & Kung, 2000) to determine the cluster centers of input attributes in their hidden layer, which outperformed radial basis function networks and vector quantization methods. Probabilistic Decision-Based Neural Networks (PDBNNs) can be considered as a special form of GMMs with trainable decision thresholds and was used in a face recognition system by Lin *et al*. where the decision threshold for each class is used to reject patterns not belonging to any unknown classes (Lin *et al.*, 1997).

4.2 Model Design

4.2.1 Architecture

Figure 4.1 (a) depicts the architecture of the proposed PRNN classifier in which each neuron in the hidden layer is represented by a Gaussian Mixture Model (GMM) and each of the neurons is represented by sigmoid activation function model at the output layer. Each parameter in this GMM has a specific interpretation and function. All weights and node threshold are given explicitly by mathematical expressions involving the defining parameters of the mixture Gaussian pdf estimates and the *a priori* class probabilities and misclassification cost.

Figure 4.1 (b) presents the architecture of each GMM when the output of a GMM is the weighted sum of G component densities. Suppose that a maximum branch factor of c has been predefined, each of the form q_i^{-1}, $i = 1,2,\ldots,c$, denotes the input from the i-th child into the current node. This operator is similar to the shift operator used in the time series representation. Thus, the recursive neural network for

the structural processing is formed as (Sperduti & Starita, 1997) and is somewhat similar to equation (3.1) and (3.2):

$$\mathbf{x} = F_n\left(\mathbf{A}q^{-1}\mathbf{y} + \mathbf{B}\mathbf{u}\right), \tag{4.1}$$

$$\mathbf{y} = F_p\left(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}\right), \tag{4.2}$$

where **x**, **u**, and **y** are the *n*-dimensional output vector of the *n* hidden layer neurons, the *m*-dimensional inputs to the neurons, and the *p*-dimensional outputs of the neurons, respectively. q^{-1} is a notation indicating the input to the node is taken from its child so that:

$$q^{-1}\mathbf{y} = \left(q_1^{-1}\mathbf{y} \quad q_2^{-1}\mathbf{y} \quad \cdots \quad q_c^{-1}\mathbf{y}\right)^T. \tag{4.3}$$

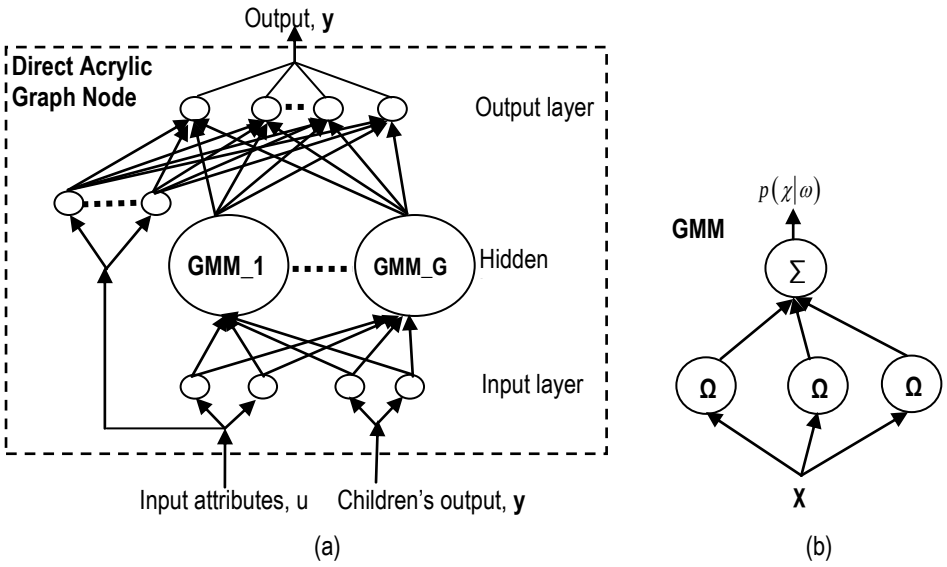


Figure 4.1 – (a) Architecture of the probabilistic based recursive neural network using a Gaussian Mixture Model. (b) Structure of a Gaussian Mixture Model, where ∑ and Ω denotes summation and Gaussian basis function operators, respectively.

The parametric matrix **A** is defined as : $\mathbf{A} = \left(\mathbf{A}^1 \quad \mathbf{A}^2 \quad \cdots \quad \mathbf{A}^c\right)$, where *c* denotes the maximum number of children in the tree, **A** is a $n \times (c \times p)$ matrix such that each $\mathbf{A}^k$, k = 1,2,…,c is a $n \times p$ matrix, which is formed by the vectors $\mathbf{a}_j^i$, j = 1,2,…n. The parameters **B**, **C**, and **D** are $(n \times m)$, $(p \times n)$ and $(p \times m)$-dimensional

matrices respectively. $F_n(\cdot)$ and $F_p(\cdot)$ are *n* and *p* dimensional vectors respectively, where their elements are defined by a nonlinear function $f(\alpha)=1/(1+e^{-\alpha})$.

Now, let *m* denote the dimension of the input features of each node in the tree, and *p* denotes the dimension of the outputs of each node. The input pattern of each GMM can be denotes by

$$\boldsymbol{\chi}=\begin{pmatrix}\mathbf{u} & q^{-1}\mathbf{y}^T\end{pmatrix}=\{x_i; i=1,2,\ldots(m+p\times c)\}, \tag{4.4}$$

where $\mathbf{u}$ is the *m*-dimensional input vector and $\mathbf{y}$ is the *k*-dimensional output vector. q^{-1} shows that the input to the node is taken from its child such that $q^{-1}\mathbf{y}$ is equivalent to equation (4.3).

Assuming the input pattern to be a structured pattern, $\boldsymbol{\chi}$ associated with class ω is a mixture of G components in Gaussian distribution,

$$p(\boldsymbol{\chi}|\omega)=\sum_{g=1}^{G}P(\Theta_g|\omega)\,p(\boldsymbol{\chi}|\omega,\Theta_g), \tag{4.5}$$

where Θ_g represents the parameters of the gth mixture component and G is the total number of mixture components. $P(\Theta_g|\omega)$ is the prior probability of cluster *g*, and is termed as the mixture coefficients of the *g*th component, which by definition can be calculated as follows:

$$\sum_{g=1}^{G}P(\Theta_g|\omega)=1. \tag{4.6}$$

In this approach $p(\chi|\omega,\Theta_g)\equiv\aleph(\mu_g,\Sigma_g)$ is the probability density function of the *g*th component, which typically is a form of Gaussian distribution with mean μ_g and covariance Σ_g, given by:

$$\aleph\left(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right) = \frac{1}{\left(2\pi\right)^{\left(m+p\times c\right)} \Big/ 2\left|\boldsymbol{\Sigma}_g\right|^{1/2}} \cdot \exp\left\{-\frac{1}{2}\left(\boldsymbol{\chi}-\boldsymbol{\mu}_g\right)\sum\nolimits_g^{-1}\left(\boldsymbol{\chi}-\boldsymbol{\mu}_g\right)^{\mathrm{T}}\right\}. \tag{4.7}$$

The outputs of the recursive network for the adaptive processing of data structures as shown in Figure 4.1 (a) is the same manner as the equation (3.2) as

$$\mathbf{y} = F_k\left(\mathbf{W}\mathbf{p} + \mathbf{V}\mathbf{u}\right). \tag{4.8}$$

where $F_k\left(\cdot\right)$ is a *k*-dimensional vector, and their elements are the nonlinear sigmoid activation function. $p = \begin{pmatrix} p_1\left(\chi|\omega\right) \\ \vdots \\ p_r\left(\chi|\omega\right) \end{pmatrix}$, **W** and **V** are the weighting parameters in $\left(p\times n\right)$ and $\left(p\times m\right)$ – dimensional matrices respectively.

4.2.2 Universal Approximation

It is proven that Multi-Layer Perceptron (MLP) is capable of approximating any function from one finite-dimensional real vector space to another such space (Hornik *et al.*, 1989). This follows the universal approximation theorem, which states that a given set of input-output training samples, $\left\{\mathbf{u}_i, \mathbf{y}_i; i = 1, 2, \cdots\right\}$, where **u** is a m vector and **y** is a p vector, it is possible to approximate the underlying mapping to an arbitrary degree of accuracy provided that we are allowed to use as many hidden layer neurons as required. MLP is capable of approximating the set of training samples which might be generated by an underlying mapping. For situation that require data structure to be adaptively processed, (Hammer & Sperschneider, 1997) have proven that the recursive neural networks can approximate mappings on structured patterns. For the set of labelled trees S in which each node has a maximum out degree of c, each node can have a maximum of c children. Each node v of the tree is assigned a

label l, such that l is an element of a finite set $\sum$, with k elements. The theorem and its proof are described in (Hammer & Sperschneider, 1997).

The local response power of the Probabilistic Neural Networks has been shown to offer possibly greater classification and approximation capabilities and the Gaussian Mixture Model (GMM) could be used as a good function approximator in many situations. Let S be a set of labelled trees in $\Re^n$ and $\mathbf{t}(\mathbf{u})$ be a target vector on S, then for any $\varepsilon > 0$ there exist G centroids $\Theta = (\Theta_1, \Theta_2, \cdots, \Theta_G)$ and an $(p \times (n+m))$ parametric matrix $\mathbf{B}$ such that the node output $\mathbf{y} = p(\mathbf{u}; \Theta, \mathbf{B})$ satisfy $|\mathbf{t}(\mathbf{u}) - p(\mathbf{u}; \Theta, \mathbf{B})| < \varepsilon$ for all $\mathbf{u} \in S$. This approximation ability of the probabilistic recursive network can be related to Parzen's approximation theory.

Such architecture is preferred over the BPTS based recursive network even though an improved learning algorithm by Cho *et al.* eliminated the long term dependency problems of the BPTS algorithm (Cho *et al.*, 2003). This is because the GMMs at the hidden layer have been proven to have better discriminatory capabilities. However in learning the means of the gradient-based BPTS algorithm, this probabilities recursive model may still encounter the problem of local minima and long term dependency. A special structural learning algorithm is proposed to overcome these problems and details are provided in the next section.

4.3 Learning Framework

Before the learning algorithms for this proposed probabilistic based recursive model are discussed, the problem of long-term dependency has first to be address. For back-propagation learning of Multi-Layer Perceptron (MLP) networks, it is well known that the parameters at the very deep layers are not updated if there are too many hidden layers (Bengio *et al.*, 1994). Errors related to the back-propagating get

multiplied by the derivative of activation functions (typically a sigmoid function), which is between zero and one. Hence, for very deep layers, the gradients formed by these derivatives could be very small. In (Bengio & Frasconi, 1996; Bengio *et al.*, 1994), the back-propagation learning problems were analytically explained, and why long-term dependencies are very difficult to solve. Bengio *et al*. (Bengio *et al.*, 1994) also stated that if the states of the recurrent MLP network stay within the vicinity of a hyperbolic attractor, it is able to robustly store information for an application of long temporal sequences. Hyperbolic attractor means the eigenvalues of the Jacobian are within the unit circle. However, they have also shown that the Jacobian at each time step will become an exponentially decreasing function if its eigenvalues are inside the unit circle. This implies that the portion of gradients becomes insignificant. This behavior is called the effect of vanishing gradient or forgetting behavior.

The learning scheme employed in this approach, can be divided into two phases. In the first phase the locally unsupervised algorithm for the GMMs. This learning phase can adopt the expectation-maximization (EM) algorithms. The second phase is the globally structural supervised learning for the recursive neural network architecture in which the gradient-based algorithms can be adopted in the structural processing. Both learning phases require several epochs to converge and the globally structural supervised learning starts after the locally unsupervised learning is converged.

4.3.1 Locally Unsupervised Learning for GMMs

The parameters of the GMMs as shown in equation (4.5) are initialized and estimated in the first learning phase. Unsupervised clustering is used to determine the parameters and the k-mean method is commonly used. The algorithm adjusts the centre of a cluster based on its distance from neighbouring input patterns in two steps:

In the first, all input patterns are examined and their closest cluster centres are determined. In the second step, each cluster centre is moved to the mean of its neighbouring patterns. The k-mean algorithm is required to decide the number of clusters before clustering. Alternatively, the learning vector quantization (LVQ) algorithm can also be used. It adaptively varies the number of clusters from class to class, but the size of clusters cannot be too large such that it incurs a high level of reproduction error, or too small such that it loses generalization accuracy. Given these constraints, it appears that the Expectation-Maximization (EM) method (Ng & McLachlan, 2004; Streit & Luginhuhl, 1994) is the most suitable for use in this locally unsupervised learning scheme. The EM method also consists of two steps: The first is the expectation (E) step and the second is the maximization (M) step. The E step computes the expectation of a likelihood function to obtain an auxiliary function while the M step maximizes the auxiliary function refined by the E step with respect to the parameters to be estimated. The EM algorithm in the locally unsupervised learning phase is defined as follows:

Using the GMM in equation (4.5), the goal of the EM learning is to maximize the log likelihood of input attribute set in structured pattern $\boldsymbol{\chi}^* = \begin{pmatrix} \boldsymbol{\chi}_1 \\ \vdots \\ \boldsymbol{\chi}_{N_T} \end{pmatrix}$,

$$\begin{aligned} \ell\left(\boldsymbol{\chi}^*, \Theta\right) &= \sum_{j=1}^{N_T} \log p\left(\boldsymbol{\chi}_j \middle| \omega\right) \\ &= \sum_{j=1}^{N_T} \sum_{g=1}^{G} \log P\left(\Theta_g \middle| \omega\right) + \log p\left(\boldsymbol{\chi}_j \middle| \omega, \Theta_g\right) \end{aligned} . \tag{4.9}$$

In E step, since we refer to the observable attributes $\boldsymbol{\chi}^*$ as "incomplete" data, we define indicator α_j^k such that it specifies which cluster the data belongs to and include it into the likelihood function in the t_L -th iteration as:

$$Q\left(\Theta,\hat{\Theta}(t_L)\right)=E\left\{\ell\left(\boldsymbol{\chi}^*,\Theta\right)\middle|\boldsymbol{\chi}^*,\hat{\Theta}(t_L)\right\}$$
$$=\sum_{j=1}^{N_T}\sum_{g=1}^{G}E\left\{\alpha_j^k,\hat{\Theta}(t_L)\right\}\left[\log P\left(\Theta_g\middle|\omega,\hat{\Theta}(t_L)\right)+\log p\left(\boldsymbol{\chi}_j\middle|\omega,\Theta_g,\hat{\Theta}(t_L)\right)\right], \quad (4.10)$$

where $\alpha_j^k=\begin{cases}1, & \text{if structured pattern } x_j \text{ belongs to cluster } k\\ 0, & \text{otherwise}\end{cases}$.

We defined $p\left(\boldsymbol{\chi}_j\middle|\omega,\Theta_g,\hat{\Theta}(t_L)\right)\equiv\aleph\left(\hat{\boldsymbol{\mu}}_g(n),\hat{\boldsymbol{\Sigma}}_g(t_L)\right)$ and $E\left\{\alpha_j^k,\hat{\Theta}(t_L)\right\}=P\left(\Theta_g\middle|\boldsymbol{\chi}_j,\hat{\Theta}(t_L)\right)$ are the conditional posterior probabilities which can be obtained by Bayes' rule:

$$P\left(\Theta_g\middle|\boldsymbol{\chi}_j,\hat{\Theta}(t_L)\right)=\frac{P\left(\Theta_g\middle|\omega\right)p\left(\boldsymbol{\chi}_j\middle|\omega,\Theta_g\right)}{\sum_{g=1}^{G}P\left(\Theta_g\middle|\omega\right)p\left(\boldsymbol{\chi}_j\middle|\omega,\Theta_g\right)}, \text{ at } t_L\text{-th iteration.} \quad (4.11)$$

In the M step, the parameters of a GMM are estimated iteratively by maximizing $Q\left(\Theta,\hat{\Theta}(t_L)\right)$ with respect to **Θ,** such that the optimized cluster mean and covariance can be calculated for the GMM models.

4.3.2 Globally Structural Supervised Learning for Recursive Network

The structural learning that is meant to occur through the whole Directed Acyclic Graph (DAG) starts after the EM learning converges in the previous phase. The goal of this second learning phase is to optimize the parameters for the entire model in a structural manner. The optimization is carried out to minimize the cost that arises from the difference (error) between the target and output values of the root nodes in the DAGs. As aforementioned, the gradient-based back-propagation learning in the structural processing suffers from problems of slow convergence, local minima and long-term dependency. Therefore, another kind of learning rule may have to apply in this proposed model.

In the second learning phase, the fine-tuning of the GMMs' decision boundaries is carried out by utilizing the target values. Reinforced and/or anti-reinforced learning techniques (Kung & Taur, 1995) were applied to update the cluster mean and covariance of each class. In the mean time, the parameters at the output layer in equation (4.8) are optimized to generalize the DAG by minimizing the total sum-square-error function in this learning phase. In this study, a penalized optimization (Cho & Chow, 1999) is selected for this globally structured learning phase to overcome the problems raised by the gradient based back-propagation algorithm. This algorithm can provide much faster convergence and avoiding the gradient vanishing in the deep trees. The penalty algorithm has been proven to be one of the global optimization algorithms for the learning of neural network which combines the local convergence properties of gradient decent method and the global convergence properties of penalized heuristic strategy to obtain an optimal solution. In case of using the penalized based method, the cost function must be written as (Cho & Chow, 1999):

$$J = \frac{1}{2}\sum_{j=1}^{N_T}\left(\mathbf{d}_j - \mathbf{a}_k\Phi_j^R\right)^T\left(\mathbf{d}_j - \mathbf{a}_k\Phi_j^R\right) + \beta\sum_{k=1}^{p}\exp\left(-\frac{\left\|\mathbf{a}_k\left(t_G - 1\right) - \mathbf{a}_k^*\right\|^2}{\alpha^2}\right), \qquad (4.12)$$

where $\mathbf{d}_j = F_p^{-1}\left(\mathbf{t}_j\right)$ which is the inverse function of the target values. $\Phi_j^R = \begin{bmatrix}\mathbf{p}_j^R \\ \mathbf{u}_j^R\end{bmatrix}$ is the j-th input pattern set at output layer of the root node and $\mathbf{a}_k = \begin{bmatrix}\mathbf{w}_k & \mathbf{v}_k\end{bmatrix}$ is the *k*-th weighting vector at output layer. $\mathbf{a}_k^*$ denotes as the local minima solution which is used to define for the penalized heuristic strategy if the convergence is stuck in a local minimum. This penalty term is used to pull the convergence out of the local minimum. The details of this method can be referred to (Cho & Chow, 1999). In this cost function, the first term is defined as a total sum-squared-error function and the second

term is defined as a Gaussian-like penalty function which is superimposed under the weight space domain. β denotes the penalty factor, which determines the influence of the penalty term against the original gradient terms. In this learning phase, the penalized based algorithm is used to learn the parameters at the output layer of the probabilistic based structured network. Consequently, the gradient equation for updating weighting parameters is:

$$\mathbf{a}_k(t_G+1) = \mathbf{a}_k(t_G) + \eta\left[-\frac{\partial J}{\partial \mathbf{a}_k}\right] \tag{4.13}$$

$$\frac{\partial J}{\partial \mathbf{a}_k} = -\left(\mathbf{d}_j - \mathbf{a}_k\left(t_G\right)\Phi_j^R\right)\Phi_j^{R^T}\frac{\partial q^{-1}\mathbf{y}_k}{\partial \mathbf{a}_k} + \frac{2}{\beta}\left|\mathbf{a}_k\left(t_G-1\right) - \mathbf{a}_k^*\right|\exp\left(-\frac{\left\|\mathbf{a}_k\left(t_G-1\right)-\mathbf{a}_k^*\right\|^2}{\beta^2}\right), \tag{4.14}$$

where the derivative $\partial q^{-1}\mathbf{y}_k / \partial \mathbf{a}_k$ is a $(r+m)\times n$ matrix with the output gradients of the child nodes with respect to the weights. Generally speaking, if the number of tree level defined as $l = 1, \cdots, L-1$, where L is the total tree levels, the derivative is defined as:

$$\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k} = \Lambda\left(\Phi_j^l\right)\cdot\left(\mathbf{a}_k(t_G)\cdot\frac{\partial q^{-(l+1)}\mathbf{y}_k}{\partial \mathbf{a}_k} + \left(\Phi_j^l\right)^T\mathbf{Q}\right), \tag{4.15}$$

where $\Lambda\left(\Phi_j\right)$ is a $(r+m)\times(r+m)$ diagonal matrix defined by the first derivative of the activation function at the output layer. Q denotes a $(r+m)\times n$ matrix with all elements being "1". Computationally, the evaluations of the above derivatives depend on the tree structure (i.e., the level of the tree) and essentially repeat the computations propagation through the structure backwardly. The total derivative with respect to the weights for the entire structure is, generally, defined as

$$\frac{\partial q^{-1}\mathbf{y}_k}{\partial \mathbf{a}_k(t_G)} = \Lambda\left(\Phi_j^1\right)\cdot\left(\Phi_j^1\right)^T\mathbf{Q} + \Lambda\left(\Phi_j^1\right)\cdot\mathbf{a}_k(t_G)\cdot\left(\Lambda\left(\Phi_j^2\right)\cdot\left(\Phi_j^2\right)^T\mathbf{Q} + \cdots\mathbf{a}_k(t_G)\cdot\left(\Lambda\left(\Phi_j^{L-1}\right)\cdot\left(\mathbf{a}_k(t_G)+\left(\Phi_j^{L-1}\right)^T\mathbf{Q}\right)\right)\right) \quad (4.16)$$

From the above discussion, suppose an extremely deep tree structure is processed by this proposed algorithm, let $L\to\infty$, so the total derivatives can be simplified by taking:

$$\lim_{L\to\infty}\frac{\partial q^{-1}\mathbf{y}_k}{\partial \mathbf{a}_k(t_G)} \approx \Lambda\left(\Phi_j^1\right)\cdot\left(\Phi_j^1\right)^T\mathbf{Q}, \quad (4.17)$$

as the expression at the second term of equation (4.16) tends to be zero because of infinite multiples of $\Lambda\left(\Phi_j\right)$ (it is noted that all elements of $\Lambda\left(\Phi_j\right)$ are between zero to one). Therefore, the gradient equation (4.14) allows to descent the original gradient at the first term and ascent the penalty function at the second term. Based on this evaluation, the effect of vanishing gradients is avoided so that the problem of long-term dependency is minimized at this learning phase.

4.3.3 Summary of the proposed learning algorithms

Step 1: Initialization

Set $t_L = 0$ and $t_G = 0$, where t_L and t_G are the iteration indexes of learning phase 1 and phase 2 respectively.

Randomly initialize the parameters $\mathbf{A}$ and Θ_g in the proposed probabilistic based recursive network.

Calculate the *priori* probability

$$P\left(\Theta_g\middle|\omega\right) = \frac{\sum_{j=1}^{N_T} P\left(\Theta_g\middle|\boldsymbol{\chi}_j,\hat{\Theta}(t_L)\right)}{N_T} \quad (4.18)$$

of each cluster using the input attributes and initial parameters.

Propagate the structured learning patterns to obtain the root output of each DAG.

Step 2: Locally unsupervised learning for GMMs (Phase 1)

Given that the input attributes $\{\boldsymbol{\chi}_1, \cdots, \boldsymbol{\chi}_{N_T}\}$ of each node in all structured patterns.

For $t_L = 1, \cdots, T_L$:

Calculate the mean and covariance parameter interatively:

$$\boldsymbol{\mu}_g(t_L+1) = \frac{\sum_{j=1}^{N_T} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_L)\right)\boldsymbol{\chi}_j}{\sum_{j=1}^{N_T} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_L)\right)}, \tag{4.19}$$

$$\boldsymbol{\Sigma}_g(t_L+1) = \frac{\sum_{j=1}^{N_T} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_L)\right)\left(\boldsymbol{\chi}_j - \boldsymbol{\mu}_g(t_L+1)\right)\left(\boldsymbol{\chi}_j - \boldsymbol{\mu}_g(t_L+1)\right)^T}{\sum_{j=1}^{N_T} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_L)\right)}, \tag{4.20}$$

Step 3: Globally supervised learning for structure network (phase 2)

For $t_G = 1, \cdots, T_G$:

Calculate the outputs $p\left(\chi \middle| \omega\right)$ of each GMM in each structured pattern and class.

Fine-tune the decision boundaries of each GMM using the reinforced/anti-reinforced learning :

$$\begin{aligned}\boldsymbol{\mu}_g\left(t_G+1\right) = \boldsymbol{\mu}_g\left(t_G\right) &+ \eta_\mu \sum_{j,\boldsymbol{\chi}_j \in D_a}^{N_t} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_G)\right)\boldsymbol{\Sigma}_g^{-1}(t_G)\left[\boldsymbol{\chi}_j - \boldsymbol{\mu}_g\left(t_G\right)\right] \\ &- \eta_\mu \sum_{j,\boldsymbol{\chi}_j \in D_b}^{N_t} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_G)\right)\boldsymbol{\Sigma}_g^{-1}(t_G)\left[\boldsymbol{\chi}_j - \boldsymbol{\mu}_g\left(t_G\right)\right]\end{aligned}, \tag{4.21}$$

$$\begin{aligned}\boldsymbol{\Sigma}_g\left(t_G+1\right) = \boldsymbol{\Sigma}_g\left(t_G\right) &+ \frac{1}{2}\eta_\sigma \sum_{j,\boldsymbol{\chi}_j \in D_a}^{N_t} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_G)\right)\left[\boldsymbol{\Gamma}_j(t_G) - \boldsymbol{\Sigma}_g^{-1}\left(t_G\right)\right] \\ &- \frac{1}{2}\eta_\sigma \sum_{j,\boldsymbol{\chi}_j \in D_b}^{N_t} P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_G)\right)\left[\boldsymbol{\Gamma}_j(n) - \boldsymbol{\Sigma}_g^{-1}\left(t_G\right)\right]\end{aligned} \tag{4.22}$$

where $\mathbf{\Gamma}_j(t_G) = \mathbf{\Sigma}_g^{-1}(t_G)\left[\boldsymbol{\chi}_j - \boldsymbol{\mu}_g(t_G)\right]\left[\boldsymbol{\chi}_j - \boldsymbol{\mu}_g(t_G)\right]^T \mathbf{\Sigma}_g^{-1}(t_G)$, $P\left(\Theta_g \middle| \boldsymbol{\chi}_j, \hat{\Theta}(t_G)\right)$ is the posterior probability as in equation (4.11). η_μ and η_σ are the learning rates while D_a and D_b are defined as the rejection and the acceptance sets respectively.

Estimate the parameters iteratively by using penalized optimization algorithm. Update the parameters iteratively:

$$\mathbf{a}_k(t_G+1) = \mathbf{a}_k(t_G) + \eta\left[-\frac{\partial J}{\partial \mathbf{a}_k}\right], \tag{4.23}$$

where η is the learning rate. The gradient is defined by equation and the total derivatives of the output gradients are determined by equation.

4.4 Some Analytical Studies

4.4.1 Decision Boundary Analysis

In the proposed probabilistic recursive model, the parameters of the GMMs control the degree of nonlinearity of the decision boundaries while the model order (the number of GMMs at the hidden layer) largely determines the discrimination capability. It is a model order selection problem and its solution is application dependent. Typically, the smallest number of GMMs should be chosen such that no loss is suffered in the classification performance. The decision boundaries are analyzed empirically and presented in the following paragraphs.

To facilitate the discussion, the decision risk is defined as:

$$p(\chi) = \min\left\{p_1(\chi), p_2(\chi), \ldots, p_n(\chi)\right\}, \tag{4.24}$$

where $\left\{p_j(\chi)\right\}$ are approximated by the equation (4.5). The minimum risk decision rule is to classify χ into that class ω having the minimum risk, that is $\omega^* = \arg\min\left\{p\left(\chi|\omega\right)\right\}$. The decision ω is the optimum Bayesian classification

decision if the probability density function of class ω is equivalent to a general mixture probability density function. The simple task of selecting the minimizing index ω^* can be performed many ways, and it has been pointed out that neural network structure can be used for this task if desired. The optimum class decision is identified by the index ω^*, such that

$$\omega^* = \arg\min\left\{p_1(\chi), p_2(\chi), \ldots, p_n(\chi)\right\}. \tag{4.25}$$

A traffic policeman signalling problem was simulated and an analysis was conducted using available data. The proposed works of the BPTS algorithms (Tsoi, 1998) used this problem to illustrate the workings of their method because they felt it presented a well-controlled environment for validation. A number of primitives are adopted, including the policeman's hat, face, body, left arm, left hand object, right arm, right hand object, skirt or pants, and pedestal. All these primitives are described by a number of input attributes, for example, their colour, shape and position and can be joined together to form the image of a "policeman" who is signalling to traffic either to "go" or to "stop". Figure 4.2 (a) provides an illustration of the traffic policeman as composed by these primitives while Figure 4.2 (b) provides a representation of its data structure.

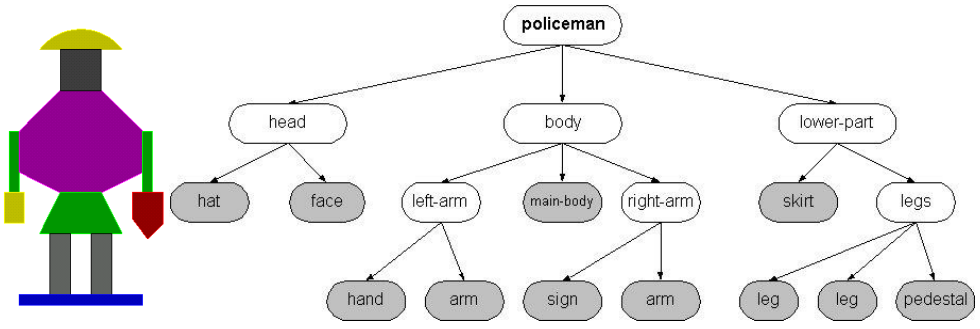


Figure 4.2 - Simulation of Traffic Policeman Signaling Situation (a) An image created by the combination of primitives. (b) A tree representation of the traffic policeman with the shaded blocks representing the primitives.

1057 samples are provided for this problem, each presenting a variation in the combination and position of the primitives. Half of the samples are related to the "go" signal, and the other half to "stop". The representation of the policeman is composed of different parts. An understanding of the words is not implied and the system is assumed to make use of the *priori* knowledge obtained through feature extraction.

Each of the two traffic policemen was divided into eleven objects. 22 features were extracted using the x- and y- coordinates of the center of mass of these 11 objects. These 22 features were then used as inputs to the learning algorithm. Five hundred structured patterns were generated for the learning set and the remaining 557 structured patterns were used for testing.

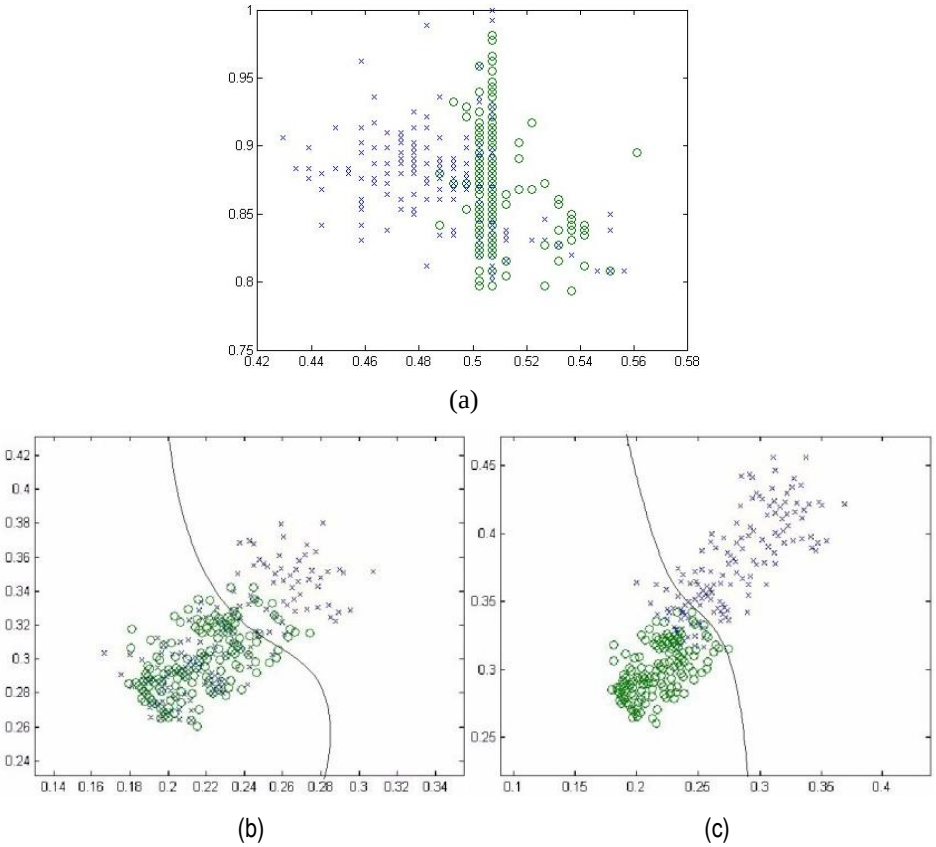


Figure 4.3 – Scatter plot of various inputs. (a) plot of input at the root node. (b) plot of the likelihood function obtained after the EM learning phase. (c) plot of the likelihood function obtained after the fine-tuning decision boundary phase.

With just two input dimensions, a scatter plot is easily used to visualize the problem. Figure 4.3 (a) is a scatter plot of the input attributes observed at the root node which maps onto the two target classes. The mark "x" represents "go" while "o" represents "stop". Figure 4.3 (b) depicts a scatter plot of likelihood function mapping obtained from equation (4.5) together with the two-class discriminant boundary functions roughly determined by the EM learning phase (phase 1). At least two GMMs had to be used to form the discriminated boundary function in this two-class problem. This figure indicates that two major clusters were likely to have been generated.

However, some of the samples on the scatter plot are not clear enough to facilitate this conclusion because the EM learning phase only registers clustering information from the observed input attributes. Further fine-tuning of the decision boundaries is essential and this was carried out in the second supervised structural learning phase (phase 2). Figure 4.3 (c) shows a scatter plot of the likelihood function obtained after the decision boundary was fine-tuned using reinforced/anti-reinforced learning. Two clusters are clearly formed and these can be mapped into the two desired classes under the optimum decision rule in equation (4.25). It is clear from the figure that the fine-tuning of the decision boundaries significantly improves both classification and decision risk performance.

4.4.2 Computational Complexity

In this section, the computational complexity of the proposed probabilistic based recursive model is compared with that of the BPTS model. Take N_v to be the number of nodes in a given input tree structured pattern. Assume also that the underlying

recursive model is specified by a state transition function *f* and by an output function *g*:

$$f : \Re^{m+cp} \to \Re^{n} \qquad\qquad g : \Re^{n+m} \to \Re^{p}.$$

The notations *n*, *m* and *p* are the number of hidden, input and output neurons respectively and *c* denotes the maximum number of children in the tree. *f* and *g* are thus configured by the $n(m+cp)$ and $p(n+m)$ parameters respectively. Ignoring the space required to maintain the input graph, the storage requirements of the BPTS model are characterized by (Frasconi *et al.*, 2001) as follows:

$$O\left(N_v n + n(m+cp) + p(n+m)\right). \tag{4.26}$$

The computational cost is determined by N_v evaluations of the state transition function *f* and by the computation of the Jacobain matrices denoted by N_E matrix-vector multiplications at the hidden layer. Thus, the computational cost of the BPTS model is given by (Frasconi *et al.*, 2001) as:

$$O\left(N_v(p(n+m)+n(m+cp)) + N_E n^2\right). \tag{4.27}$$

Because our proposed probabilistic based recursive learning algorithm is a hybrid involving both locally unsupervised and globally supervised learning, its computation and storage requirements are more demanding than that of the BPTS algorithm. The proposed algorithm requires the computation of the parameters of the GMMs at the unsupervised learning phase, the computation of the fine-tuning in the decision boundary functions, as well as the computation of the parameters at the output layer. One thing this algorithm does not require is the iterative computation of the Jacobain matrices. This reduces its computational complexity significantly and keeps it a level below that of the BPTS algorithm. The storage requirement of the proposed probabilistic model is as follows:

$$O\left(N_v G + G(m+cp) + p(G+m)\right), \tag{4.28}$$

where G is the number of Gaussian Basis Function operators. The computational cost of the probabilistic based recursive model is derived in the following manner:

$$O\left(G(m+cp)^2\right) + O\left(2N_v G(m+cp)(1+m+cp) + N_v p(n+m)^2\right). \tag{4.29}$$

Note that the first portion of the computational cost function relates to the phase 1 learning process. The second portion relates to phase 2 learning. The storage and computational cost requirements of this model are compared with that of the original recursive model. It appears that the computational complexity of the recursive neural networks is application-dependent and dominated by the matrix multiplications related to the Jacobians of the hidden layer function. In contrast, the computational complexity of the probabilistic based model is application-independent and dominated by the number of Gaussian basis function operations.

4.4.3 Convergence Analysis

In this proposed algorithm, the learning scheme is divided into two phases, i.e. locally unsupervised learning phase and globally structural supervised learning phase. In the locally unsupervised learning phase, the EM algorithm is used to determine the parameters in GMMs, hence the cluster parameters of each class can be learnt from the input attributes. The convergence of the EM algorithm for this locally unsupervised learning has been proved in (Dempster *et al.*, 1977) and guaranteed to local maximum. That is, in each iteration, the estimated parameters provide an increase in the likelihood function until a local maximum achieved. However, there is no guarantee that the convergence will be to a global maximum as, for likelihood functions with multiple maxima, convergence will be to a local maximum which depends on initialization. In this proposed model, it is not necessary to guarantee the

global convergence be achieved because the EM algorithm only uses to determine the clustering information (or discriminative information) in the observed input attributes. The entire model will further learn iteratively in the second phase, i.e. the globally structural supervised learning, to achieve the global minimum of the cost function.

For the globally supervised learning phase, we make use of the penalty algorithm to determine the weighting parameters at the output layer by means of minimizing the total sum-square-error function together with a Gaussian-like penalty function. Since the penalized optimization is an unconstrained optimization method in finite dimensional space, it is not guaranteed to converge to the global minimum of the cost function, but it is globally convergent in the sense that it provides an uphill force whenever the convergence is trapped in local minima. Moreover, we also deduce that the convergence behaviour of this structured algorithm is influenced by the statistical characteristics of the input pattern set at the output layer Φ_j and the value assigned to the learning parameters η and β in equations (4.13) and (4.14) respectively. Therefore, we may state, for any condition, that provides the input pattern set Φ_j, we have to select the proper learning parameters for the structured algorithm to be convergent. There are two distinct aspects of the convergence problem in this structured algorithm:

1. The structured algorithm is said to be convergent in the mean if the mean value of the weight vector $\mathbf{a}_k(n)$ approaches the optimum solution $\mathbf{a}_o$ as n approaches infinity; that is, $E\{\mathbf{a}_k(n)\} \to \mathbf{a}_o$.
2. The structured algorithm is said to be convergent in the mean square if the mean-square value of the error vector, $\mathbf{e}(n) = \mathbf{d}_j - \mathbf{a}_k(n)\Phi_j^R$ approaches a

small constant value as n approaches infinity; that is, $E\{\mathbf{e}(n)^T\mathbf{e}(n)\} \to \text{constant}$.

It turns out that two conditions are made for selecting the learning parameters of η and β.

Condition 1

Assume that the weight vector $\mathbf{a}_k$ computed by the structured algorithm is uncorrelated with the input pattern set Φ_j, as shown by $E\{\mathbf{a}_k(n)\Phi_j\} = 0$. By taking the expectation of both sides of equation (4.13) and then applying this assumption, we may get the condition for the learning parameter η as:

$$0 < \eta < \frac{1}{\lambda_{\max}}, \tag{4.30}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{R}_\Phi^*$.

Proof:

By taking the expectation in equation (4.13) and as $E\{\mathbf{a}_k(n)\Phi_j\} = 0$, we get

$$E\{\mathbf{a}_k(n+1)\} = [\mathbf{I} - \eta\mathbf{R}_\Phi\Gamma]E\{\mathbf{a}_k(n)\} + \eta\mathbf{r}_{d\Phi}\Gamma, \tag{4.31}$$

where we can define:

$$\mathbf{R}_\Phi = E\{\Phi_j^R\Phi_j^{RT}\}, \tag{4.32}$$

$$\mathbf{r}_{d\Phi} = E\{\mathbf{d}_j\Phi_j^R\}, \tag{4.33}$$

$$\text{and } \Gamma = E\left\{\frac{\partial q^{-1}\mathbf{y}_k}{\partial \mathbf{a}_k}\right\}. \tag{4.34}$$

We refer to $\mathbf{R}_\Phi$ as the correlation matrix of the input pattern set and to $\mathbf{r}_{d\Phi}$ as the cross-correlation between the input pattern and the desired output. In the mean

time, if we assume that $\Gamma = E\left\{\Lambda\left(\Phi_j^1\right)\Phi_j^1\right\}$ as it is applied in a deep tree structure, so we can define a new matrix $\mathbf{R}_\Phi^* = \mathbf{R}_\Phi\Gamma$ and $\mathbf{r}_{d\Phi}^* = \mathbf{r}_{d\Phi}\Gamma$. In order to find the condition for the convergence in the mean, we can make use of an orthogonal similarity transformation on the matrix $\mathbf{R}_\Phi^*$ by

$$\mathbf{R}_\Phi^* = \mathbf{P}\Lambda\mathbf{P}^T, \tag{4.35}$$

where Λ is a diagonal matrix made up of the eigenvalues of the matrix $\mathbf{R}_\Phi^*$ and $\mathbf{P}$ is an orthogonal matrix whose columns are the associated eigenvectors of $\mathbf{R}_\Phi^*$. By the Wiener-Hopf filtering, we can define:

$$\mathbf{r}_{d\Phi}^* = \mathbf{R}_\Phi^*\mathbf{a}_o, \tag{4.36}$$

where $\mathbf{a}_o$ is the optimum solution. Using the above equation for $\mathbf{r}_{d\Phi}^*$ substituting the orthogonal similarity transformation of equation (4.35) into equation (4.31), we get

$$E\left\{\mathbf{a}_k(n+1)\right\}\mathbf{P}^T = \left[\mathbf{I} - \eta\mathbf{P}\Lambda\mathbf{P}^T\right]E\left\{\mathbf{a}_k(n)\right\}\mathbf{P}^T + \eta\Lambda\mathbf{a}_o\mathbf{P}^T. \tag{4.37}$$

Let $\mathbf{w}(n)$ be defined as a transformed version of the deviation between the $E\left\{\mathbf{a}_k(n)\right\}$ and $\mathbf{a}_o$. We may have the affine transformation as:

$$E\left\{\mathbf{a}_k(n)\right\} = \mathbf{P}\mathbf{w}(n) + \mathbf{a}_o, \tag{4.38}$$

$$\text{and } E\left\{\mathbf{a}_k(n+1)\right\} = \mathbf{P}\mathbf{w}(n+1) + \mathbf{a}_o. \tag{4.39}$$

So, from equation (4.38), we simply to have

$$\begin{aligned}\left(\mathbf{P}\mathbf{w}(n+1) + \mathbf{a}_o\right)\mathbf{P}^T &= \left[\mathbf{I} - \eta\mathbf{P}\Lambda\mathbf{P}^T\right]\left(\mathbf{P}\mathbf{w}(n) + \mathbf{a}_o\right)\mathbf{P}^T + \eta\Lambda\mathbf{a}_o\mathbf{P}^T \\ \mathbf{w}(n+1) + \mathbf{a}_o\mathbf{P}^T &= \mathbf{w}(n) - \eta\Lambda\mathbf{w}_k(n) + \mathbf{a}_o\mathbf{P}^T - \eta\Lambda\mathbf{a}_o\mathbf{P}^T + \eta\Lambda\mathbf{a}_o\mathbf{P}^T. \\ \mathbf{w}(n+1) &= \mathbf{w}(n)\left[\mathbf{I} - \eta\Lambda\right]\end{aligned} \tag{4.40}$$

The above equation can represent a system of uncoupled homogeneous first-order difference equations as shown:

$$w_j(n+1) = w_j(n)\left(1 - \eta\lambda_j\right), \quad j = 1, 2, \ldots, (r+m), \tag{4.41}$$

where the λ_j are the eigenvalues of the matrix $\mathbf{R}_{\Phi}^{*}$ and $w_j(n)$ is the j-th element of the vector $\mathbf{w}(n)$. For the algorithm to be convergent in the mean, we require that for an arbitrary choice of the initial value of $w_j(n)$ the following condition be satisfied:

$$\left|1-\eta\lambda_j\right|<1. \tag{4.42}$$

Under this condition, if $w_j(n)\to 0$ as $n\to\infty$, so we can define the selection of the learning parameter η as follow

$$0<\eta<\frac{1}{\lambda_{\max}}, \tag{4.43}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{R}_{\Phi}^{*}$.

Condition 2

If $E\left\{\mathbf{e}(n)^T\mathbf{e}(n)\right\}\to$ constant as n tends to infinity, so it implicit that $\Delta J=J(n+1)-J(n)\le 0$. By expanding the cost function defined in equation (4.12) in a first order Taylor series, we may get the condition for the penalty factor β as:

$$0<\beta\le\left|2\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^{*}\right|\left[\mathbf{e}(n)\Phi_j^{R}\Lambda\left(\Phi_j^{1}\right)\Phi_j^{1}\mathbf{Q}\right]^{-1}\right|. \tag{4.44}$$

Proof:

The cost function defined in equation (4.12) can be expanded by a first order Taylor series as:

$$J(n+1)=J(n)+\left[\frac{\partial J}{\partial\mathbf{a}_k}\right]\Delta\mathbf{a}_k, \tag{4.45}$$

where $\Delta\mathbf{a}_k=\eta\left[-\frac{\partial J}{\partial\mathbf{a}_k}\right]$. By the convergence theorem of the second distinct, $\Delta J\le 0$ is given, so we define:

$$\left[\frac{\partial J}{\partial \mathbf{a}_k}\right]\Delta\mathbf{a}_k \le 0, \tag{4.46}$$

Substituting equations (4.14) and (4.17) into (4.46), we get,

$$\eta\left\|-\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}+\frac{2}{\beta}\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\exp\left(-\frac{\left\|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right\|^2}{\beta^2}\right)\right\|^2 \le 0, \tag{4.47}$$

and as η is supposed to be positive value, thus we have:

$$\begin{aligned}&-\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}+\frac{2}{\beta}\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\exp\left(-\frac{\left\|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right\|^2}{\beta^2}\right)\le 0\\ \Rightarrow\quad &\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}\ge\frac{2}{\beta}\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\exp\left(-\frac{\left\|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right\|^2}{\beta^2}\right).\end{aligned} \tag{4.48}$$

Taking the logarithm in both sides to get rid of *exp*, we have,

$$\log_e\left\{\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}\right\}\ge\log_e\left\{\frac{2}{\beta}\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\right\}-\frac{\left\|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right\|^2}{\beta^2}. \tag{4.49}$$

Initially, we assume that $|\beta| >> 0$ and is a quite large value, so $\beta^2 \to \infty$ then $\frac{1}{\beta^2} \to 0$, therefore

$$\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}\ge\frac{2}{\beta_{UB}}\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right| \tag{4.50}$$

$$\Rightarrow\ \beta_{UB}\le 2\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\left[\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}\right]^{-1}. \tag{4.51}$$

So, we can choose the penalty factor β as follows

$$0<\beta\le\left|2\left|\mathbf{a}_k(n-1)-\mathbf{a}_k^*\right|\left[\left(\mathbf{d}_j-\mathbf{a}_k(n)\Phi_j^R\right)\Phi_j^R\frac{\partial q^{-l}\mathbf{y}_k}{\partial \mathbf{a}_k}\right]^{-1}\right|. \tag{4.52}$$

Assume that it is working under a deep tree structure; equation (4.17) can substitute into equation (4.52) as:

$$0 < \beta \le \left| 2\left|\mathbf{a}_k(n-1) - \mathbf{a}_k^*\right| \left[\mathbf{e}(n)\Phi_j^R \Lambda\left(\Phi_j^1\right)\Phi_j^{1T}\mathbf{Q}\right]^{-1} \right|. \quad (4.53)$$

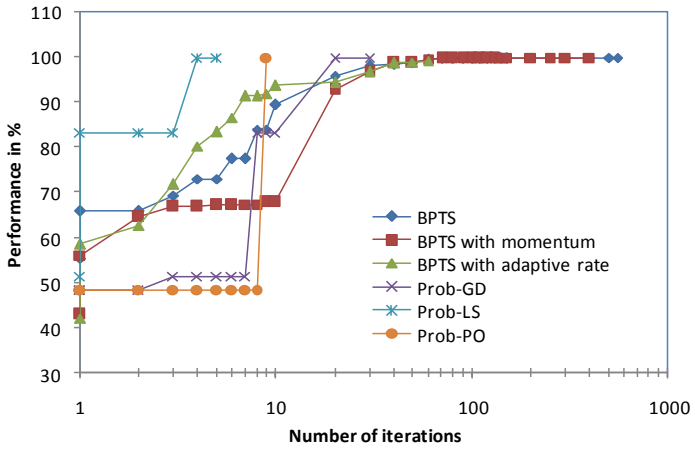
4.5 Experimental Results and Discussion

This section presents an evaluation of the performances of the proposed probabilistic based recursive neural network in its task of adaptively processing data structures. Two experiments were conducted to validate the model's performance; they are Traffic Policeman Signalling problem and Natural Scene Image classification problem. In each case, every node of the DAG was encoded by a single hidden layer network with architecture as described in Section 4.2. In this empirical analysis of the decision boundaries (Section 4.4.1), the number of Gaussian operations could either be set by the number of classes or the size of the input attributes, whichever is more appropriate. The algorithm was initialized by taking all learning parameters in the range of -1.0 to 1.0 under uniform distribution. The algorithm was then implemented by Matlab 6.1 running under a P4 processor with 2.4 Ghz clock speed. The proposed algorithm's performance was benchmarked against the BPTS model and other well-known machine learning classifiers.

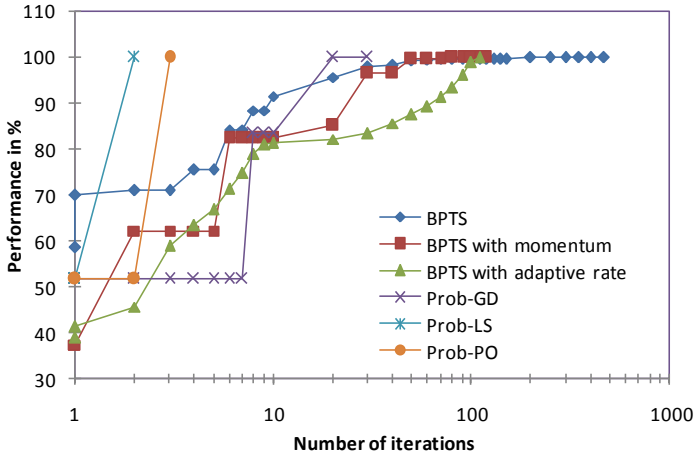
4.5.1 Simulation of the Traffic policeman signaling problem

As there are only two possible actions to be input and classified in this problem, it follows that only two input and output nodes as a result. However, the number of hidden nodes (i.e. number of GMMs in our model) may vary. In this study, we used four and six hidden nodes to sufficiently generalize the problem. The performance of the proposed probabilistic based recursive network at the second phase was investigated by using two other variants, i.e., gradient decent (GD) optimization and least squares (LS) optimization to compare with the BPTS type algorithms.

Figure 4.4 (a) and Figure 4.4 (b) clearly illustrates the advantage of the proposed algorithms in facilitating faster convergence speeds over the ten required iterations. In addition, the proposed approach appears to require a much shorter learning time than BPTS type algorithms (approx. 10 vs 30 seconds). This is despite the fact that our proposed algorithm incurs a relatively high computational cost from the processing of the Penalized Optimization (PO) algorithm through each iterations as well as from the EM learning that occurs in the first phase.



(a)



(b)

Figure 4.4 – (a) Convergence performances of the different algorithms with 4 hidden nodes in the traffic policeman signalling simulation. (b) Convergence performances of the different algorithms with 6 hidden nodes in the traffic policeman signaling simulation. (BPTS : Back-Propagation Through Structures, Prob-GD: Probabilistic based gradient descent algorithm, Prob-LS: probabilistic based least squares algorithm, Prob-PO: probabilistic based Penalized Optimization algorithm).

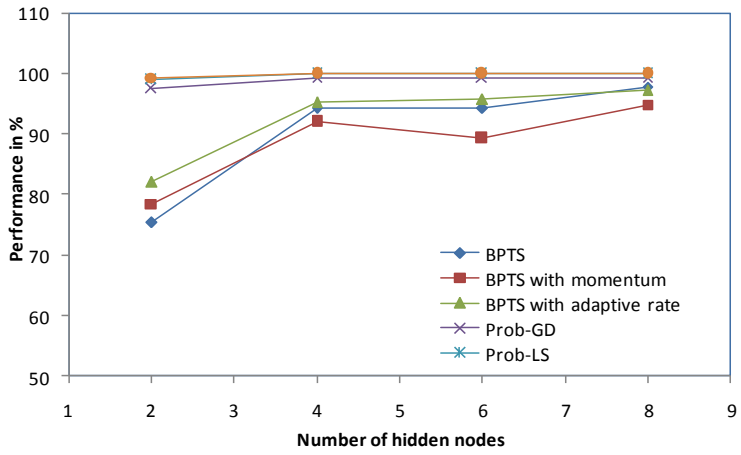
Table 4.1 tabulates the computation time required for training under the different learning algorithms and recursive models. The results show that the

probabilistic recursive model trained using the Least Squares and PO algorithms yielded better performance than the BPTS learning algorithms. The PO algorithm required only 54.6% of the time required by the BPTS with the adaptive learning rate algorithm for the network configuration containing four hidden nodes. In the architecture using six hidden nodes, the time required by the PO algorithm was only 12.6% of that required by the BPTS with an adaptive learning rate.

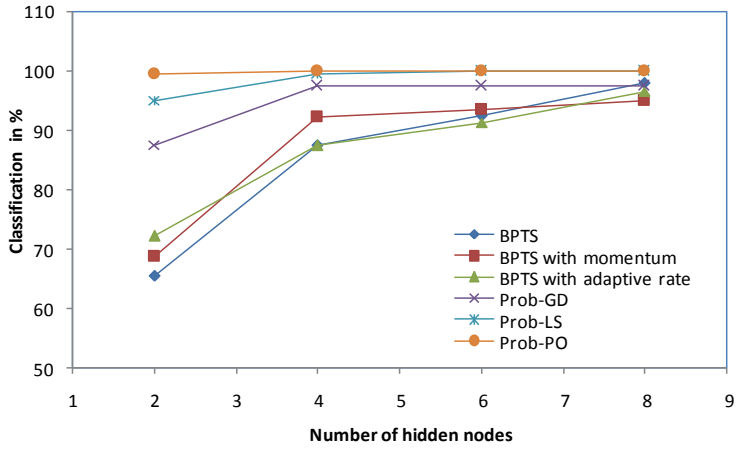
Table 4.1 – Computation time taken for training by different learning algorithms. *n* - number of hidden nodes, *t* - Computation time for training.

n	Models and algorithms	*t*
4	Recursive Neural Networks (BPTS algorithm)	40 sec.
	Recursive Neural Networks (BPTS with momentum adjustment)	40.7 sec.
	Recursive Neural Networks (BPTS with adaptive learning rate)	28.7 sec.
	Probabilistic Recursive Model (Gradient Descent)	24.6 sec.
	Probabilistic Recursive Model (Least Squares)	10.85 sec.
	Probabilistic Recursive Model (Penalized Optimization)	15.66 sec.
6	Recursive Neural Networks (BPTS algorithm)	193 sec
	Recursive Neural Networks (BPTS with momentum adjustment)	51.6 sec.
	Recursive Neural Networks (BPTS with adaptive learning rate)	55.08 sec.
	Probabilistic Recursive Model (Gradient Descent)	30.9 sec.
	Probabilistic Recursive Model (Least Squares)	6.84 sec.
	Probabilistic Recursive Model (Penalized Optimization)	6.93 sec.

Figure 4.5 (a) and (b) shows the results obtained from the use of different numbers of hidden nodes. By incorporating both discriminative and structural information in the model, the proposed algorithms yielded what was clearly the best solution with an accuracy rate of classification that reached almost 100%.



(a)



(b)

Figure 4.5 – (a) Training performances , (b) Classification performances of the different algorithms with varying number of hidden nodes. (Prob-GD: Probabilistic based gradient descent algorithm, Prob-LS: probabilistic based least squares algorithm, Prob-PO: probabilistic based Penalized Optimization).

4.6.2 Natural Scene Image Classification Simulation

One of the challenges in computer vision is natural scene image understanding. Research have shown that designing a generic computational model that can learn concepts from images and automatically interpreting these images is highly difficult (Smeulders *et al.*, 2000). In many image representation methodologies, objects in an image are simply represented by its pixel values (Jain, 1989). The pattern recognition algorithms utilised by either symbolic or statistical pattern representation incorporating with machine-learning models (Comon, 1994; Field, 1987; Tarr & Bulthoff, 1995) would attempt to distinguish between the objects. The main goal is to understand a particular given scene. It is usually assumed that an image includes a

number of objects. However, there are many possible relationships among objects in the image that they were never taken into account. Therefore, if some primitives could be extracted from an image, thus the relationships among the objects would be more transparent.

Typical structural representation approach is a region-based representation of images by means of Binary Space Partition (BSP) tree (Radha *et al.*, 1996; Salembier & Garrido, 2000). This potentially offers an accurate representation, which involves a number of regions that is much lower than the number of original pixels. The extraction of the relevant regions is typically obtained by a region-based segmentation in which the algorithm can extract the interesting regions of images. Once the regions of interest have been extracted, a node is added to the tree for each of these regions. Relevant regions to describe the objects can be merged together based on the merging strategy. This binary tree structure can be elaborated to a semantic representation which nodes correspond to the regions of the image and arcs represent the relationships among regions.

On the other hand, Li and Wang (Li & Wang, 2003) demonstrated that a content-based image retrieval system, called ALIP, is able to interpret pictures for indexing in form of linguistic automatically. The ALIP uses a 2D Multi-resolution Hidden Markov Model (2D-MHMM) to characterize images that yielded about 63% classification accuracy for a 10-class natural image database. Other algorithms proposed in (Forsyth & Fleck, 1999; Matas *et al.*, 1995) might use layout feature descriptions to be captured in a graph or any other ordered set of feature values with their relationships to characterize images. So far, however, there is no unique or generic model that is able to generalize those above representation properly.

BSP tree structure representation

The idea of creating and processing binary space partition (BSP) tree structure-based image representation is an attempt to take benefit from the attractive features of the segmentation results. Most of the studies are started from the terminated nodes and merge two similar neighbouring regions associated with the child nodes based on their contents. This merging is iteratively operated by a recursive algorithm as described in Chapter 2 (Section 2.4).

Figure 4.6 - An example of region merging to create a binary tree, five regions created by the segmentation method.

For instance, a natural scene image shown in Figure 4.6 has five regions segmented. The algorithm is able to merge them within four steps. In the first step, suppose that the pair of most similar regions is regions '1' and '2', which can be merged to create 'a'. In the second step, node '3' is merged with region '5' to create 'b' corresponding to the foreground. Then, the created region 'a' is merged with region '4' to form region 'c' corresponding to the background region. Finally, node 'c' is merged with region 'b' to create region 'd' which is the root node corresponding to the whole image. The merging sequence becomes:

$$a = O(1,2) \rightarrow b = O(3,5) \rightarrow c = O(a,4) \rightarrow d = O(b,c), \tag{4.54}$$

and the tree constructed is shown in Figure 4.7.

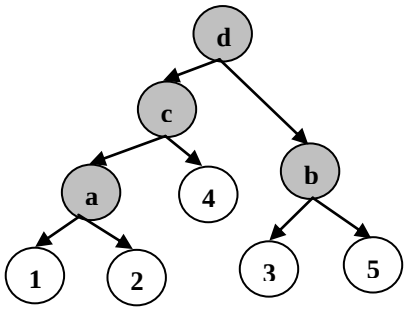


Figure 4.7 - An example of region merging to create a BSP tree, five regions created by the segmentation method.

Note that any two tree-structures can be distinguished because they have different skeleton, or if they have the same skeleton, but they have different node attributes (i.e., different image features).

Performance Evaluations

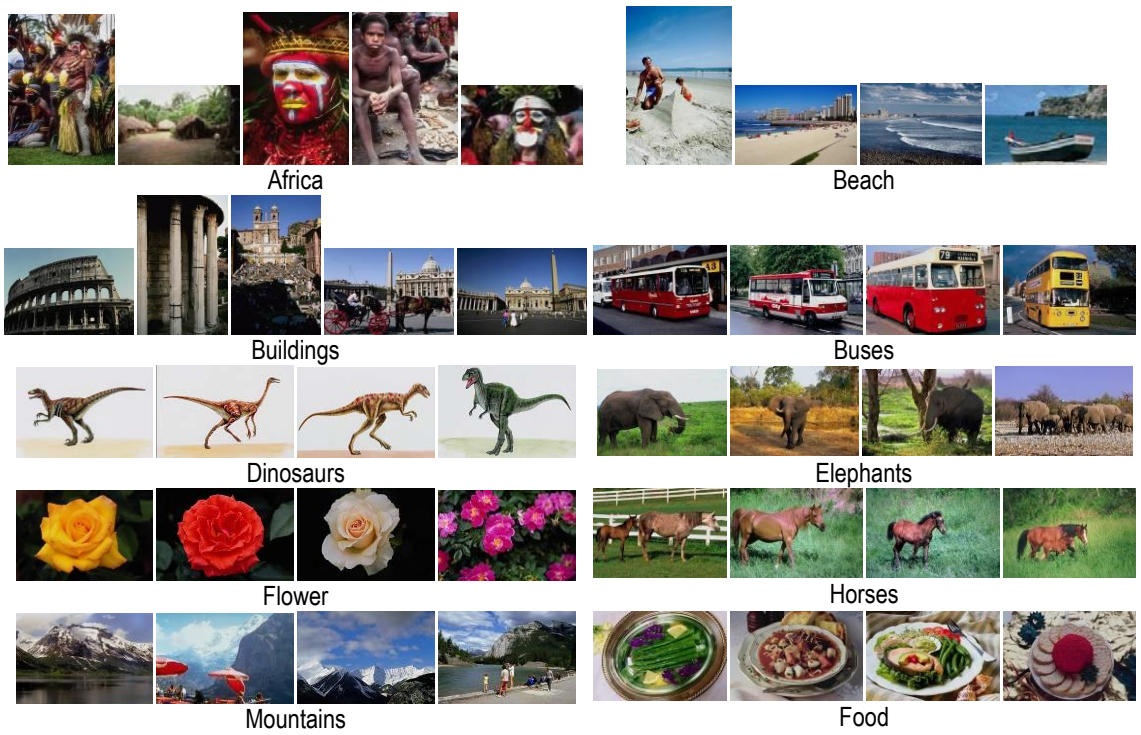


Figure 4.8 - Samples images of ten categories of Natural Scene Images.

This section evaluates the performance of the proposed probabilistic recursive neural network (PRNN) model in an image classification problem. A region-based tree structure representation is used to deal with the problems of image classification and

retrieval. The idea is to simulate the human perception of a real world scene in which both the entities in the environment and their relationship to each other contribute to an understanding of content structure. Images for use in this simulation were downloaded from the database reported in (Li & Wang, 2003) and some samples are shown in Figure 4.8. The database used in this case contains 1000 different images and they are classified into ten categories of 100 images each. A Binary Tree (BT) is used to represent the semantics such that the tree's nodes correspond to the regions of the image and its arcs represent the relationship among regions. The construction of the BT is based on the three steps, i.e. Segmentation, Merging, and Feature Extraction.

Figure 4.9 - Image segmentation. Top: Segmented image, Bottom: Five segmented regions.

Segmentation is a process of identifying regions of interest and objects in images depicting various scenes. We first employed an automatic colour image segmentation method, called JSEG (Deng & Manjunath, 2001), that introduces a merging criterion and pixel labels to help minimize the cost of partitioning the image. The pixel labels are derived from colour quantization, which is a process of determining the number of colour classes in an image. As shown in Figure 4.9, colour quantization results in the segmentation of the top image into five regions. Once this is done, the BT representation is established by merging pairs of homogeneous regions. Trees of segmented images are built in a bottom-up fashion. The merging criterion is based on an examination of the entropy of all pairs of regions to determine

the highest of all. After a pair of regions is merged, a new region is formed and the merging process continues onto new region sets. The merging process terminates only when the last pair of regions merge to create an entire image.

After creating the BT structure that represents each image, the features of each region act as labels that are computed and attached to the corresponding nodes. The visual features could include colours and textures that are used to characterize image content. The colour features were extracted from the colour histograms in Hue-Saturation-Intensity (HSI) colour space. This colour space describes the colour properties and the components of colour features are defined as: $\tau_{color} = [h_{1...4} \quad s_{1...4} \quad v_{1...4}]$, where $h_{1...4}$, $s_{1...4}$, and $v_{1...4}$ are the probabilities of the four most dominant peaks in the HSI histograms respectively.

However, in order to describe a region in detail, it is necessary to use texture together with colour. The mean and variance of neighbouring pixel luminance were employed to characterize the texture and were computed as:

$$\mu(x,y) = \frac{1}{MN}\sum_{v=0}^{M-1}\sum_{u=0}^{N-1} I(x+u, y+v) \text{ and} \tag{4.55}$$

$$\sigma^2(x,y) = \frac{1}{MN}\sum_{v=0}^{M-1}\sum_{u=0}^{N-1} [I(x+u, y+v) - \mu(x,y)]^2 , \tag{4.56}$$

where μ and σ^2 are the neighbourhood means and variances respectively. The texture feature vector is then defined as: $\tau_{texture} = [\mu \quad \sigma^2]$. The feature, in this case, has a total of 14 dimensions (i.e. 12 for color and 2 for texture), which is equivalent to the number of input attributes to the recursive network for each BT node. M and N denote the window size in which they are used to determine the number of neighboring pixels.

A BT structure was created to represent each image and a set of BT patterns was generated from the database. Among the generated BT structure patterns, more complex and deeper BT structures were used to represent images rich in semantic content. BT structures were generated in the range of four to six levels to represent all types of images obtained from the natural scene database.

Five hundred images were used in this study for training and the remaining five hundreds were used for testing. As fourteen input attributes had to be extracted and ten categories had to be classified, the configuration of the recursive network was set as 14-8-10 such that 14, eight and ten neurons were assigned in the input, hidden and output layers respectively. The classification result was obtained by choosing the best parameter set from 20 independent trainings under different initial conditions.

Experimental Results

The classification results of the proposed probabilistic approach are shown in Table 4.2. Each row lists the percentage of images in each category that were classified appropriately, as well as the percentage in each category that were erroneously placed. The numbers on the diagonal show the classification accuracy achieved in every category. The average rate of classification success across all these ten categories is about 92%.

Our proposed approach was then compared with a similar simulation carried out using 2D multi-resolution hidden Markov model (2D-MHMM) reported in (Li & Wang, 2003) and another involving the BPTS model. The 2D-MHMM was used with flat-vectors input such that some regularities inherently associated with the data structures were broken and less significant generalization results were yielded. The BPTS model suffered from poor convergence and long-term dependence problems

and also resulted in a relatively low classification rate. The results of this comparison are shown in Table 4.3. The PRNN model clearly outperformed the other two models on the classification task.

Table 4.2 – A confusion matrix of image classification by the PRNN model.

%	Africa	Beach	Build-ings	Buses	Dino-saurs	Ele-phants	Flowers	Horses	Mount-ains	Food
Africa	**94**	0	0	0	0	2	0	0	0	4
Beach	2	**91**	0	0	0	4	1	0	2	2
Buildings	1	0	**95**	0	0	3	0	0	0	1
Buses	0	0	1	**93**	0	3	0	0	0	3
Dinosaurs	0	1	0	0	**91**	1	0	6	0	2
Elephants	1	1	0	4	0	**93**	8	0	0	2
Flowers	2	1	0	0	0	0	**97**	0	0	0
Horses	0	0	0	0	5	5	3	**83**	3	1
Mountains	2	0	0	1	3	0	0	0	**94**	0
Food	2	2	0	1	0	0	2	1	0	**92**

Table 4.3 – Comparative results of the image categorization.

	2D-MHMM	BPTS model	PRNN model
Overall classification rate	63.6%	70.1%	**92.3%**

Another form of validation of the performance of the proposed model was obtained by the Receiver-Operating-Characteristic (ROC) curves depicted in Figure 4.10. In the figure, Type I error denotes when the desired class is classified erroneously (for example, Africa is classified as Elephants) while Type II error occurs when an unwanted classes is erroneously deemed to be the desired object. The occurrences of both types of error were averaged over the 20 independent runs. Those runs were trained under different initial settings so as to minimize sensitivity to the initialization. As observed from Figure 4.10, Type II error drops very fast in relation to low levels of Type I error and remains low as the Type I error rate increases. An almost ideal ROC curve is achieved, i.e. one that is L-shaped along the two error axes. Other tested models did not do well on this measure and the results

show that the probabilistic based model's ability to correctly distinguish wanted classes from unwanted ones is clearly superior.

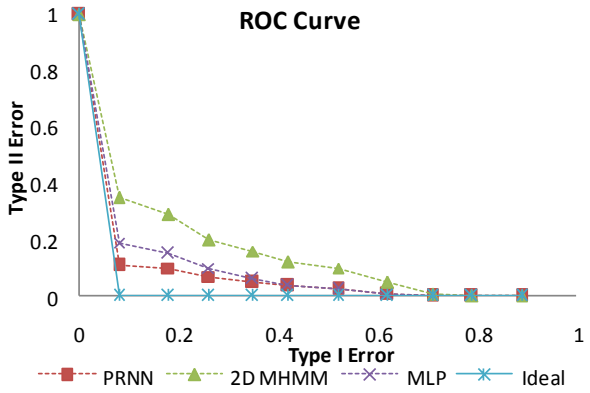


Figure 4.10 – ROC curves related to the classification results of the various models.

4.6 Conclusions

Probabilistic Recursive Neural Network (PRNN) model is a novel contribution to the structural pattern recognition. Unlike conventional connectionist and statistical approaches which rely on static representations of data resulting in vector of features, the probabilistic based recursive model proposed in this chapter allows patterns to be properly represented by directed graphs or trees. These patterns are subsequently processed using specific recursive neural networks. Existing supervised/unsupervised recursive networks for structural processing cannot directly acquire the required discriminative information from target patterns. As a result, their class discrimination ability would be more or less degraded by the absence of *a priori* class characterization. Moreover, it is extremely difficult for the currently pervasive Back Propagation Through Structures (BPTS) learning algorithm to learn deep tree structures because of the problem of long-term dependency.

In this study, the model proposed for classifying the structured patterns involves architecture represented by a set of Gaussian Mixture Models (GMMs) at the hidden layer. A set of weighted inputs was added to the sigmoid function model at the output layer. The proposed architecture allows discriminative information to be utilized during learning performed in an unsupervised manner. This unsupervised learning process harnesses the expectation-maximization (EM) algorithm to learn the GMM's parameters. The decision boundaries of the GMMs were fine-tuned by the use of reinforced and/or anti-reinforced learning techniques. After obtaining the optimum parameters of the GMMs, the weighting parameters at the output layer of the sigmoid function model were trained in a supervised manner. A gradient descent method was originally used but the problems of slow convergence and long-term dependency still existed. The penalized optimization method was found to yield much better results than that obtained from the simple least square method.

An empirical study was conducted using two experiments, namely traffic policeman simulation and natural scene image classification, benchmarking against BPTS typed algorithms. The experiment results showed that the model out-performed the BPTS in terms of learning time. It also yielded the highest classification accuracy rate for the policeman simulation experiment. It also obtained a recognition rate of about 95% for natural image classification.

Chapter 5

Local Experts Organization Model

Chapter 4 presented a probabilistic recursive neural network (PRNN) model for adaptive processing of tree structures. In this PRNN model, the learning framework utilized a hybrid learning system with penalized optimization (PO) to learn the structural patterns unsupervised in locally, but remains supervised in globally. This learning framework is able to address the two problems particularly in learning a deep tree structure, they are slow convergence speed and long term dependence problem, which have ridden the original Back Propagation Through Structures (BPTS) model.

Although this model is able to yield excellent performance as shown in the previous chapter, the PRNN model still encounters the following problems:

1. The high computational cost required for learning the structure patterns using penalized optimization and EM algorithm.
2. The high sensitivity of initialization to the model parameters **A** and Θ_g influenced to the generalization results consistency.

This chapter suggests a novel model to relax the above limitations. The motivation for this model is that feature length is not always consistent for recognizing any kind of images. As shown in Figure 1.1 of Chapter 1, for instance, the eyes of the subjects are occluded by the sunglasses and the eyes related features are un-extractable, which means that we are unable to obtain any information from

there. For the feature representation, we can either put the null value to the feature vector corresponding to these areas or ignore the part of features. But the later might cause the length of the feature to vary with respect to what algorithm being used. Since the length of the feature vector is not consistent, the number of features extracted is often dependent on the performance of image segmentation. In order to deal with this problem, a so-called Local Experts Organization (LEO) model is proposed in this chapter. The LEO model is inspired by the processing of information in the brain and the natural hierarchical model presented in natural organization, where workers stated as Local Experts report to their supervisor stated as Fusion Classifier, whom in turn reports to upper management stated as Global Fusion Classifier. Using this model, the system should be less affected when nodes are missing due to the inconsistent feature length. Each node is made up of network with the hybrid of Local Experts and Fusion Classifier.

The major contribution of this chapter is to provide the offering of the local experts organization (LEO) model to overcome the limitations of the PRNN model. The LEO architecture is a hybrid structure that uses support vector machine (SVM) and reduced multivariate polynomial (RM) classifier as local experts as well as fusion classifier respectively. It is supposed that using this hybrid structure, the learning time required for processing the tree structures is able to reduce in certain degree comparing to the PRNN model in the previous chapter. Additionally, such LEO model is able to generalize structural patterns to organize the feature extracted from various regions of a natural scene image.

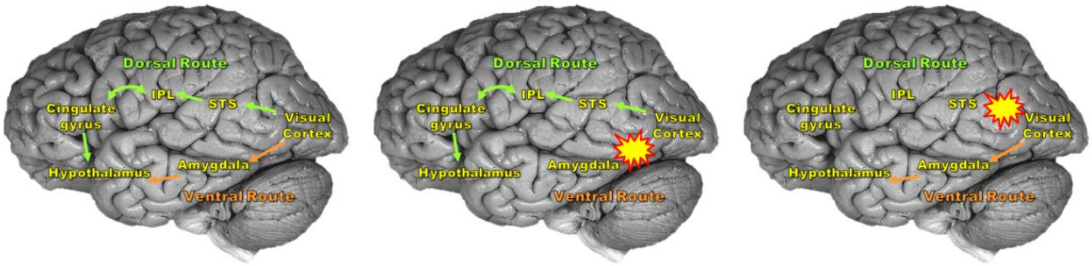
This chapter is organized as follows: Section 5.1 describes the background ideas of the motivation for developing the LEO model. Section 5.2 describes the architecture details of the LEO model and its associated problems. The optimization

of the model parameters is described in Section 5.3. Section 5.4 discusses to use SVM as the Local Experts in the LEO model, whereas Section 5.5 discusses to employ the reduced multivariate polynomial classifier as a Fusion Classifier in the LEO model. Section 5.6 discusses the experiments performed to benchmark the LEO model against other well-known classifier models as well as the BPTS and PRNN models. Finally, the conclusions are drawn in Section 5.7.

5.1 Motivation

In the human brain, we make inference and decisions based on various stimuli arriving from various neurological pathways. An example of this behaviour can be observed during face recognition. When a person sees a face, neural signal flows from both the dorsal and ventral routes in the human brain are taken before reaching the hypothalamus where a decision made on the face is a familiar or unfamiliar face as shown in Figure 5.1 (a). The Dorsal routes are responsible for the covert recognition, and the ventral route is used for overt recognition. Neuroscientist has shown that both paths are essential for normal face recognition (Ellis & Young, 1990). Ellis and Young suggested that there might be a link between prosopagnosia and Capgras delusion patients (Ellis & Young, 1990). Prosopagnosia and Capgras delusion patients have been known to have their ventral route (passing from visual cortex through the amygdala to the limbic system) severed from the result of brain injury as shown in Figure 5.1 (b), simultaneously recognize a face and, at the same time, deny its authenticity (Ellis & Lewis, 2001). According to this hypothesis, if prosopagnosia is the result of damage to the system responsible for generating conscious face recognition, sometimes leaving an unconscious or covert mechanism intact, than Capgras delusion might arise when the reverse occurs, that is, an intact overt system, coupled with a malfunctioning covert system. In Figure 5.1 (c), the

autonomic covert dorsal route is damage causing Capgras delusion (Ellis & Young, 1990).



(a) (b) (c)

Figure 5.1 – Neuroanatomical account of face processing (a) Normal face processing. The green route shows the covert dorsal route via the IPL (inferior parietal lobule) and the STS (superior temporal sulcus). The red route is the overt ventral route to recognition. (b) In prosopagnosia the overt ventral route is damage, hence face recognition is compromised. (c) This account can also be applied to explain Capgras delusion, where damage is postulated to be in the covert dorsal route. Adopted from (Haxby *et al.*, 2000).

Another example which demonstrates that the brain perceives patterns through various pathways can be found in the Thatcher illusion. In "Thatcherized" faces, the eyes and mouth regions are turned upside-down as shown in Figure 5.2 (d), (e) and (f). A dual processing theory explains that the face recognition is mainly based on the processing of local/featural and configural information (Bartlett & Searcy, 1993). Hence, when the face is presented inverted in Figure 5.2 (f), the face looked fairly normal and an inversion of the features is not readily noticed. However, the results looked grotesque in an upright face as shown in Figure 5.2 (d).

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

Figure 5.2 – Demonstration of Thatcher illusion for the three orientations (a), (b) and (c) shows the original picture (d), (e) and (f) shows the Thatcher version of the picture in upright, ninety degree, inverted orientation respectively.

In our artificial life, sharing tasks are usually found in a typical human resource organization structure as shown in Figure 5.3 that every person is an expert in their own domain. For instance, if there is a problem in an enterprise solution that

was developed from the R&D department, all the engineers would provide individual analysis report based on the part of the solution they are involved in. They would submit their report to their team leader, who would compile and filter the report before submitting to the R&D manager. The R&D manager would submit a compiled report to the CIO with regards to the problem in the solution. The CIO would get the advisement from the other IT managers and submit a report to the CEO. The CEO would have to make a decision on whether to terminate the solution or patch the solution, based on inputs from the CFO and COO.

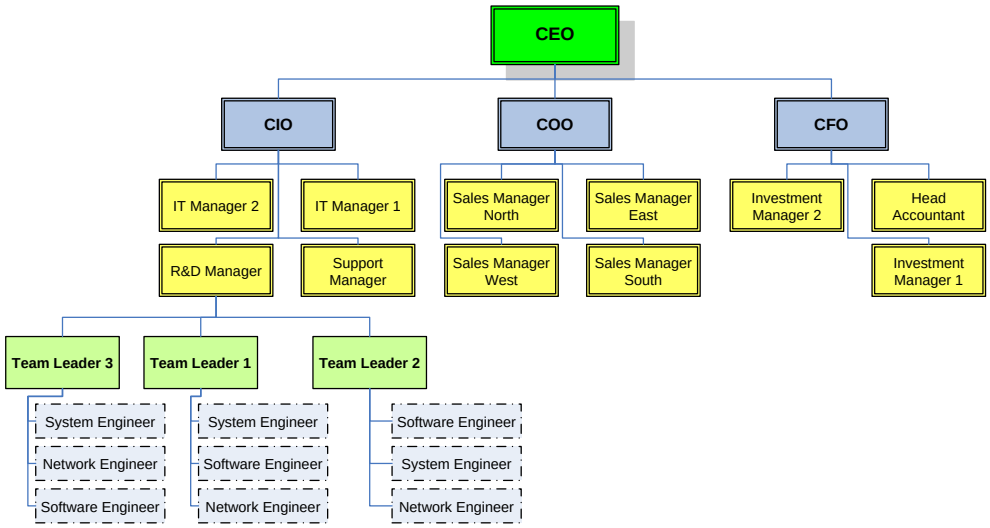


Figure 5.3 – Organization chart of a typical IT company.

In this chapter, we employ the same idea of the artificial human organization chart such that the strength of such disparate node is that each node is specialized in the task it is responsible for. Each of the nodes are experts in their own domain, and the branch fusion classifier values weights the inputs from each of the Local Expert, as well as, its own domain expert values and presents the decision as the output from these fusion classifiers. Such design is known as our proposed Local Experts Organization (LEO) model. The detail discussion of this LEO model will be presented in the following sections.

5.2 LEO Model Design

Suppose that a tree structure is shown in Figure 5.4(a) to illustrate the LEO based structural processing for a natural scene image as shown in Figure 2.1. The number of the nodes in Figure 5.4(a) is dependant up on how the features of an image are represented in a tree structure. Each leaf node of the tree represents each individual object and the root node represents the whole image. In general, each feature attached to each node becomes an input attribute in the LEO model. And also, the tree becomes the hierarchy to convert the input into most likely a time series representation (Tsoi, 1998). Details of the training methodology can be found in Chapter 3 of this thesis. The output $\vec{Y}$, from each LEO node is the local regression relationship, and the parent node uses this information as one of its inputs $\vec{y}$. Based on the children's node output, and together with the Local Expert (LE) output, the parent node would be able to determine its own local winner.

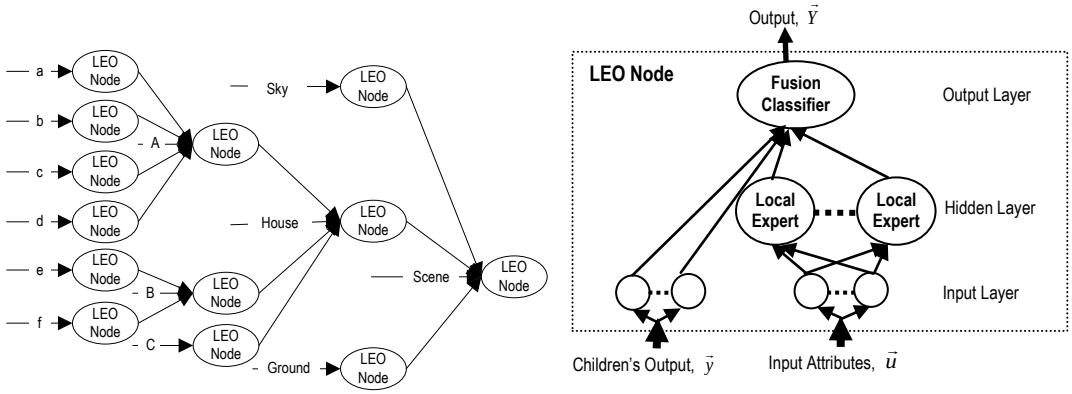


Figure 5.4 – (a) Directed Acyclic Graph (DAG) representation of features extracted from the House Scene in (Tsoi, 1998) for LEO network, (b) The architecture of a LEO node.

Figure 5.4(b) shows the architecture of the LEO node where exists the inter-connections between Local Experts (LE) and Fusion Classifier (FC). In this architecture, the children's output can be expressed as:

$$\vec{y} = \left[y_{1,1}, \ldots, y_{1,m}, y_{2,1}, \ldots, y_{n,m} \right], \tag{5.1}$$

where m is the number of output classes and n is the number of children in which parent node is connected to. At the terminal nodes (the nodes that do not have any child nodes), $\vec{y}$ is set to be a $n \times m$ zero vector. The input attributes to the LEO node can be expressed as:

$$\vec{u} = [u_1 \cdots u_i] \tag{5.2}$$

where i is the number of features for a given node. The LEO node's output can be expressed as:

$$\vec{Y} = \left[P(\omega_1 | (\vec{u}, \vec{y})), P(\omega_2 | (\vec{u}, \vec{y})), \cdots, P(\omega_m | (\vec{u}, \vec{y})) \right] \tag{5.3}$$

where ω presents the output class. The output from each of the Local Expert (LE) can be expressed as the posterior probability for each of the class, i.e.:

$$P(\omega_m | \vec{u}) = \frac{p(\vec{u} | \omega_m) P(\omega_m)}{p(\vec{u})}. \tag{5.4}$$

The inputs to the Fusion classifier can be expressed as:

$$\vec{v} = \left[p(\omega_1 | \vec{u}), \ldots, p(\omega_m | \vec{u}), \vec{y} \right] \tag{5.5}$$

5.3 Parameters Optimization

Firstly, all parameters of the LEO model are initialized at the pre-defined levels (initializing at pre-defined levels help to reduce the optimization search boundary). Then, the parameters of the LE are optimized in the fashion of node-by-node manner. The error of each node in the current state is computed to compare with that of the previous iteration. If the difference between both errors is smaller than zero, the optimization procedures of the LE are continuously performed until the difference becomes greater than or equal to zero. Then, the LE parameters are stored and proceed to the Fusion Classifier (FC) processing. The FC is also optimized as the same manner as the LE. The overall accuracy is calculated at the root node. The

difference between the error of current state and the pervious state is used to determine whether the optimization is stopped. Once the difference is greater than or equal to zero, the optimization is stopped and the FC parameters are stored. The optimization framework of the LEO classifier is described as below in Algorithm 5.1 and the details of the Local Experts and the Fusion Classifier are described in the following sub-sections.

Algorithm 5.1. - The optimization of Local Experts Organization algorithm

Begin
Initialize: *Generate parameters at a pre-defined level for LE and FC*
While*(optimize is* **NOT** *true)*
For *each node in the entire tree*
Train Local Experts with sample data
Calculate LE Error rate as:

$$Error = P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)}$$

Train Fusion Classifier with Local Experts output as defined in equation (5.4)
Output(FC)=Test FC with LE output
End For
error rate = Output(FC) of root node
If *error rate is greater than goal*
Then
Adjust parameters for LE and FC
Else
Set optimize is true
End While
End

5.4 Local Experts

In this model, Support Vector Machines (SVM) are employed as they fulfilled the role of the Local Experts. Support Vector Machines (SVM) is a hyperplane that separates a set of positive examples from a negative example with maximal margin, was invented by Vapnik in 1979 (Vapnik, 1982). Considering the problem of separating the set of training vectors belonging to two separate classes, $(x_1, y_1),\ldots,(x_n, y_n)$, where instances $(\vec{x}_i \in \Re^n)$, with labels $y_i \in \{-1,+1\}$. The output of a linear SVM is

$u = \vec{w}\vec{x} - b$, where w is the normal vector to the hyperplane, x is the input vector and b is the threshold. The separating hyperplane is $u = 0$, and it is optimal when the set of vectors is separated without error and maximal margin.

Cortes and Vapnik (Cortes & Vapnik, 1995) defined the optimization statement for maximizing margin as:

$$\min_{\vec{w},b,\vec{\xi}} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i , \tag{5.6}$$

subject to $y_i u_i \geq 1 - \xi_i, \forall i$, where ξ_i are slack variables that permit margin failure. N is the number of training examples. According to Platt (Platt, 1998), the training is expressed as a minimization of dual Lagrange multipliers :

$$\min_{\alpha} \Psi(\alpha) = \min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{5.7}$$

subject to $0 \leq \alpha_i \leq C, \forall i$ and $\sum_{i=1}^{N} y_i\alpha_i = 0$, where e is the vector of all ones, C is the upper bound that must be greater than zero, Q is an N by N positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(\vec{x}_i, \vec{x}_j)$, and $K(\vec{x}_i, \vec{x}_j) \equiv \phi(\vec{x}_i)^T \phi(\vec{x}_j)$ is the kernel function that measures the similarity or distance between the input vector $\vec{x}$ and stored training vector $\vec{x}_j$. The function ϕ maps the training vectors $\vec{x}_i$ into higher dimensional space. The decision function is defined as:

$$\text{sgn}\left(\sum_{i=1}^{N} y_i\alpha_i K(\vec{x}_i, \vec{x}) - b\right), \tag{5.8}$$

where $\vec{x}$ is the input vector.

Equation (5.7) forms a quadratic programming (QP) problem that arises from the SVMs. Osuna *et al.* (1997) theorem proved that a large QP problem can be broken down into a series of small QP sub-problems. As long as at least one example

that violates the Karush-Kuhn-Tucker (KKT) conditions are added to the examples of the previous sub-problem, each step will reduce the overall objective function and maintain a feasible point that obeys all of the constraints. Therefore, a sequence of QP sub-problems that always adds at least one violator will be guaranteed to converge. Osuna *et al*. (Osuna *et al.*, 1997) also suggested keeping a constant size matrix for every QP sub-problem, which implies adding and deleting the same number of examples at every step. Using a constant-size matrix will allow the training on arbitrarily sized data sets. Based on Osuna's theorem, Platt (Platt, 1998) invented the learning algorithm called Sequential Minimal Optimization (SMO) which decompose the overall QP problem into QP sub-problems and for solving them using an analytic QP step.

SMO chooses to solve the smallest possible optimization problem at every step, which involves two Lagrange multipliers to jointly optimize (Platt, 1998). SMO finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. The advantage of SMO over other algorithms lies in the fact that solving for two Lagrange multipliers can be done analytically. Therefore, numerical QP optimization is avoided. In addition, as SMO uses no matrix algorithms, therefore, it requires no extra matrix storage (Platt, 1998). Thus, very large SVM training problems can fit inside the memory of an ordinary personal computer.

5.5 Fusion Classifier

In the proposed LEO model, a form of Polynomial classifier is used as the Fusion Classifier. Multivariate Polynomial (MP) model being tractable for optimization, sensitivity analysis, and prediction of confidence intervals, provides an effective way to describe complex nonlinear input-output relationship. However, for high-dimensional and high-order systems, multivariate polynomial regression becomes

impractical due to its prohibitive number of product terms. Toh *et al*. proposed a parametric reduced multivariate polynomial model to circumvent this dimension explosion problem, and demonstrated good performance in multimodal biometric decision fusion applications (Toh, Yau *et al.*, 2004). The general multivariate polynomial model is defined as follows:

$$g(\boldsymbol{\lambda}, \mathbf{x}) = \sum_{i}^{K} \lambda_i x_1^{n1} x_2^{n2} \cdots x_l^{nl} \,, \tag{5.9}$$

where summation is taken over all nonnegative integers $n_1, n_2, \cdots, n_l$ for which $n_1 + n_2 + \cdots + n_l \le r$ with r being the order of approximation. $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]^T$ is the parameter vector to be estimated and the regression vector $\mathbf{x} = [x_1, \ldots, x_l]^T$ containing l inputs. K is the total number of terms in $g(\boldsymbol{\lambda}, \mathbf{x})$.

A second-order bivarate polynomial model ($r = 2$ and $l = 2$) is given by:

$$g(\boldsymbol{\lambda}, \mathbf{x}) = \boldsymbol{\lambda}^T \mathbf{p}(\mathbf{x}), \text{ where,} \tag{5.10}$$

$$\boldsymbol{\lambda} = [\lambda_1 \quad \cdots \quad \lambda_6]^T, \text{ and,} \tag{5.11}$$

$$\mathbf{p}(\mathbf{x}) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_1 x_2 \quad x_2^2]^T. \tag{5.12}$$

Given m data points with $m > K$ (assume $K = 6$) and using the least-square error minimization is given by:

$$v(\boldsymbol{\lambda}, \mathbf{x}) = \sum_{i=1}^{m} [y_i - g(\boldsymbol{\lambda}, \mathbf{x}_i)]^2 = [\mathbf{y} - \mathbf{P}\boldsymbol{\lambda}]^T [\mathbf{y} - \mathbf{P}\boldsymbol{\lambda}]. \tag{5.13}$$

The parameter vector $\boldsymbol{\lambda}$ can be estimated by,

$$\boldsymbol{\lambda} = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{y}, \tag{5.14}$$

where $\mathbf{P} \in \Re^{m \times k}$ denotes the Jacobian matrix of $\mathbf{p}(x)$:

$$\mathbf{P}=\begin{bmatrix}1 & x_{1,1} & x_{2,1} & x_{1,1}^2 & x_{1,1}x_{2,1} & x_{2,1}^2\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ 1 & x_{1,m} & x_{1,m} & x_{1,m}^2 & x_{1,m}x_{2,m} & x_{2,m}^2\end{bmatrix}, \tag{5.15}$$

and $\mathbf{y}=\left[y_1,\ldots,y_m\right]^T$ is the known inference vector from the training data. The first and second subscripts of the matrix elements $x_{j,k}$, $\left(j=1,2,\ \ k=1,\cdots,m\right)$ indicate the number of inputs and instances, respectively. From equation (5.13), the following is formed:

$$v\left(\boldsymbol{\lambda},\mathbf{x}\right)=\sum_{i=1}^{m}\left[y_i-g\left(\boldsymbol{\lambda},\mathbf{x}_i\right)\right]^2+d\left\|\boldsymbol{\lambda}\right\|_2^2=\left[\mathbf{y}-P\boldsymbol{\lambda}\right]^T\left[\mathbf{y}-P\boldsymbol{\lambda}\right]+d\boldsymbol{\lambda}^T\boldsymbol{\lambda}, \tag{5.16}$$

where $\left\|\cdot\right\|_2$ denotes the l_2-norm and d is the regularization constant.

The following nonlinear estimation model is usually considered to significantly reduce the huge number of terms in multivariate polynomials,

$$f\left(\boldsymbol{\lambda},\mathbf{x}\right)=\lambda_0+\sum_{j=1}^{r}\left(\lambda_{j1}x_1+\lambda_{j2}x_2+\cdots+\lambda_{jl}x_l\right)^j, \tag{5.17}$$

where the weight parameters $\left(\lambda_{jk},j=1,\cdots,r,\ \ k=1,\cdots,l\right)$ are part of the nonlinear estimation model. l denotes the number of input dimensions and r denotes the order of model.

Toh *et al*. (2004) highlighted that solutions derived from such a nonlinear estimation model have no guarantee that the solutions are optimal. By using Mean Value Theorem, the problem was overcome. The Reduced Multivariate (RM) model is written as the following form:

$$\begin{aligned}f\left(\boldsymbol{\lambda},\mathbf{x}\right)=\lambda_0+\sum_{k=1}^{r}\sum_{j=1}^{l}\lambda_{kj}x_j^k+\sum_{j=1}^{r}\lambda_{rl+j}\left(x_1+x_2+\ldots+x_l\right)^j\\ +\sum_{j=2}^{r}\left(\boldsymbol{\lambda}_j^T\cdot\mathbf{x}\right)\left(x_1+x_2+\cdots+x_l\right)^{j-1}\end{aligned}, \tag{5.18}$$

where l and $r\geq 2$. The number of terms in this model is defined as:

$$K = 1 + r + l(2r - 1). \tag{5.19}$$

Minimizing the objective function in equation (5.18), the following is obtained:

$$\boldsymbol{\lambda} = \left(P^T P + d\mathrm{I}\right)^{-1} P^T \mathbf{y}, \tag{5.20}$$

where, $\mathbf{y} \in \Re^{m \times 1}$ and I is a $(K \times K)$ identity matrix.

The above minimization used by least squares optimization might be suffered from the ill-posed problem if the dimension of vector $\mathbf{p}(\mathbf{x})$ is much larger than the number of given data points, i.e. $K < m$. One way to resolve this problem may need to have a local discriminator to reduce the dimension. In this approach, SVM is used as a local discriminator to produce a local regression output in which the dimension is reduced to be less than the number of training patterns that is able to fulfill the requirement of the RMP model.

The output for each LEO node with respect to equation (5.20) can be expressed to be:

$$\mathbf{y}^m = \left[f(\lambda_1, \mathbf{x}), \cdots, f(\lambda_n, \mathbf{x}) \right] = \mathbf{P}\boldsymbol{\lambda}, \tag{5.21}$$

where the largest element of $\mathbf{y}^0$ will be the output class. m is the node number and $m = 0$ represents this is the root node output. The vector $\mathbf{x}$ can be defined as:

$$\mathbf{x} = [p, y_1, y_2, \cdots, y_n], \tag{5.22}$$

where n is the number of children that the LEO node is connected to. p is the probability estimates from the multi-class SVM. In order to obtain a probability estimate from a multi-class SVM, the SVM must work in regression. However, traditional SVM offers only classification predicts as class label without any probability information. Lin and Weng (2004) have extended SVM to work in regression and producing probability estimates as results. The pair-wise class probability estimates from this multi-class SVM can be expressed as:

$$r_{ij} \approx \frac{1}{1+e^{Af+B}}, \tag{5.23}$$

where A and B are estimated by minimizing the negative log-likelihood function using known training data and their decision values f. Wu *et al*. (2004) have shown in their second approach that solving the following optimization problem, the value of p_i from the values of r_{ij} can be obtained:

$$J = \min_{p} \frac{1}{2} \sum_{i=1}^{k} \sum_{j:j\neq i} \left(r_{ji} p_i - r_{ij} p_j \right)^2, \tag{5.24}$$

subject to $\sum_{i=1}^{k} p_i = 1,\ p_i \geq 0,\ \forall i$. The objective function can be formulated as:

$$J = \min_{p} \frac{1}{2} \mathbf{p}^T Q \mathbf{p}, \text{ where,} \tag{5.25}$$

$$Q_{ij} = \begin{cases} \sum_{s:s\neq i} r_{si}^2 & if \quad i = j \\ -r_{ji} r_{ij} & if \quad i \neq j \end{cases}. \tag{5.26}$$

For this convex problem, the optimality condition where there is a scalar g becomes:

$$\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ g \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}, \tag{5.27}$$

where e is a $k \times 1$ vector of all ones, 0 is a $k \times 1$ vector of all zeros, and g is the Lagrangian multiplier of the equality constant $\sum_{i=1}^{k} p_i = 1$. Hence,

$$-\mathbf{p}^T Q \mathbf{p} = -\mathbf{p}^T Q \left(-g Q^{-1} \mathbf{e} \right) = b \mathbf{p}^T \mathbf{e} = g, \tag{5.28}$$

and the solution p satisfy the condition:

$$Q_{tt} p_t + \sum_{j:j\neq t} Q_{tj} p_j - \mathbf{p}^T Q \mathbf{p} = 0, \text{ for any } t. \tag{5.29}$$

A Polynomial kernel, $K\left(x_i, x_j\right) = \left(\gamma \left(x_i^{\mathrm{T}} x_j + 1 \right)\right)^{\vartheta}$, is used in the multi-class SVM (Wu *et al.*, 2004) for this proposed model. In this LEO architecture, three

parameters need to be determined: the parameters C and γ for the multi-class SVM and the r parameter for the Fusion Classifier. The d is kept constant at 1×10^{-4} for simplicity reasons.

5.6 Experimental Results and Discussion

5.6.1 Simulation of the Traffic Policeman signaling problem

The traffic police analysis was based on the database described in Section 4.5.1, where a total of 1057 samples were created for simulating the actions of a traffic policeman. Half of the samples were for the "GO" signal, and the remaining for "STOP". A total of 557 samples were used for training and 500 samples for testing. The LEO model was benchmarked against the PRNN model described in the previous chapter, Back Propagation Through Structures (BPTS) algorithm (Goller & Kuchler, 1996). Traditional classifiers such as Support Vector Machines (SVM) (Platt, 1998) and Decision Tree (C4.5) (Quinlan, 1993) were also used for benchmarking.

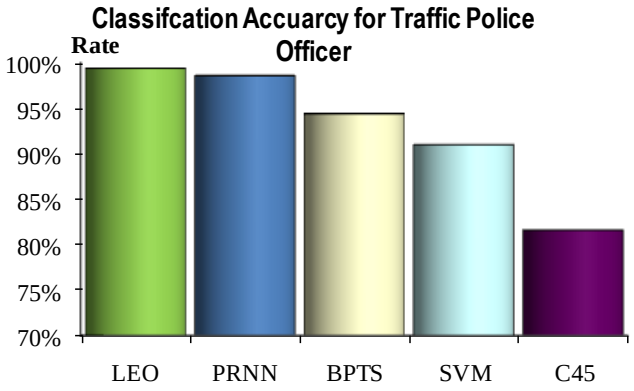


Figure 5.5 - Classification Accuracy for Traffic Police Officer using Local Expert Organization Model (LEO). LEO – Local Expert Organization, PRNN - (probabilistic recursive neural network), BPTS – (back propagation through structures), SVM – support vector machine, C4.5 decision tree.

The SVM and C4.5 algorithm used was based on the WEKA package (Witten & Frank, 2005), in which the kernel function of the SVM is used as the same

polynomial function as the proposed LEO model with the complexity parameter at 1.0 and gamma parameter at 0.01. The C4.5 decision tree model was used 3 folds tree pruning with confidence factor of 0.25. Experiment results as shown in Figure 5.5 shows that the LEO model outperforms the other tested classifiers in which the LEO model achieved a classification rate of 99% whereas the PRNN and the BPTS models might only achieve 98% and 94% respectively.

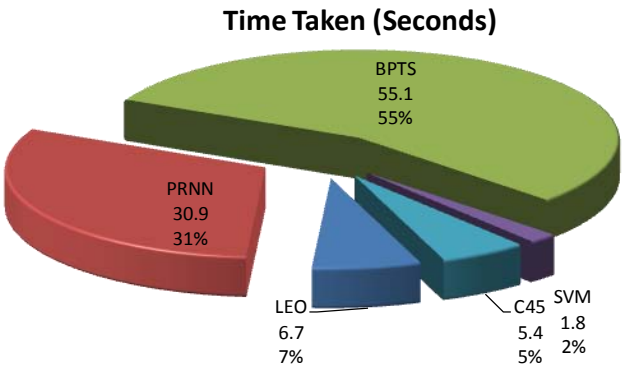


Figure 5.6 – Time taken for training Traffic Police Officer by various models (LEO – Local Experts Organization, SVM- Support Vector Machine, C45 – Decision Tree, BPTS – Back Propagation Through Structures, PRNN – Probabilistic Recursive Neural Network).

Additionally, the LEO model is benchmarked against other models in terms of the learning speed. Figure 5.6 shows the results of time taken for training by various classification models. It can be observed that the LEO model used the least time for training as compared with other adaptive processing of tree structure models although the time taken for training by both SVM and C45 models are much less than the LEO model. Precisely, the LEO model used 6 seconds to train as compared to 30 seconds for the PRNN model and 55 seconds for the BPTS model. The LEO model is able to train using significantly lesser time than the BPTS model as the tasks of training multiple Local Experts in LEO model can be performed in parallel. The BPTS and PRNN model is still needed to process in sequential order as the training algorithm

follows a feed-forward mechanism. Apparently, the complexity of LEO model's training algorithm is much simpler than that of these two models.

5.6.2 Natural Scenery Images Classification

Based on the natural scene experiment presented in Chapter 4, seven hundred images were used in this study for training and the remaining three hundreds were used for testing. As fourteen input attributes had to be extracted and ten categories had to be classified, the configuration of the LEO model was set as 14 input vectors to generalize 10 classes problem. The classification results were obtained by 5-fold cross-validations with 10 independent trainings under different initial parameter setting. A confusion matrix of the 5-fold cross-validations by the LEO model is shown in Table 5.1. Each row lists the total number of images in each category running by the 5-fold validations (each fold contains 30 images) that were classified appropriately, as well as the total number in each category that were erroneously placed. The numbers on the diagonal show the corrected images achieved in every category. The average rate of classification success across all these ten categories is about 83%. The classification rate achieved by the LEO model is slightly lower than the rate achieved by the PRNN model as shown in previous chapter, but it is still better than the other well-known classifiers (see the Table 5.2 below).

The LEO model was then compared with a similar simulation carried out using other well-known classifiers, such as Support Vector Machines (SVM) (Platt, 1998) Back Propagation Through Structures (BPTS) (Goller & Kuchler, 1996), K-Nearest Neighbour (KNN) (Aha & Kibler, 1991), Decision Tree (C4.5) (Quinlan, 1993), RBF network, Naïve Bayesian (NB) multinomial classifier, which were performed from the WEKA package (Witten & Frank, 2005). The kernel function of the SVM is used as the same polynomial function as the LEO model with the

complexity parameter at 1.0 and gamma parameter at 0.01. The C4.5 decision tree model was used 3 folds tree pruning with confidence factor of 0.25. All of these tested classifiers were used with flat-vectors input such that some regularities inherently associated with the tree structures were broken and less significant generalization results were yielded. Some of those models might suffer from poor convergence and resulted in a relatively low classification rate.

Table 5.1 - Image Classification Confusion Matrix of the LEO model

%	Africa	Beach	Build-ings	Buses	Dino-saurs	Ele-phants	Flowers	Horses	Mount-ains	Food
Africa	**117**	2	4	0	0	4	7	4	1	11
Beach	3	**120**	5	4	1	2	1	5	9	0
Buildings	3	5	**114**	3	1	3	3	5	9	4
Buses	5	5	5	**127**	0	1	0	2	1	4
Dinosaurs	1	1	1	0	**145**	0	2	0	0	0
Elephants	1	6	5	3	1	**125**	0	3	5	2
Flowers	3	1	0	0	1	1	**135**	3	4	2
Horses	1	6	1	1	0	7	2	**129**	2	1
Mountains	4	12	7	3	0	4	2	3	**112**	3
Food	3	2	4	7	1	4	5	4	2	**118**

Table 5.2 - Classification rates averaged by 5-fold cross validations obtained from different classifiers (LEO – Local Experts Organization, PRNN – Probabilistic Recursive Neural Network, BPTS – Back Propagation Through Structures, SVM – Support Vector Machine, KNN – K-Nearest Neighbor, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial)

Image Variations	No change (%)	Noise (%)	Blur (%)	Brightness (%)	Darkness (%)
LEO	**82.80**	**69.10**	**77.35**	**74.05**	**73.60**
PRNN	92.33	68.4	75.64	72.39	71.20
BPTS	69.75	64.20	68.90	64.95	63.15
SVM	62.53	51.75	62.60	59.50	59.40
KNN	66.60	58.75	62.20	58.25	60.20
C45	62.33	55.20	56.60	54.30	55.05
RBF	21.00	18.00	18.20	18.50	18.35
NBM	24.33	27.18	24.05	23.70	24.00

Table 5.2 summarizes the classification results obtained from different classifiers. Extensive experiments were also conducted to compare their performance in the classification under different scenarios, which are image variations in intensity

levels, blur levels and noise levels. BPTS model was observed to have yielded better performance than the conventional classifiers model since it carried out the processing of tree structure which offers multi-level and, multi-node analysis, whereas conventional models carried out the single flat vector analysis. In the case of using the LEO model, the obtained performance is even better than that of the BPTS model; therefore it is clear that the LEO model outperforms the other models on this classification task.

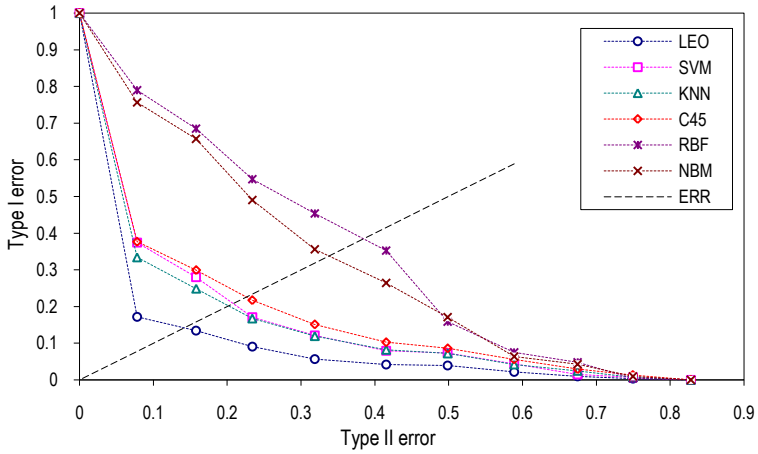


Figure 5.7 - Receiver-Operating-Characteristic (ROC) curves of the natural scene image classification obtained from different classifiers (LEO – Local Experts Organization, SVM – Support Vector Machine, KNN – K-Nearest Neighbour, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial)

Another form of validation was conducted to obtain the Receiver-Operating-Characteristic (ROC) curves depicted in Figure 5.7. Two error metrics are used for this validation. Type I error occurs when the desired class is classified erroneously (for example, Africa is classified as Elephants) while Type II error occurs when an unwanted class is erroneously deemed to be the desired class. The occurrence of both types of error was averaged over the 5-fold cross validation with 10 independent runs that were trained under different initial settings so as to minimize sensitivity to the initialization. In the graph of Figure 5.7, the EER line denotes the case of Equal Error Rates, where Type I errors equal to Type II errors. An ideal ROC curve could be in L-shaped line along the two errors in order to keep the EER low. As observed from

Figure 5.7, in the LEO model, Type I error drops very fast in relation to low levels of Type II error and remains low as the Type II error rate increases. Other tested models did not do well on this measure and the results show that the LEO model is clearly superior in correctly distinguishing wanted classes from unwanted ones.

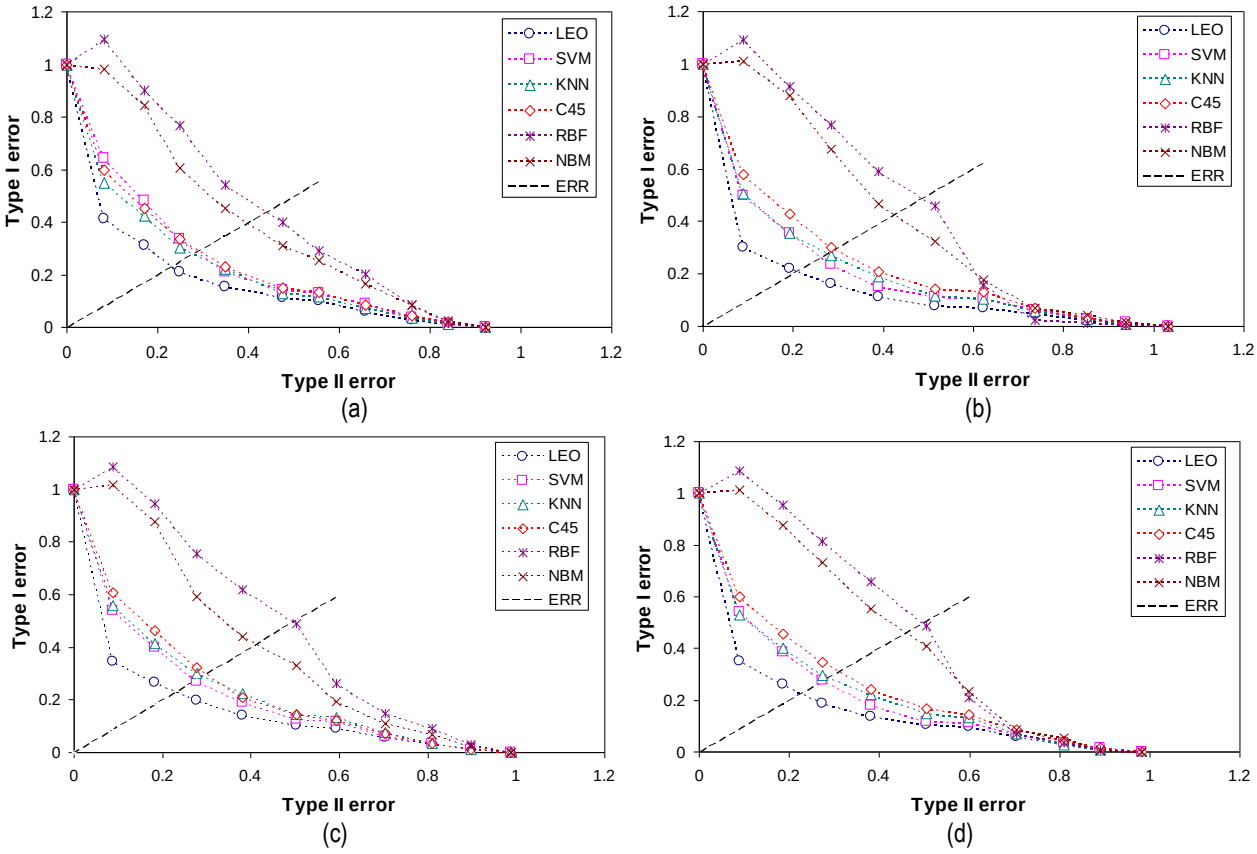


Figure 5.8 - Receiver-Operating-Characteristic (ROC) curves of the natural scene image classification obtained from different classifiers (LEO – Local Experts Organization, SVM – Support Vector Machine, KNN – K-Nearest Neighbour, C45 – Decision Tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial) under different scenarios of (a) noise variation, (b) blur variations, (c) brightness variations, (d) darkness variations.

Further extensive experiments were conducted to test the robustness of the LEO model under different image variation scenarios; intensity levels, blur levels and noise levels; the results were presented using the ROC curves as shown in Figure 5.8. Although the results show that the change in the ROC curves corresponds to the significance of the image alternations, the LEO model is relatively robust to the unknown image alternations. The lowest EER under different scenarios are still obtained from the LEO model amongst the other EERs obtained from various tested

models. This proved that the LEO model is able to more robust in the problem domain of natural scene image classification.

In addition, the proposed LEO model is benchmarked against other models including the BPTS model for tree structure processing in terms of training speed. Figure 5.9 shows the results of time taken for training by various classification models. It can be observed that the proposed LEO model took training significantly less than the training taken by the BPTS model. It is due to the fact that the task of training multiple Local Experts in the LEO model is performed in parallel, whereas the BPTS model is still processed in sequential order.

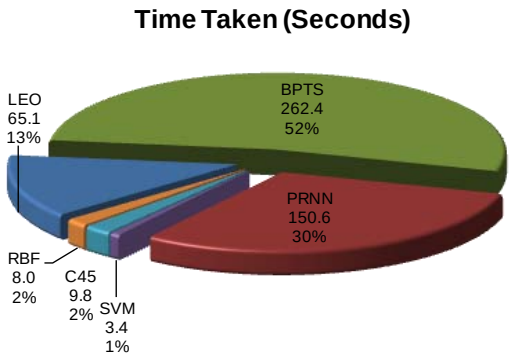


Figure 5.9 – Time taken for training for Natural Image classification database by the various models (LEO – Local Experts Organization, SVM – Support Vector Machine, C45 – Decision Tree, BPTS – Back Propagation Through Structures, PRNN – Probabilistic Recursive Neural Network).

5.7 Conclusion

This chapter presented the Local Experts Organization (LEO) model be a novel contribution to the structural pattern recognition. The LEO model is introduced to solve the two latent problems with using conventional connectionist models for performing tree structure based representation, i.e., long training time and inconsistency for each training cycle due to the use of random initialization parameters. Unlike conventional connectionist and statistical approaches that rely on

static representations of data resulting in vectors of features, the approach proposed in this chapter allows patterns to be properly represented by directed graphs or trees. These patterns are subsequently processed using specific tree structured models. LEO model was inspired by how the decision was made based on hierarchical organization in nature. In a human brain, analysis of a facial image is performed through two pathways before a decision is made about the identity of the face. Also in a typical organization, decisions are made at various levels of the organization before the final decision is made at the top of the hierarchy. The rationale for the proposed hybrid approach is that a Local Expert (LE) has the capability of learning in high dimensional spaces, and a Fusion Classifier (FC) is used for branch level decision making. Both of them incorporated together to process the tree structure representation in traffic policeman simulation and natural image classification.

The validations of different image scenarios for classification were conducted. The promising performance achieved by the LEO model illustrates that it is able to use for structural pattern recognition across a range of situations.

Chapter 6

Facial Image Understanding and Interpretation: A Cognitive Approach

Two cognitive connectionist models have been proposed for representation of tree structures. As described in Chapter 2, the tree structure is able to present an opportunity for preserving the relationship information of the various facial components which are present in global and local features representation for face and emotion recognition. This chapter describes an application on how these adaptive processing models can be applied in the area of face and emotion recognition.

(a) (b) (c)

Figure 6.1 – Example on how a feature locator would detect the location of the primary features under different scenario. (a) perfect unobstructed frontal image (b) subject wearing a pair of sunglasses (c) subject wearing a scarf.

In a typical face recognition system, the face is first detected and the location of the eyes and nose are detected as reference points for feature extractions. The subject has his face covered by the sunglasses as shown in Figure 6.1 (b) and by scarf as shown in Figure 6.1 (b). This causes the feature detection failure to locate the

correct positions (center) of the features. Most system would continue to extract features from these wrong locations, and represented using flat feature vector. Traditional algorithms use these flat vectors without relationship information is unable to differentiate between poor features and good features, which may result in achieving low recognition rates.

This chapter describes two novel tree structures for representing facial components, namely Human Face Tree Structure (HFTS) and FacE Emotion Tree Structures (FEETS), to use for the application of face recognition and facial emotion recognition respectively. The HFTS can be considered as a subset of the FEETS, as FEETS requires more detail information to be extracted from the cheek and mouth. These regions can be considered as non-discriminative features for face recognition. The tree structure as described in Chapter 2 is capable to describing relationship information between the features. The proposed feature extraction algorithm was based on Gabor wavelets transformation. Both holistic and local features were extracted and represented by Gabor features. Hierarchical information present in the holistic and the local features can be represented using tree structures. In the previous chapter, there were demonstrations that the adaptive processing of tree structures is more robust to noise. Using hierarchical information in features, we attempt to improve the recognition rates for images that are plagued by the following problems as:

1. Environmental lighting;
2. Pose variations;
3. Obstructions such as wearing sunglasses or scarf as shown in Figure 6.1.

The major contribution in this chapter is to provide the offering of the formation of the Human Face Tree Structures (HFTS) and FacE Emotion Tree

Structures (FEETS). The methodology and representation of the features can be found in section 6.3, 6.4, and 6.6. It is supposed that the proposed model is more robust when it comes to missing feature locations. An Asian Emotion database (Wong & Cho, 2007) was created to test the robustness of the model using a larger database. The details on the creation process of the database can be found in Appendix A1.

This chapter is organized as follow: Section 6.1 describes the biometric application as well as the challenges faced by current systems. Section 6.2 describes how emotion can be recognized using facial images. Section 6.3 discusses on how a brain inspired model is used for face recognition and emotion recognition. Gabor wavelets are used for feature extraction. The features extracted are transformed to form the Human Face tree structure and Face Emotion Tree structure. In Section 6.5, procedures on how the ORL (Samaria & Harter, 1994) and YALE (Georghiades *et al.*, 2001) databases are prepared for face recognition experiments are documented. Similarly, JAFFE (Lyons *et al.*, 1998; Lyons *et al.*, 1999), CMU (Cohn *et al.*, 1997) and Asian Emotion databases (Wong & Cho, 2007) are prepared for facial emotion recognition experiments. Section 6.6 discusses the experiments performed to benchmark the HFTS and FEETS processed using PRNN and LEO against other well-known classifier models. Finally, the conclusions are drawn in Section 6.7.

6.1 Background of Traditional Recognition Methods

6.1.1 Facial Recognition for Biometrics

Face recognition involves computer to recognize personal identity based on either geometric or statistical features which are directly derived from face images. For over

30 years, extensive research works were conducted by psychophysicists, neuroscientists and engineers on various aspects of face recognition by human and machine (Zhao *et al.*, 2003). With all image-based biometrics, accurate detection, isolation, and registration of the subject within the image frame are critical and necessary steps before the recognition processing. The face detection (Yang *et al.*, 2002) and background removal steps are referred to as a segmentation process (Woodward *et al.*, 2003). The main challenge is that the recognition system has to be invariant to both external changes, such as environmental light, person's pose, person's position and distance between the camera and person and internal deformations, such as facial expression, aging, and makeup (Ekman, 1992).

The performance of existing techniques is still inconsistent and often results differ greatly from those obtaining from human visual perception. Many researchers have explored geometrical features based methods for face recognition. Kanade (Kanade, 1973) presented an automatic feature extraction method based on ratios of distance and reported a recognition rate of between 45-75% with a database of 20 people. Some other methods employ the use of local feature from face images, and often needs the localisation of fiducial points (Gökberk *et al.*, 2003). Brunelli and Poggio extracted a set of geometrical features (Brunelli & Poggio, 1993) using independently matching templates in the 3 key fiducial points such as nose width and length, mouth position, and chin shape in which this method could achieve 90% recognition rate on a database of 47 people. Gao and Leung (Gao & Leung, 2002) have proposed a Line-Edge Map (LEM) features for face coding and used the line segment Hausdorff distance measurement for the human face recognition. The results are encouraging with a single public database with over 90% recognition rate under lighting variations but below 75% under pose variations. However, such geometrical

featuring methods would be dependent on the accuracy of the feature-locating algorithm. Singh *et al*. proposed using a robust skin colour based algorithm to locate the fiducial points of the face (Singh *et al.*, 2003) and achieved recognition accuracy of 96% for frontal images. But, relative low recognition rates were obtained from various posture images. Wiskott *et al*. have proposed using Localised Gabor Jets (Wiskott *et al.*, 1997) and represented in the form of Elastic Bunch Graph approach. The method used a total of about 45 facial landmarks such as the pupils, the corners of the mouth, tip of the nose the top and bottom of ears, etc. By using the bunch graph, learning weights could be given to more discriminative and more robust nodes (Wiskott *et al.*, 1997), so that they were able to achieve 6.5% improvement in terms of recognition accuracy on a gallery size of 130-150 probe images with different poses. However, the accuracy of such geometrical feature based methods is basically dominated by the performance of face detection algorithms. Face detection algorithm such as Viola-Jones detection algorithm (Viola & Jones, 2001) has a detection rate of 77.8% on MIT test set (Rowley *et al.*, 1998) containing 23 images with 149 faces. Liu proposed using Bayesian Discriminating Features (Liu, 2003) for multiple face detection which achieved 98.5% detection accuracy with one false detections using FERET database (Phillips *et al.*, 1997).

Not only the geometrical features based methods can be used for face identification, but statistical approaches are also employed for processing the face image under low-level dimension. Eigenfaces are used to represent every face as a vector of weights in the database (Turk & Pentland, 1991). The approaches are obtained by projecting the image into Eigenface space by a simple inner product operation. Apart from using either statistical or geometrical approaches, neural

network based feature extraction has been proposed to develop a compact internal representation of faces (Er *et al.*, 2002; Lawrence *et al.*, 1997; Lin *et al.*, 1997).

Most facial recognition systems use a set of feature vectors to represent facial images in their database. However there is lacking of describing the relationship between the feature vectors. In most common image recognition systems, images are simply represented by its pixel values (Jain, 1989), and the recognition algorithm uses this representation (Jing *et al.*, 2003; Liu & Wechsler, 2002, 2003) and attempts to distinguish between the images.

6.1.2 Emotion Recognition

In a human society, relationships with friends, colleagues and family (Ekman, 2004; Levine, 2007) are carefully maintained by the ability to understand, interpret and react to emotions. There is growing evidence that emotions play a part in decision making, perception and empathic understanding in a way that affects intelligent functions (Bechara *et al.*, 2000; Isen, 2000; Perlovsky, 2001). Most people are able to interpret the emotions expressed by others all the times, but there are people who lack this ability, such as people diagnosed along the autism spectrum (Baron-Cohen, 1995). Capgras' syndrome patients (Ellis & Lewis, 2001) who suffered from head trauma and have the link between their visual cortex and limbic system severed, thinks that family and friends are replaced by impostors because they are unable to feel any emotions from that person. This is due to the covert recognition process (emotion response stimuli) is disconnected and the overt recognition process (people identity nodes) are still connected (Ellis & Lewis, 2001).

Regarding the basic human emotion, Ekman has identified six basic categories of emotions (Ekman, 2004) (i.e. fear, anger, sadness, surprise, disgust and joy) which

human is able to express easily on his face. Figure 6.2 shows a subject expressing different facial expressions by controlling his facial muscles to represent the six basic emotions. Such emotions are revealed earlier through facial expression than people verbalize or even realize their emotion states (Tian *et al.*, 2001). Ekman has shown how to use a facial expression to identify a lie (Ekman, 1991). Cognitive interpretations of emotions are known to be innate and universal to all humans regardless of cultures (Ekman, 1999; Thompson, 1941). Besides, detecting suspicious behaviours through human emotions is also an active research in the field of security. Wrongdoers, with reason to falsify information or documents or carry illegal hidden objects, will be under stress and detection apprehension. They will exhibit some forms of emotional betrayal of their intentions.

Figure 6.2 - Six basic emotions (from left to right): anger, joy, sadness, surprise, fear and disgust (images taken from NTU Asian Emotion database).

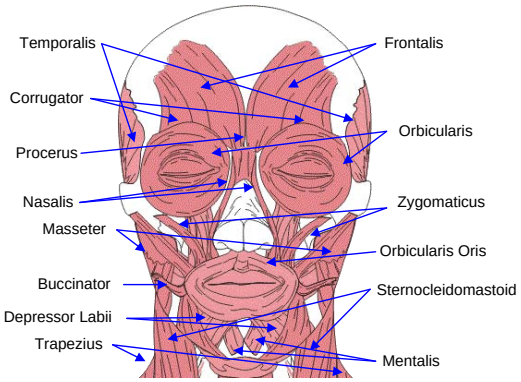


Figure 6.3 – Various muscles used in making facial expressions in facial emotion.

Models and automated systems have been created to recognize the emotional states from facial expressions. The typical method - Facial Action Coding System

(FACS) (Ekman & Friesen, 1978) for measuring facial movements in behavioural science, was developed by Ekman and Friesen in 1977. The facial muscles shown in Figure 6.3 are used by FACS to rate the 46 Action Units. Other methods such as electromyography, which directly measures the electrical signals generated by the facial muscles and deducing the facial behaviour from it, are both obtrusive and non-comprehensive. According to the survey in (Ekman & Rosenberg, 1997), FACS uses 46 defined Action Units (the details are described in Annex A4) to correspond into each independent motion of the face. However this model takes over 100 hours of training to achieve minimal competency for a human expert (Donato *et al.*, 1999). Faster automation approaches, such as measurement of facial motion through optic flow (Mase, 1991; Rosenblum *et al.*, 1996) and analysis of surface textures based on principal component analysis (PCA) (Lanitis *et al.*, 1997). Other techniques include using Gabor wavelets (Daugman, 1988), linear discriminant analysis (Belhumeur *et al.*, 1996), local feature analysis (Penev & Atick, 1996), and independent component analysis (Bartlett & Sejnowski, 1997), however, such methodologies may fail to handle large and complex task like recognizing human emotion because either they take too long to train a system or they take too much memory.

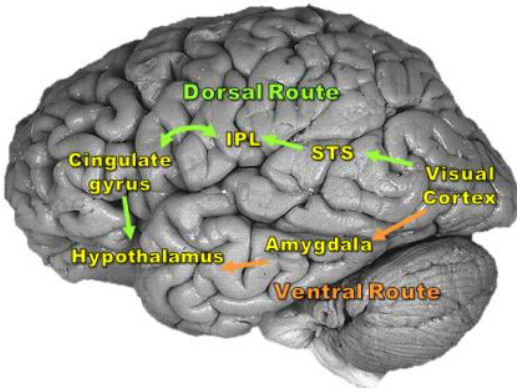
6.3 Framework Design

6.3.1 System Architecture

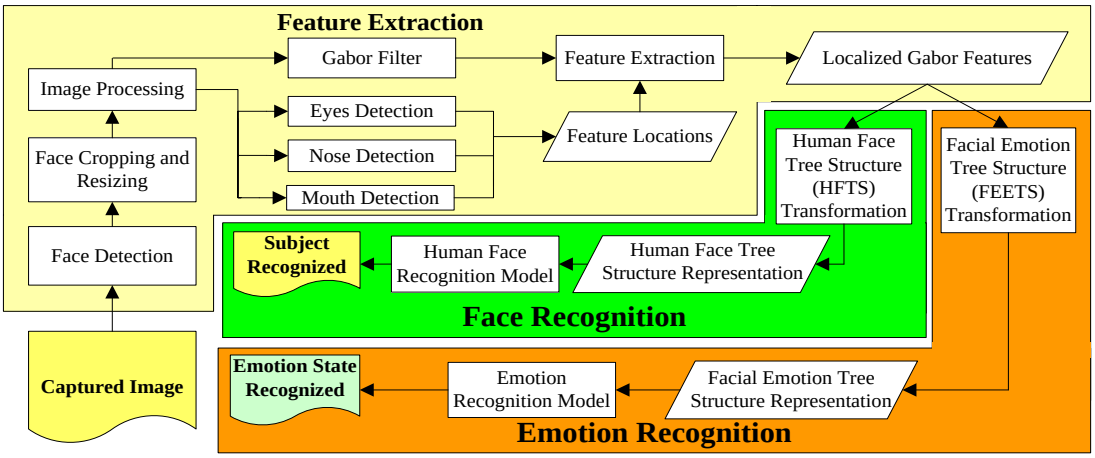
Figure 6.4(a) shows the two pathways, Dorsal and Ventral Route, in a human brain which are responsible for face recognition and emotion recognition respectively. In the human brain, the visual cortex process the images captured by the retina. The visual cortex is divided into six functionally distinct layers, labelled 1 through 6. V1 neurons consist of tiled sets of selective spatiotemporal filters. V2 receives strong feedforward connections from V1. Anatomically, V2 cells are tune to simple properties such as orientation, spatial frequency and color. V3 contains neurons that respond to different combinations of visual stimulus. V4 is one area which receives strong feedforward input from V2. V4 is the first area in the ventral stream to show strong attention modulation. The visual cortex forms the feature extraction region in the brain for images received by the retina. The dorsal route receives features extracted from the V1 area of the visual cortex into the parietal lobe. Evidently (Perrett & Mistlin, 1990), the neurons in the inferior temporal cortex and the superior temporal sulcus are used for face recognition. The ventral route receives features from the V1, V2 and V4 into the areas of the inferior temporal lobe. It also has strong connections to the medial temporal lobe (which stores long-term memories), and the limbic system (which controls emotions).

In this section, Figure 6.4(b) presents a framework which represents the neuro-cognitive structure of the brain for emotion recognition in facial expression (Taylor et al., 2003). The model has a branching structure that emphasizes a distinction between the representation of invariant aspects of faces, which underlie recognition of unique identity, and the representation of changeable aspects of face. The underlie perception of information that facilitates social communication. Within this brain-

inspired framework, the face detection and feature extraction block is built to model the functions of the visual cortex. The model has a hierarchical structure that distinguishes a core system for the visual analysis of faces and an extended system that processes the meaning of the information obtained from faces.



(a)



(b)

Figure 6.4 – Brain inspired model for Human Face Interpretation. (a) Signal path in the brain for face processing. The green route shows the covert dorsal route via the IPL (inferior parietal lobule) and the STS (superior temporal sulcus). The orange route is the overt ventral route to recognition. (b) Proposed recognition system for face processing. The yellow block mimics the visual cortex functions. The green block mimics the Ventral Route function of face recognition. The orange block mimics the Dorsal Route function of emotion recognition.

Pre-processing steps attempt to detect and track the face. We used the Intel Open CV library's face and feature detector to locate the face and the 4 fiducial points (i.e. Eyes, Nose and Mouth). The Open CV face and feature detector implements the Viola and Jones algorithm (Viola & Jones, 2001), which is a statistical approach for object detection. The algorithm uses an AdaBoost (Freund & Schapire, 1995)

classifier with Haar-like features because they are computed similar to the coefficients in Haar wavelet transforms, for detecting the face and fiducial points. The classifier is trained on images of fixed size, and detection is performed by sliding window search algorithm. For detection of faces with different sizes, the classifier has the ability to scale as well. Each feature is described by the template, its coordinate relative to the search window origin and size of the feature.

Gabor wavelet transform is used as a feature extractor with an approximate experimental filter response profile encountered in cortical neurons (Liu & Wechsler, 2003). Gabor wavelet has capability to capture the properties of spatial localization, orientation selectivity, spatial frequency selectivity, and quadrature phase relationship.

The green block in Figure 6.4(b) depicts a process flow diagram to show how the proposed Human Face Tree Structure (HFTS) representation is formed from the local gabor feature vector. The objective of this system is to extract relevant features from the facial image and hence the system is able to recognize the subject based on these features extracted. The features in HFTS are somehow similar to those extracted by the V1 region of the visual cortex. The human face recognition model is represented by either the PRNN or LEO model that mimics the functions of the various brain sections responsible for the accurate and robust face recognition. The orange block depicts the emotion recognition process to show how the proposed FacE Emotion Tree Structure (FEETS) representation is formed from the original gabor feature vector. The features in the FEETS are somehow similar to those extracted by the V1, V2 and V4 regions of the visual cortex. The amygdale is closely connected with hypothalamic and midbrain motivational areas and is involved in all emotional responses, from the most primitive to the most cognitively driven (Gaffan & Murray, 1990). The multiple features received from the various parts of the visual cortex

suggested the signals are presented in a hierarchical structure to the amygdale. Both the PRNN and LEO models have demonstrated strong capability for the adaptive processing of facial structures. These models have been used to adaptively process the HFTS and FEETS.

6.3.2 Feature Extraction

The face image can be captured from any form of video or image source. At first, the face detection module is used to separate the face from the background as well as the body. Several well known techniques such as PLANNING by Kelly (Kelly, 1970) for automatic extraction of head and body outlines from an image and subsequently the locations of eyes, nose and mouth. A real-time face tracker (Yang & Waibel, 1996) using hue detection for skin colour could be used for face detection in case of colour images are used. The face detection step provides us with a rectangle head boundary, which includes the whole face area. Having correctly detected the face from the rest of the input image, then the face needs to be cropped from the background and resized to be normalized against the other images encoded by the system. Image processing techniques can be used to improve the image's quality, for instance, noise removal through median filters could be used for noisy environment condition. Simple histogram stretching maybe used to enhance the image contrast for the images in the developed system.

For the feature extraction, five main techniques can be used for, i.e., dimensionality reduction transforms, discrete cosine transform (DCT), Gabor wavelet, spectrofaces and fractal image coding. Karhunen-Loeve transform, known as Principal Component Analysis expansion for representation (Kirby & Sirovich, 1990; Sirovich & Kirby, 1987), is used for dimensionality reduction transformation. Linear

Discriminant Analysis (LDA), Fisher Discriminant Analysis (FDA) and Independent Discriminant Analysis (IDA) are perhaps the most popular techniques (Aras *et al.*, 2004) to generate a set of the most discriminant features so that different classes of training data can be classified. Discrete Cosine Transform is not only used in Joint Picture Expert Group (JPEG) for image compression, but it could also be used for feature extraction as DCT transform images from spatial domain to the frequency domain by means of sinusoidal basis functions. Gabor wavelets has the capability to capture the properties of spatial localization, orientation selectivity, spatial frequency selectivity, and quadrature phase relationship, such that it seems to be a good approximation to filter response profiles encountered experimentally in cortical neurons (Liu & Wechsler, 2003). Hence, it is a good feature extraction technique for face processing because of its biological relevance and computational properties. Figure 6.5 shows the responses of two different facial images for one of the selected Gabor filters. Appendix A3 shows the details Gabor filter response for the 7 selected Gabor filters.

Figure 6.5 – Gabor wavelet response of the 6 basic emotions.

In the proposed system, Gabor feature vectors including global and local features representations are extracted. The global features are said to be more accurate in frontal views of face. It also provides the holistic analysis of the image and they do not depend on the accurate location of the fiducial points of the face. Meanwhile, local features provide the robustness of accurate face recognition whenever the faces

are in different postures. For localized feature extraction, four key fiducial points located at eyes, nose and mouth are needed to detect to form the basic reference locations of the faces. Brunelli and Poggio (Brunelli & Poggio, 1993) used independently matching templates for finding the 4 key fiducial points.

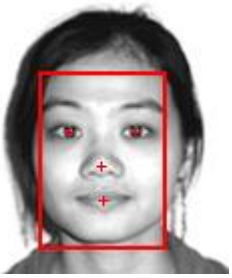
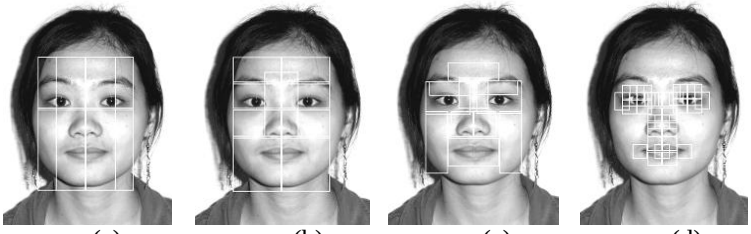


Figure 6.6 - Four primary feature locations and entire face region. Crosses denote the centre of fiducial points. Rectangle box denotes of region of interest.

The Intel Open CV library's face and feature detector is used to locate the face and the 4 fiducial points (i.e. Eyes, Nose and Mouth) as shown in Figure 6.6. The Intel Open CV face and feature detector implements the Viola and Jones algorithm (Viola & Jones, 2001), which uses an AdaBoost (Freund & Schapire, 1995) classifier with Haar-like features for detecting the face and fiducial points. The left, right, top and bottom regions of these 4 fiducial points are used to define the extended feature components.



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

Figure 6.7 - Sixty feature regions denoted by rectangle boxes at various detail levels (from left to right). (a) level 2: upper, lower, left, right and centre region of face. (b) level 3: forehead, left and right eye, eyes, nose, mouth, left and right cheek and nostril. (c) level 4: forehead, left and right eye, eyes, nose, mouth, left and right cheek, left and right nose. (d) detail features of various regions of interests.

Figure 6.7 describe the extended feature components used for facial emotion recognition, whereas face recognition uses a subset of these features. These extended components as described by Gabor feature vector are selected as they contained the

most important information of the human face for the purpose of identification. The shapes of the eyes, bridge of the nose, tip of the nose and lips do not change easily without surgery (Cho & Wong, 2007). The eyebrows are not considered to be an important feature, because the eyebrows could easily be altered by shaving/trimming.

The location points of the left and right eye features are being derived from the location of the centre of the left eye and right eye denoted by the coordinates (x_{LE}, y_{LE}) and (x_{RE}, y_{RE}) respectively. The location of the nose bridge is the middle point of the left and right eye on the X-axis. The nose feature locations are derived from the location of tip of the nose denoted by the coordinate of (x_{NS}, y_{NS}). The locations of lips features are derived from the center of lips coordinate of (x_{LS}, y_{LS}).

6.3.3 Localized Gabor Feature Extraction

A pre-defined global filter based on the two-dimensional Gabor wavelets $g(x, y)$ is used, which can be defined as follows (Manjunath & Ma, 1996):

$$g(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right], \tag{6.1}$$

where parameters $W = U_h$, $\sigma_x = 2\sigma_u/\pi$ and $\sigma_y = 2\sigma_v/\pi$. $\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}}$, $\sigma_v = \tan\left(\frac{\pi}{2K}\right)\left[U_h - 2\ln\left(\frac{\sigma_u^2}{U_h}\right)\right]\left[2\ln 2 - \frac{(2\ln 2)^2\sigma_u^2}{U_h^2}\right]^{-\frac{1}{2}}$ and K is the total number of orientations, $a = (U_h/U_l)^{-\frac{1}{s-1}}$ and s is the number of scales in the multi-resolution decomposition. U_h and U_l denote the lower and upper center frequencies respectively.

Given an image $I(x, y)$, the Gabor wavelet transformed is defined as follows:

$$W_{mn}(x,y) = \int I(x_1, y_1)\, g_{mn}{}^*(x - x_1, y - y_1)\, dx_1 dy_1 . \tag{6.2}$$

The subscript *m* denotes the size of the filter bank in terms of number of orientation. The subscript n denotes the size of the filter bank in terms of number of scales. The values of $m = 7$ and $n = 1$ is used in this system. The localized Gabor Feature $F(x_F, y_F)$ can be expressed as a sub-matrix of the holistic Gabor wavelet output from equation (6.2),

$$F_{mn}(x_F, y_F) = W_{mn}\begin{bmatrix} (x_F, y_F) & \cdots & (x_{F+S}, y_F) \\ \vdots & \vdots & \vdots \\ (x_F, y_{F+S}) & \cdots & (x_{F+S}, y_{F+S}) \end{bmatrix}, \tag{6.3}$$

where *s* defines the size of the feature area. The x_F and y_F can be defined respectively as:

$$x_F = x_{RF} + c, \tag{6.4}$$

$$y_F = y_{RF} + c, \tag{6.5}$$

where the subscript "RF" refers to the relative centre location coordinates that can be labelled as "LE", "RE", "NS" or "LS" for the different facial components. The mean and standard deviation of the convolution output is used as the representation for classification purpose:

$$\mu_{mn} = \int\int \left|F_{mn}(x_F, y_F)\right| dxdy, \text{ and} \tag{6.6}$$

$$\sigma_{mn} = \sqrt{\int\int \left(\left|F_{mn}(x_F, y_F)\right| - \mu_{mn}\right)^2 dxdy}. \tag{6.7}$$

The LGF vector of each image can be formed as:

$$\vec{X} = \left[\vec{F}^0, \vec{F}^1, \vec{F}^2, ..., \vec{F}^t\right] \tag{6.8}$$

Each of the features, i.e., F^n, is a vector of features extracted using equation (6.6) and (6.7) from the sub-matrix of the convolution output for the image with Gabor filter bank. The superscript *n* denotes the set of features derive from each of

the t feature location. The number of extended features, t, used for face and emotion recognition is 38 and 60 respectively.

$$\vec{F}^n = \left[\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, \ldots, \mu_{mn}, \sigma_{mn}\right]. \tag{6.9}$$

6.3.4 Gabor to Tree Structure Representation

After the feature vectors are obtained from the face images, face and emotion recognition classifiers are essential to make use of the feature vectors to discriminate and identify each of the faces and emotions. Various pattern classifiers such as Support Vector Machines (SVM) (Platt, 1998), K-nearest neighbors (K-NN) (Aha & Kibler, 1991), Naïve Bayes Algorithm (John & Langley, 1995) and Artificial Neural Networks could be used for recognition stage of the system. SVM is a learning technique developed by Vapnik (Vapnik, 1995) which was strongly motivated by results of statistically learning theory. SVM operates on the principle of induction, known as structural risk minimization, which minimizes the upper bound of the generalization error. K-nearest neighbors method is a nonparametric technique in pattern recognition, which is used to generate k numbers of nearest neighbors rules for classification. Naïve Bayes algorithm is based on the Bayesian decision theory, which is a fundamental statistical approach to the problem of pattern classification. This approach is based on quantifying the tradeoffs between various classification decisions using probability and the costs that accompany such decisions. Most of these techniques often lose to generalize the relationship information between the features. Structural pattern classification techniques described in Chapter 2 such as adaptive processing of tree structures (Sperduti & Starita, 1997), makes use of the relationship information between features to recognize tree structures patterns more robustly.

Since extracting localized Gabor feature vectors to represent facial images do not provide the description of the relationship between the feature vectors, Wiskott *et al*. has proposed a method for generalising representation of faces by using elastic bunch graph (Wiskott *et al.*, 1997). This method attempted to provide more correlation information about the features. It appears that the method produces a rather good generalization as it could detect face features rather well. The down side to this method would be that the bunch graph is rather big as it takes the convolution output from each Gabor jet. Hence it would have a feature vector of about 180,000 features for 8 orientations and 5 scales of filter combination. They are obtained from 45 feature locations and their window size is of 10 x 10 pixels. On the contrary, the proposed method uses a feature vector of 546 features derived from a 7 orientations and 1 scale Gabor filter bank. In this approach, after extracting the localized features, human faces can be represented by a tree structure model based on the entire face acting as a root node and localized features like eyes, nose and mouth acting as its branches as shown in Figure 6.8. The branch nodes are labelled as $\vec{F}^0, \vec{F}^1, \vec{F}^2, ..., \vec{F}^{38}$, which are corresponding to each facial component. The arc between the two nodes corresponds to the object relationship, and extracted features are attached to the corresponding nodes. In this facial tree structure representation, two contributions are achieved in order to eliminate the problems raised by the conventional localized feature representations. Firstly, this representation can make full use of the relationship information in the tree structure during recognition, whereas the elastic bunch graph is unable (Cho & Wong, 2007). Secondly, although human facial features may not be altered easily by surgery, the features appearance can strongly vary in the image with changing in facial expression, illuminations and pose (Cho & Wong, 2007). The statistical moments derived from Gabor wavelet convolution to

describe each facial feature suffer from these variations. In this approach, the localized Gabor feature extraction incorporating with tree structure representation is used to maintain the robust face recognition in terms of accuracy and generalization even if some parts of features appearance are changed or missed.

Ekman's facial action coding system (FACS) (Ekman & Friesen, 1978) provided the foundation of dominant facial components of interest for recognizing human facial expression as shown in Figure 6.3. The details of the Action units used by FACS encoding system are presented in Annex A4. In this system, similar concept of facial components of interest to extract the facial features to represent human face. In addition, tree structure is used to present the relationship information between the features from coarse to fine detail, which are represented by the Localized Gabor Feature vector.

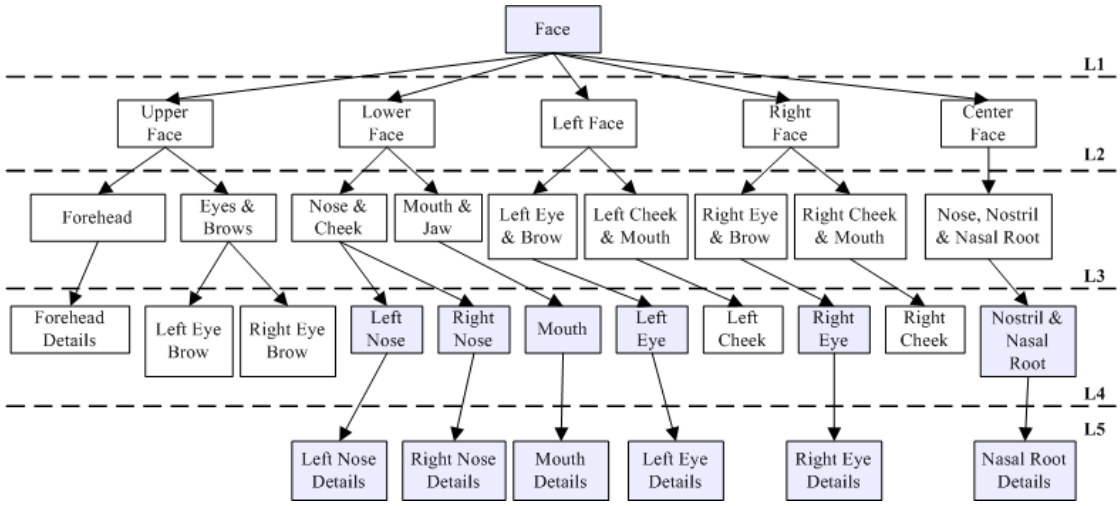


Figure 6.8 – A Typical FEETS representation of a Human Face. Blue nodes are nodes in the used for HFTS representation.

Using this approach, the facial emotion can be represented by a 5 level deep tree structure model as shown in Figure 6.8. The entire face region acts as a root node and localized features upper and lower face and left, right and centre of the face become the second level branch nodes. At the third level nodes, the forehead, eyes,

nose, mouth and cheek area become the corresponding branch nodes. At the fourth level, the forehead, eyes, eyebrows, nose, cheeks and mouth act as the branching nodes to the third level nodes. Sub-detail features from the 4-key fiducial points from the leaves of the tree structure. The leaf nodes have been grouped together as shown in Figure 6.8. There are about 8-9 features in most of these groups. The HFTS features are a subset of the FEETS representation, which is somehow similar to how the dorsal route receives signal from the V1 region of the visual cortex.

6.4 Database Preparation

6.4.1 Biometric Facial Recognition

For experiments of biometric facial recognition, two public face databases are used to evaluate the performance of the proposed method in terms of classification and verification capabilities. One of the evaluations is carried out by the YALE Face Database B (Georghiades *et al.*, 2001). In the database, it was divided into several subsets based on the different lighting conditions and posture orientations. An optimisation procedure was conducted using these data sets, which contains 5760 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses x 64 illumination conditions). Forty images out of the total 64 images of each pose and person were selected. So, there are 3,600 images for ten persons in the database to be used. The test dataset was split into two major groups, as determined by the lighting conditions, and into various subsets as determined by variation of postures. Normal and extreme lighting conditions are respectively termed as light source directions of the optical axis are less than or equals to 20° and 45°, shown in Figure 6.9 and Figure 6.10 respectively. Moreover, Figure 6.11 shows various poses

taken in the YALE Face Database. Pose 0 is the frontal images meaning the viewing direction is in angle of 0° with the camera optical axis. Poses 1, 2, 3, 4, and 5 were about 12° from the camera optical axis, while pose 6, 7, and 8 were about 24 degrees from the axis.

Figure 6.9 - Normal lighting conditions.

Figure 6.10 – Extreme lighting conditions.

Pose 0 Pose 1 Pose 2 Pose 3 Pose 4 Pose 5 Pose 6 Pose 7 Pose 8

Figure 6.11 – Pose 0 to 8 in the YALE Face Database.

Verification performance can be evaluated by plotting in false acceptance rates. We tested the model on another widely used face database, namely ORL Database (Samaria & Harter, 1994), which comprises of 10 different images per person. A total of 40 persons are involved in the database. Five images out of the 10 images were used as the training image and the other five images were used for testing. The original images of the size of 92 x 112 pixels are shown in Figure 6.12.

Figure 6.12 – Original Images of 92 x 112 pixels of various persons in the ORL Database.

Pose 1 Pose 2 Pose 3 Pose 4 Pose 5 Pose 6 Pose 7 Pose 8 Pose 9 Pose 10

Figure 6.13 – Pose 1 to 10 of the ORL Database, they are cropped and resized.

In order to extract the facial region, the images were cropped out from the original pictures. They were then scaled to the dimensions of 100 x 100 pixels. Figure 6.13 shows the faces in different poses being cropped out and scaled up to 100 x 100 pixels. The location of the eyes, nose and centre of lips can then be detected properly.

6.4.2 Emotion Recognition

For the experiments of emotion recognition, two public domain facial expression databases were used to evaluate the performance of our approach in this paper. The Japanese Female Facial Expression (JAFFE) Database compiled by the Psychology Department in Kyushu University, Japan (Lyons *et al.*, 1998; Lyons *et al.*, 1999) contains 213 images of 7 facial expressions (including neutral) posed by ten Japanese actresses. The original images are of the size 256 x 256 pixels as shown in Figure 6.14.

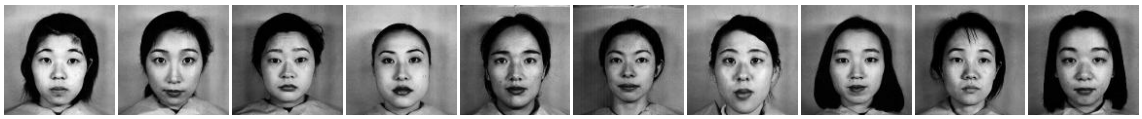


Figure 6.14 – Original image of 256 x 256 pixels of various persons in JAFFE database.

Another facial database used is the Cohn-Kanade AU-Coded Facial Expression Database (CMU Database) by Robotics Institute in Carnegie Mellon University, USA (Cohn *et al.*, 1997), which consists of about 2000 image sequences from over 200 subjects, aged from 18-30 years. In this database, sixty-five percent are female, fifteen percent are African-American, and three percent are Asian or Latino.

The original images are of the size 640 x 480 pixels as shown in Figure 6.15. The CMU Database has been used by the correct identification of the Action Units in the FACS systems.

Figure 6.15 – Original Images of 640 x 480 pixels in Cohn-Kanade AU-Code Face Expression Database (a) happy (AU 6+12+25) (b) anger (AU 4+L14+17) (c) disgust (AU 4+7+17+23+24) (d) fear (AU 1+2+5d+25+27) (e) happy (AU 6+12+16+25) (f) disgust (AU 15d +17e+B22) (g) anger (AU 4+6+7+9d+17b) (h) anger (AU 4+17+23+24) (i) fear (AU 1+2+5+25+27) (j) happy (AU 6+12+25) (k) sad (AU 25) (l) fear (AU 1+2+5+16+20+25) (m) disgust (AU 4+6+7+9d+17d+25).

To the best of our knowledge, few investigations have been conducted on analyzing face emotion behaviour among the different races in the Asian population. Most of the publicly available emotion database contains images that are captured from video recording or stored in low-resolution quality. The closet Asian emotion database is the JAFFE database. A 3D facial expression database by Yin *et. al*. (Yin *et al.*, 2006) contains 100 subjects in various emotions from various races found in the America. The development of our database was designed to capture high-resolution 2D facial images for various races, age groups and gender found in the Asian population in seven emotional states and in 3 different poses. As Singapore is in the heart of Asia and has a high mixture of different races in Asia, we have

collected our data from our University. The detail of the creation process is found in the Appendix A1 of this thesis. The Asian Emotion database is the third database used in the experiments, and contains 4947 images from 153 persons in 6 basic emotions and 1 neutral emotion.

All of the 213 images from the JAFFE database and only 1079 images of 20 subjects from the CMU database were selected and used for our experiments. These images are in frontal pose taken under normal lighting conditions. The face regions were cropped out from the original images and resized to 200 x 200 pixels, in the sense that the location of the eyes, nose and center of mouth can be easily detected. The proposed model is benchmarked under subject-dependent and subject-independent conditions.

For the JAFFE database, the training set of subject-dependent condition contains 143 images of the 10 persons to express 7 facial expressions. The testing set contains the remaining 70 images covering all the subjects and their expressions. The CMU database does not come with an emotion metadata for all the images. However, a set of AU values is given for each of the image. We performed an AU lookup for the relative emotions for each image. As the database was created for pseudo expressions rather than emotions, some of the images do not have a clear emotion classification and they were subsequently excluded from this experiment. We ignored to use and classify surprise expression in this experiment because it is difficult to find any image that was identified as surprise expression according to its AU even though some of them were looked like surprise with expressing the jaw drop and the eyebrow raise. Those expressing are also appearing at a facial image that was identified as fear emotion. A total of 77 images, which is the minimum number for one of the emotion class, for each of the 5 remaining emotions from the database were

used in this experiment. For the CMU database, the training set of subjects-dependent condition comprises of 317 images for 5 basic facial expressions. The testing set contains the remaining 58 images covering all the persons and their expressions. For the Asian Emotion database, the training set of subjects-dependent condition comprises of 3901 images for 6 basic facial expressions. The testing set contains the remaining 1046 images.

Table 6.1 – Distribution of Training and Test images.

Subjects	JAFFE		CMU		Asian Emotion	
	Training	Test	Training	Test	Training	Test
Dependent	143	70	317	58	3901	1046
Independent	170	43	310	75	3957	990

The purpose of having subjects-independent dataset is to test the robustness of the approaches in recognizing an emotion of subjects, which is not found in the training set. The experiments were run using 5-fold cross validation method, which means that different subjects were used for training and testing for each run. For the JAFFE database, we used 8 subjects with all 170 images in various expressions as for the training set. The testing set contains the remaining 43 images. For the CMU database, we performed 5-fold cross validation and in each fold we used 310 images for training and 75 image for testing. For the Asian Emotion database, we performed 5-fold cross validation and in each fold we used 3957 images for training and 990 images for testing. None of the subjects used in the training set would appear in the test set during this validation. Table 6.1 is tabulated the distribution of the training and testing sets under subjects-dependent and subjects-independent conditions of these three databases.

6.5 Experimental Results and Discussion

This section presents an evaluation of the performances of the proposed HFTS and FEETS representation with the PRNN and LEO models. Several experiments were conducted to validate the model's performance. They are the biometric facial recognition problem and emotion recognition problem described in Section 6.5.1 and 6.5.2 respectively. The ORL and Yale database is used to benchmarked for the adaptive processing of HFTS model against various traditional classifier models. For face recognition, the key indicator for performance evaluations, the accuracy rate is defined as:

$$\frac{\text{Total Number of Correctly Classified Cases}}{\text{Total Number of Test Cases}}, \tag{6.10}$$

the verification rate is defined as:

$$\frac{\text{Total Number of Positive Class Identified}}{\text{Total Number of Positive Class}}, \tag{6.11}$$

and the false accepted rate is defined as:

$$\frac{\text{Total Number of Negative Classes Identified as Positive Class}}{\text{Total Number of Test Cases}}. \tag{6.12}$$

The purpose is to evaluate the performance of the systems used as an authentication tool. The evaluation results are presented by showing the trade-off between the verification rate and the false acceptance rate for each of the persons in the database.

The JAFFE, CMU and Asian Emotion database is used to study the performance of the adaptive processing of FEETS model against various traditional classifier models. The FEETS model is also compared with the traditional QuadTree to demonstrate the effectiveness of FEETS model in terms of number of features extracted and the scalability of the model. The adaptive processing of FEETS

recognition performance of the emotion recognition system is benchmarked against traditional classifiers.

For addressing the problem of obstructions on the face image for recognition, a system robustness experiment was conducted in Section 6.5.2. Sunglasses and scarf were manually inserted on the test images to simulate obstructions. Empirical studies show that the FEETS representation is more robust in recognizing face emotions. In section 6.5.3, a literature survey is conducted to compare the various emotion recognition engines against the FEETS adaptively processed using both LEO and PRNN models presented in the previous chapters.

6.5.1 Biometric Facial Recognition

Classification Performance

This section presents the classification performance of the proposed Human Face Tree Structure (HFTS) representation with the proposed PRNN and LEO models which benchmarked against the other well-known classifiers such as Support Vector Machine (SVM), K-nearest neighbour (KNN) (Aha & Kibler, 1991) and Naïve Bayes algorithm (John & Langley, 1995). All of these tested classifiers are used flat feature vectors as input patterns for classification. We employed the Weka 3.4 package (Witten & Frank, 2005) to implement the tested models whereas our proposed HFTS model was implemented by Matlab 6.5. John Platt's sequential minimal optimization algorithm (Platt, 1998) is used as a training algorithm for SVM classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. Multi-class problems are solved using pairwise classification. In the multi-class case, the predicted probabilities are coupled using Hastie and Tibshirani's pairwise coupling method

(Hastie & Tibshirani, 1998). KNN was implemented based on (Aha & Kibler, 1991), which uses normalized Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance. If multiple instances have the same (smallest) distance to the test instance, the first one found is used for classification.

Table 6.2 – Performance of HFTS model against other methods for different lighting conditions.

Method	Normal Lighting		Extreme Lighting	
	Mean	Deviation	Mean	Deviation
SVM	81.97%	11.37%	65.52%	9.82%
KNN	75.40%	13.02%	69.40%	9.43%
MLP	81.56%	3.04%	73.16%	4.55%
Naïve Bayes	72.35%	0%	52.15%	0%
PRNN	95.69%	1.37%	83.35%	2.76%
LEO	96.30%	1.24%	84.52%	2.68%

Table 6.3 – Performance of HFTS model against other methods for various poses.

Method	Frontal		Pose 2-6		Pose 7-9		All Pose	
	Mean	Dev	Mean	Dev	Mean	Dev	Mean	Dev
SVM	67.88%	13.21%	77.41%	8.37%	74.35%	11.12%	75.34%	9.68%
KNN	74.74%	10.48%	76.63%	8.67%	66.06%	14.83%	72.15%	10.91%
MLP	81.16%	2.98%	77.34%	1.52%	74.19%	3.84%	76.74%	1.73%
Naïve Bayes	62.53%	0%	91.03%	0%	49.45%	0%	64.11%	0%
PRNN	89.41%	2.91%	90.15%	1.52%	88.87%	2.53%	89.64%	1.28%
LEO	90.24%	2.85%	90.65%	1.48%	89.98%	2.45%	90.83%	1.15%

We also benchmarked with the PRNN and LEO models by using flat vector representation instead of the proposed HFTS representation as input features to the Back Propagation Through Structures (BPTS) algorithm. Table 6.2 and Table 6.3 demonstrate the classification results under the different lighting conditions and various poses respectively. The comparative results show that the proposed models exhibit a better performance with average classification rate of 95% for the normal lighting conditions and 83% for the extreme lighting conditions. It also shows that the proposed models perform better in pose variation recognition, having above 89%

recognition rate for various poses. The proposed cognitive and adaptive models have better generalization in the sense that it can handle more pose variations as highlighted by the results in Table 6.3.

We tested the models for robustness whenever some fiducial points could not be detected. For instance, the eyes feature detector had failed to detect the locations of eyes. The comparative results as shown in Figure 6.16 illustrating that the performance of the proposed models can achieve relative high accuracy of about 75% if the fiducial points of eyes had failed to detection, whereas the other classifiers are unable to achieve the similar result. The proposed method is able to maintain high recognition accuracy as feature relationship information is retained in the HFTS representation, as compared with other methods which uses a flat vector as input.

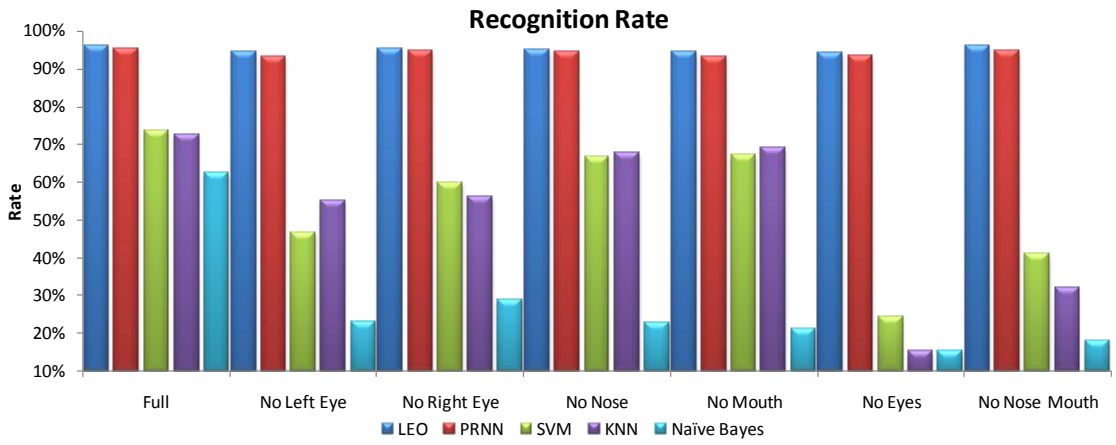


Figure 6.16 – Overall performance for HFTS model against other methods for missing fiducial points.

Verification Performance

Verification performance was being tested by each individual in the database enrolled with a number of images as the positive class and a number of random images from the other individual to form the negative class. Several approaches have been proposed for the evaluation on verification performance using ORL database. For instances, in (Samaria & Harter, 1994), a stochastic based hidden markov model (HMM) was employed for face identification that a verification error rate of 13% was

obtained. Lawerence *et. al.* (Lawrence *et al.*, 1997) proposed a convolutional neural network (CNN) approach and obtained the best error rate of 3.83%. Guo *et. al.* (Guo *et al.*, 2000) proposed to use the support vector machines (SVM) on classification of the ORL database, and achieved 3.0% and 1.5% for mean and lowest error rates respectively. All these approaches incorporating with holistic feature extraction are well-known to be used for face authentication. In our evaluations, the results are benchmarked in Table 6.4 that the proposed HFTS model is able to achieve the lowest error rate of 1% for face authentication. It is an evidence to show that the proposed HFTS model is able to develop a more accurate authentication system. Moreover, we compared the model with the other tested classifiers, such as SVM, KNN, MLP and Naïve Bayes.

Table 6.4 – Benchmarking with the other well-known models for face authentication.

Model	Error rate
Hidden Markov Model	13%
Convolution Neural Network	3.8%
Support Vector Machine	3%
The Proposed Model	1%

Table 6.5 – Verification Performance of the proposed model against the tested classifiers.

Method	Accuracy		Verification Rate		False Acceptance Rate	
	Mean	Deviation	Mean	Deviation	Mean	Deviation
SVM	97.18%	1.74%	48.13%	21.47%	2.09%	1.66%
KNN	96.80%	1.90%	54.38%	26.49%	2.57%	1.75%
MLP	96.94%	2.02%	63.32%	26.53%	1.96%	1.62%
Naïve Bayes	96.82%	3.42%	36.88%	25.31%	2.30%	3.44%
The Proposed Model	99%	0.22%	96.88%	10.11%	0.1%	0.06%

Figure 6.17(a) shows the recognition rate, Figure 6.17(b) shows the verification rate, and Figure 6.17(c) shows the false acceptance rates for each person

in ORL database. It is observed that the model obtained the lowest false acceptance rate of 0.1%. A low false acceptance rate is critically important as it governs the amount of impostors that are successfully authenticated as the user. The overall verification performances against the other tested classifiers are tabulated in Table 6.5 to show that the proposed HFTS model produces significantly encouraging results in terms of accuracy rate, verification rate and false acceptance rate.

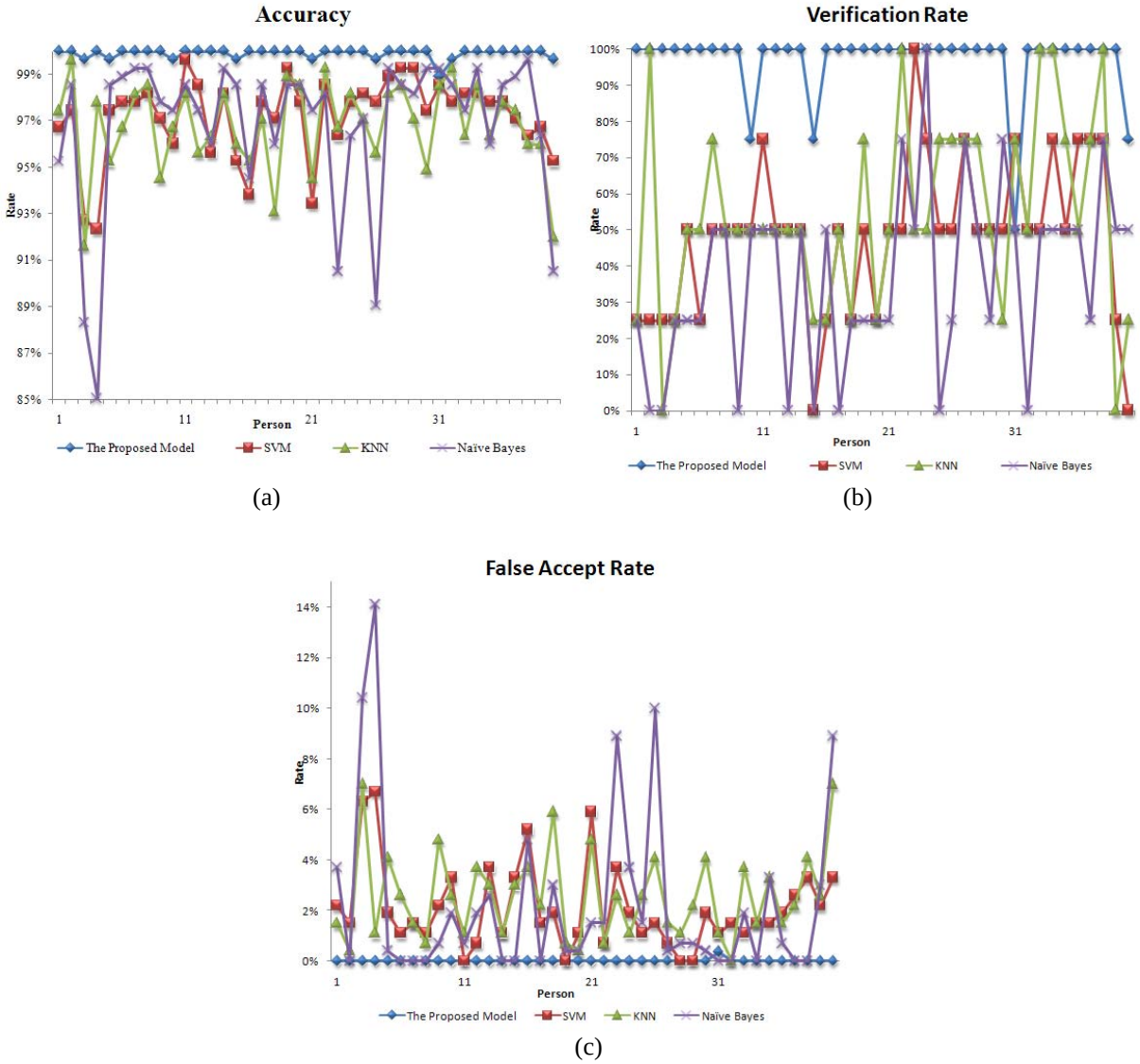


Figure 6.17 – Recognition, verification, and false accept rate for each person in the ORL database obtained by different classifier

6.5.2 Emotion Recognition

Tree structure representation: FEETS vs Quadtree

In this section, the evaluation of the features representations by the proposed FEETS is presented for emotion recognition using facial expression. Another type of tree structures called Quadtree was implemented to use in this experiment for the representations of facial images. Quadtree is a well known form of tree structures in which the entire face image (cropped from the background) might use as the root node in the Quadtree. The entire face image (200x200 pixels) is then divided into 4 equal quadrants and the features extended from these 4 quadrants are labelled for the second level of the Quadtree. Each of the quadrants is further divided into 4 quadrants, i.e. 16 squares are formed in the entire face. The features extracted from these 16 squares are also labeled for the third level of the Quadtree. Each of the labelled nodes of the sub-quadrants is connected to their parent quadrant nodes. The expansion of the Quadtree is continued until the desired tree is expected. In our study, the desired Quadtree is in four level depths which is sufficient to present all the details of face components. Features extracted in the Quadtree structure are based on the Localized Gabor Features vectors extraction.

Table 6.6 – Recall and generalization rates of FEETS vs 4-level QuadTree using PRNN model on JAFFE database. Dataset A – Subject Dependent, Dataset B – Subject Independent.

Datasets	QuadTree				FEETS			
	Recall Rate		Generalization Rate		Recall Rate		Generalization Rate	
	Average	Dev	Average	Dev	Average	Dev	Average	Dev
Dataset A	95.00%	0.25%	87.28%	0.44%	96.57%	0.39%	87.21%	0.32%
Dataset B	95.12%	0.60%	68.49%	2.2%	92.47%	0.31%	83.84%	1.41%

The comparative results between the Quadtree and FEETS are tabulated in Table 6.6. The recall rate shows the recognition rate when the system is tested with the training set, this is to highlight the performance of the system when the same input

is used. The generalization rate shows the recognition rate of the system when a completely different test set (i.e. does not include samples in the training data) is used to evaluate the system. The obtained results demonstrated that the FEETS has a better generalization than that of the QuadTree, particularly in the condition of subject independent. We also evaluated the recognition rate of the two different tree structures and the relationship to the depth of the tree structures nodes. The performances are presented in Figure 6.18. This is slightly improvement for the Quadtree if the number of tree level increases up to four for the emotion recognition under the subjects-dependent condition. However, for the condition of subjects-independent set, there is no significant improvement by the Quadtree representation even if the depth of the trees was increased. As FEETS representation make use of the localized feature extraction method, the feature might obtain to be closer to the corresponding facial components of interest than a generic Quadtree, hence the higher recognition rate could be yielded from the FEETS representation.

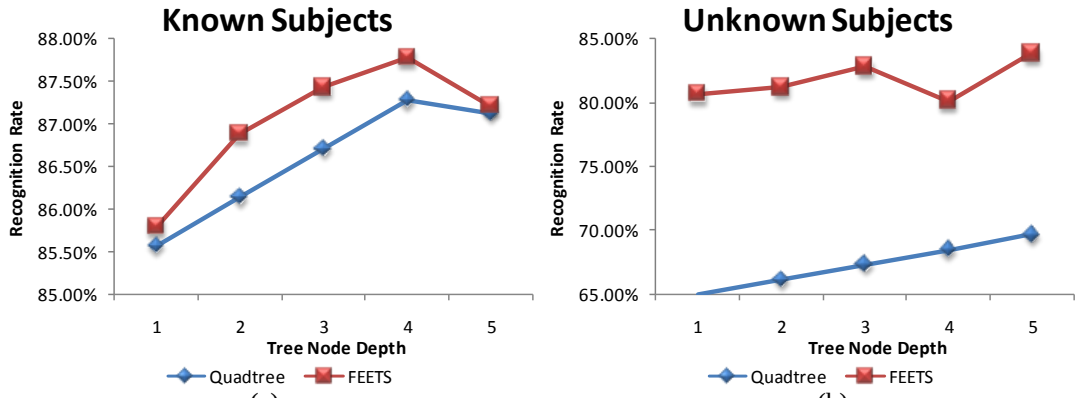


Figure 6.18 – Performances of FEETS vs QuadTree using different number of tree levels on JAFFE database.

Moreover, Table 6.7 shows the performance of these two tree structures features and processed using conventional classifers such as SVM, KNN and Naïve Bayes. The results demonstrate that the performances of the FEETS representation is

better than that of the Quadtree representation, particularly using advanced classifiers such as, the SVM or KNN classifier for emotion recognition.

Table 6.7 – QuadTree vs FEETS on CMU database benchmarking with other classifiers. Dataset A – Subject Dependent, Dataset B – Subject Independent.

Classifiers	QuadTree		FEETS	
	Dataset A	Dataset B	Dataset A	Dataset B
SVM	74.13%	34.54%	89.65%	87.06%
KNN	72.41%	32.98%	87.93%	85.24%
Naïve Bayes	55.17%	13.76%	72.07%	68.15%
PRNN	85.45%	56.10%	97.65%	95.87%

Recognition Performances

In this section, the performances of the capability of our proposed emotion recognition method are evaluated. As shown in the results, the proposed PRNN model is able to yield about 87% accuracy for the subjects-dependent condition by using the FEETS representations. The result shows likely to be similar to the recognition results obtained from the SVM and KNN classifiers. All the comparative results are tabulated in Table 6.8. Moreover, the robustness of our proposed model is investigated by recognizing emotions of the subjects-independent conditions. Again, about 83% classification rate in average is achieved by using our proposed model whereas the other classifiers can only yield about 20% accuracy for emotion recognition of the subjects who never trained by the system. The result implies that the LEO model with the FEETS representation has higher generalization capability than the other models. Table 6.10 shows the comparative results conducted by the CMU database. Again, the results reveal that the proposed cognitive model acts a good classifier even for emotional states in CMU database were not exactly defined. The confusion matrix presenting the interclass recognition error rate for each of the emotions is shown in Table 6.9 and Table 6.11 for JAFFE and CMU datasets

respectively. Each row lists the number of images in one emotion recognized to each of the 5 or 7 emotions by our method. Figure 6.20 shows the emotion response using the FEETS processed using the PRNN model from the relative image Figure 6.19. The histogram response shows that the model is able to response correctly to the various images showing each emotion.

Table 6.8 – Performances of the proposed model benchmarking against other models on JAFFE database.

Method	Subject-Dependent				Subject-Independent			
	Average	Max	Min	Dev	Average	Max	Min	Dev
Navie Bayes	58.57%	58.57%	58.57%	0.00%	9.30%	9.30%	9.30%	0.00%
SVM	80.50%	87.14%	70.89%	3.12%	20.50%	27.91%	12.58%	4.37%
KNN	75.53%	84.29%	67.68%	3.65%	19.18%	27.91%	12.35%	5.03%
PRNN	87.21%	88.57%	87.14%	0.32%	83.84%	86.05%	79.07%	1.41%
LEO	88.26%	89.02%	87.32%	0.25%	84.35%	87.39%	80.53%	1.23%

Table 6.9 – Confusion Matrices showing the interclass recognition errors using PRNN model in JAFFE.

	Subject-Dependent							Subject-Independent						
	AN	DI	FE	HA	NE	SA	SU	AN	DI	FE	HA	NE	SA	SU
Anger	7	1	0	0	0	2	0	4	0	1	0	1	0	0
Disgust	0	10	0	0	0	0	0	0	6	0	0	0	0	0
Fear	1	0	6	0	1	2	0	0	0	6	0	1	0	0
Happy	0	0	0	10	0	0	0	0	0	0	6	0	0	0
Neutral	0	0	0	0	10	0	0	0	1	1	0	4	0	0
Sad	1	0	1	0	0	8	0	0	0	1	0	0	5	0
Surprise	0	0	0	0	0	0	10	0	0	0	0	1	0	5
Overall error rate	12.8%							16.3%						

Table 6.10 – Performances of the proposed model against other models on CMU database.

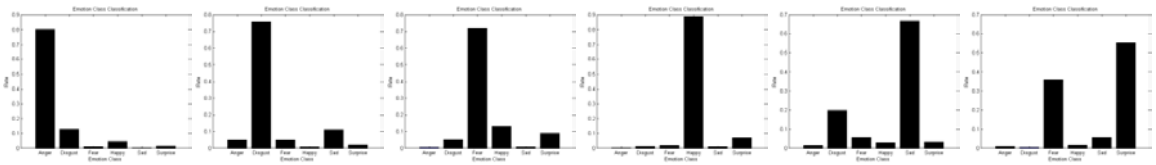
Method	Subject-Dependent				Subject-Independent			
	Average	Max	Min	Dev	Average	Max	Min	Dev
Naïve Bayes	72.07%	86.21%	63.79%	8.74%	68.15%	88.31%	51.94%	14.08%
SVM	89.65%	96.55%	82.76%	5.45%	87.06%	97.40%	81.03%	6.42%
KNN	87.93%	96.55%	81.03%	6.67%	85.24%	96.62%	79.48%	6.76%
PRNN	97.65%	99.5%	95.27%	1.77%	95.87%	98.18%	94.40%	1.43%
LEO	98.52%	99.8%	96.48%	1.43%	96.33%	99.21%	95.25%	1.29%

Table 6.11 – Confusion Matrices showing the interclass recognition errors using PRNN model in CMU.

Emotion	Subject-Dependent					Subject-Independent				
	AN	DI	FE	HA	SA	AN	DI	FE	HA	SA
Anger	12	0	0	0	0	74	0	0	1	0
Disgust	0	12	0	0	0	0	73	0	1	0
Fear	0	0	12	0	0	2	0	73	0	1
Happy	0	0	0	11	0	0	0	0	75	0
Sad	0	0	0	0	11	0	0	0	0	75
Overall error rate	0%					1.59%				

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

Figure 6.19 - Six Basic Emotions a) Anger, b) Disgust, c) Fear, d) Happy, e) Sad and f) Surprise.



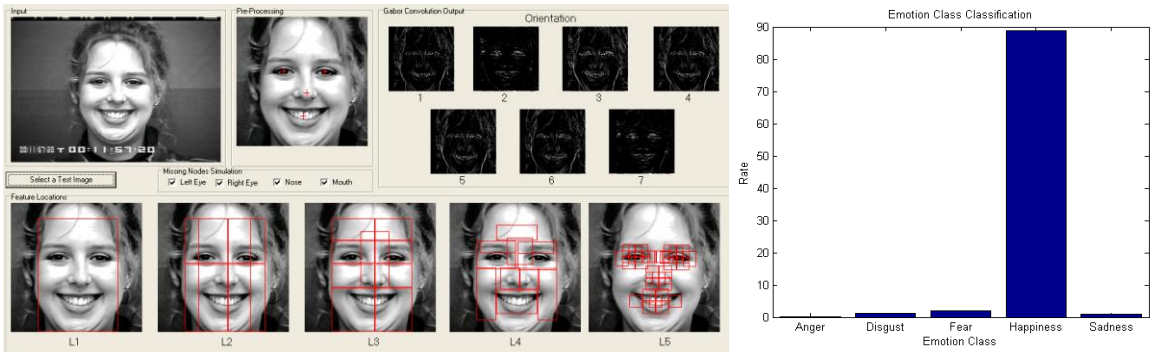
a) b) c) d) e) f)

Figure 6.20 - Output response of PRNN for the Six Basic Emotions a) Anger, b) Disgust, c) Fear, d) Happy, e) Sad and f) Surprise.

System robustness

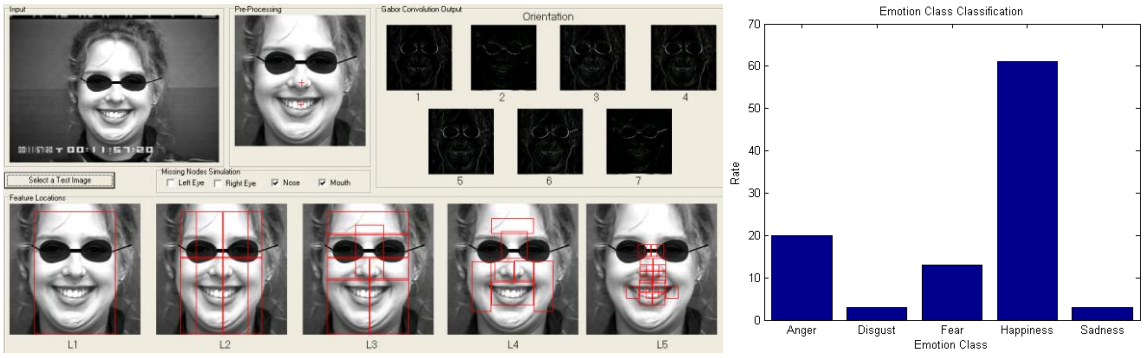
We also evaluated the robustness of our proposed method in case of the fiducial point detection failure by simulating some tested subjects in the CMU dataset in which the facial components might be covered by artifacts, e.g., sunglasses and veil, as shown in Figure 6.22 and Figure 6.23 respectively. These artifacts were drawn onto the test images, and the artifacts were not presented in the original CMU image dataset. Figure 6.21 shows the same subject without any artifacts. The histogram response shown in the same figure manifests the output response of the PRNN model by processing the tree structure. The highest bar represents the highest response of the respective emotions which indicates that such emotion is detected. We observed that

if the eyes were covered, for example shown in Figure 6.22, the output response of the model indicated that there is an increase in possibility of the emotion expressed from the tested image being anger, disgust and fear, which all are said to be negative emotion.



(a) (b)

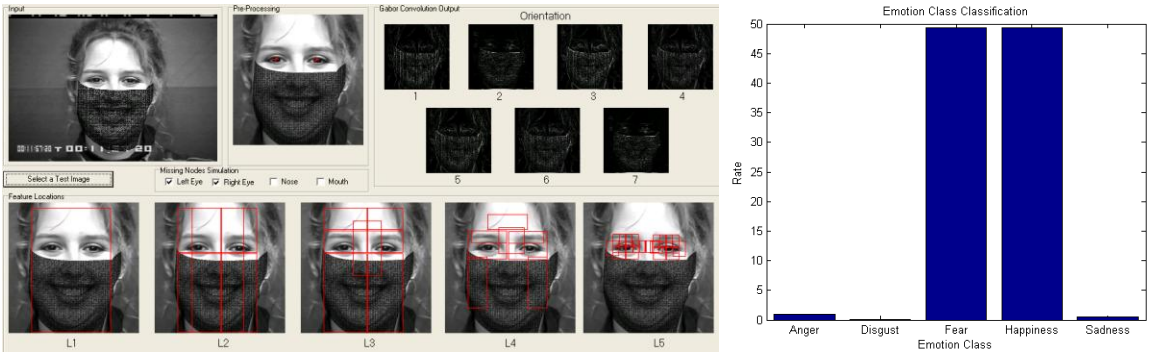
Figure 6.21 – (a) Subject without any artifacts; (b) Histogram showing the output response of the PRNN model.



(a) (b)

Figure 6.22 – (a) Subject wearing sunglasses, eye detection is unable to detect locations of eyes; (b) Histogram showing the output response of the PRNN model.

On the other hand, if the nose and mouth were covered by a veil as shown in Figure 6.23, the model is unable to clearly identify an emotion expressed in the image with artifacts. The response obtained by generalizing the eyes features indicated that the subject is most likely to be experiencing fear or happiness, as the most indicative feature being the mouth for happiness was masked. The facial information is insufficient to identify the expression even by the human perceiving.



(a) (b)

Figure 6.23 – (a) Subject wearing veil, nose and mouth detection is unable to detect locations of nose and mouth; (b) Histogram showing the output response of the PRNN.

Table 6.12 - Performance of face emotions recognition for missing fiducial points in subject-dependent and subjects-independent conditions. Set A – Subject Dependent, Set B – Subject Independent.

Method	FEETS		SVM		KNN		Naïve Bayes	
Conditions	**Set A**	**Set B**	**Set A**	**Set B**	**Set A**	**Set B**	**Set A**	**Set B**
Full	97.65%	95.87%	89.65%	87.06%	87.93%	85.24%	72.07%	68.15%
No Left Eye	97.65%	94.40%	48.28%	49.87%	79.31%	50.67%	44.83%	27.20%
No Right Eye	96.28%	92.53%	29.31%	20.80%	87.93%	61.33%	31.03%	24.27%
No Nose	97.65%	93.88%	62.07%	33.33%	77.59%	59.47%	37.93%	26.67%
No Mouth	97.65%	92.00%	53.45%	26.93%	84.48%	55.47%	50.00%	24.80%
No Eyes	96.28%	88.00%	25.86%	22.40%	37.93%	27.73%	18.97%	22.13%
No Nose Mouth	97.65%	91.47%	41.38%	36.53%	58.62%	28.27%	29.31%	27.73%

Table 6.12 shows the performance of our approach against the other methods for the emotion recognition in case of missing features. The obtained results indicated that our proposed system is more robust than the other classifiers, as it adopted an analysis of the facial features from global to local by the tree structure representation, which encodes the relationship information between the extracted features. The impact of losing local features due to missing fiducial points during feature extraction is less significant than the other methods. The proposed model uses top down analysis, so that the impact of losing a child node would be less significant as the analysis of the facial features could be contributed on the other nodes in the tree structure. The top 3 levels of the FEETS do not depend on the location of the fiducial points, hence these nodes can be created when there are missing fiducial points.

The experiment was repeated using Asian Emotion Database. We have used all the 4947 images from our database containing 153 persons in 6 basic emotions and 1 neutral emotion. For this experiment, we created 2 datasets, i.e., subject-dependent dataset (Dataset A) and subject-independent dataset (Dataset B). Subject-dependent dataset would be able to show how well the system performed when the system knows how each of the subject's pseudo emotions looked. The training images contain 3901 images of all 153 subjects in various emotions. The testing set is using the remaining 1046 images. Subject-independent dataset would be able to evaluate the performance of this system when it is used in a situation where, prior knowledge of subject is not available, as the system does not know how the evaluated subject's pseudo emotions will look. We performed 5-folds cross validation (each fold contains 120 subjects as the training set and remaining 30 subjects as test set) in this evaluation, so that all the 150 subjects will be trained and tested. Table 6.13 shows the performance of the various models using the Asian emotion database.

Table 6.13 - Performance of FEETS/LEO model against other classifiers. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial). Dataset A – Subject Dependent, Dataset B – Subject Independent.

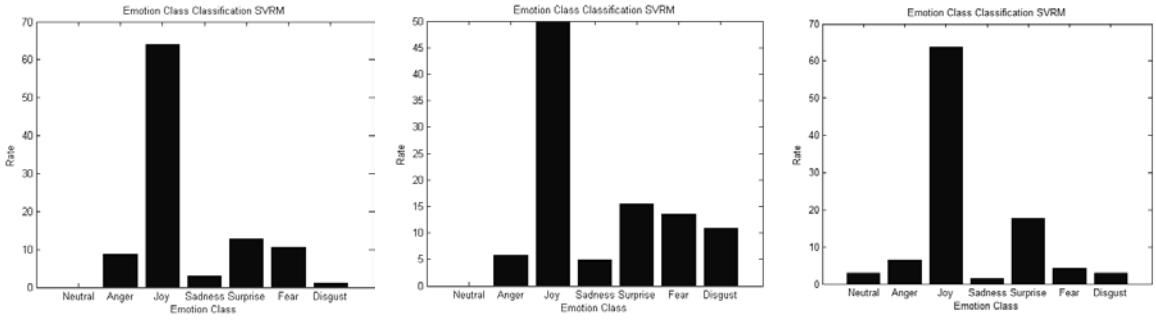
	LEO	PRNN	SVM	C45	RBF	NBM
Dataset A	77.3%	62.5%	52.9%	45.8%	30.5%	16.1%
Dataset B	57.0%	56.8%	50.0%	41.4%	31.7%	15.7%

A further evaluation to the system against extreme conditions where feature detection failed completely was conducted. Feature loss could be considered a high occurrence error for any facial or image recognition problem, as subjects might not always be in perfect view of the camera, for example, objects occlusion or self-occlusion. Figure 6.24 shows some extreme situations where feature detector will fail. Figure 6.24 (b) shows a subject wearing a veil, in which it will cause nose and mouth

detector failed to detect the corresponding feature locations. Similarly, wearing sunglasses as shown in Figure 6.24 (c) will cause eyes detection failure to detect the eyes feature.

(a) (b) (c)

Figure 6.24 - (a) Subject without any artifact, (b) subject wearing a veil, (c) subject wearing sunglasses.



(a) (b) (c)

Figure 6.25 - Likelihood histogram showing LEO output response for fig. 13 scenarios, (a) subject without any artifacts, (b) subject wearing a veil, (c) subject wearing sunglasses.

Those are able to evaluate the system's robustness when features are lost due to failure to detect feature locations. In this experiment, undetected features were padded with zeros in order to retain the similar length for the feature vector, as well as the shape of the FEETS representation. The system was evaluated against various degrees of loss, i.e. single eye missing, nose missing, mouth missing, as well as multiple feature loss, i.e. eyes missing, nose and mouth missing. Using the perfect features location and full features as a training set, we evaluated the system when tested with missing features. Figure 6.25 shows the corresponding responses from the LEO model for each of the scenarios in Figure 6.24. It is obvious that the 'Joy' emotion was detected by the proposed LEO model even though there are the situations of the features hiding by the artifacts.

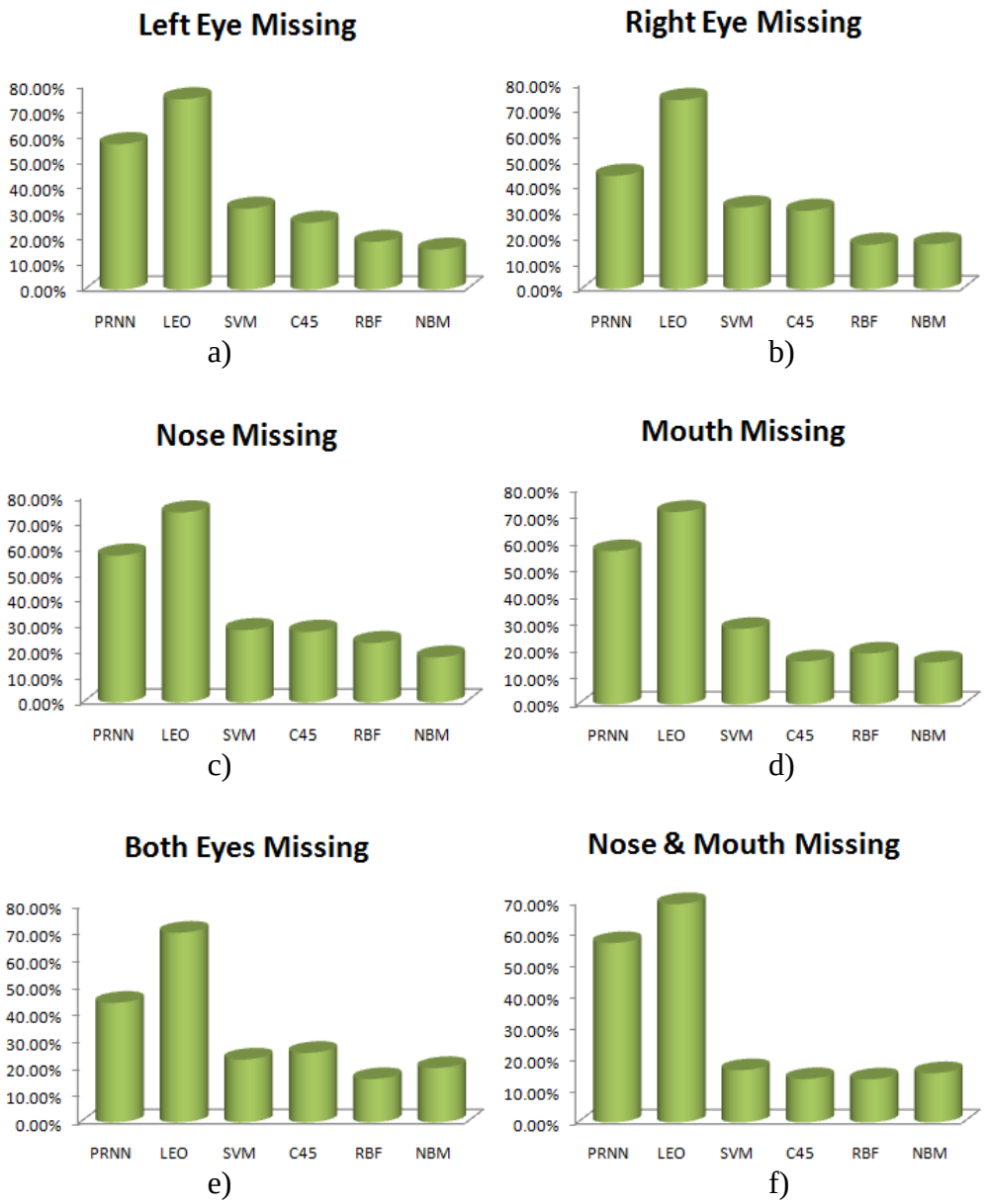


Figure 6.26 - Performance results of missing features evaluation by the LEO model against other models for dataset A (subject dependent). (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial).

Figure 6.26 and Figure 6.27 show the performance of the LEO model against other approaches using both Dataset A and Dataset B respectively. Experiments in Dataset B were run under 5-folds cross validation to ensure that all the subjects are trained and tested in turns. The experimental results show that the proposed LEO model is more robust when features were getting lost. We observe that the performance of the LEO model is better than the others when features were missing

rather than features were selected from a wrong region as shown in Figure 6.30. It is because of a missing feature condition in the LEO model that the affected node could be removed from the FEETS representation. A recommendation is drawn that features locations with a low confidence levels should be considered as undetected locations as divination in the feature location generates more noise into the system during evaluation.

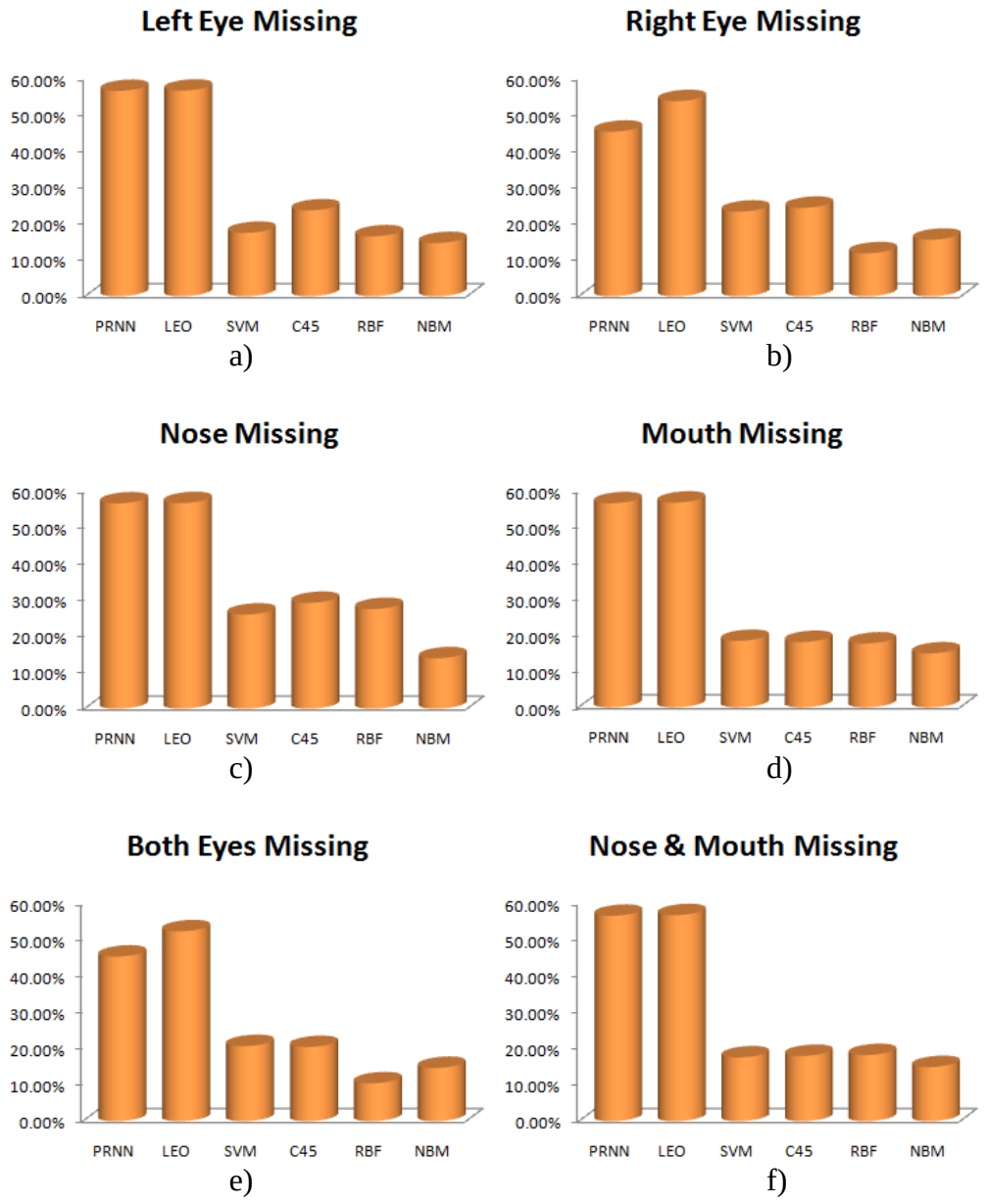


Figure 6.27 - Performance results of missing features evaluation by the LEO model against other models for dataset B (subject independent). (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial).

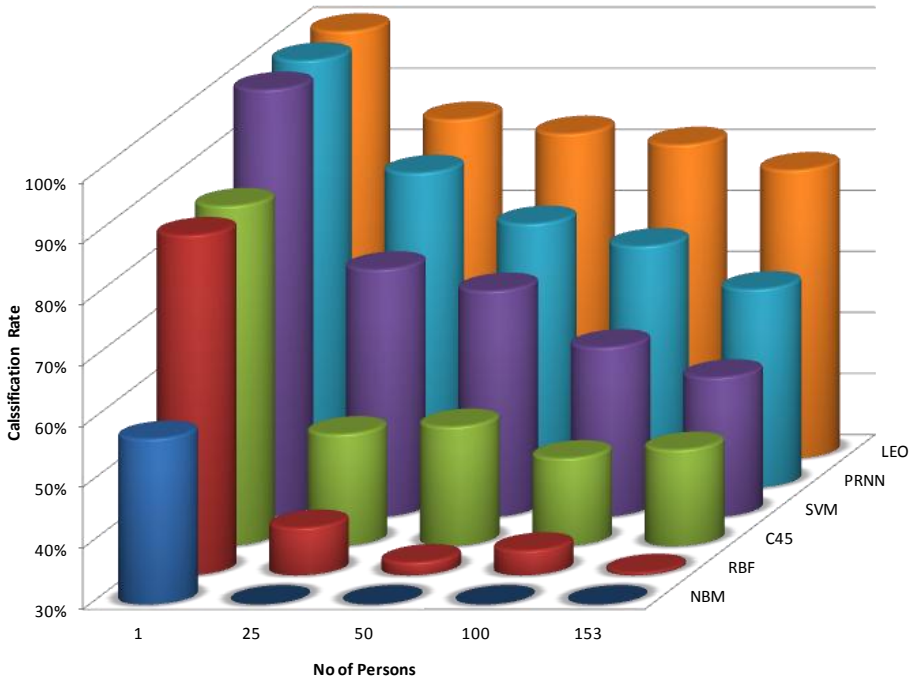


Figure 6.28 - Scalability performance of model using different numbers of persons in training and testing set. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial).

Moreover, the scalability of the LEO model by training and testing the model using different number of persons in the training data and test data was investigated. We tested the performance under 1, 25, 50, 100 and 153 persons in the training and test data, where the experiment was performed using Dataset A. Figure 6.28 shows that the LEO model is more scalable than the other models. We observe that most of the classification models are able to obtain high recognition rate when there is small number of person (may be less than 10). However, there is a huge performance difference when the number of subjects is increase to 25. This is due to the increasing in the number of variations created by various people for their pseudo emotion expressions. We observed that the LEO model is robust in when the number of subject increased, as the feature-to-feature relationship is preserved in the FEETS representation.

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

Figure 6.29 - Examples of features detection errors. (a) feature location is 5 pixel or less off from the ideal center of features, (b) features are 6 to 10 pixels off from the ideal center of features, (c) 11 to 15 pixels off from the ideal center of features, (d) more than 16 pixels off the ideal center of features.

We benchmarked our two proposed models, i.e., the PRNN and LEO models. We also performed empirical studies using other well-known classifiers such as Support Vector Machine (SVM), C4.5 Tree (C45), Gaussian Radial Basis Function network (RBF network), and Naïve Bayesian Multimodal (NBM) classifier (Mccallum & Nigam, 1998), which were performed from the weka package (Witten & Frank, 2005). All of these tested classifiers were used with flat-vectors input such that some regularities inherently associated with the tree structures were broken and less significant generalization results were yielded. Some of those models might suffer from poor convergence and resulted in a relatively low classification rate. Table 6.13 summarized the performance of these models against other classification model for perfect feature location detection in the face emotion recognition system. Two datasets, subjects dependent and independent, were being used and the results showed that the LEO model is able to yield a better performance that the other models.

An evaluation on how well our system performed when there is noise in the accuracy of the feature detectors, i.e., error in locating the center of features, was conducted. Figure 6.29 shows some examples of error in locating the center of features at various degrees of error. Under normal situations, it is necessary to consider error levels, less than 15 pixels off the center of feature locations, as a common error in feature detection processing. We benchmarked our system

performance for this type of noise by using the Dataset A, against other classification models. The results as shown in Figure 6.30 demonstrate that the hierarchical learning model, i.e. the LEO model, is the most robust and less subjective to this type of noise. It also showed that traditional classification models might suffer from noise in the feature detection inaccuracy existing in the facial emotion recognition system.

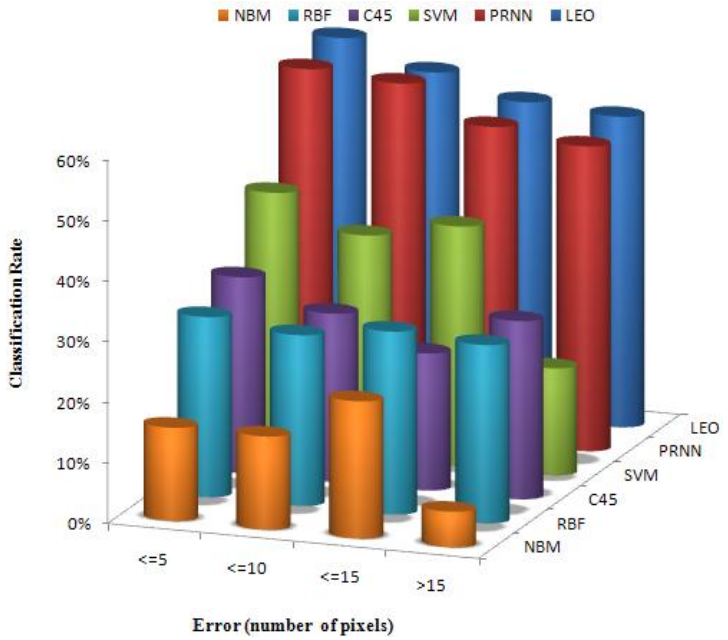


Figure 6.30 - Chart showing the performance of the classification models for detecting error in the fiduical point locations. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SVM – Support Vector Machine, KNN – K-Nearest Neighbor, C45 – decision tree, RBF – Radial Basis Function, NBM - Naïve Bayesian Multinomial).

Figure 6.31 - Chart showing the amount of time taken during training for the two models. Note that the codes in these models are not optimized, and are running based on both Matlab platform and from WEKA package. (LEO – Local Experts Organization, PRNN –Probabilistic Recursive Neural Network, SMO – Sequential Minimum Optimization, C45 – decision tree)

We evaluated if the LEO model meets the objective of reducing the training time needed for processing FEETS representation. The LEO model was benchmarked in terms of time required for training against PRNN model as well as other classification models, and result in Figure 6.31 showed that the LEO model required about 2291 seconds as compared to 57934 seconds using the PRNN model.

6.5.3 Comparison with other recognition approaches

Actually, it is difficult to directly compare the results of the proposed method with other facial emotion recognition approaches since most of the research works were conducted under different types of testing and using different sets of data. They all are using either a static image or a sequence of images of one expression. Due to such difficulties, we discuss and compare here only based on the results of their independent tests of static images. The most recent recognition results (i.e. from 2002 to 2006) are tabulated in Table 6.14 and compared with the proposed method. It is observed that high recognition rates can be achieved when the numbers of human subjects, emotion types and testing images are small.

Table 6.14 - Comparison with other recognition approaches.

Approaches	Database Nature	Numbers of testing images	Recognition rates
Abbound & Davoine	10 subjects and 7 recognized emotions	70	83.3%
Ma & Khorasani	60 subjects and 4 recognized emotions	80	93.8%
Zheng et al.	JAFFE (10 subjects and 6 emotions)	183	77.1%
Zheng et al.	EKMAN (14 subjects and 6 emotions)	96	79.2%
Guo & Dyer	JAFFE (10 subjects and 6 emotions)	213	91.0%
Wu et al.	10 subjects and 6 emotions recognized	183	83.2%
Proposed Model	**JAFFE (10 subjects and 7 emotions)**	**113**	**85.2%**
Proposed Model	**CMU (20 subjects and 5 emotions)**	**133**	**96.5%**

From Table 6.14 Abbound and Davoine (Abboud & Davoine, 2004) used 70 unknown images of 10 subjects for recognition of 7 emotions (i.e. the neutral emotion

was included). A mean accuracy rate of 83.3% was achieved. Ma and Khorasani (Ma & Khorasani, 2004) proposed using a constructive feedforward neural network for recognizing 5 basic emotions including neutral. A recognition rate of 93.8% on a database of 60 subjects was achieved in which 40 subjects were used for training and 20 subjects were used for testing but without including neutral emotion in the testing. Guo and Dyer (Guo & Dyer, 2005) employed 213 images of 7 emotions from JAFFE database as experimental data and reported an accuracy of 91.0%. Moreover, Zheng et al. (Zheng *et al.*, 2006) proposed to use kernel based correlation analysis to recognize the emotions from both JAFFE and EKMAN databases where 77.1% and 79.2% recognition rates were obtained respectively. Wu et al. in (Wu *et al.*, 2005) reported an accuracy of 83.2% by classifying six basic emotions from 10 subjects. Our approach has produced recognition rates of 87.2% and 97.6% for JAFFE and CMU databases respectively. The result using JAFFE database is slightly slower than the results obtained from the approaches of Ma and Khorasani (Ma & Khorasani, 2004) and Guo and Dyer (Guo & Dyer, 2005). However, they had used larger numbers of subjects for training (40 subjects required) and less numbers of emotions for recognizing (4 emotions only). We have employed less numbers of subjects for training (8 subjects only for CMU database) and more numbers of emotions for recognition (7 emotions with recognizing neutral emotion). Hence, our results are more robust and working better for generalization than the others.

6.6 Conclusion

Human Face Tree Structure (HFTS) and FacE Emotion Tree Structure (FEETS) are novel contribution to the facial interpretation and understanding domain. Unlike flat vector data representation, HFTS and FEETS allow patterns to be properly

represented by directed graphs or trees. These patterns are subsequently processed using Probabilistic Recursive Neural Network (PRNN) model presented in Chapter 4 and Local Experts Organization (LEO) model presented in Chapter 5.

In this study, the model proposed for classifying the facial features is inspired by the neural pathway of the human brain for face processing. The images are filtered using Gabor wavelets, and the local features were extracted based on the location reference to the center of fiducial points. The fiducial points were detected and, the features extracted were transformed to HFTS and FEETS depending on the application the images were used for.

Empirical studies were conducted to demonstrate that the model was able to improve the recognition rates of face recognition systems that were plagued by the problems of harsh lighting conditions as well as pose variations. The results showed that the average classification rate processed by the models were 96% for normal lighting conditions and 83% for the extreme lighting conditions. It also shows that the proposed HFTS model performs better in pose variation recognition, having above 89% recognition rate for various poses.

Benchmarking results for emotion recognition system using the FEETS model show that the architecture is a stronger than traditional methods for representing data as well as processing them. The FEETS representation when processing using PRNN and LEO models yields a recognition rate of above 98% for CMU database for subject dependant dataset. On a large database such as Asian Emotion database, the recognition rate for subject dependent dataset for FEETS yields of above 77%. Further studies were conducted to show the robustness of the system when it comes to variation of location of the fiducial points to the accuracy rate of the system. Results showed that the FEETS processing using PRNN or LEO model have the highest

accuracy rates as compared to other models and was the least affected by the location deviation.

The third major problem in facial recognition system is the presence of occluding objects such as wearing sunglasses or scarf. Experiment results shows that the FEETS representation is able to accurately recognise the images with an accuracy of more than 65% as compared to traditional classifiers which yields less than 30%. The FEETS model is more robust than other conventional classification models while facial features are lost due to undetected key feature locations. It is also concluded that the LEO model is the most scalable in large database system for face emotion recognition. Further studies should be carried out to enhance the effective of self-evolving learning as such enhanced model would have significant use in real-time classification and regression problems.

Chapter 7

Conclusions and Future Research

This chapter summarizes and concludes the research works that carried out, the research issues as well as the obtained achievements of this thesis. Finally, the possible solutions are suggested for further studies along with the research in the future.

7.1 Summary

The goal of this thesis is to investigate the potential of cognitive connectionist models for solving recognition problems with erratic patterns. The research first identified and investigated various feature representation methods in both structured and unstructured manners. Then, the general idea of the adaptive processing of tree structures approach was described to represent and recognize structured patterns. The model, learning algorithm and its problems were presented. In addition, some recent advanced techniques, for example, an improved BPTS learning algorithm, unsupervised model as well as genetic evolution processing of tree structures, were also discussed. Apart from those developments, some other important issues are necessary to be addressed in this research, for examples, the discriminative capability of the connectionist model, the initialization sensitivity, the potential of connectionist

models for pattern recognition. Those issues have been investigated and described in this thesis in which the summaries are as follows:

7.1.1. The Cognitive Connectionist Models

Chapter 4 presented using a novel approach of applying probabilistic based recursive neural network (PRNN) for adaptive processing of tree structures. Two problems encountered by BPTS algorithm were addressed. The major contributions of this chapter lies in the use of the Gaussian Mixture Model (GMM) architecture at the hidden layer and use of recursive neurons at the output layers for adaptive processing of data structures. Conceptual and empirical analysis demonstrated the effectiveness of this model in terms of decision boundary, computational complexity and convergence analysis. Empirical studies were conducted using two experiments, namely Traffic policeman simulation and natural scene image classification. The BPTS typed algorithms were used to benchmark with the PRNN model. The experimental results showed that the PRNN lead the BPTS algorithm by 3 folds (10 vs 30s) for learning time respectively. It also obtained encouraging classification accuracy for both the policeman simulation and natural scene image classification experiments. Nevertheless, it is found that the PRNN model encounters the problem of high computational cost required for learning the structured patterns using the penalized optimization and EM algorithm. Moreover, the high sensitivity of the initialization to the model parameters might influence the generalization results consistency.

Chapter 5 presented another novel connectionist architecture called Local Experts Organization (LEO) model, which is inspired by the brain's cognitive functions to recognize objects at different parts of the human brain. The major

contribution of this LEO model is to overcome the limitations of the PRNN model. The LEO architecture is a hybrid structure that uses support vector machine (SVM) and reduced multivariate polynomial (RM) classifier as the local expert as well as the fusion classifier respectively. The benchmarking results using the traffic policeman simulation shows that the learning time is one-fifth of that using the PRNN model and ten times faster than that of the BPTS algorithm. The reason for this speed improvement lies in the parallel processing capability of the LEO model for learning the nodes as compared to the sequential order needed in both BPTS and PRNN models. LEO model also scored the lowest Equal Error Rate (EER) under different image variations scenarios amongst the other EERs obtained from various tested models. It is proven that the LEO model is able to be more robust in the problem domain of natural scene image classification.

7.1.2. Facial Processing Applications

Chapter 6 described how the PRNN and LEO models can be applied to facial image understanding and interpretation. The major challenges of face recognition systems are variations of operating environment, such as lighting, pose variations, and obstructions for which sunglasses and/or scarf is presented in the subjects. Those variations would cause most face recognition systems to failure. The major contribution in this research comes from the offering of the Human Face Tree Structure (HFTS) and FacE Emotion Tree Structure (FEETS) to represent the facial image for personal identification and emotion recognition respectively. In this study, the models proposed for classifying the facial features are inspired by the neural pathway of the human brain for face processing. The methodology and representation of the features can be found in this case study chapter. Empirical studies were conducted to demonstrate that the models were able to improve the recognition rates

of face recognition systems that were plagued by the problems of harsh lighting conditions as well as pose variations.

In addition, an Asian Emotion Database was also created to test the robustness of the models. In the study of face emotion recognition, the FEETS representation is an extension from the HFTS representation and the extracted features for this representation are based on the FACS guidelines on the key locations of facial emotion traits. The benchmarking results showed that the PRNN and LEO models outperform the other classifiers such as, SVM, KNN, Naïve Bayes, RBF and C45, which are different from our approach that used the unstructured data representation.

7.1.3. Recognition of erratic patterns

Chapter 6, we also demonstrated that a human face when occluding with sunglasses or veil, using structured feature representation with adaptive processing by PRNN and LEO models are able to recognize the images with an accuracy of more than 65% as compared to less than 30% from traditional classifiers using unstructured features. It showed that the approach is more robust than the other unstructured feature representation while facial features are lost due to undetected key feature locations. It is also found that the LEO model is the most scalable in large database system for face emotion recognition as the training time required is nonlinearly proportional to the size of the database as compared to the other models. A comparison study was performed to compare with other known approaches to emotion recognition using similar dataset. Our models also outperform the existing models in that arena.

In essence, the works reported in this thesis conclude that using structured feature representation and adaptively processed using cognitive connectionist models such as the proposed PRNN and LEO models, the successful recognition rates achieved by face recognition systems are higher than those obtained using

unstructured feature representation. Structured feature representation is somewhat similar to that of the human's cognitive functions where various areas of the brain is trained and specialized in performing a specific function. Both PRNN and LEO models have the potential to be applied to other applications where erratic patterns are presented.

7.2 Future Research

Though promising results are achieved on several key issues in the investigation of using cognitive connectionist models for recognition of erratic patterns, many possible extensions still exist to improve the model further.

- **Self-evolving learning**

  The learning algorithm in both the PRNN and LEO model can be extended to use self-evolving learning algorithms as such enhanced models would have significant use in real-time classification and regression problems.

- **Automatic Extraction of structure from unstructured data**

  As demonstrated, the potential of structured data representation in the training of the data is enormous. However, prior knowledge of the data sampled is required to create a manual data hierarchy of the features extracted. Could an automated system such as decision trees be capable of extracting relational data information and somewhat preserving the structure of features relationship to that created by a manual operator?

- **Processing of Elastic Bunch Graph using PRNN and LEO models**

  Could a weighted tree structure such as Elastic Bunch Graph (EBG) be processed using PRNN and LEO models? The EBG offers automated feature location as well as feature extraction process, which are currently lacking in the HFTS and FEETS model for facial image understanding and interpretation. The EBG uses

weights instead of activated patterns to describe the relationship information presented in the face image. It could potentially attempt the question of whether the human brain's cognitive functions which are activity patterns without any weights to each of the nodes and branch inputs. It is known that a cognitive decision from an event might be influenced by previous experiences; for example, the human's body natural reaction to fear is fight or flight. However, when an adult sees a pit bull terror barking at him, his body would tell him to flight, but he/she would remain still based on the knowledge that flight would only lead him to injuries. Similarly weighted features might further enhance the performance of the system, by suppressing and amplifying nodes in the tree structures.

References

Abboud, B., & Davoine, F. (2004). *Appearance factorization based facial expression recognition and synthesis.* Paper presented at the 17th Int. Conference on Pattern Recognition.

Aha, D., & Kibler, D. (1991). Instance based learning algorithms. Machine Learning. *Machine Learning, 6*, 37-66.

Aras, S., Subramanian, A., & Zhang, Z. (2004). *Face Recognition*: University of Buffalo.

Baron-Cohen, S. (1995). *Mindblindness: An Essay on Autism and Theory of Mind*.

Bartlett, J. C., & Searcy, J. (1993). Inversion and Configuration of Faces. *Cognitive Psychology, 25*, 281-316.

Bartlett, M. S., & Sejnowski, T. (1997). Viewpoint invariant face recognition using independent component analysis and attractor networks. In M. Mozer, M. Jordan & T. Petsche (Eds.), *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Bechara, A., Damasio, H., & Damasio, A. R. (2000). Emotion, Decision making and the Orbitofrontal Cortex. *Cerebral Cortex*.

Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1996). EigenFaces vs. FisherFaces: Recognition Using Class Specific Linear Projection. *IEEE Trans. Pattern Anal. Machine Intell., 19*(7), 711-720.

Bengio, Y., & Frasconi, P. (1996). Input-Output HMMs for sequence processing. *IEEE Trans. Neural Networks, 7*, 1231-1249.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long Term Dependencies with Gradient Descent is difficult. *IEEE Trans. on Neural Networks, 5*(2), 157-166.

Bingham, E., & Hyvarinen, A. (2000). A Fast Fixed-Point Algorithm for Independent Component Analysis of Complex Valued Signal. *International Journal of Neural Systems, 10*(1), 1-8.

Brunelli, R., & Poggio, T. (1993). Face Recognition: Features versus templates. *IEEE Trans. Pattern Anal. Machine Intell., 15*, 1042-1052.

Cho, S.-Y., Chi, Z., Siu, W.-C., & Tsoi, A. C. (2003). An Improved Algorithm for learning long-term dependency problems in adaptive processing of data structures. *IEEE Trans. on Neural Networks, 14*(4), 781-793.

Cho, S.-Y., & Chow, T. W. S. (1999). Training Multilayer Neural Networks Using Fast Global Learning Algorithm - Least Squares and Penalized Optimization Methods. *Neurocomputing, 25*(1-3), 115-131.

Cho, S.-Y., & Wong, J.-J. (2007). Human face recognition by adaptive processing of tree structures representation. *Neural Computing and Applications*.

Cho, S. Y., & Chi, Z. (2005). Genetic Evolution Processing of Data Structures for Image Classification. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 216-231.

Cho, S. Y., & Chow, T. W. S. (1998). A layer-by-layer least squares based recurrent networks: Stalling and escape. *Neural Processing Letter, 7*(1), 15-25.

Cohn, J. F., Zlochower, A. J., Lien, J. J., Wu, Y.-T., & Kanade, T. (1997). *Automated face coding: A computer-vision based method of facial expression analysis.* Paper presented at the 7th European Conference on Facial Expression Measurement and Meaning.

Comon, P. (1994). Independent Component Analysis, a new concept? *Signal Processing, 36*, 287-314.

Cortes, C., & Vapnik, V. N. (1995). Support Vector Networks. *Machine Learning, 20*, 273-297.

Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., *et al.* (2001). Emotion Recognition in Human-computer Interaction. *IEEE Signal Processing Magazine, 18*(1), 32-80.

Craw, I., Costen, N., Kato, T., Robertson, G., & Akamatsu, S. (1995). *Automatic face recognition: Combining configuration and texture.* Paper presented at the

International workshop on Automatic Face and Gesture Recognition IWAFGR, Zurich.

Daugman, J. G. (1988). Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Trans. Pattern Anal. Machine Intell., 36*, 1169-1179.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc., Ser. B, 39*(1), 1-38.

Deng, Y., & Manjunath, B. S. (2001). Unsupervised Segmentation of Color-Texture Regions in Images and Video. *IEEE Trans. Pattern Anal. Machine Intell., 23*(8), 800-810.

Donato, G., Bartlett, M. S., Hager, J. C., Ekman, P., & Sejnowski, T. J. (1999). Classifying Facial Actions. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 21*(10), 974-989.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification 2nd Edition*: A Wiley-Interscience Publication.

Ekman, P. (1991). *Telling Lies*. New York: W.W. Norton.

Ekman, P. (1992). Facial Expression of Emotion: An Old Controversy and New Findings. *Philosophical Transactions of the Royal Society of London, 335*, 63-69.

Ekman, P. (1999). Facial Expressions. In T. Dalgleish & M. Powers (Eds.), *Handbook of Cognition and Emotion*. New York: John Wiley & Sons Ltd.

Ekman, P. (2004). *Emotions Revealed*. New York: Henry Holt and Company LLC.

Ekman, P., & Friesen, W. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, CA: Consulting Psychologist Press.

Ekman, P., & Rosenberg, E. L. (1997). *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression using the Facial Action Coding System (FACS)*. New York: Oxford University Press.

Ellis, H. D., & Lewis, M. B. (2001). Capgras delusion: a window on face recognition. *Trends in Cognitive Science, 5*, 149-156.

Ellis, H. D., & Young, A. W. (1990). Accounting for delusional misidentifications. *The British Journal of Psychiatry, 157*, 239-248.

Er, M. J., Wu, S., & Lu, J. (2002). Face Recognition with Radial Basis Function (RBF) Neural Networks. *IEEE Trans. Neural Networks, 13*(3), 697-710.

Field, D. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Amer. A, 4*(12), 2379-2394.

Forsyth, D. A., & Fleck, M. M. (1999). Automatic Detection of Human Nudes. *Internation Journal of Comuter Vision, 32*(1), 63-77.

Frasconi, P., Gori, M., Kuchler, A., & Sperduti, A. (2001). *From sequences to data structures: Theory and applications*. New York.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A General Framework for Adaptive Processing of Data Structures. *IEEE Trans. Neural Networks, 9*, 768-785.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of online learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95* (pp. 23-37): Springer-Verlag.

Gaffan, D., & Murray, E. A. (1990). Amygdalar interaction with the mediodorsal nucleus of the thalamus and ventromedial prefrontal cortex in stimulus-reward associative learning in the monkey. *Journal of Neuroscience, 10*, 3479-3493.

Gao, Y., & Leung, M. K. H. (2002). Face Recognition Using Line Edge Map. *IEEE Trans. Pattern Anal. Machine Intell., 24*(6), 764-779.

Georghiades, A. S., Belhumeur, P. N., & Kriegman, D. J. (2001). From Few to Many: illumination Cone Models for Face Recognition under Variable Lighting and Pose. *IEEE Trans. On Pattern Anal. And Machine Intell., 23*(6), 643-660.

Giles, C. L., & Gori, M. (1998). *Adaptive Processing of Sequences and Data Structures*. New York: Springer-Verlag.

Gökberk, B., Irfanoglu, M. O., Akarun, L., & Alpaydin, E. (2003). *Optimal Global Kernel Location Selection For Face Recognition.* Paper presented at the IEEE International Conference on Image Processing.

Goller, C., Gori, M., & Maggini, M. (1999). *Feature extraction from data structures with unsupervised recursive neural networks.* Paper presented at the International Joint Conference on Neural Networks, Washington, DC.

Goller, C., & Kuchler, A. (1996). *Learning Task-dependent distributed representations by back-propagation through structures.* Paper presented at the IEEE Int. Conf. Neural Networks.

Gori, M., & Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*, 76-86.

Guo, G., & Dyer, C. (2005). Learning from examples in the small case: face expression recognition systems. *IEEE TRansactions on Systems, Man and Cybernetics, Part B, 35*(3), 477-488.

Guo, G., Li, S. Z., & Chan, K. (2000). *Face Recognition by Support Vector Machines.* Paper presented at the IEEE Conference on Automatic Face and Gesture Recognition.

Hagenbuchner, M., & Tsoi, A. C. (2003). A Self-Organizing Map for Adaptive Processing of Structured Data. *IEEE Transactions on Neural Networks, 14*(3), 491-505.

Hammer, B. (2000). Learning with Recurrent Neural Network. In *Springer Lecture Notes in Control and Information Sciences* (Vol. 254). New York: Springer-Verlag.

Hammer, B., A., M., Sperduti, A., & M., S. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing, 57*, 3-35.

Hammer, B., & Sperschneider, V. (1997). *Neural networks can approximate mappings on structured objects.* Paper presented at the 2nd Int. Conf. Computational Intelligence Neuroscience.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *Annals of Statistics, 26*(2), 451-471.

Haxby, J. V., Hoffman, E. A., & Gobbini, M. I. (2000). The distributed human neural system for face perception. *Trends in Congitive Science, 4*, 223-233.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer Feedforward network are universal approximation. *Neural Networks, 2*, 359-366.

Isen, A. M. (2000). Positive Affect and Decision Making. In *Handbook of Emotions* (pp. 417-435). New York: Guilford Press.

Jain, A. (1989). *Fundamentals of Digital Image Processing*. NJ: Prentice Hall.

Jing, X.-Y., Zhang, D., & Yao, Y.-F. (2003). Improvements on the linear discrimination technique with application to face recognition. *Patter Recognition Letters, 24*, 2695-2701.

John, G. H., & Langley, P. (1995). *Estimating Continuous Distributions in Bayesian Classifiers.* Paper presented at the The Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo.

Kanade, T. (1973). *Picture processing by computer complex and recognition of human faces.* Unpublished Ph.D Dissertation, Kyoto University, Japan.

Kelly, M. D. (1970). *Visual Identification of people by computer*. CA: Standford.

Kirby, M., & Sirovich, L. (1990). Application of the Karhunen-Loeve Procedure for Characterization of Human Faces. *IEEE. Trans on Pattern Analysis and Machine Intelligence, 12*.

Kohonen, T. (1995). *Self-organizing maps* (Vol. 30): Berlin: Springer.

Kung, S. Y., & Taur, J. S. (1995). Decision-Based Neural Networks with Signal/Image classification applications. *IEEE Trans. Neural Networks, 6*, 170-181.

Lanitis, A., Taylor, C., & Cootes, T. (1997). Automatic interpretation and coding of face images using flexible models. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 19*(7), 743-756.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolution neural network approach. *IEEE Trans. Neural Networks, 8*, 98-113.

Levine, D. S. (2007). How Does the brain create, change, and selectively override its rules of conduct? In L. I. Perlovsky & R. Kozma (Eds.), *Neurodynamics of Cognition and Consciousness* (pp. 163-181): Springer.

Li, J., & Wang, J. Z. (2003). Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Machine Intell., 25*(9), 1075-1088.

Lin, C.-J., & Weng, R. C. (2004). *Simple Probabilistic Predictions for Support Vector Regression*: Department of Computer Science, National Taiwan University.

Lin, S. H., Kung, S. Y., & Lin, L. J. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. on Neural Networks, Special Issue on Biometric Identification, 8*(1), 114-132.

Liu, C. (2003). A Bayesian Discriminating Features Method for Face Detection. *IEEE Trans. Pattern Anal. Machine Intell., 25*(6), 725-740.

Liu, C., & Wechsler, H. (2002). Gabor Feature Based Classification using the Enhanced Fisher Linear Discriminant Model for Face Recognition. *IEEE Trans. on Image Processing, 11*(4), 467-472.

Liu, C., & Wechsler, H. (2003). Independent Component Analysis of Gabor Features for Face Recognition. *IEEE Transactions on neural networks, 14*(4), 919-928.

Lyons, M. J., Akamatsu, S., Kamachi, M., & Gyoba, J. (1998, 14-16 April). *Coding Facial Expressions with Gabor Wavelets.* Paper presented at the Third IEEE International Conference on Automatic Face and Gesture Recognition, Nara Japan.

Lyons, M. J., Budynek, J., & Akamatsu, S. (1999). Automatic Classification of Single Facial Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(12), 1357-1362.

Ma, L., & Khorasani, K. (2004). Facial Expression recognition using constructive feedforward neural networks. *IEEE TRansactions on Systems, Man and Cybernetics, Part B, 34*(3), 1588-1595.

Mak, M. W., & Kung, S. Y. (2000). Estimation of elliptical basis function parameters by the EM algorithms with application to speaker verification. *IEEE Trans. Neural Networks, 11*(4), 961-969.

Manjunath, B. S., & Ma, W. Y. (1996). Texture Features for Browsing and Retrieval of Image Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(8), 837-842.

Mase, K. (1991). Recognition of facial expression from optical flow. *IEICE Transactions E, 74*(10), 3474-3483.

Matas, J., Marik, R., & Kittler, J. (1995). *On Representation and Matchinig of Multi-Colored Objects.* Paper presented at the Fifth International Conference on Computer Vision.

Mccallum, A., & Nigam, K. (1998). *A Comparison of Event Models for Naive Bayes Text Classification.* Paper presented at the International Conference on Machine Learning.

Moghaddam, B., & Pentland, A. P. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(7), 696-710.

Ng, S. K., & McLachlan, G. J. (2004). Using the EM algorithm to train neural networks: Misconceptions and a new algorithm for multiclass classification. *IEEE Transactions on Neural Networks, 15*(3), 738-749.

Osuna, E., Freund, R., & Girosi, F. (1997). *An Improved Training Algorithm for Support Vector Machines.* Paper presented at the IEEE. Neural Network for Signal Processing.

Penev, P. S., & Atick, J. J. (1996). Local feature analysis: a general statistical theory for object representation. *Network: Computation in Neural Systems, 7*(3), 477-500.

Perlovsky, L. I. (2001). *Neural networks and Intellect: using model-based concepts* (3rd printing ed.). New York: Oxford University Press.

Perrett, D., & Mistlin, A. (1990). Perception of facial characteristics by monkeys. In W. Stebbins & M. Berkley (Eds.), *Comparative Perception* (Vol. 2, pp. 187-215). New York: John Wiley & Sons.

Phillips, P. J., Moon, H., Rauss, P., & Rizvi, S. A. (1997). *The FERET Evaluation methodology for Face Recognition Algorithms.* Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition.

Platt, J. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Scholkopf, C. Burges & A. Smola (Eds.), *Advances in Kernel Methods - Suppoort Vector Learning* (pp. 185-208): MIT Press.

Pollack, J. B. (1990). Recursive Distributed Representations. *Artificial Intelligence, 46*(1-2), 77-106.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Radha, H., Vetterli, M., & Leonardi, R. (1996). Image compression using binary space partitioning trees. *IEEE Transactions on Image Processing, 5*(2), 1610-1624.

Roberts, S., & Tarassenko, L. (1994). A probabilistic resource allocating network for novelty detection. *Neural Computation, 6*, 270-284.

Rosenblum, M., Yacoob, Y., & Davis, L. (1996). Human expression recognition from motion using a radial basis function network architecture. *IEEE Trans. Neural Networks, 7*(5), 1121-1138.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural Network based Face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 20*, 20-38.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*: MIT Press.

Salembier, P., & Garrido, L. (2000). Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval. *IEEE Trans. on Image Processing, 9*(4), 561-576.

Samaria, F., & Harter, A. C. (1994). *Parameterisation of a Stochastic Model for Human Face Identification.* Paper presented at the Second IEEE Workshop Applications of Computer Vision.

Sebe, N., Lew, M., Cohen, I., Sun, Y., Gevers, T., & Huang, T. (2004). *Authentic facial expression analysis.* Paper presented at the International Conference on Automatic Face and Gesture Recognition, Seoul, Korea.

Singh, S. K., Chauhan, D. S., Vatsa, M., & Singh, R. (2003). A Robust Skin Color Based Face Detection Algorithm. *Tamkang Journal of Science and Engineering, 6*(4), 227-234.

Sirovich, L., & Kirby, M. (1987). Low-Dimensional Procedure for the Characterization of Human Face. *Journal of the Optical Society of America, 4*, 519-524.

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., & R.Jain. (2000). Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. Pattern Anal. Machine Intell., 22*(12), 1349-1380.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for classification of structures. *IEEE Trans. Neural Networks, 8*, 714-735.

Streit, D. F., & Luginhuhl, T. E. (1994). Maximum likelihood training of probabilistic neural networks. *IEEE Trans. on Neural Networks, 5*(5), 764-783.

Tarr, M. J., & Bulthoff, H. H. (1995). Is Human Object Recognition Better Described by Geon Structural Descriptions or by Multiple Views - Comment on Biederman and Gerhardstein (1993). *Journal of Experimental Psychology: Human Perception and Performance, 21*, 71-86.

Taylor, J. G., Fragopanogos, N., R.Cowie, Douglas-Cowie, E., Fotinea, S.-E., & Kollias, S. (2003). An Emotion Recognition Architecture based on human brain structure. In *Lecture Notes in Computer Science* (Vol. 2714, pp. 1133-1142).

Thompson, J. (1941). Development of facial expression of emotion in blind and seeing children. *Archives of Psychology, 37*.

Thompson, P. (1980). Margaret Thatcher - A New Illusion. *Perception, 9*, 483-484.

Tian, Y.-L., Kanade, T., & Cohn, J. F. (2001). Recognizing Action Units for Facial Expression Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 23*(2), 1-18.

Toh, K.-A., Tran, Q.-L., & Srinivasan, D. (2004). Benchmarking a Reduced Multivariate Polynomial Patter Classifier. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 26*(6), 740-755.

Toh, K.-A., Yau, W.-Y., & Jiang, X. (2004). A Reduced Multivariate Polynomials Model for Multi-Modal Biometrics and Classifiers Fusion. *IEEE Trans. Circuits and Systems for Video Technology, 14*(2), 224- 233.

Tsoi, A. C. (1998). *Adaptive Processing of Data Structure : An Expository Overview and Comments*. Australia: Faculty Informatics, Univ. Wollongong, Wollongong.

Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience, 3*, 71-86.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Viola, P., & Jones, M. (2001, July 13, 2001). *Robust Real-time Object Detection.* Paper presented at the Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling, Vancouver, Canada.

Voegtlin, T. (2000). *Context quantization and contextual self-organizing maps.* Paper presented at the International Joint Conference on Neural Networks.

Wiskott, L., Fellous, J.-M., Kruger, N., & Malsburg, C. v. d. (1997). Face Recognition by Elastic Bunch Graph Matching. *IEEE Trans. Pattern Anal. Machine Intell., 19*(7), 775-779.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann, San Francisco.

Wong, J.-J., & Cho, S.-Y. (2007). A Brain-Inspired Model for Recognizing Human Emotional States from Facial Expression. In L. I. Perlovsky & R. Kozma (Eds.), *Neurodynamics of Cognition and Consciousness* (pp. 233-254): Springer.

Woodward, J. J. D., Orlans, N. M., & T.Higgins, P. (2003). *Identity Assurance in the Information Age Biometrics*. Mc Graw Hill: Osborne.

Wu, S., & Chow, T. W. S. (2005). PRSOM: A New Visualization Method by Hybridizing Multidimensional Scaling and Self-Organizing Map. *IEEE Transactions on Neural Networks, 16*(6), 1362-1380.

Wu, T.-F., Lin, C.-J., & Weng, R. C. (2004). Probability Estimates for Multi-Class Classification by Pairwise Coupling. *Journal of Machine Learning Research, 5*, 975-1005.

Wu, Y., Liu, H., & Zha, H. (2005). *Modeling facial expression space for recognition.* Paper presented at the Int. Conference on intelligent Robots and Systems.

Yang, J., & Waibel, A. (1996). *A Real-Time Face Tracker.* Paper presented at the IEEE Workshop on Applications of Computer Vision, Saratosa, FL, USA.

Yang, M.-H., Kriegman, D. J., & Ahuja, N. (2002). Detecting Faces in Images: A Survey. *IEEE Trans. Pattern Anal. Machine Intell., 24*(1), 34-58.

Yin, H. (2002). ViSOM: A novel method for multivariate data projection and structure visualization. *IEEE Transactions on Neural Networks, 13*(1), 237-243.

Yin, L., Wei, X., Shun, Y., Wang, J., & Rosato, M. J. (2006). *A 3D Facial Expression Database for Facial Behavior Research.* Paper presented at the 7th International Conference on Automatic Face and Gesture Recognition.

Young, A. W., Hellawell, D., & Hay, D. C. (1987). Configurational information in face perception. *Perception, 16*, 747-759.

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face Recognition: A Literature Survey. *ACM Computing Surveys, 35*(4), 399-458.

Zheng, W., Zhou, X., Zou, C., & Zhao, C. (2006). Facial Expression recognition using Kernel Canonical Correlation Analysis (KCCA). *IEEE Transactions on Neural Networks, 17*(1), 233-238.

Appendix

A1 ASIAN EMOTION DATABASE

To the best of our knowledge, few investigations have been conducted on analyzing face emotion behaviour among the different races in the Asian population. Most of the publicly available emotion database contains images that are captured from video recording or stored in low-resolution quality. The closest Asian emotion database is the Japanese Female Emotion Database (Lyons *et al.*, 1999), and it contains 213 images of 7 facial expressions (including neutral) posed by 10 Japanese actresses. A 3D Facial expression database (Yin *et al.*, 2006) by Yin *et al*. contains 100 subjects in various emotions from various races found in the America. The development of the database was designed to capture high-resolution 2D facial images for various races, age groups and genders found in the Asian population in seven emotional states and in 3 different poses. As Singapore is in the heart of Asia and has a high mixture of different races in Asia, Nanyang Technological University became the source for data collection. Currently, the database is opened to public to freely be downloaded from:

http://www.ntu.edu.sg/sce/labs/forse/Asian%20Emotion%20Database.htm.

A1.1 Creation of Asian Emotion Database

Figure A.1 – Photograph station setup for the creation of Asian Emotion Database.

A photograph station was set up as shown in Figure A.1 in a public venue, and invited volunteers (both female and male) of all races and age groups from the public to participate in the data collection exercises. The images were captured using a 5 mega-pixels Ricoh R1V digital camera using ISO 64 settings with flash from the camera. Subjects were sitting down at 143cm away against a white background from the camera. The images were cropped and scaled from the original 2560 x 1920 pixels facial to around 900 x 1024 pixels for archival. The images are annotated, processed, and stored each image in 24-bit colour Bitmap format as a ground truth. The filenames of the images were encoded in the following manner:

XXXXX_X_X_XX_X_X_X_XXX.bmp Filename

1 2 3 4 5 6 7 8 Section

Each section of the filename stores the following information in ascending order: Subject ID, age group, gender, race, and consent to publication, emotions, pose, and sample number. Using this filename format, the data is searchable by query. Along with the images, we included the location of the centre of eyes, nose and mouth and stored in an 8-bit integer format file in a corresponding filename with "loc" as filename extension as a ground truth.

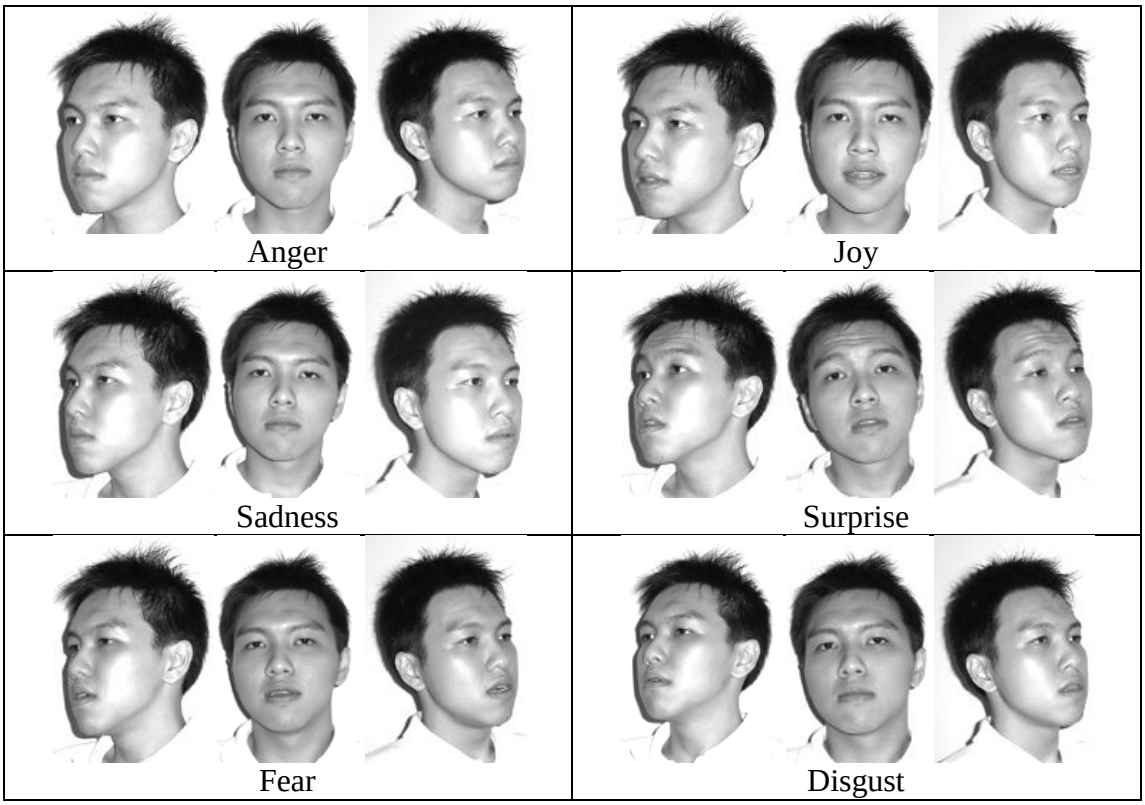


Figure A.2 - Subject in 6 different expressions in 3 different poses

Each subject was instructed to sit in front of the camera. They were requested to perform 7 expressions, i.e. Neutral, Anger, Joy, Sadness, Surprise, Fear, and Disgust. As the digital camera is unable to capture dynamic facial expressions, the subject is required to perform each expression for a short period. Ideally, a video clip could be used for eliciting a genuine emotion state of subjects. However, it is difficult to provide such a setup, especially for emotions such as sadness and fear (Sebe et al., 2004). Cowie *et al.* (Cowie et al., 2001) quotes, displays of intense emotion or "pure"

primary emotions rarely happened. Due to the time constraints and the limitation of setup, the subjects were asked to perform pseudo-emotions. The subjects were also asked to position themselves, 45 degrees to the left and right, so we could capture these expressions for half profile poses as shown in Figure A.2.

A1.2 Statistics of Participants

The Asian Emotion Database contains around 4947 images for 7 different facial expressions and 3 different poses from 153 subjects, who participated in the data collection exercises over a period of 4 days. Out of the 153 subjects, 64 of them have given consent to their images for use in publications. The Asian Emotion Database has facial images from various races, including, Chinese, Malay, Indian, Thai, Vietnamese, Indonesian, Iranian and Caucasian. 72% of the subject belongs to the Chinese race, 7% Indian, 7% Vietnamese, 5% Caucasian, 4% Malay and 4% others. Table A.1 shows the detail distribution of our database according to Race and Gender.

Table A.1 - Distribution of subjects according to race and gender

	Female	Male	Total
Chinese	33	77	110
Malay	5	2	7
Indian	3	8	11
Vietnamese	1	10	11
Caucasian	1	7	8
Others	3	3	6
Total	46	107	153

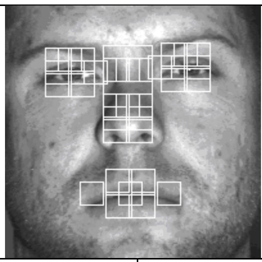
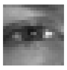
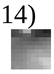
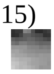
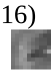
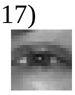
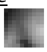
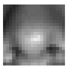
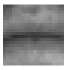
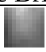
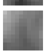
Table A.2 shows the majority of the subjects about 72% were of the 20-29 year old age group, as well as the detail breakdown of the database according to Race and Age group. Due to time constraints of our participants, we were only able to get 27 of them to pose for the 2 half profile poses for all expressions.

Table A.2 - Distribution of subjects according to race and age group

Age (years)	<20	20-29	30-39	40-49	>=50	Total
Chinese	4	80	20	6	0	110
Malay	0	6	0	2	0	7
Indian	1	5	2	2	1	11
Vietnamese	0	11	0	0	0	11
Caucasian	0	4	1	3	0	8
Others	3	3	0	0	0	6
Total	8	109	23	12	1	153

A2 38 Facial Feature Components of HFTS

Table A.3 – 38 Facial features components used in HFTS representation.

	
Left Eye 1.  Left Corner of Left Eye (F01) 2.  Right Corner of Left Eye (F02) 3.  Top Center of Left Eye (F03) 4.  Top Left Corner of Left Eye (F04) 5.  Top Right Corner of Left Eye (F05) 6.  Bottom Left Corner of Left Eye (F06) 7.  Bottom Right Corner of Left Eye (F07) 8.  Extreme Right Corner of Left Eye (F08) 9.  Whole of Left Eye (F09)	Right Eye Chapter 1  Left Corner of Right Eye (F10) Chapter 2  Right Corner of Right Eye (F11) Chapter 3  Top Center of Right Eye (F12) Chapter 4  Top Left Corner of Right Eye (F13) Chapter 5  Top Right Corner of Right Eye (F14) Chapter 6  Bottom Left Corner of Right Eye (F15) Chapter 7  Bottom Right Corner of Right Eye (F16) Chapter 8  Extreme Left Corner of Right Eye (F17) Chapter 9  Whole of Right Eye (F18)
Nose 1.  Left Corner of Nose Tip (F19) 2.  Right Corner of Nose Tip (F20) 3.  Top Center of Nose Tip (F21) 4.  Top Right of Nose Tip (F22) 5.  Top Left of Nose Tip (F23) 6.  Bottom Left of Nose Tip (F24) 7.  Bottom Right of Nose Tip (F25) 8.  Entire Nose Tip (F26)	Lips 1.  Top Left of center of Lips (F27) 2.  Top Right of center of Lips (F28) 3.  Bottom Left of center of Lips (F29) 4.  Bottom Right of center of Lips (F30) 5.  Left Corner of Lips (F31) 6.  Right Corner of Lips (F32) 7.  Center of Lips (F33) 8.  Whole of center of Lips (F34)
Nose Bridge 1.  Center of Nose Bridge (F35) 2.  Left Corner of Nose Bridge (F36) 3.  Right Corner of Nose Bridge (F37) 4.  Whole of Nose Bridge (F38)	

A3 Gabor Filter Response on 6 Basic Emotions

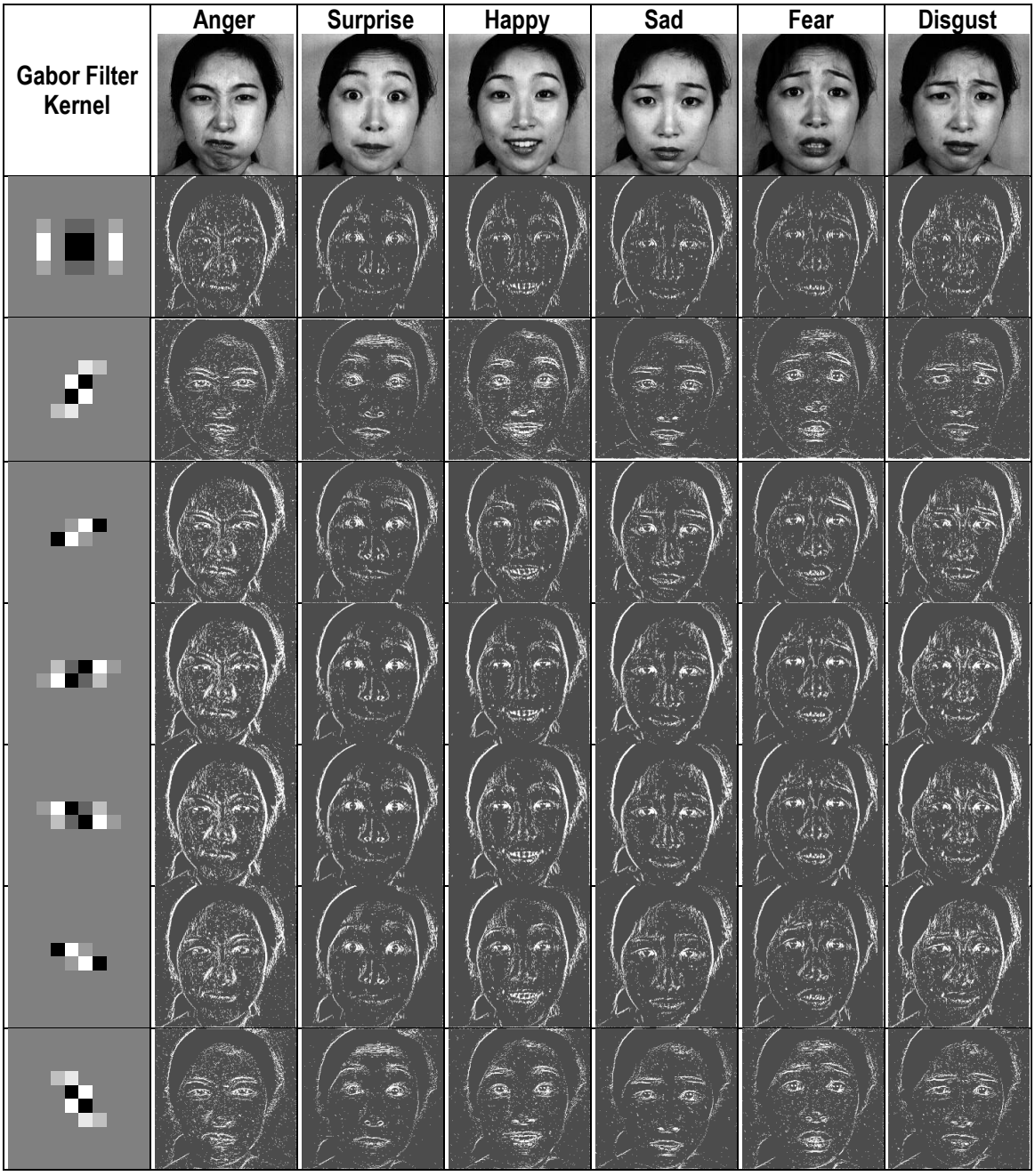


Figure A.3 – Examples of Gabor wavelets and their corresponding convoluted images.

A4 FACS Action Units

Table A.4 – FACS AU Definition.

AU	FACS Name	Muscular Basis
1	Inner Brow Raiser	Frontalis, Pars Medialis
2	Outer Brow Raiser	Frontalis, Pars Lateralis
4	Brow Lowerer	Depressor Glabellae; Depressor Supercilli; Corrugator
5	Upper Lid Raiser	Levator Palpebrae Superioris
6	Cheek Raiser	Orbicularis Oculi, Pars Orbitalis
7	Lid Tightener	Orbicularis Oculi, Pars Palebralis
8	Lips Toward Each Other	Orbicularis Oris
9	Nose Wrinkler	Levator Labii Superioris, Alaeque Nasi
10	Upper Lip Raiser	Levator Labii Superioris, Caput Infraorbitalis
11	Nasolabial Furrow Deepener	Zygomatic Minor
12	Lip Corner Puller	Zygomatic Major
13	Cheek Puffer	Caninus
14	Dimpler	Buccinnator
15	Lip Corner Depressor	Triangularis
16	Lower Lip Depressor	Depressor Labii
17	Chin Raiser	Mentalis
18	Lip Puckerer	Incisivii Labii Superioris; Incisivii Labii Inferioris
20	Lip Stretcher	Risorius
22	Lip Funneler	Orbicularis Oris
23	Lip Tightner	Orbicularis Oris
24	Lip Pressor	Orbicularis Oris
25	Lips Part	Depressor Labii, or Relaxation of Mentalis or Orbicularis Oris
26	Jaw Drop	Masetter; Temporal and Internal Pterygoid Relaxed
27	Mouth Stretch	Pterygoids; Digastric
28	Lip Suck	Orbicularis Oris
38	Nostril Dilator	Nasalis, Pars Alaris
39	Nostril Compressor	Nasalis, Pars Transversa and Depressor Septi Nasi
41	Lid Droop	Relaxation of Levator Palpebrae Superioris
42	Slit	Orbicularis Oculi
43	Eyes Closed	Relaxation of Levator Palpebrae Superioris
44	Squint	Orbicularis Oculi, Pars Palpebralis
45	Blink	Relaxation of Levator Palpebrae and Contraction of Orbicularis Oculi, Pars Palpebralis
46	Wink	Orbicularis Oculi