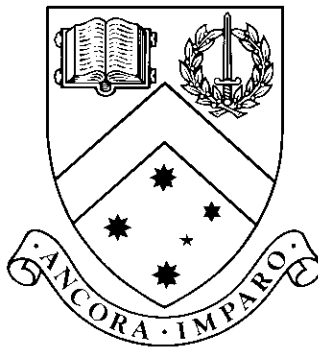


Recognising Patterns in Large Data Sets: A Distributed Approach

by

Anang Hudaya Muhamad Amin,
BTech.(Hons) Information Technology (UTP)
Master of Network Computing (Monash)



Thesis

Submitted by Anang Hudaya Muhamad Amin
for fulfillment of the Requirements for the Degree of
Doctor of Philosophy (0190)

**Clayton School of Information Technology
Monash University**

January, 2011

© Copyright

by

Anang Hudaya Muhamad Amin

2011

For my wife. Thanks for being with me through this journey.

Contents

List of Tables	ix
List of Figures	xii
Abstract	xx
List of Publications	xxiii
Acknowledgments	xxvi
1 Introduction	1
1.1 Pattern Recognition Concept and Theories	4
1.2 Scalability in Pattern Recognition	6
1.3 Distributed Pattern Recognition	8
1.3.1 Top-Down vs. Bottom-Up Approaches	10
1.3.2 Distributed Multi-Feature Recognition	13
1.3.3 Resource-Awareness	14
1.4 Motivation and Aims	14
1.5 Hypothesis and Research Objectives	16
1.6 Research Contributions	18
1.7 Thesis Outline	19
2 Pattern Recognition and Distributed Approach	23
2.1 Neural Network/Machine Learning Approach	25
2.1.1 Scalability and Adaptability of Neural Networks	28
2.1.2 Convolutional Neural Network	33
2.2 Scalability Evaluation for Neural Network Approaches	37

2.2.1	Storage Capacity Analysis	37
2.2.2	Communication Frequency Evaluation	38
2.3	Key Components for Scalable Pattern Recognition	40
2.3.1	Learning Mechanism	40
2.3.2	Processing Approach	41
2.3.3	Training Procedure	43
2.4	Pattern Distribution Techniques	43
2.4.1	Subpattern Distribution Technique	45
2.4.2	Set Distribution Technique	46
2.5	Graph Neuron for Scalable Recognition Scheme	47
2.5.1	Graph-based Pattern Recognition	47
2.5.2	GN Architecture and Pattern Representation	49
2.5.3	GN Complexity Estimation	55
2.5.4	Crosstalk Issue in GN	56
2.6	Hierarchical Graph Neuron (HGN)	59
2.6.1	Size of HGN Network	61
2.6.2	HGN Recognition Procedure	62
2.6.3	HGN Complexity Estimation	64
2.6.4	HGN Solution to Crosstalk Problem	67
2.6.5	Scalability in HGN Approach	69
2.7	Distributed Approach for HGN	72
2.7.1	Distributed Approach Design	74
2.7.2	Non-Uniform Approach	76
2.7.3	Uniform Approach	81
2.8	Conclusions	86
3	Distributed Hierarchical Graph Neuron	89
3.1	DHGN for Distributed Pattern Recognition	90
3.1.1	GN Associative Memory Concept	91
3.1.2	System Architecture	92
3.1.3	Dual-Phase Recognition Procedure	97
3.1.4	Bias Array Design	105

3.1.5	Collaborative-Comparison Learning (CCL)	106
3.2	Dimensionality Reduction in Pattern Pre-Processing	108
3.2.1	Structural Reduction	108
3.2.2	Content Reduction	111
3.2.3	Case Study: DHGN Image Recognition based on Binary Signature	111
3.3	Analysis and Evaluation	119
3.3.1	Complexity Evaluation	119
3.3.2	Scalability Analysis	127
3.4	Pattern Recognition Simulation and Results	137
3.4.1	Binary Character Pattern Recognition	138
3.4.2	Recognition Test on Binary Images	141
3.4.3	Performance Test on Binary Images	142
3.5	Multi-Value DHGN Model	143
3.5.1	Complexity Estimation	147
3.5.2	Recognition Accuracy	152
3.5.3	Summary	154
3.6	Conclusions	155
4	Multi-Feature Pattern Recognition: A Distributed Approach	159
4.1	Data Features for Pattern Recognition	161
4.1.1	Common Approaches to Feature-based Pattern Recognition	162
4.1.2	Pattern Recognition (PR) using Multiple Features	163
4.2	DHGN Multi-Feature Recognition	164
4.2.1	System Architecture	164
4.2.2	Recognition Analysis	165
4.2.3	Complexity Estimation	166
4.3	Greyscale Image Recognition using Multi-Feature Approach	170
4.3.1	Multi-Feature DHGN for Facial Image Recognition	171
4.3.2	Recognition Accuracy Analysis	172
4.3.3	Results and Discussion	174
4.3.4	Concluding Remarks	180
4.4	Handwritten Object Recognition with Multiple Features	181

4.4.1	The Data Set	182
4.4.2	Classification Procedures	182
4.4.3	Results and Discussion	185
4.4.4	Concluding Remarks	188
4.5	Conclusions	189
5	Resource Considerations for Distributed PR	191
5.1	Message-Passing Model for DHGN	193
5.1.1	Process Actions	194
5.1.2	System Synchronisation	197
5.2	Network Granularity Analysis and Evaluation	198
5.2.1	DHGN Configurations for Adaptive Granularity	199
5.2.2	Performance Analysis	202
5.2.3	Clustered DHGN for Coarse-Grained Networks	207
5.2.4	Fully-Distributed DHGN for Fine-Grained Networks	212
5.2.5	Summary	219
5.3	DHGN Fault Tolerance Mechanism	221
5.3.1	Subnet Fault	222
5.3.2	Node Fault	224
5.4	Conclusions	225
6	Distributed PR Applications within Fine-Grained Networks	227
6.1	WSN Event Detection	229
6.1.1	WSN Deployment Issues for Event Detection	231
6.1.2	Summary	234
6.2	Integrated DHGN-WSN Scheme	235
6.2.1	Dimensionality Reduction in Sensory Data	237
6.2.2	DHGN Event Classification	238
6.2.3	Performance Metrics: Memory Utilisation	240
6.3	Case Study: Forest Fire Detection using Integrated DHGN-WSN	241
6.3.1	Existing Approaches	241
6.3.2	Dimensionality Reduction on FPMC Values	242
6.3.3	Methodology	243

6.3.4	Classification/Recognition Results	244
6.3.5	Spatio-Temporal Analysis of Event Data	246
6.3.6	Summary	247
6.4	Conclusions	249
7	Conclusions	251
7.1	Summary	251
7.2	Research Contributions	253
7.3	Future Research	257
7.3.1	Algorithm-Specific Research	257
7.3.2	Application-Specific Research	258
	References	261
	Appendix A Extended Analysis and Results	279
A.1	Greyscale Image Recognition	279
A.1.1	The Dataset	279
A.1.2	Extended Results	279
A.2	Negative Image Detection using DHGN	282
A.2.1	Subpattern Size Reduction	282
A.2.2	Negative Image Recognition	284
A.2.3	Recognition Tests and Results	284
A.2.4	Discussions	286
	Appendix B Algorithms and Pseudocodes	297
B.1	Single-Value (Original) DHGN	297
B.2	Multi-Value DHGN	300

List of Tables

2.1	Store and recall responses of a GN Array.	53
2.2	Bias array entries for all active GN in HGN composition when pattern $wwwz$ is introduced.	68
2.3	Bias array entries for all active GN in HGN composition when pattern $zwwy$ is introduced.	68
2.4	Bias array entries for all active GN in HGN composition when pattern $wwxy$ is introduced.	69
2.5	Character representations of 35-bit patterns using a horizontal scanning approach.	77
3.1	Recalled indices retrieved from all DHGN subnets after each pattern input.	104
3.2	Colour composition values correspond to each bin b_j	113
3.3	Detailed binary signatures for image in Figure 3.10.	114
3.4	Detailed binary signatures of block image in Figure 3.12.	115
3.5	Big-O notations for Hopfield network and DHGN implementation in network generation stage.	120
3.6	Big-O notations for Hopfield network in recognition stage.	121
3.7	Big-O notation for DHGN implementation in recognition stage.	122
3.8	DHGN computational complexity terms.	128
3.9	Total possible bias entries for each layer within a HGN network for 55-bits binary patterns.	134
3.10	Comparison between the total possible maximum bias array size for HGN and DHGN implementations for binary pattern recognition with different pattern sizes.	135

3.11	Comparison between HGN and DHGN implementations with regards to the number of messages communicated per pattern recognition.	136
3.12	Average recognition time for each subnet in DHGN network for a given 16kb binary image.	143
3.13	Representations for the amount for each different types of communication performed in MV-DHGN message-passing model.	145
3.14	MV-DHGN bias array estimation terms.	153
3.15	Dataset collection comprising of random binary subpatterns.	153
4.1	Example of data obtained from SI modules, in the form of error values for each feature.	168
4.2	BPNN execution parameters.	177
4.3	Discretisation on feature data values using variable-binning methods.	183
4.4	Sample of Zernike moment's feature patterns obtained using discretisation.	184
4.5	Multi-feature DHGN architecture parameter details.	184
4.6	Recognition accuracy parameters and their respective representations.	185
4.7	Results of the classification decision on 4 different features of numeral character objects.	186
4.8	Comparative analysis on error values between DHGN and other classifiers for similar data set with respective features.	187
5.1	Representations for different process actions in DHGN message-passing model.	195
5.2	Comparison between fine-grained and coarse-grained networks.	199
5.3	DHGN subnet associative array structure after subpatterns 00001 and 11111 have been memorised.	202
5.4	Threshold classes with respective value range used in the tests.	216
5.5	Comparative analysis on recognition accuracy parameters between DHGN and other classifiers for event recognition using sensory data obtained from three wireless sensors (Smart-It 1, Smart-It 2, and Smart-It 5).	218
6.1	Berkeley Mica Mote sensor node specifications.	231
6.2	Example of a simple temperature readings with respective binary signature.	238
6.3	Ignition potential versus FFMC value.	242
6.4	Modified FFMC classification for DHGN event detection scheme.	242

6.5	Sensory data with allocated binary signature bits.	244
6.6	Temperature threshold ranges with respective binary signatures.	245
6.7	Training data set in the form of specific threshold ranges used in classification test. Binary digits in brackets represent signature for the respective data range.	245
6.8	Comparison on classification accuracy between DHGN and Kohonen SOM classifiers for forest fire detection. Different numbers of training data were used for each SOM implementation.	245
A.1	Results of image recognition test using DHGN recognition scheme with 9-bits input subpatterns.	285
A.2	Results of image recognition test using DHGN recognition scheme with 7-bits input subpatterns. There is an improvement on the accuracy of this DHGN scheme.	285
A.3	Results of image recognition test using a recognition scheme with 5-bits input subpatterns. DHGN is able to recall all the characters using minimum voting scheme on both dataset II and III.	286

List of Figures

1.1	Conceptual representation of conventional pattern recognition scheme.	5
1.2	Distributed pattern recognition with process farming approach.	11
1.3	Distributed pattern recognition with process pipelining approach.	12
2.1	Generic neural network architecture for pattern recognition with two layers of hidden neurons.	25
2.2	The general topology of a Hopfield Network.	30
2.3	General topology of Kohonen network.	32
2.4	SVM classification using optimal hyperplane for separable classes. The SVs lies on the dotted lines.	34
2.5	Estimated number of signals/messages generated, C by each neuron within a single layer in common neural network schemes involving different number of iterations.	39
2.6	Comparison of processing speed-up estimation between recognition pro- cesses on different parallel fractions (P) with an increase in the number of parallel processors used.	42
2.7	A comparison between subpattern and pattern subset distribution techniques.	44
2.8	A labelled graph with a vertex set $V = \{1, 2, 3, 4, 5, 6, 7\}$ and edge set $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 7\}, \{2, 6\}, \{5, 7\}, \{1, 7\}\}$	48
2.9	A two-dimensional GN network for binary pattern with 5-bit size.	49
2.10	GN network activation from input pattern “ <i>ABBAB</i> ”.	50
2.11	GN recognition process with bias array illustration for different input patterns.	52
2.12	Maximum bias array analysis for GN implementation on increasing different number of pattern elements and pattern size.	57
2.13	Crosstalk phenomenon illustration on input patterns in GN network.	58

2.14	Crosstalk phenomenon in GN pattern recognition.	58
2.15	Hierarchical Graph Neuron (HGN) with binary pattern of size 7 bits.	59
2.16	HGN composition of 2- and 3-dimension for pattern size 49 and 147.	60
2.17	Analysis on the effects of increasing pattern size over total maximum bias array size within HGN for 1- and 2-dimensional compositions.	62
2.18	HGN composition for crosstalk example (Section 2.3.4).	67
2.19	Growth rate of GNs in HGN composition with increasing number of different pattern elements and pattern size.	70
2.20	HGN decomposition into a number hosts within a physical network.	71
2.21	HGN Decomposition into distributed HGN sub-networks. The HGN net- work is decomposed into three HGN subnets.	73
2.22	Comparison between HGN and distributed HGN for increasing size of pat- terns. 7-bit pattern segment is used for each HGN subnet in distributed HGN scheme.	74
2.23	Test character representations in 7-by-5 1-bit format.	75
2.24	Character bitmap image representations.	75
2.25	Non-uniform distributed HGN approach with 7-21-7 compositions for 35- element patterns with two possible values.	76
2.26	The HGN subnets successfully store the bitmap pattern for character ‘I’ at index value of ‘2’ after having stored the bit map pattern for character ‘A’ at index value of ‘1’.	78
2.27	Results for introducing 1-bit distortion pattern of character ‘A’. The first HGN subnet shows that a new subpattern has been found (with assigned index 0) while other compositions correctly recall this as the pattern asso- ciated with index 1 (bitmap pattern of ‘A’).	78
2.28	Character set used in distributed HGN simulation on pattern recognition.	79
2.29	Comparison on recall accuracy between non-uniform distributed HGN and HGN pattern recognition schemes on different distortion levels applied to character set <i>A, I, J, S, X, and Z</i>	80
2.30	A one-bit distortion occurring within the overall input pattern ‘A’ stays encapsulated within the left composition.	81
2.31	Uniform distributed model composition for analysing 35-bit binary patterns.	82

2.32	Comparison on recall accuracy between uniform distributed HGN and HGN pattern recognition schemes on different distortion levels applied to character set <i>A, I, J, S, X, and Z</i>	83
2.33	Encapsulation effect within uniform model when processing a pattern of character ‘ <i>A</i> ’ with 2-bit distortion. The effects of the distortions are localised within the two compositions on the left and do not influence the findings of the remaining compositions.	84
2.34	Comparison on recall accuracy between uniform and non-uniform distributed HGN pattern recognition schemes on different distortion levels applied to character set <i>A, I, J, S, X, and Z</i>	85
2.35	The effects of a 1-bit distortion in the pattern get localised within the uniform distributed compositions (lower) whereas the effects of the distorted pattern are propagated along the right side of the entire HGN composition, leading to a false conclusion.	86
3.1	DHGN framework for distributed pattern recognition.	93
3.2	Analogical representation of DHGN distributed pattern recognition scheme.	98
3.3	DHGN pattern recognition process workflow. This diagram represents DHGN network with 3-layer subnets.	100
3.4	Samples of binary character images.	103
3.5	GN node abstract representation showing its storage framework.	106
3.6	Collaborative-comparison learning approach for one-dimensional pattern “ABCDE”. Each activated graph neuron (GN) stores the signals received by its adjacent neurons.	107
3.7	Structural reduction on binary character images into one-dimensional bit-string representation.	109
3.8	Character patterns with structural and random distortions.	110
3.9	Results of DHGN pattern recognition on structural and random distorted character patterns.	110
3.10	Block image with four different colours.	113
3.11	DHGN implementation for colour recognition using binary signature.	114
3.12	Block image is divided into grids with equivalent sizes.	115

3.13	Different levels of noise and rotational distortion for greyscale image of Lena with respective colour histograms, used in the image recognition test.	116
3.14	Total recall and error rates for DHGN greyscale image recognition on 40 images using colour feature analysis.	117
3.15	Transformation of global colour histogram of image Baboon from original image to 3-quantisation level image.	118
3.16	Comparison between SVM and DHGN total recall rates for greyscale image recognition.	119
3.17	Big-O notation comparisons for processes within Hopfield network recognition stage.	122
3.18	Complexity measurement of SOM's weight initialisation process.	123
3.19	Complexity measurement of SOM's BMU calculation process.	123
3.20	Complexity measurement of DHGN's network generation process.	124
3.21	Complexity measurement of DHGN's classification process.	125
3.22	The charts showing the total maximum possible bias entries for each layer and within individual nodes for 55-bit binary patterns using the HGN approach.	135
3.23	Comparison between the total cumulative bias entries in DHGN and HGN implementations. The subnet for DHGN is for handling subpattern with size 5 in this comparison.	136
3.24	A comparison of communication costs, for each of the GN node, within HGN and DHGN.	137
3.25	Original binary character patterns used in the recognition tests.	138
3.26	Bit representation for binary character patterns used in the recognition test.	138
3.27	Comparison among different character representation for DHGN pattern recognition.	139
3.28	Eight different levels of random distortion applied to binary character patterns.	140
3.29	Results for binary character pattern recognition with DHGN.	141
3.30	Comparison between One- and Two-Level DHGN pattern recognition on binary characters.	142
3.31	Heterogeneous binary images used in the image recognition test.	143

3.32	Four different levels of Gaussian noise added to image Lena.	144
3.33	Results of the image recognition test with ten binary images stored.	145
3.34	Results of the image recognition test with twenty binary images stored.	146
3.35	Total recognition time for each DHGN subnet in binary pattern recognition with different number of subpatterns derived from 16KB binary images.	147
3.36	Multi-Value DHGN for binary pattern recognition on 5-bit patterns.	148
3.37	Comparison between DHGN (with binary values) and MV-DHGN distributed pattern recognition models on cost of communications.	149
3.38	DHGN compositions with DHGN and MV-DHGN subnets.	150
3.39	Processing capacity estimation for both DHGN and MV-DHGN implemen- tations.	151
3.40	Comparative processing capacity estimation for DHGN and MV-DHGN on the impact of increasing number of different pattern elements.	152
3.41	Estimated cumulative bias array at all levels within a subnet for MV-DHGN and DHGN implementations on binary 55-bits subpatterns.	154
3.42	Recognition test results obtained from the simulation using binary character patterns A, I, J, S, X, and Z.	155
3.43	Percentage of majority votes for each character for different sets of distortion rates.	156
3.44	Recognition time per subpattern for different subpattern size and number of random subpatterns.	157
4.1	Data feature representation for a set of images.	162
4.2	DHGN multi-feature recognition scheme that is made up of a collection of DHGN networks for analysing patterns using multiple sets of features.	165
4.3	Estimated execution time for minimum voting function within coordinator node for 10000 pattern classes with increasing number of features.	169
4.4	Edge map generation using Sobel's edge detection technique when applied to an original greyscale facial image.	172
4.5	Fifty different individuals in the face image dataset obtained from the Face Recognition Data.	173

4.6	Error values obtained for 50 facial image classes from original DHGN implementation using only colour feature on 1000 test images.	175
4.7	Analysis on error values obtained from recognition simulation on greyscale value feature using DHGN and BPNN.	176
4.8	Comparison on error values between greyscale value and edge feature on 50 facial images obtained using DHGN multi-feature scheme.	178
4.9	Analysis on error values obtained from recognition simulation on greyscale and edge features using Multi-Feature DHGN and BPNN.	179
4.10	Total execution time for each subnet in DHGN network for edge recognition process.	180
4.11	Store/recall time for each subpattern within each DHGN subnet for edge recognition process.	181
5.1	DHGN process actions within a message-passing model for binary pattern recognition on 5-bit patterns.	196
5.2	State chart for DHGN implementation in synchronous and asynchronous communications.	198
5.3	Fully-distributed DHGN configuration for fine-grained network.	200
5.4	Clustered DHGN Configuration for coarse-grained network where each DHGN node is capable of performing the entire subpattern recognition processes.	201
5.5	Effects of increasing binary pattern length on the total execution time for clustered DHGN implementation on HPC machines.	203
5.6	Average recall time for each subpattern in clustered DHGN implementation on HPC machines.	204
5.7	Recall time for a single binary subpattern in DHGN fully-distributed configuration implementation over different subpattern sizes and quantities.	205
5.8	Average store/recall time for DHGN implementation on binary subpattern with different length and quantities.	206
5.9	Effects on the average store/recall time for DHGN implementation with increasing binary subpattern length.	207
5.10	Proposed commodity grid-based distributed pattern recognition framework.	209
5.11	DPR-Commodity Grid workflow.	210

5.12	Framework for commodity-grid based pattern recognition.	211
5.13	The Karajan grid engine architecture.	212
5.14	DHGN distributed event detection framework.	214
5.15	Process workflow for the proposed event detection scheme using fully-distributed DHGN algorithm	215
5.16	DHGN event detection results for test using 1690 sensor datasets (x-axis). Note that Smart-It 3 and Smart-It 4 produce non-events since noise and light exposure readings are well below the threshold values (<i>T-Light</i> and <i>T-Noise</i>).	217
5.17	Recognition time for each sensor data in 1690 sensor datasets (x-axis) for all Smart-It nodes using DHGN distributed pattern recognition scheme. . .	220
5.18	Subpattern-division method for task redelegation in DHGN fault manage- ment scheme.	223
5.19	Size and subpattern length of subnet using subpattern-division method on DHGN network with increasing subnet faults.	224
6.1	A generic wireless sensor node architecture.	230
6.2	Sensor nodes placement within a Cartesian grid. Each node is allocated to a specific grid area.	236
6.3	A process workflow for DHGN distributed event detection within WSN. . .	239
6.4	Maximum memory consumption for each DHGN subnet for different pattern sizes. DHGN uses minimum memory space with small pattern size.	241
6.5	Analysis on learning iteration between Kohonen SOM and DHGN for dif- ferent number of classes used in training.	246
6.6	Analysis of event data triggered by the sensor nodes and received by the base station.	247
6.7	Process workflow for the proposed spatio-temporal event detection scheme using DHGN.	248
A.1	40 heterogeneous greyscale images used in DHGN image recognition test. These images belongs to 5 different classes, namely Babboon, Lena, Gold- hill, Peppers, and Camera.	280

A.2	Image histograms for 40 greyscale images used in DHGN image recognition test. Each image comprises 256-greyscale levels.	281
A.3	Error values obtained for each quantisation level from recognition test on 5 classes of greyscale images with noise distortion and rotation.	288
A.4	Recall values obtained for each quantisation level from recognition test on 5 classes of greyscale images with noise distortion and rotation.	289
A.5	Binary character image 'E' with its negative.	289
A.6	Recognition results using DHGN image recognition scheme. The similarities represent the minimum voting obtained from all subnets. The grey areas within the test patterns show the pixel value of test pattern that is similar to stored pattern.	290
A.7	Extended recognition results showing the effect of input subpattern size to the minimum voting scheme.	291
A.8	Datasets used in DHGN image recognition tests.	292
A.9	Character-to-maximum vote ratio for characters A, E, F, L, O, and Q in Dataset I retrieved using DHGN-9 recognition scheme. Note that the star represent column for respective test character.	293
A.10	Character-to-maximum vote ratio for characters A, E, F, L, O, and Q in Dataset III retrieved using different DHGN input subpattern size. Note that the star represents column for respective test character.	294
A.11	Comparison between percentage of recall and number of subnets against the input subpattern size in DHGN pattern recognition with minimum voting scheme.	295
B.1	Context diagram showing important functions within DHGN implementation.	297

Abstract

Advancements in computer architecture, high speed networks, and sensor/data capture technologies have the potential to generate vast amounts of information and bring in new forms of data processing. Unlike the early computations that worked with small chunks of data, contemporary computing infrastructure is able to generate and store large - petabytes - of data for day-to-day operations. These data may arise from high-dimensional images used in medical diagnosis to millions of multi-sensor data collected for the detection of natural events, these large-scale and complex data are increasingly becoming a common phenomenon. This poses a question of whether our ability to recognise and process these data, matches our ability to generate them. This question will be addressed, by looking at the capability of existing recognition schemes to scale up with this outgrowth of data. A different perspective is needed to meet the challenges posed by the so called data deluge. So this thesis take a view which is somewhat outside the conventional approaches, such as statistical computations and deterministic learning schemes, this research considers the bringing together strengths of high performance and parallel computing to artificial intelligence and machine learning and thus proposes a distributed processing approach for scalable pattern recognition.

The research has identified two important issues related to scalability in pattern recognition. These are complexity of learning algorithm and dependency on single processing (CPU-centric) scheme. Scalability in regards to pattern recognition, can be defined as the growth in the capability of pattern recognition algorithms to process large-scale data sets rapidly and with an acceptable level of accuracy. To scale up the recognition process, a pattern recognition system should acquire simple learning mechanisms and the ability to parallelise and distribute its processes for analysis of increasingly large and complex patterns.

This thesis describes a new form of pattern recognition by enabling recognition procedure to be synthesised into a large number of loosely-coupled processes, using a fast single-cycle learning associative memory algorithm. This algorithm implements a divide-and-distribute approach on patterns, hence reducing the processing load capacity per compute node. By using this algorithm, patterns arising from diverse sources e.g. high resolution images and sensor readings may be distributed across parallel computational networks for recognition purposes using a generic framework. Furthermore, the approach enables the recognition process to be scaled up for increasing size and dimension of patterns, given sufficient processing capacity available in hand. Apart from this, a single-cycle learning mechanism being applied in this scheme allows recognition to be performed in a fast and responsive manner, without affecting the level of accuracy of the recogniser. The learning mechanism enables memorisation of a pattern within a single pass, therefore, adding more patterns to the scheme does not affect its performance and accuracy. A series of tests have been performed on recognition accuracy and computational complexity using different types of patterns ranging from facial images to sensor readings. This was done to study the accuracy and scalability of the distributed pattern recognition scheme. The results of these analyses have indicated that the proposed scheme is highly scalable, enables fast/online learning, and is able to achieve accuracy that is comparable to well known machine learning techniques.

After addressing the scalability and performance aspects, this thesis deals with pattern complexity by including pattern recognition applications with multiple features. With the recognition process implemented in a distributed manner, the capacity for allowing more features to be added is possible. The proposed multi-feature approach provides an effective scheme that is capable to accommodate multiple pattern features within the analysis process. This is essential in data mining applications that involve complex data, such as biomedical images containing numerous features. The distributed multi-feature approach using single-cycle learning algorithm demonstrates high recall accuracy in the recognition simulations involving complex images.

Finally, this thesis investigates the scheme's adaptability to different levels of network granularity and discovers important factors for the scalability of the pattern recognition scheme. This allows the recognition scheme to be deployed in different network conditions, ranging from coarse-grained networks such as computational grids, to fine-grained systems,

including wireless sensor networks (WSNs). By acquiring resource-awareness, the proposed distributed pattern recogniser can be deployed in different kinds of applications on different network platforms, creating a generic scheme for pattern recognition. Further analysis on adaptive network granularity feature of distributed single-cycle learning pattern recognition scheme was conducted as a case study to examine the effectiveness and efficiency of the proposed approach for distributed event detection within fine-grained WSN networks. The outcomes of the study indicate that the distributed pattern recognition approach is well-suited for performing event detection using the divide-and-distribute approach with the in-network parallel processing mechanism within a resource-constrained environment. Furthermore, the ability to perform recognition using a simple learning mechanism, enables each sensor node to perform complex applications such as event detection. As a result, this research may give a new insight for applications involving large-scale event detection including forest-fire detection and structural health monitoring (SHM) for mega-structures.

List of Publications

Publications arising from this thesis include:

Book Chapters

- **Khan, A.I. and Muhamad Amin, A.H. and Raja Mahmood, R.A. (2010)**, An On-line Scheme for Threat Detection Within Mobile Ad Hoc Networks. In *Mobile Intelligence: Mobile Computing and Computational Intelligence*. L.T. Yang, et al., Eds. 2010, John Wiley & Sons.
- **Khan, A.I. and Muhamad Amin, A.H. (2009)**, Integrating Sensory Data within a Structural Analysis Grid. In *Parallel, Distributed and Grid Computing for Engineering*. B.H.V. Topping and P. Iványi, Eds. 2009, Saxe-Coburg Publications.
- **Khan, A.I. and Muhamad Amin, A.H. and Raja Mahmood, R.A. (In Print)**, Lightweight Event Detection Scheme using Distributed Hierarchical Graph Neuron in Wireless Sensor Networks. in *Wireless Sensor Network*. In-Tech Publications.

Journals

- **Raja Mahmood, R.A. and Muhamad Amin, A.H. and Khan, A.I. (2008)**, A Lightweight, Fast and Efficient Distributed Hierarchical Graph Neuron-based Pattern Classifier. In *International Journal of Intelligent Engineering and Systems*. vol. 1, no. 4, pp. 9-17, December 2008.

Conference Proceedings

- **Muhamad Amin, A.H. and Khan, A.I. (In Print)**, A Divide-and-Distribute Approach to Single-Cycle Learning HGN Network for Pattern Recognition. in *The 11th International Conference on Control, Automation, Robotics and Vision, ICARCV 2010*. 7 December 2010 to 11 December 2010, Singapore.
- **Muhamad Amin, A.H. and Khan, A.I and Raja Mahmood, R.A. (2009)**, A distributed event detection scheme for wireless sensor networks. in *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2009)*. 14 December 2009 to 16 December 2009, ACM Press, New York NY USA, pp. 295-299.
- **Muhamad Amin, A.H. and Khan, A.I. (2009)**, Collaborative-comparison learning for complex event detection using Distributed Hierarchical Graph Neuron (DHGN) approach in Wireless Sensor Network. in *Proceedings of the 22nd Australasian Joint Conference AI 2009: Advances in Artificial Intelligence*. 1 December 2009 to 4 December 2009, Springer-Verlag, Berlin Germany, pp. 111-120.
- **Muhamad Amin, A.H. and Khan, A.I. (2008)**, Parallel Pattern Recognition Using a Single-Cycle Learning Approach within Wireless Sensor Networks. in *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*. pp.305-308, 1-4 Dec. 2008.
- **Muhamad Amin, A.H. and Khan, A.I. (2008)**, Commodity-Grid Based Distributed Pattern Recognition Framework. in *Proceedings of the Sixth Australasian Workshop on Grid Computing and E-Research - Volume 82*. (Wollongong, NSW, Australia, January 01 - 01, 2008). W. Kelly and P. Roe, Eds. Conferences in Research and Practice in Information Technology Series, vol. 333. Australian Computer Society, Darlinghurst, Australia, 27-34.
- **Khan, A.I. and Muhamad Amin, A.H. (2007)**, One Shot Associative Memory Method for Distorted Pattern Recognition. in *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia*. December 2-6, 2007, Proceedings, vol. 4830, M. A. Orgun and J. Thornton, Eds.: Springer, 2007, pp. 705-709.

Recognising Patterns in Large Data Sets: A Distributed Approach

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Anang Hudaya Muhamad Amin
January 31, 2011

Acknowledgments

First and foremost, my humble thanks to God, who is the most Beneficent and the most Merciful for the endless help for me to complete this thesis.

PhD. is a journey, and along with it there are people who have been continuously giving me support, advice and help to complete this journey. I would like to thank Dr. Asad I. Khan and Prof. Bala Srinivasan for their relentless help and advice in completing my study. I would like to express my gratitude towards them for their encouragement and assistance in completing my thesis.

Special thanks and deep appreciation to Dr. Asad I. Khan, for all the advice and support throughout the duration of my study. You have been an inspiration and guidance to me, through all the easy and difficult times. Your share of expertise and experience are highly valuable and acknowledged.

My deep gratitude to my family for being there with me throughout this journey. To Mak, Abang Hasuna, Kak Shah, Kak Jue, Abang Fazlur, Kak Andong, Abang Nazeri and all the kids, you guys have been a good company. My neverending thanks and love should be conveyed to Shahrul Badariah, my sayang, for her continuous love and support and always there for me through easy and hard times.

Finally, I would like to thank everyone who helped to make this possible. My friends and colleagues. It has been an incredible journey of self-discovery.

To Arwah Abah & Arwah Papa. May you two be given a place in Jannah. Thanks for everything. Al-Fatihah.

Anang Hudaya Muhamad Amin

Monash University

January 2011

Chapter 1

Introduction

The recent development of computing technology has brought forward the ability of generating huge volumes of highly-complex data. This is consistent with Moore's Law of exponential increase in computing power and solid-state memory (Moore, 2000), in which it stated that "[T]he complexity for minimum component costs has increased at a rate of roughly a factor of two per year... Certainly over the short term this rate can be expected to continue, if not to increase" [pp. 57]. Even though this was initially referred to the transistor counts within a processor, the effect of this law seems to be applicable in almost all area of computing, including data generation and analysis. In essence, our existing capability to generate data seems to outstrip our capability to analyse it.

The outgrowth of data, commonly known as data deluge, has significant implications about the existing developments of computing applications (Hey and Trefethen, 2003). According to Anderson (2008), the chief editor of *Wired* magazine,

“[S]ixty years ago, digital computers made information readable. Twenty years ago, the Internet made it reachable. Ten years ago, the first search engine crawlers made it a single database. Now Google and like-minded companies are sifting through the most measured age in history, treating this massive corpus as a laboratory of the human condition”.

In this research, two key contribution factors for this large-scale data generation have been identified. They are data storage technologies and the sophisticated approaches in data capture and sensors.

The development of data storage technology provides the ability to store large volumes of data. This increases the possibility of large-scale data generation. For instance, the development of aggregated storage framework introduced by Vazhkudai, Ma, Freeh, Strickland, Tammineedi and Scott (2005) has shown an initiative to enable the storage of large volume of data using an existing distributed computing architecture. In addition, Beck, Dongarra and Plank (2005) have proposed the grid solution using Internet Backplane Protocol (IBP) for large volume resource data storage.

The development of powerful data-capture instruments and sensors has also led a massive production of large-scale and highly-complex data. This is usually the case in scientific applications such as biomedical image recognition and satellite imaging technology. As mentioned in the work of Fox, Aktas, Aydin, Donnellan, Gadgil, Granat, Pallickara, Parker, Pierce, Oh, Rundle, Sayar and Scharber (2005), the development of sophisticated data-capture instruments and sensors, such as the Large Hadron Collider and Interferometric Synthetic Aperture Radar (InSAR) in high energy physics, has led to the generation of large-volume and multi-dimensional data in that field. In addition, Lewis, Hall, Hufton, Evans, Menk, Arfelli, Rigon, Tromba, Dance, Ellis, Evans, Jacobs, Pinder and Rogers (2003) have demonstrated the medical image creation using X-ray techniques that produce high-resolution and high-dimensional images.

These state-of-the-art data capture and storage technology are the key factors that have driven existing computing technology towards generation of highly-complex and large-scale data. However, it is impractical for data analysts to manually analyse and explore this data without the assistance of highly-sophisticated computational tools. The capabilities of existing applications for data mining and analysis so far have not achieved their fullest potential. This is mainly due to the algorithmic complexity of existing data mining applications. For instance, a complexity of a decision tree classification tool can range from $O(n \log n)$ to $O(n^2)$ or worse, depending on the type of pruning applied (Kamath and Musick, 2000). These kinds of algorithms are computationally expensive and infeasible for large data sets.

With the advent of distributed computing, distributed data storage and processing capabilities have also contributed to the development of cloud computing as a new paradigm. Cloud computing can be defined as a pay-per-use paradigm for providing services over the

Internet in a scalable manner. This new trend of computing can potentially make it possible to achieve capability of conducting such large-scale data processing. This kind of processing is prevalent, as organisations are moving towards cloud computing for their day-to-day operations. Nevertheless, existing data management and processing schemes are incapable of providing an efficient mechanism for deployment within cloud. Some of the concerns include inadequate capability to parallelise data workload, security concerns as a result of storing data at an untrusted host, and weak data replication functionality, as described by Abadi (2009).

In order to obtain useful information from data, it is important for applications to extract features or patterns. This pattern extraction from data is commonly known as data mining. It involves the process of uncovering patterns, associations, anomalies, and significant structures and events in data. One of the key areas in data mining is pattern recognition.

Pattern recognition is a common analysis tool for a wide range of applications including business decision-making, medical diagnosis, and scientific exploration. It provides an avenue for valuable information to be retrieved from raw data obtained through simulation, experimentation, or diagnosis. With the extent of existing data collection technologies, gap between data collection and data analysis capabilities is widening rapidly.

This thesis describes fundamental research on scalability for pattern recognition. Scalability in the context of pattern recognition can be defined as the ability to either handle growing amounts of patterns in a graceful manner or to be readily enlarged. This definition has been derived from the common definition of scalability as described by Bondi (2000). The scalability issues of the existing pattern recognition schemes for large-scale data deployment will be addressed. A number of different approaches will be extensively reviewed and a solution for the scalability problem will be proposed. The main contribution of this thesis lies in the knowledge, design, and implementation of scalable approach for pattern recognition involving complex and large-scale patterns.

The rest of the chapter will introduce a brief background literature as well as the motivation, aims, and hypothesis of this thesis. In addition, this chapter will also describe objectives and contributions of this research. The composition of this chapter is as follows: Section 1.1 provides a brief introduction on pattern recognition concept and theories. Section 1.2 presents a discussion on scalability consideration in pattern recognition. In

this section, issues related to scalability of existing pattern recognition schemes will be examined. In Section 1.3, distributed approach for pattern recognition as a solution for scalability problem in pattern recognition will be observed. The motivation and aims of this research will be presented in Section 1.4. Section 1.5. provides a description on the hypothesis and objectives of this research, while Section 1.6. lists all contributions made from this research. Finally, Section 1.7 gives an outline of the thesis.

1.1 Pattern Recognition Concept and Theories

Pattern is a quantitative or structural description of an object or some other entity of interest (Bow, 2002). The study of pattern recognition can be traced back as early as the 1950s, when digital computers were just starting to be used for information processing. Pattern recognition as being defined by Schalkoff (1991), is the science that concerns the description or classification of measurements. In other words, pattern recognition is the study of theories, methods, and techniques for description and classification of measurements. In the context of this thesis, pattern recognition could be defined as a system involving description and classification of measurements. The measurements could be of different types, including images, sensory readings, and data obtained from experimentation. Pattern recognition may also be categorised into three categories: information reduction, information mapping, and information labelling. Information reduction involves the reduction of information from huge sets of data into clusters or groups of information, while information mapping and information labelling involve categorisation of data based on specific labels or criteria. The data categorisation could be achieved through two different approaches as being described by Jain, Duin and Mao (2000). These approaches are supervised and unsupervised classification/recognition. In supervised classification, the pattern recognition involves classification of pattern into predefined classes, whereas in unsupervised classification, the pattern is assigned to a hitherto unknown class.

Pattern recognition task may be defined as a composition of three major components: data acquisition, data pre-processing, and decision classification as shown in Figure 1.1. Data acquisition phase involves the process of gathering data from surroundings or from an experiment that is being conducted. For instance, an analog signal retrieved from a transceiver which is being transformed into its digital format via a transducer for computer

processing. The output of this stage is a set of measured data as shown in the figure as x_{phy} . These data will then be used as an input to the data pre-processing stage.

Data pre-processing is a stage for data refinement and extraction for specific usage within the pattern recognition application. Some examples of pre-processing in pattern recognition include applications of mathematical operators such as morphological and Laplacian operators for image enhancement. Data pre-processing also includes feature extraction process. This process is important as to reduce the amount of data obtained from the data acquisition stage. The extracted data must be manageable and sufficient to represent discriminatory information for identification (Bow, 2002). Feature extraction is also an essential mechanism to identify inherent characteristics or features found within data object that has been acquired. For instance, edge detection technique is used to extract information on the edges of object within a particular image. Algorithms such as Sobel (Kimmel, Shaked, Elad and Sobel, 2005) and Canny (Canny, 1986) edge detectors have been commonly used to extract edge features. The output of this pre-processing stage is a set of refined data or features, x_{feat} as shown in Figure 1.1 for classification purposes.

The decision classification stage is crucial in pattern recognition, as it determines the reliability of the pattern recognition system itself. Reliability of any pattern recognition schemes could be analysed through its accuracy in generating correct classification. This accuracy could be measured through different criteria, including classification error, acceptance error, mean square error, and receiver operating characteristic (ROC) curve. Classification stage also focuses on the process of labelling and determining pattern's feature into specific groups or classes. In supervised classification, the labels or classes have been predefined by the user. On the other hand, unsupervised classification builds up the labels or classes from different groups of patterns introduced into the system.

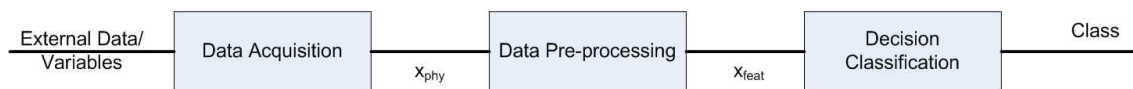


Figure 1.1: Conceptual representation of conventional pattern recognition scheme.

Pattern recognition is an important tool in evaluating and analysing large-scale data that have been produced in a wide area of applications. Nevertheless, current approaches incur excessive computational complexity to adapt to these large and highly-complex data

sets. There are a number of barriers that need to be addressed when dealing with these data sets, in regards to the pattern recognition implementation. These include:

- i. *Size of data:* With increasing size of data, existing pattern recognition schemes must be able to manage data in an efficient manner with specific concerns on storage and transport. The methods in which data should be stored and communicated within a recognition process must take into account the size of data sets used.
- ii. *Dimensions of data:* The sophisticated approaches in data capture technology have enabled the extraction of highly-dimensional data from the environment. In this context, pattern recognition applications must be able to cater for different dimensionalities of data in their implementations.
- iii. *Algorithmic complexity:* Existing pattern recognition schemes are powerful and have the ability provide highly-accurate solutions. Nevertheless, they incur high algorithmic complexity in their implementations. This phenomenon is mainly due to their iterative nature, as well as complex mathematical foundations. Some algorithms are exponential and infeasible for large-scale data. Furthermore, due to their expensive computations, existing pattern recognition schemes also can be computationally time-consuming, especially when dealing with data with large size and dimension.

These barriers are the common factors in determining the scalability of a particular pattern recognition approach. Each approach must be able to address increasing size and dimensionality of data, while minimising its complexity. In this regards, there is a need to evaluate some of the possible approaches towards adding scalability to pattern recognition schemes.

1.2 Scalability in Pattern Recognition

Scalability is an important factor in today's pattern recognition approaches. Existing outgrowth of data in daily's usage shows that there is a need to maintain the growth of the capability of existing algorithms to serve these large-scale data. For example, according to Anderson (2008), for every 72 minutes, there is one petabyte of data processed by Google's server. This value will definitely keep on increasing as the storage and processing mechanisms advances rapidly. The question of scalability as described by Pal and Mitra

(2004), whether the pattern recognition algorithm can process large data sets efficiently, while building from them the best possible models.

There are several techniques to scale up pattern recognition algorithms for large-scale data sets. These techniques could be divided into a number of approaches:

- i. *Data Approach*: This kind of technique modifies the data, prior to the recognition process. Some of the techniques include data reduction (Chow and Huang, 2008), dimensionality reduction (Rueda and Herrera, 2008), and data partitioning (Kbir, Maalimi, Benslimane and Benkirane, 2000). The aim of this approach is to minimise the size and dimension of data for efficient recognition. However, this approach may undermine the importance of data integrity in its approach through representation of large data domain using small data set.
- ii. *Learning Approach*: Pattern recognition algorithms mainly require a learning mechanism in its procedure. This mechanism may be computationally expensive. Hence, reducing the complexity of the learning mechanism is an objective of scalability. Examples of improving scalability using learning approach include active learning (Cheng and Wang, 2007) and incremental learning (Schlimmer and Granger, 1986). A significant limitation of this approach is that it may affect the accuracy of the algorithm in order to achieve fast and simple learning capabilities.
- iii. *Distributed Computing Approach*: The advancement in networking technologies have enabled large-scale computations to be performed within the body of a network itself. Rapid development in high performance computing and grid technology allows collaboration of resources to work for a specific application. In this context, existing pattern recognition algorithms may be implemented on distributed computing platform through the use of parallel processing. Some of the examples of scalable pattern recognition schemes using this approach include the works carried out by (Li, Tang, Xia and Wang, 2005; Khan, 2002).

Distributed computing approach for scaling existing pattern recognition algorithms is seen to be a potential optimum solution. However, some of the existing algorithms are highly-complex and difficult to parallelise. Developments of neural network algorithms for pattern recognition have shown an interesting insight on the implementation of pattern

recognition in distributed computing. Neural networks in its nature are formed through the collaboration of computational nodes known as neuron. Nevertheless, the integration between these two components is still in infancy, due to the tightly-coupled nature of existing neural network schemes. Furthermore, neural networks have been developed since 1950s with the early introduction of perceptron by Rosenblatt (1957). It has been initially conceived for single-processing (CPU-centric) architecture, with high dependency on iterative techniques. Therefore, more initiatives need to be carried out in order to attain the effectiveness and efficiency of neural network algorithms for pattern recognition using distributed computing approach.

Distributed computing may provide a seemingly unlimited scalability towards large-scale processing, given such a rapid advancement in existing distributed processing technology. Implementations of distributed pattern recognition scheme are possible. with the use of a simple, computationally inexpensive, and embarrassingly parallel pattern recognition algorithm on distributed computing. Nevertheless, there are some limitations towards this approach that need to be addressed. These include storage capacity consideration for different kinds of network, high internode communications, and network/node failure.

1.3 Distributed Pattern Recognition

One of the proposed solutions for the implementation of large-scale pattern recognition is distributed pattern recognition (Morrill, 1998). Distributed pattern recognition can be defined as an extension of existing pattern recognition approaches in which the recognition process is delegated across a distributed system. There are basically two main components involved in distributed pattern recognition: distributed pattern recognition algorithms, and the distributed architecture for pattern recognition.

Most of the initiatives on distributed pattern recognition have been focusing on providing distributed architecture for pattern recognition (Hsiao, Sung and Fan, 2002; Guoqing, Songcan and Jun, 1992; Nagy, 2005; Al-Hertani and Ilow, 2005; Choi and Oh, 2006). Consequently, this creates such a high dependency on hardware implementation. Hence, the issue of scalability in this context has yet to be solved. This is mainly due to inflexibility

of the existing distributed pattern recognition to be implemented across different architectural platforms and network environments, providing high capability for large-scale recognition deployments.

Development of distributed pattern recognition schemes that are based solely on algorithmic approach, independent of any hardware implementation, has yet to be fully realised. Although there are some existing research on the implementation of distributed approach for existing pattern recognition schemes (Talukder, Sheikh and Chandramouli, 2004; Khan and Mihailescu, 2004; Turkoglu and Arslan, 2001; Garai and Chaudhuri, 2007), these studies merely focusing on manipulating the methods in which this particular algorithm performed its recognition function (from sequential to parallel mechanism). Furthermore, existing distributed approaches have yet to be able to reduce the computational complexity of their respective algorithms, to be deployed in a distributed environment. In addition, these studies plainly lack consideration of the communication costs incurred due to highly-iterative features of existing pattern recognition schemes.

A new form of distributed pattern recognition scheme is needed, that has the flexibility to be deployed in various network environments, while being able to maintain low computational cost in its function. This scheme must be developed from an algorithmic perspective, capable of providing highly-accurate recognition function and scalable towards increasing size, amount, and complexity of data.

The deployment of pattern recognition applications for large-scale data sets is an open issue that needs to be addressed. Several approaches have been proposed, including the techniques such as data reduction (Chow and Huang, 2008), active learning (Cheng and Wang, 2007) and distributed approach (Li et al., 2005; Khan, 2002) in pattern recognition. Nevertheless, a common denominator for this is the algorithmic complexity of existing pattern recognition schemes. The distributed approach for pattern recognition offers a significant advantage for large-scale data analysis because it has the ability to provide extensive support for resource availability for increasing size, complexity, and amount of data. The ultimate goal for any distributed pattern recognition is to be able to perform a large-scale analysis to extract useful information from a huge collection of data.

Distributed pattern recognition remains a relatively unexplored area since pattern recognition has been considered highly problem specific, hence bearing little prospect as a generic commodity application. This is mainly due to the problem of complexity in existing

pattern recognition algorithms, and these problems limit their distribution factor. Several initiatives have been made to parallelise and distribute a pattern recognition algorithm to work on a distributed system. However, this kind of implementation faces a significant hurdle in the process of parallelisation.

The following subsections will further described some of the characteristics and applications of distributed pattern recognition, together with its related approaches.

1.3.1 Top-Down vs. Bottom-Up Approaches

Existing distributed pattern recognition schemes have been designed and deployed using a top-down approach, in which relatively CPU-centric (or sequential-based) algorithms were modified and enhanced to perform in a distributed manner. Furthermore, existing schemes tend to apply distribution mechanism partially, i.e. in the context of training and validation. Some of these examples include feed-forward neural networks and self-organising maps. Different kinds of distribution approaches have also been considered, which are described below (Foo, Saratchandran and Sundararajan, 1995):

- i. *Process Farming*: In this approach, recognition process is distributed across a number of parallel processors. Each processor carries out a training process using a copy of the algorithm being deployed, as shown in Figure 1.2. In this configuration, each processing network consists of a master node and several worker nodes. Each worker nodes performs training or recognition process independently of each other. However, for each cycle, updates (in terms of bias weight and errors) must be sent to the master node for evaluation/adjustment purposes. This process will be iteratively performed until the optimum bias weight and error value have been achieved by the network. The training dataset used in this approach is divided into a number of training subsets for each processor used. Therefore, each processor would perform this training procedure on a subset of the overall data.
- ii. *Pipelining*: Recognition procedure involving pipelining approach follows an incremental method, in which the training process is conducted subsequently, following a pipeline procedure as shown in Figure 1.3. In this context, each processor also contains a copy of the algorithm and perform a recognition process on a particular training subset. However, the weight and error changes are modified and evaluated,

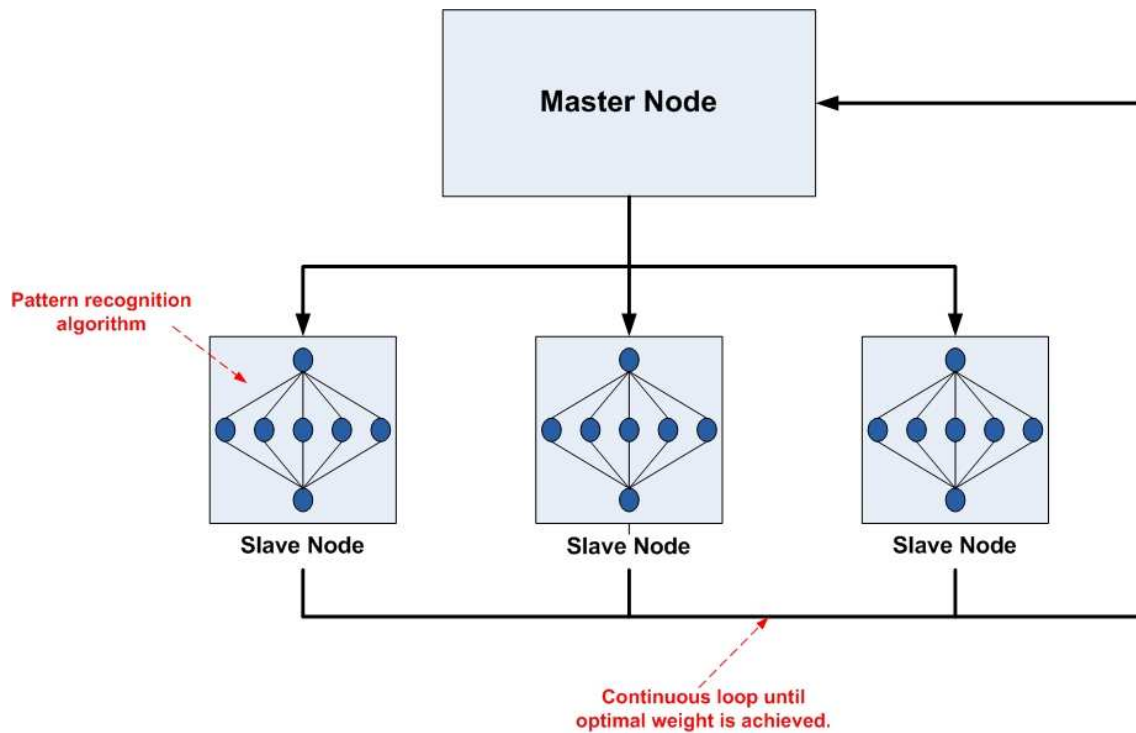


Figure 1.2: Distributed pattern recognition with process farming approach.

each time these values are passed from one processor to another, and being added into each weight and error calculations involved.

The top-down approach towards distributed pattern recognition has several limitations including:

- i. *Recall disintegrity*: Distribution of training dataset into a number of subsets would create a disintegrity in the training process itself, due to vertical splitting of data. This will influence the actual recall process, due to significant difference between the weight changes produced by the algorithm on highly-cohesive training set, i.e. training data that hardly classified due to dissimilar feature values; and loosely-cohesive data, i.e. data that is easily classified and clustered.
- ii. *Highly-Congestive Network*: Algorithms such as feed-forward neural network and Hopfield network are highly-iterative (in terms of training cycles to obtain optimum output) in nature. Large number of iterations in training/recognition process would lead to massive communication exchanges within any distributed environment. Therefore, it would create a highly-congested network.

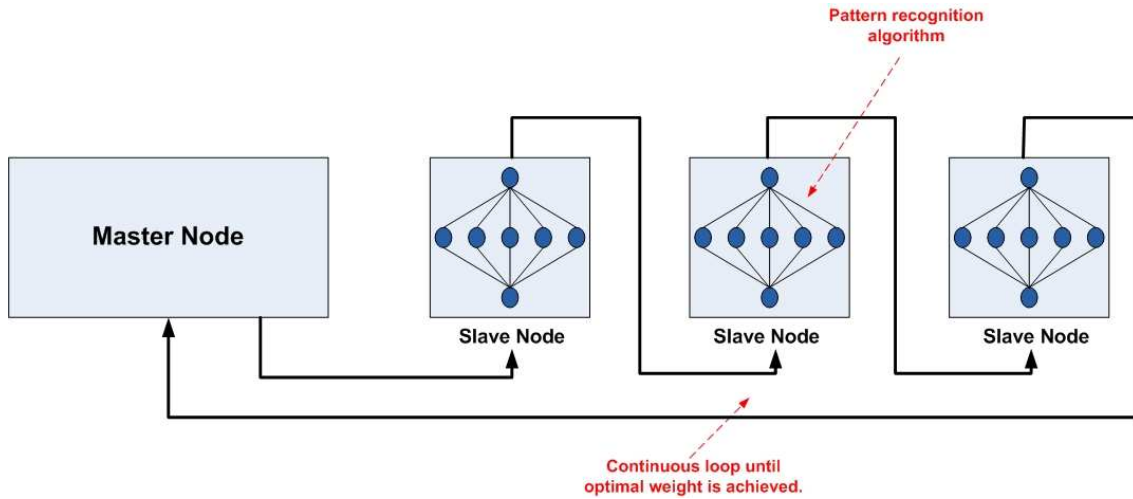


Figure 1.3: Distributed pattern recognition with process pipelining approach.

- iii. *Unchanged Level of Complexity:* The approach used in existing distributed pattern recognition schemes mainly apply actual pattern recognition process at a smaller scale, i.e. using similar algorithm with smaller training space. Hence, the complexity of the algorithm remains. The difference lies in the fact that executing recognition process at a smaller scale may improve the algorithm's performance time through the reduction of the amount of training data used. However, this is not always the case as the processing time also depends on the number of learning cycles implemented for each recognition process. While the complexity of the algorithm remains unchanged, it is hard to estimate its resource requirements, and thus it may not be applicable for resource-constrained networks such as wireless sensor networks (WSNs).

Of all other known distributed pattern recognition schemes, Graph Neuron (GN) (Khan, 2002) is shown to be effective in a distributed manner using a bottom-up approach. GN implements distributed associative memory concept, in which the processing element (or neuron) may be distributed across a computational network. The bottom-up approach in the GN distributed scheme simply means that the algorithm itself is readily capable of performing pattern recognition in a distributed environment through its in-network processing capability. In this context, each neuron within a GN infrastructure could be matched to a single physical processor within a network. In addition, GN adopts a single-cycle learning approach, in which for each pattern introduced into the network, its recognition procedure only performs learning mechanism within a single pass. GN

performs its recognition procedure using a scalable graph-matching approach, and hence does not require weight and error adjustments as in other neural network algorithms.

1.3.2 Distributed Multi-Feature Recognition

Conventional pattern recognition schemes as shown in Figure 1.1 commonly involve analysis of a single feature with a number of parameters that are linked together to form a pattern vector. This technique requires a careful selection of feature that best represents the whole pattern population. The use of multiple features in recognising patterns would perhaps enhance the accuracy of the recognition approach. Efforts in implementing pattern recognition using multiple features have been introduced through a number of studies including (Favata and Srikantan, 2002; Cao, Ahmadi and Shridhar, 1995; Zhao, Huang and Sun, 2004). Existing pattern recognition schemes involving multiple features mainly adopt the highly-complex algorithms such as radial-basis function neural network (RBFNN) and back-propagation neural network (BPNN).

Existing techniques primarily rely on separate mechanism for each feature recognition to be conducted. The research conducted by (Zhao et al., 2004) have shown the use of neural networks committee machine that forms a collection of decisions obtained by a group of neural network algorithms that perform face recognition on different features. Cao et al. (1995) on the other hand, performs a feature-combination approach in their handwritten recognition scheme. The multiple features approach has shown to produce high recall efficiency. However, this comes together with complex and iterative nature of the algorithms being used. As a result, the performance of the schemes may be affected, due to delays in the recognition process.

Deployment of multi-feature recognition using distributed approach has yet to be fully-realised, due to expensive computational costs of existing pattern recognition algorithms. This study envision a distributed recognition scheme that is capable of providing real-time recognition using multiple features with high accuracy. This distributed multi-feature recognition together with the availability of high-performance networks may be highly-beneficial for applications involving real-time feedback such as face identification for security purposes.

1.3.3 Resource-Awareness

Current approaches towards implementing pattern recognition algorithms in distributed environment have purposely targeted towards improving the performance time, as well as to provide scalability for increasing amount and dimension of data. Nevertheless, these approaches are overburdened by their highly-complex computations, and require significant quantity of resources to perform in a distributed manner. For example, for a single processor to perform a recognition process using a Hopfield network with n neurons, its computational complexity is equivalent to $O(n \log n)$. Hence, it is important for the network to acquire sufficient computational resources for the algorithm to perform at its best condition. However, this is not always the case in different kinds of computational networks available in the current technological climate.

The important aspect that is still lacking in the existing distributed pattern recognition schemes is resource-awareness. It is essential for any distributed scheme to consider its computational and storage incurrences, due to different levels of granularity of networks. With the extent of application development ranging from complex data mining process to event detection, the use of distributed systems have been widely applied in day-to-day operations ranging from high performance computing networks to lightweight and resource-constrained WSN networks. In this vein, a dynamic and robust distributed pattern recognition scheme should be able to perform under different network granularity hence, providing an expandable coverage for different kinds of applications.

Existing distributed schemes have shown to work well on resource-abundant networks such as the grid and high performance networks. However, when dealing with scarce resources, these schemes might fail due to either insufficient power, computational resources, or storage capacity. It is the intention of this research to take up a further investigation of the capability of a distributed pattern recognition scheme, for deployment within different levels of network granularity.

1.4 Motivation and Aims

As previously mentioned, the distributed approach has been identified as a potential solution for scalability issue within existing pattern recognition schemes. Nevertheless, some

limitations have also been identified. These include improper implementations of tightly-coupled recognition algorithm using a top-down approach.

The main motivation for the research work of this thesis lies in the need for a bottom-up approach for the distributed pattern recognition scheme with resource-awareness and ability to perform multi-feature recognition, instead of redeveloping existing top-down approaches in applying a CPU-centric pattern recognition algorithm in a distributed manner. It is essential to observe algorithmic development from different perspectives. This research will address this issue by looking from the point of distributed processing, in which, all the processing elements within a network may form a pattern recognition procedure using a fully-distributed algorithm. This is somewhat different from the traditional point of view, in which the development of distributed approach has been observed from the algorithmic viewpoint.

As described earlier, Graph Neuron (GN) as a pattern recognition scheme has shown its higher ability to perform in a distributed manner, with its single-cycle learning and in-network processing capabilities. However, GN has so far only being implemented on WSN infrastructure for energy-efficient event detection (Baqer, Khan and Baig, 2005) and intrusion detection (Baig, Baqer and Khan, 2006). An extension of GN algorithm known as Hierarchical Graph Neuron (HGN) has been introduced by Nasution and Khan (2008) which offers high scalability and accuracy for distorted pattern recognition. Furthermore, HGN eliminates a limitation of GN by accommodating overall view of patterns in its recognition procedure. In this work, further analysis on the capabilities of HGN for distributed pattern recognition scheme will be conducted.

The aim of this thesis is to design fully-distributed and scalable pattern recognition schemes for large-scale data analysis and evaluation. Different approaches for pattern recognition are compared, and their capabilities for deployment within a computational network is observed, and a conclusion that the most appropriate approach is to use a lightweight and loosely-coupled recognition algorithm with in-network processing capability and single-cycle learning mechanism is made. An intention of this research is to demonstrate that a simple recognition mechanism using a pattern-matching concept would scale up for large and highly-complex data sets, given a sufficient amount of network resources

are available. The distributed model developed for pattern recognition using the bottom-up approach should be able to produce an acceptable level of accuracy, while preserving its scalability for increasing size, amount, and complexity of data.

In addition, this thesis will also investigate a distributed pattern recognition deployment for different computational networks, based upon their granularity, ranging from coarse- to fine-grained networks. The effectiveness and efficiency of the proposed model in different network conditions will be evaluated. This study is essential in order to provide a dynamic utilisation of the proposed model for different network configurations. Furthermore, existing pattern recognition applications have been widely deployed in different areas ranging from image recognition to event detection. Hence, the ability for distributed pattern recognition to suit the available network resources is distinctively important.

1.5 Hypothesis and Research Objectives

Based upon the research motivation and aims described in previous section, the main hypothesis of this research is that a distributed pattern recognition scheme with single-cycle learning and the bottom-up approach, will remain scalable for any given size or dimensions of data, given that sufficient computational resources are available. Furthermore, this research will also be looking into resource-awareness and multi-feature aspects of distributed pattern recognition scheme. In order to achieve the aims of this research and prove its hypothesis, a number of objectives have been formulated as follows:

- i. To conduct a review on current distributed pattern recognition schemes. A number of initiatives have been carried out in performing pattern recognition using a distributed manner. Issues related to the top-down approach that has been implemented in existing distributed schemes will be investigated. The ultimate goal for this objective is to build up a foundation of knowledge on distributed pattern recognition and the proposed solution towards existing problems its currently facing.
- ii. To conduct an evaluation of a particular recognition scheme with the bottom-up approach. This will improve the understanding of its processes, together with its capabilities and limitations. An investigation will also be conducted on the capability of this algorithm to perform recognition procedure in distributed approach. This aims to derive a proposal on distributed pattern recognition scheme that is

deployable in distributed environment. In achieving this, a plan has been made to discover a number of important aspects of the proposed distributed approach, including its computational complexity, learning mechanism, data representation, and distribution mechanism. An additional intention is to benchmark the proposed scheme with other established neural network algorithms including Hopfield network, Kohonen SOM, and instantaneously-trained neural networks (ITNNs), in regards to their computational complexity.

- iii. To investigate the scalability of distributed pattern recognition approach. An important aspect in any distributed pattern recognition scheme is the ability to scale up for any given amount and dimension of data. The scalability factor of the proposed approach will be investigated, which will also provide an analysis of its complexity. This is to prove that the complexity of this current approach remains constant, or growing linearly with an increase in the amount and complexity of data. Two important parameters for scalability will be used in the analysis involving recognition time and recall accuracy of the proposed scheme.
- iv. To investigate capability and effectiveness of the proposed distributed approach for implementation with multiple features. In this objective, A proposal of multi-feature recognition using a distributed pattern recognition approach will be reviewed. Existing multi-feature recognition approaches heavily rely on separate complex algorithmical mechanism for each feature. A distributed approach using a singular mechanism has yet to be explored. An outcome of this objective would be a proposed multi-feature distributed recognition scheme that capable of analysing complex patterns. This research will also look at feature distribution and recall accuracy of the proposed scheme.
- v. To evaluate storage and processing efficiency for the proposed distributed approach for pattern recognition within a resource-constrained environment. An analysis of resource-awareness factor for our proposed algorithm will be carried out. A proof of compatibility of our proposed approach for deployment within different levels of network granularity will be demonstrated. In achieving this objective, further investigation will also be made into its storage cost, in terms of its estimated data representation size. In addition, the computational cost requirement, i.e. the amount

of processing nodes required for a given recognition task involving patterns with different sizes and dimensions will also be analysed. This thesis therefore plans to evaluate the best data representation for effective pattern storage and processing.

1.6 Research Contributions

The contributions of this thesis can be summarised as follows:

- i. A divide-and-distribute approach on patterns for recognition purposes will be proposed. Pattern distribution approach in this work is different from existing technique. A distribution technique known as horizontal splitting will be used. This technique involves a distribution of pattern into subpatterns, rather than the commonly used vertical splitting technique in which distribution of pattern data is made into smaller pattern subset. This research extends the works on existing single-cycle learning recognition algorithm implementation using this approach, and will additionally observe its capability to reduce recognition error through the effect known as error encapsulation.
- ii. A distributed pattern recognition scheme will be proposed, which will implements a bottom-up algorithmic distribution approach. The scheme is distributed single-cycle learning pattern recognition algorithm with parallel in-network processing capability, which offers high scalability and recognition accuracy for distorted pattern recognition with low computational complexity.
- iii. A collaborative-comparison learning (CCL) technique will be proposed for fast and resource-effective recognition procedure for distributed pattern recognition implementation. CCL adopts a scalable graph-matching technique with single-cycle recognition procedure. The capability of CCL to learn patterns with low storage requirement and complexity will be demonstrated.
- iv. An extension towards data representation for distributed pattern recognition algorithm is introduced, which implements binary signature scheme for data representation and dimensionality reduction. This scheme enables data to be represented in a simple binary format with high representation factor. Moreover, it helps reducing

the complexity requirement for data analysis for the proposed distributed pattern recognition scheme.

- v. A new distributed multi-feature recognition scheme will be introduced. This scheme performs recognition of features in distributed manner with low computational complexity and high classification accuracy. This research will perform a simulation on this scheme for face recognition using a number of different features. The results indicated high accuracy for increasing number of features used in our proposed scheme, while maintaining relatively-low algorithmic complexity.
- vi. Two computational configurations namely fine-grained and coarse-grained distributed pattern recognition schemes will be presented for different levels of network granularity. Fine-grained scheme is used for recognition involving large number of processing nodes with low computational capability such as in WSN network, while coarse-grained scheme may be implemented in resourceful networks such as computational grid. Both configurations perform similar recognition techniques, and achieve high recognition accuracy for distorted pattern recognition.

1.7 Thesis Outline

The rest of the thesis is organised as follows. In Chapter 2, a review of different approaches to pattern recognition will be conducted, and a detailed evaluation of the scalability of these approaches will be presented. In addition, different distribution techniques that are applicable for scaling up these recognition algorithms will be identified. This chapter also specifically provides a review of Graph Neuron (GN) and Hierarchical Graph Neuron (HGN) algorithms that implement effective parallel pattern matching techniques for distorted patterns. In addition, a detailed elaboration on the concepts, features, and limitations of these algorithms will be provided. The main intention of Chapter 2 is to focus attention on the proposed distributed approach for pattern recognition algorithm. This chapter also demonstrates different possible forms of distribution for the algorithm deployment within a distributed network.

In Chapter 3, a novel distributed approach for pattern recognition scheme will be presented. This chapter extensively describes the features of the scheme proposed in this

thesis. The description includes its overall architecture, learning mechanism, and evaluation of the possible implementation platforms. In addition, a discussion on pre-processing requirements will also be presented. Both collaborative-comparison learning (CCL) technique and binary signature scheme implemented in this thesis will be discussed. This chapter will also demonstrate a significant enhancement on collective decision technique for distributed approach using the proposed majority/minority voting approach. Both complexity and scalability analyses on the distributed scheme are performed, in comparison with other pattern recognition algorithms. A series of experimental analyses on the pattern recognition capability of the proposed distributed scheme will also be described.

In Chapter 4, a discussion on a distributed pattern recognition scheme involving multiple-feature patterns will be conducted. In this chapter, the capabilities of DHGN scheme for analysing complex patterns such as facial images using a distributed multi-feature recognition approach will be demonstrated. This research performs a simulation of this approach in a face recognition application and handwritten numeral character objects. Several enhancements of the proposed scheme are also presented.

The deployment of distributed pattern recognition so far has yet to consider the constraints on computational resources. In Chapter 5, the results of an investigation into the scheme proposed in this thesis to determine the performance capabilities within both highly-resourceful computational networks and resource-constrained environment such as a wireless sensor network (WSN) will be presented. A discussion on the message-passing model of the proposed scheme will be presented, and a number of possible implementations and configurations of pattern recognition in different network environments will be proposed. In addition, this chapter also details the communication cost analysis of the proposed scheme for efficient pattern recognition implementation on a decentralised and fully-distributed networks.

As continuation of the discussion related to resource-awareness of the proposed DHGN distributed pattern recognition scheme, Chapter 6 looks specifically into the applications of DHGN within a resource-constrained WSN network. Capabilities of DHGN in performing distributed recognition will be put into critical applications such as event detection within fine-grained WSN environment. A case study on DHGN for forest fire detection within WSN network will also be presented.

Finally, Chapter 7 summarises all the contributions made through the thesis, and elaborates some potential ideas for possible future research.

Chapter 2

Pattern Recognition and Distributed Approach

Distributed computing approach offers seemingly unlimited scalability towards pattern growth with the rapid advent of network computing technology that enables processing to be performed within a body of a network, rather than concentrating on exhaustive single-CPU utilisation. Nevertheless, existing approaches are still lagging behind, due to tightly-coupled recognition algorithms being implemented, as well as iterative processing means to achieve significantly accurate results.

The neural network approach offers a promising tool for large-scale pattern recognition. This is mainly being reflected by its ability to perform parallel computations using interconnected neurons. However, there are also several issues related to its implementation. These include convergence problems, complex iterative learning procedures, and low scalability with regards to the training data required for optimum recognition.

The aim of this chapter is to undertake a review of current approaches in pattern recognition for large-scale data. An analysis on the scalability issue within existing schemes will also be considered. This chapter intends to establish the need for scalable pattern recognition model to enable large-scale data analysis to be performed. In addition, this chapter will also provide a foundation for the proposed distributed approach in scalable pattern recognition.

The objectives of this chapter are:

- i. To analyse the scalability of existing pattern recognition approaches using neural network algorithms.
- ii. To provide a discussion on key components of a scalable pattern recognition scheme.
- iii. To conduct a review of learning mechanisms for pattern recognition using neural network algorithms.
- iv. To provide a justification on the adoption of distributed approach as a mean to enhance scalability of pattern recognition algorithms.
- v. To propose a distributed approach for deployment within a single-cycle learning algorithm for large-scale data recognition and analysis.

This chapter has been structured as follows. Section 2.1 presents a discussion on the scalability of existing neural network/machine learning approaches for pattern recognition. Section 2.2 will then present further detailed descriptions on the evaluations that have been carried out, in regards to the scalability factor of existing neural network approaches. Section 2.3 describes some of the key components for scalable pattern recognition. Consequently, Section 2.4 discusses different pattern distribution techniques that can be employed in any distributed pattern recognition systems. Section 2.5 initiates the main focus of this thesis, through a discussion on GN algorithm for scalable pattern recognition, followed by further exploration of the knowledge of this algorithm for distributed pattern recognition implementation. Based on the reviews conducted, Graph Neuron (GN) algorithm exhibits single-cycle learning and in-network processing capabilities, and has been implemented in a number of pattern recognition applications in wireless sensor networks (Baquer et al., 2005; Baig et al., 2006). This research intends to further explore this algorithm for the possibility of deployment in the proposed scalable distributed pattern recognition scheme. Section 2.6 extends the analysis on GN-based algorithm known as Hierarchical Graph Neuron (HGN). HGN is an associative memory algorithm that implements GN-based recognition in a hierarchical mode, to obtain an overall view of the pattern's structure. A review on HGN and its capability and limitation to provide scalable scheme for pattern recognition will also be included. In Section 2.7, the discussion on HGN is extended by looking at possible distribution mechanisms for HGN implementation, as to increase its scalability for large-scale pattern recognition. Furthermore, this section leads

to this doctoral research on distributed pattern recognition. Finally, Section 2.8 concludes the chapter.

2.1 Neural Network/Machine Learning Approach

The neural network approach was defined in the 1950s with the introduction of the Perceptron approach by Rosenblatt (1957). The concept of an artificial neural network (ANN) (Hecht-Nielsen, 1989; Hopfield and Tank, 1985; Kohonen, 2000) is fundamental to neural computing, which emerged from the knowledge and understanding of how biological neural systems work with regard to information storage and manipulation (Schalkoff, 1991). Neural networks can be considered as massive parallel computing systems consisting of a large number of small processors known as neurons that interconnect with each other (Jain et al., 2000). One of the benefits of ANN-based pattern recognition is that it allows the system to learn and adapt to the nature of data. By having this adaptive feature, the recognition schemes are able to be used in a wide range of applications. Hence, it is capable of offering a more scalable approach for large-scale recognition. Figure 2.1 shows a generic example of neural network structure for pattern recognition.

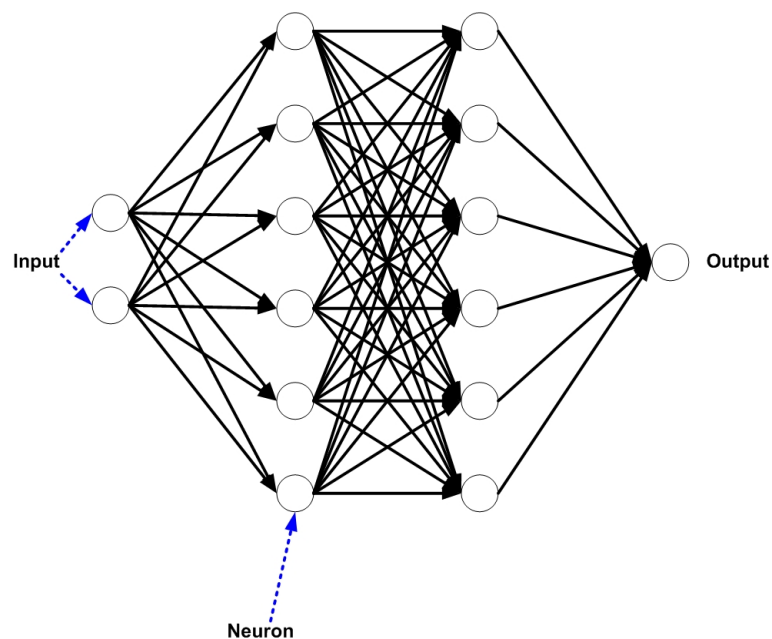


Figure 2.1: Generic neural network architecture for pattern recognition with two layers of hidden neurons.

The pattern recognition applications using neural network approach rely heavily on the learning algorithm being adopted. It is essential as to determine the efficiency and accuracy of pattern store and recall operations. Learning algorithms allow neural networks to learn complex non-linear input-output relationships, use sequential training procedures, and adapt themselves to the data (Jain et al., 2000). Prominent approaches in learning algorithms include Hebbian learning (Hebb, 1988) and incremental learning (Schlimmer and Granger, 1986). Further discussions on these learning mechanisms will be presented in Section 2.3.1.

Apart from neural networks, machine learning is also a widely-used approach for pattern recognition. Machine learning as defined by Nilsson (1996) refers to changes in the systems that perform tasks associated with artificial intelligence (AI). Machine learning is also considered a deterministic approach for pattern recognition. It is also commonly used in conjunction with other neural network schemes. For instance, Kernel-based Associative Memory (KAM) (Nowicki and Dekhtyarenko, 2004) implements kernel machine as the learning tool for its pattern recognition process. Other machine learning algorithms including support vector machines (SVMs) (Cortes and Vapnik, 1995) are also being used as a pattern recogniser.

Implementations of machine learning approaches for pattern recognition require a priori knowledge on the types of functions or kernel machines to be used for a specific recognition domain. For instance, linear SVM has the ability to perform classification on binary problems, and is not suitable for multi-class domains. In addition, the machine learning approach also commonly requires extensive and iterative learning procedures to obtain the best parameter estimation for the machine to work on a specific set of data. These two issues in machine learning approach affect its feasibility in providing scalable and generic pattern recognition scheme.

Neural network (and machine learning) approaches offers low levels of both scalability and adaptability, based upon the reviews of existing algorithms and techniques applied. This evaluation is based upon the following criterias of neural network approaches:

- i. Some neural network/machine learning approaches can be conducted within a parallel environment. E.g. Hopfield network and Feed-Forward neural network. This parallelisation capability enables recognition to be conducted on large-scale data.

However, the complexity of these algorithms hinder its capability to perform pattern recognition in a purely-parallel manner

- ii. Enhancement in unsupervised machine learning schemes such as K-mean clustering algorithm provides an opportunity for heterogeneous patterns and data to be used in recognition processes. Nevertheless, these algorithms require strenuous training and complex recognition procedure.
- iii. Limited storage capacity - E.g. an estimate of $0.138N$ random patterns, where N represents the number of units in the network, which could be stored by Hopfield network for optimum recognition (Sulehria and Zhang, 2008).
- iv. Some neural networks have a capability to learn from the data used in the recognition process. However, the learning process require large number of similar data for memorisation.

In addition to these factors, neural network/machine learning approaches also suffer from a number of issues, including:

- i. Iterative procedure in weight adjustment for many neural network schemes such as feed-forward network during data training imposes significant delay in processing.
- ii. Over-fitting problem - Small training set is not capable of representing large-scale actual data.
- iii. Recognition function within neural networks or machine learning works only for a specific problem within the recognition domain. Network retraining is required for different sets of problems.

The reviews obtained from the literature have shown that neural network/machine learning approach outweighs other approaches with regards to the scalability and adaptability issues. Nevertheless, a syntactical approach, for instance, is also considered to be adaptive towards wide range of problems, owing to its grammatical inference technique. On the other hand, the machine learning or neural networks could be considered as the best candidates for scalable and adaptive pattern recognition scheme, due to its capability to use different kinds of functions and representations. Furthermore, some of these approaches have a capability to perform in parallel environment, where recognition loads

are able to be distributed across computational nodes. Nevertheless, they require extensive and complex computations in deriving the best solution for recognition process. The following subsection describes some of the examples of neural network/machine learning algorithms that have been applied in pattern recognition domain.

2.1.1 Scalability and Adaptability of Neural Networks

This subsection outlines the major approaches for pattern recognition applications using neural networks or machine learning techniques. An attempt is made to describe several machine learning and neural networks algorithms that are commonly used in pattern recognition. The discussions within this section encircled towards understanding both approaches and exploring issues related to their implementations. The focus aims to explain the scalability and adaptability factors within both machine learning and neural network approaches. To achieve this, several examples of pattern recognition algorithms will be reviewed.

An examination of the literature revealed that neural network and machine learning approaches offer a highly-reliable deterministic pattern recognition scheme (Vivanco, Demko and Pizzi, 2005). Furthermore, these approaches provide a medium for the system to learn from existing data or patterns and adapt to their conditions. However, their relative computational complexity has been considered as a major bottleneck for large-scale implementation. This is due to the fact that most common algorithms in machine learning (including neural networks) apply extensive and highly iterative training procedures. The discussion outlined in this section will cover the underlying principles of both approaches.

In this subsection, scalability and adaptability aspects of each of the approaches will also be reviewed. This review will firstly discuss some of the algorithms related to neural network and machine learning approaches, followed by the provision of several examples of neural network implementations for pattern recognition, including some examples of a form of neural network approach known as associative memory (AM). In addition, scalability and adaptability factors of each algorithm will be examined, including the analysis of some machine learning algorithms for recognition purposes.

Feed-forward Neural Network

Feed-forward network design acts as a basis for more complex neural networks such as Hopfield Network and Kohonen Self-Organising Network. It offers a convenient approach for making auto-associations between an input layer and an output layer in computational problem solving such as in pattern recognition (Nadal, 1989). However, there are some constraints related to its implementation in the classification process. These include its sensitivity towards the training parameters, training speed, nonlinear classification function, overtraining sensitivity, and regularisation requirement (Jain et al., 2000). According to Kalos (2005), the feed-forward neural networks are highly specialised, thus making it difficult to be accepted in the mainstream applications. Furthermore, he stated two problems relating to the implementations of these neural networks. These include a difficulty in interpreting the results and trial-and-error nature of the process required to design its architecture. The difficulty in predicting the results lies in the fact that the results obtained from the network must be cross-validated, in order to select the best results. Moreover, the difficulty in designing the network architecture is due to the instability of the number of hidden layers required for a given dataset. In relation to these problems, the weight adjustments mechanism between input-output neurons also influenced the complexity of the algorithm.

Scalability and adaptability are among the major issues in feed-forward network. Due to its computationally intensive operations, feed-forward network requires extensive training procedure as to obtain optimum recognition accuracy for a given data set. Nevertheless, its architecture provides a possibility for parallel implementation, to improve its scalability for large-scale processing. Feed-forward network is an acyclic network in which the nodes are only connected with nodes in subsequent or previous layer. There are a number of initiatives that have been carried out to parallelise this neural network. These include the works by Kumar, Shekhar and Amin (1994) and Foo et al. (1995). Their work has been focusing on parallelising the backpropagation learning for feed-forward network. Parallelism for feed-forward neural network could be achieved. Nevertheless, it incurs additional requirements, including specific machine architecture such as hypercubes and transputer array. In addition, serial feed-forward network has shown better benchmark performance than parallel scheme for small training data as described by Kumar et al. (1994).

With a requirement for specific machine architecture, complex computation and other limitations being described previously, feed-forward network is not a suitable candidate for a scalable pattern recognition scheme. However, its ability to parallelise its structure is considered to be an opportunity that could be taken into consideration.

Hopfield Network

The Hopfield network is a supervised recurrent neural network approach based on the work of Hopfield and Tank (1985). The Hopfield network provides an alternative approach for solving complex computational problems such as combinatorial optimisation. The Hopfield network has also been proven to provide a better solution to the Travelling Salesman Problems (TSP) and pattern recognition. Figure 2.2 illustrates the general topology of a Hopfield network which has been adapted from van der Smagt (1990).

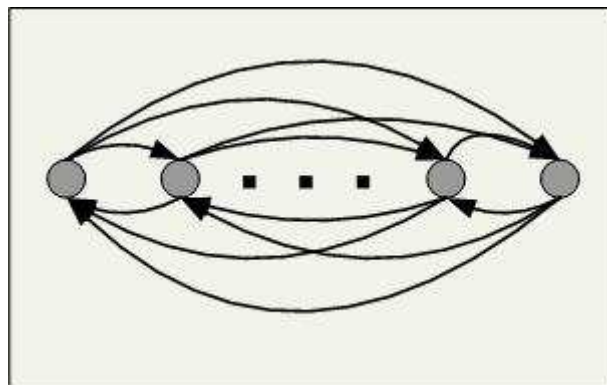


Figure 2.2: The general topology of a Hopfield Network.

According to Kim, Yoon, Kim, Park, Ntuen and Sohn (1992), Hopfield networks can be categorised into two types: Discrete Hopfield Network (DHN) and Continuous Hopfield Network (CHN). DHN is a stochastic model and it offers simple implementation and fast processing. However, DHN produces the results in approximation, since it uses binary values for the states of neurons. Therefore, DHN would not be able to provide a precise solution in a pattern recognition application. On the other hand, CHN offers a near-optimal solution using a differential equation approach. This is actually a burden for CHN since it requires longer time for its simulation. Hence, CHN is not suitable for pattern recognition application which requires fast recognition, such as in biometric pattern recognition.

The Hopfield network suffers from several problems, including convergence problems that lead to less than optimal solutions being provided (Li et al., 2005). In addition,

Hopfield networks have a scalability issue with regard to the storage of biased patterns (Löwe, 1999). Sulehria and Zhang (2008) stated that the size of Hopfield network memory is a round $0.138N$, where N represents the number of neurons within the network. This limitation is mainly due to the phenomenon known as spurious local minima. On the other hand, the Hopfield network has been adopted for solving optimisation problems due to some of its advantages, including massive parallelism, convenient hardware implementation, and the neural network architecture (Li et al., 2005).

Due to its storage limitation and convergence problem, Hopfield neural network is not suitable for pattern recognition involving large-scale data sets. Regardless of its capability to perform recognition processes in parallel, its finite memory capacity hinders its capability to perform large-scale recognition, as been demonstrated by the research work of Wilson (2009).

Kohonen Self-Organising Network

Introduced by Teuvo Kohonen, Kohonen map (Kohonen, 2000) (or Kohonen network) is an unsupervised neural network algorithm that can be used for patterns clustering and classification. Kohonen derived the network from the competitive learning algorithm (Rumelhart and Zipser, 1988), which consists of an N -unit input layer and an M -unit output layer, in which M equals to the number of cluster groups required. Kohonen represents the network in terms of ordered maps, as in the brain. Kohonen network forms the existing approach known as Self-Organizing Maps (SOMs). SOM is an algorithm for mapping high-dimensional data space to lower-dimensional data space. SOM takes benefit of dimension reduction in classification process. The uniqueness of SOM is such that the neurons are well-represented in the form of geometrical dimension (Giorgetti, Gupta and Manes, 2007). Figure 2.3 shows the general topology of Kohonen network. Note that each node is connected to the inputs.

Kohonen network has been implemented in wide area of applications. These include geoinformatics (Zaremba, St-Laurent, Niemann and Richardson, 2000), bioinformatics (Wang, Zheng and Azuaje, 2007), finance (Blazejewski and Coggins, 2004), information retrieval (Lin, Soergel and Marchionini, 1991), and wireless technology (Giorgetti et al., 2007). Some recent developments of Kohonen network in the classification approach can be seen through the work of Berglund and Sitte (2006) in parameterless SOM,

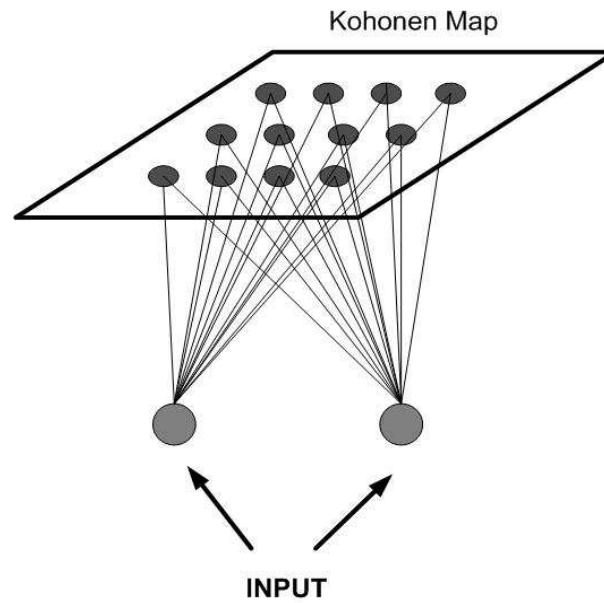


Figure 2.3: General topology of Kohonen network.

Cheung and Law (2007) in Rival-Model Penalised SOM, and Wu and Chow (2007) in Self-organising and self-evolving neurons.

Self organizing map (SOM) is a popular tool for clustering and visualisation of high-dimensional data. However, the processes involved require numerous iterations on each neuron, and thus incurred additional computational costs to be burdened. Its computational complexity increases significantly as with an increase in the dimension of data. In solving this issue, existing dimension reduction techniques have commonly been adopted. However, this adds to its processing time. Apart from the curse of dimensionality issue, learning rate determination is also considered as an issue, since the initial value of learning rate is essential in determining the efficiency of the map (Cheung and Law, 2007). These two problems related to Kohonen SOM implementation limit its capability to provide a scalable approach for pattern recognition. Nevertheless, it can be considered as an adaptive scheme, due to its availability to represent different types of data using a single form of representation.

Radial Basis Functions (RBF) Neural Network

RBF Neural Network (RBFNN) implements radial basis functions such as Gaussian function in its activation function. RBFNN is conceptually similar to K-nearest neighbour

(k-NN) model. Its underlying principle in recognition process is such that the class prediction for a given pattern is based upon the patterns having closely-related features to it. RBFNN network formation is almost similar to feed-forward neural network. RBFNN has been adopted in several multi-class classification applications including the works of Yang and Paindavoine (2003) in real-time face tracking and identity verification, and Ng, Dorado, Yeung, Pedrycz and Izquierdo (2007) in multi-class image classification based on MPEG-7 descriptors.

In regards to the scalability and adaptability issues, it is impractical for RBFNN to be deployed for large-scale recognition. This is due to the fact that there are several problems concerning its implementation. These include its generalisation problem. In this context, it is difficult to train the network with limited amount of training data for such a large-scale actual data. In addition, usual training approaches for RBFNN involve methods to find cluster centres in the feature space to be used in RBF function. One common approach involves the use of K-means clustering. This approach is computationally expensive and extensively iterative. The difficulty also arises in determining the best cluster centres for a given input space.

2.1.2 Convolutional Neural Network

Convolutional neural network (CNN) has been originally introduced by LeCun and Bengio (1995). It has been applied in a number of pattern recognition applications, including handwritten character recognition (Lecun, Bottou, Bengio and Haffner, 1998), visual documents (Simard, Steinkraus and Platt, 2003) and face recognition (Lawrence, Giles and Tsoi, 1996). It is based upon the multi-layer property of neural networks. According to Bouvrie (2006), This particular kind of neural network assumes to learn filters, in a data-driven fashion as a means to extract features describing the inputs. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal pre-processing. It performs gradient-based learning for pattern memorisation. CNN offers fast recognition procedure. However, according to Nebauer (1998), CNN requires large number of training set to achieve generalisation.

Support Vector Machine (SVM)

In recent years, support vector machines (SVMs) have been considered as an attractive alternative to multilayer feed-forward neural networks for data classification and regression (Casali, Costantini, Perfetti and Ricci, 2006). SVM has been found to be very robust in many applications, including optical character recognition (OCR), text categorisation, and face detection in images. SVM can be categorised as a type of kernel methods (Tsang, Kwok and Zurada, 2006). In classification technique, SVM implements optimal separating hyperplane for classification. SVM finds the best separating (maximal margin) hyperplane between the two (binary) classes of training samples in the feature space. Figure 2.4, adapted from Casali et al (2006), illustrates the use of hyperplane in SVM classification.

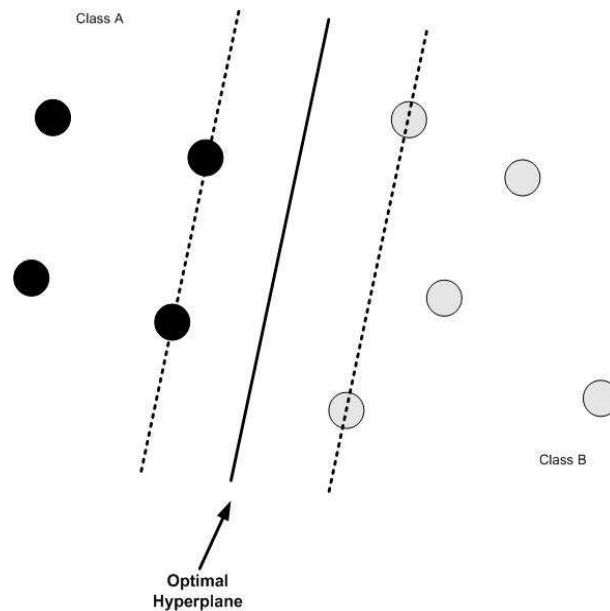


Figure 2.4: SVM classification using optimal hyperplane for separable classes. The SVs lie on the dotted lines.

SVM has been originally used for binary problems (Mavroforakis and Theodoridis, 2006). Some of the advantages of SVM in solving binary problems include the fact that SVM offers unique solution. In addition SVM provides good generalisation properties of solution. It also composed of sound theoretical foundation, based on learning theory and optimization theory.

Some of the limitations of SVM have been mentioned in other research (Huang, Mao, Siew and Huang, 2005; Dong, Krzyzak and Suen, 2005). These include the slow test phase as compared to other learning methods. In addition, the computational costs in

SVM could become very expensive, given the number of support vectors (SVs) becomes large. On the scalability issue, the training kernel matrix used in SVM grows quadratically with the size of the data set. Therefore, with an increase in data set, the training kernel matrix will also be increased significantly. This scalability issue has also been mentioned in (Nguyen and Ho, 2006; Fei and Liu, 2006). The initiatives to multi-class classification using SVM has also been implemented widely (Mavroforakis and Theodoridis, 2006; Fei and Liu, 2006).

Morphological Associative Memory (MAM)

Morphological Associative Memory (MAM) is a neural network algorithm which derived from the theory of image algebra (Ritter and Sussner, 1996). MAM is a type of artificial neural networks that follows the similar operations as existing neural network approaches. The main difference is in the determination of activation value of each neuron. In MAM, the multiplication and addition operations in neural network are replaced by the maximum-minimum and addition operations (Ritter, Sussner and Diaz-de Leon, 1998). This provides non-linearity towards the determination of activation value for each of the neurons, and hence, offers possibility for it to be used in non-linear applications.

MAM has been applied in a number of pattern recognition applications. These include works by Sussner and Valle (2006), Akyama and Kikuti (2001), and Hattori, Fukui and Ito (2002). MAM implementation in pattern recognition offers several advantages, as compared to other neural network approaches. MAM offers a one-shot pattern convergence and recall. Therefore, it provides fast recognition time. In addition, MAM provides perfect recall condition, even with an increase in the number of patterns stored. It also gives perfect recalls for noisy and distorted patterns.

The problems involved with MAM implementation in pattern recognition include the incapability to handle erosive noise with maximum operation and incapability to handle dilative noise with minimum operation. This leads to the problem of handling patterns having both erosive and dilative noises. To solve this problem, MAM uses kernel patterns (Ritter et al., 1998) that comprise unique patterns that reflect the individual patterns being stored. The use of kernel patterns does solve the dilative and erosive noises issue. However, the problem lies in deciding the kernel pattern to be used. This indirectly hinders

the scalability of MAM in pattern recognition applications. Furthermore, MAM's accuracy in pattern recognition is highly affected by its memory size (Sussner and Valle, 2006).

Hamming Associative Memory

Hamming associative memory (Hassoun and Watta, 1996) is a neural network algorithm with simple content-addressable memory approach. Its implementation involves the use of Hamming distance function to find the nearest pattern matches within a pattern recognition process. Hamming associative memory has a high error correction capability, however, problems lie within its slow retrieval speed and impractical hardware implementation (Ikeda, Watta, Artiklar and Hassoun, 2001).

The design of Hamming associative memory involves a series of procedures to obtain the nearest input-output matches. The following procedures describe the Hamming associative memory implementation in pattern retrieval:

- i. Given an input memory key $p \in \{0, 1\}^N$, stored pattern vector $\{p^1, p^2, \dots, p^m\}$ and test pattern p^t , find a Hamming distance $d(p^i, p^t)$, where $i = \{1, m\}$.
- ii. Compute Hamming distances $d_k = d(p^i, p^k)$, where $k = \{1, m\}$.
- iii. Find the minimum distance $d_{min} = \min(d_1, d_2, \dots, d_m)$.
- iv. Output the fundamental memory $p^t = p^k$, where $d_k = d_{min}$.

Hamming associative memory has also been used for high-performance associative memory approach in the works of Ikeda et al. (2001) and Mu, Watta and Hassoun (2007) in decoupled Hamming memory.

The reviews of existing approaches in neural network/machine learning for pattern recognition have indicated issues related with scalability of the existing schemes. These include requirements for large-training data sets, complex learning functions, and limited memory capacity for accurate recognition. Machine learning approaches in specific, is hardly scalable due to its complex function or kernel, and its architecture that follows single-processing approach (also known as CPU-centric approach). Unlike neural network approaches that have a capability for decentralised/distributed processing (through collaboration of neurons). Nevertheless, existing neural network implementations are still unable to gain this capability, due to their complexity and highly iterative procedures.

The following section provides a description of a scalability evaluation for some of neural network approaches as described in this section.

2.2 Scalability Evaluation for Neural Network Approaches

Scalability in general, could be achieved using a distributed approach. Therefore, the scalability factors for the pattern recognition schemes could also be derived from the scalability requirements for any distributed systems. In this analysis, there are two important factors that have been determined. These are storage capacity and communication frequency for neural network implementation. These two factors have been proposed based on the scalability requirements for distributed systems as been explained by Srinivas and Janakiram (2005). The following subsections will further discussed these two factors in relation to the neural network approaches that have been described in the previous section.

2.2.1 Storage Capacity Analysis

The baseline evaluation for storage capacity is based on the effect of an increase in the number of patterns stored within a given network. In achieving this, the memory capacity of each processing node for pattern storage is analysed. The importance of this memory capacity in recognition approach lies in its ability to provide scalable storage medium for large-scale patterns. The analysis will consider the effect of the pattern quantity against the size of the memory required per node, within a given neural network.

Existing neural networks rely largely on the weight calculations in their recognition processes. In this context, each processing node would have a collection of weight-input values stored within its memory. In a simplest form, for a given P different pattern, the size of the memory M could be determined using the following equation 2.1:

$$M = \sum_{n=1}^p w_n i_n \quad (2.1)$$

Where w represents the correlated weight and i represents the input value for each stored pattern n of P . This kind of memory consumption effect occurred in different neural network schemes, including feed-forward neural network, Hopfield network, Radial Basis Function Neural Network (RBFNN), Morphological Associative Memory (MAM), and Hamming Associative Memory. In the case of Hopfield network, its accuracy will

significantly deteriorate if the number of patterns stored is greater than $0.138N$, where N represents the number of nodes used.

Kohonen SOM's memory representation is different from the other neural networks being discussed. For each node in the SOM lattice, pattern is represented using vector-weight representation. In this context, each node will store a set of weights for a particular pattern vector. Thus, for a pattern vector with dimensions d , hence there will also be equivalent number of w weight values, where $w = d$.

2.2.2 Communication Frequency Evaluation

Communication frequency in neural network implementation is specifically related to the number of communications (in the form of messages or signals) that is projected by a single node (or neuron) towards other nodes within a given network. In this chapter, a preliminary analysis is carried out to determine the frequency of messages or signals exchange within a network using different neural network approaches. High communication frequency would lead to network congestion in actual implementation, and thus limiting the scalability of the recognition implementation. Therefore, for a scalable network, it is important that the communication frequency is kept to minimum.

Communications between nodes in existing neural networks such as feed-forward, Hopfield, and RBF neural networks are highly-iterative in nature. This is due to their common weight adjustment/ feedback methods in generating an optimum result during recognition processes for a single pattern/ pattern vector. In addition, each node's communication frequency relies on the number of nodes per layer within multi-layer networks. Given a multi-layer network with n nodes per layer, for each pattern, the number of messages/signals communicated C , by each node could be derived using the following equation 2.2:

$$C = nw \tag{2.2}$$

Where w is the number of iterations for weight adjustment. In this perspective, an increase in the size of the network and the weight adjustment iteration would lead to high number of signals being projected. Therefore, this would lead to inefficiency in providing a scalable scheme for recognition purposes. Figure 2.5 illustrates this phenomenon.

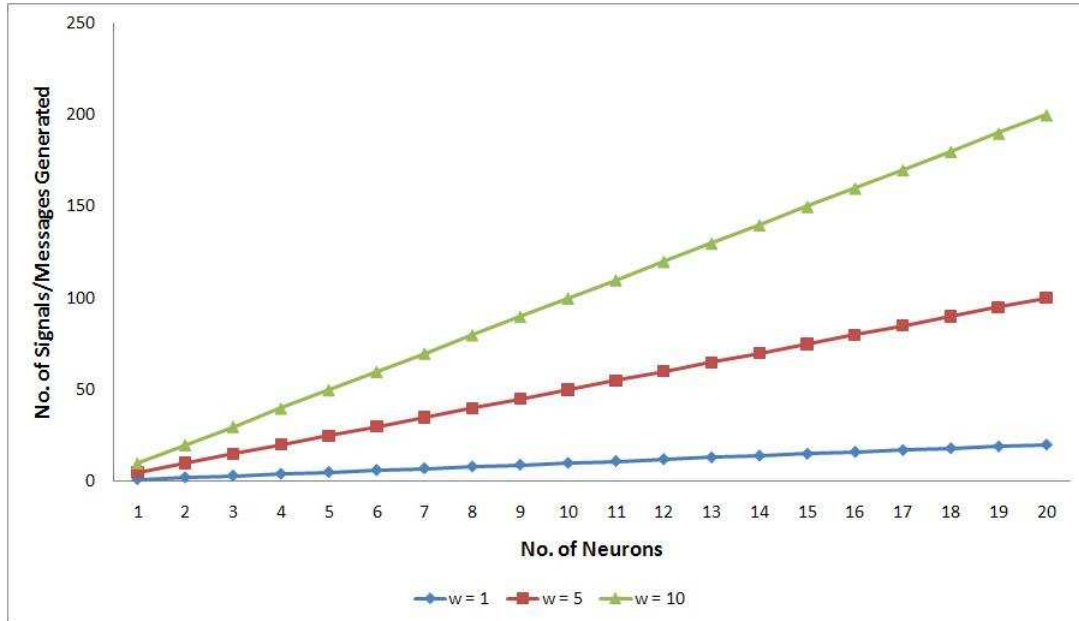


Figure 2.5: Estimated number of signals/messages generated, C by each neuron within a single layer in common neural network schemes involving different number of iterations.

Some of the associative memory (AM) schemes for pattern recognition, including morphological and Hamming associative memories offers one-shot learning procedure. This kind of procedure reduces a need for iterative process in deriving optimum recognition results. Furthermore, this kind of neural network performs lattice-based operations in which communications between each node is kept to minimum, while operations are being done in singular manner, i.e. no collaboration between nodes involved. This effect is also experienced within a Kohonen SOM network.

Existing neural network schemes mostly fail to scale up due to their complex nature and iterative learning procedures. Furthermore, the training-validation-test mechanism introduced produces significant delays in execution. It also creates strong dependency between training and test data. Hence, there is a need to consider an algorithm that limits its complexity and training-test data dependency.

Graph Neuron (GN) is a graph-based associative memory algorithm (Khan and Mihailescu, 2004). GN is highly scalable and implements single-cycle learning for pattern recognition (Khan, Isreb and Spindler, 2004; Nasution, Khan and Kendall, 2005). Furthermore, GN adopts an in-network processing approach in which the computational processes occur within the body of the network itself. GN has been proposed for several pattern recognition implementations (Nasution et al., 2005; Baqer et al., 2005; Baig et al., 2006).

This research will further investigate this algorithm and uncover its potential towards a scalable approach for pattern recognition. In this chapter, this algorithm and its derivatives will be explored in detail. In addition, the capability of GN-based algorithm for distributed pattern recognition will also be discussed.

2.3 Key Components for Scalable Pattern Recognition

From the reviews of the algorithms in Section 2.2 and analysis that has been carried out in Section 2.3, several key components for a scalable pattern recognition scheme are proposed. These include simple learning mechanism, distributed processing approach, and single-cycle training procedure. These three components affect the scalability of pattern recognition algorithm. For instance, single-processing or CPU-centric approach limits the capability of the algorithm to process large number of patterns in minimum time allocation. The following subsections explore these three key factors for scalable pattern recognition scheme.

2.3.1 Learning Mechanism

In pattern recognition, learning approaches play an important role in determining the efficiency and accuracy of pattern store and recall operations. Prominent approaches include Hebbian learning (Hebb, 1988), incremental learning (Schlimmer and Granger, 1986), and one-shot learning. Hebbian learning is a classical learning technique, based on the synaptic plasticity concept, where in output of a neuron has a significant impact on other neurons' inputs. Hebbian learning is a well-known technique for spatiotemporal pattern recognition within auto-associative neural networks. However, Hebbian learning can lead to saturation and “catastrophic forgetting”, which makes this learning technique less scalable. Most of the existing neural network algorithms implement Hebbian learning, including Hopfield, and feed-forward neural networks. A simple form of Hebbian learning follows the rule:

$$w_{ab} = x_a x_b \tag{2.3}$$

Where w_{ab} represents the weight connecting neuron b to a . x_a and x_b represent both the input of neuron a and postsynaptic response of neuron b .

Incremental learning on the other hand, has been developed to solve the scalability issue in pattern recognition (Song, Liu, Zhang and Yang, 2008). It simplifies the problem of large training sets, specifically in machine learning algorithms such as Support Vector Machine (SVM). In incremental learning, training data is divided into several subsets. Each data subset undergoes training phase at any one particular time. The results from each of these training sessions are subsequently combined to form the actual results. This approach increases the scalability of the algorithm, when large numbers of training patterns are being used. However, it still imposes problems when dealing with large-scale patterns. Higher computational resources are required to process larger patterns. Furthermore this approach tends to be tightly-coupled and requires costly kernel function computations (Schlimmer and Granger, 1986).

One-shot learning is a type of learning that has been developed to offer a capability for a system to learn information with initial minimum amount of data required. Existing implementations of this learning mechanism have been introduced using probabilistic approach such as Bayesian classifier (Fei-Fei, Fergus and Perona, 2006; Miller, Matsakis and Viola, 2000). Categories of object could be learned from a small dataset. The one-shot learning will then learn from the information obtained from these categories. This one-shot learning approach is somehow emulating incremental learning, in the sense that learning process continues with introduction of new patterns. Graph Neuron (GN) (Khan and Mihailescu, 2004) also implements one-shot learning with a conceptually different perspective. GN implements its learning algorithm through neuron-adjacency comparison approach.

2.3.2 Processing Approach

In the previous section, several distribution approaches in pattern recognition have been discussed. Distribution of input space within pattern recognition algorithm enables improvement of processing speed. Current trends in recognition approach indicate a move towards parallel processing, in which recognition processes are carried out in parallel for different datasets.

Existing neural network recognition schemes including Hopfield network (Hopfield and Tank, 1985), Back-propagation neural network (BPNN) (Wythoff, 1993), convolutional neural network (LeCun and Bengio, 1995), and fuzzy neural network (Kasabov, 1996)

are iterative in nature and therefore time and resource intensive. These factors limit the capability of the existing recognition scheme to scale up with an increase in the quantity and size of the patterns stored. Furthermore, existing neural network schemes are tightly-coupled and have been purely developed for single-processor environment. It has been proven through numerous analyses that parallel processing offers higher speed-up in execution of processes. This follows the Amdahl's law, in which stated that parallel processing could achieve maximum speed, given higher fraction of tasks that could be parallelized. Figure 2.6 shows the estimated increase in speed of processes with different parallel portions. Note that as the parallel fraction of the tasks increases, it will also increase the speed of the processes.

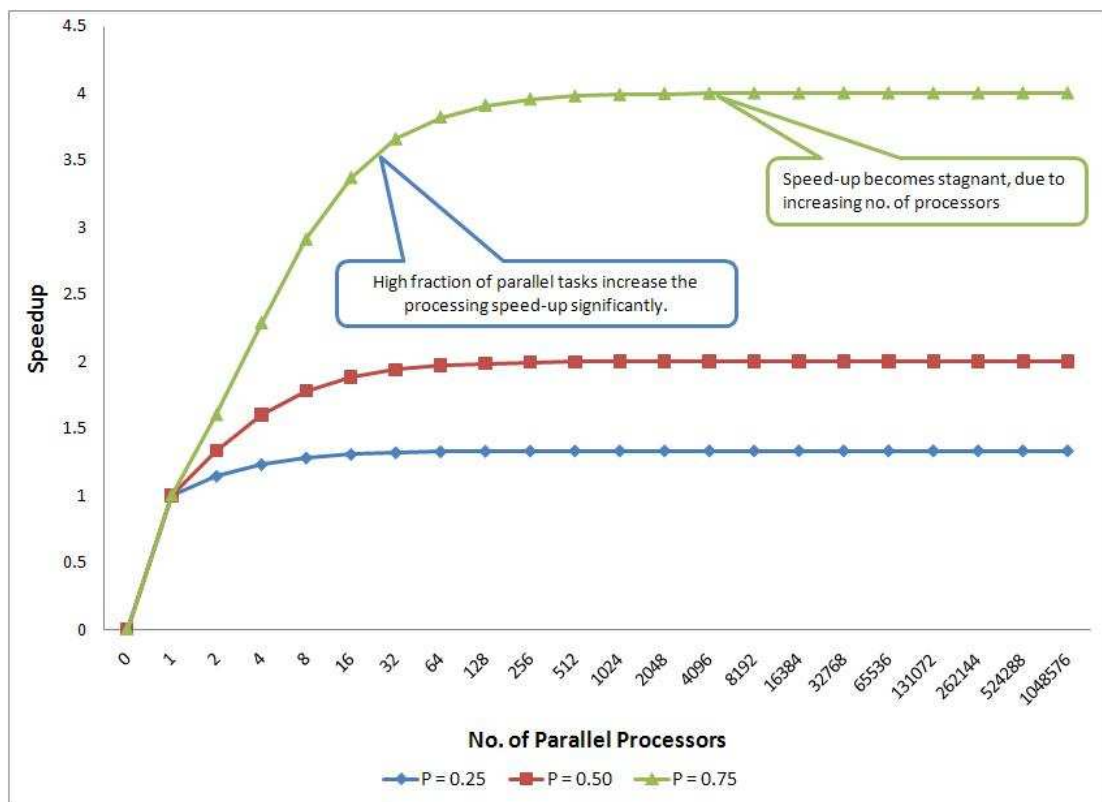


Figure 2.6: Comparison of processing speed-up estimation between recognition processes on different parallel fractions (P) with an increase in the number of parallel processors used.

The performance of parallel and distributed processing outplays the single-processing approach in the sense that it provides a fast processing mechanism. Nevertheless, in pattern recognition, to obtain an embarrassingly-parallel approach is hard to achieve, due to the nature of the recognition algorithms being deployed.

2.3.3 Training Procedure

Training in recognition context, is a process of building up the algorithm for actual recognition process. It allows the algorithm to learn from a sample dataset, before the actual recognition takes place. Training can be achieved from small or large training datasets, depending on the requirements of the recognition algorithm. In addition, training could be performed in a multi-cycle or single-cycle manner.

Existing deterministic pattern recognition algorithms usually require large training datasets for generalisation purposes. In this view, the training dataset should have all the characteristics of the actual data. However, this is not usually the case. Furthermore, current approaches involve multi-cycle training, due to the nature of learning mechanism being discussed earlier.

Single-cycle training in learning has been introduced by Khan [65] in Graph Neuron (GN) implementation. The learning involves recognising adjacency values between neurons, rather than revising weight values between nodes as in the Hebbian and Incremental learning approaches. The training in GN is conducted within a single-cycle, allowing faster recognition process.

2.4 Pattern Distribution Techniques

Implementations of existing neural network/machine learning approaches for pattern recognition have shown some limitations, as discussed in Section 2.1. These include generalisation problem and complex learning mechanism as demonstrated in RBFNN implementation, extensive iterations and slow processing speed such as in SOM and Hamming associative memory, tightly-coupled functions in MAM, and computationally intensive procedures in almost all the algorithms that have been discussed earlier. These limitations unavoidably affect the scalability of those approaches for real-time and large-scale recognition deployments. Furthermore, existing approaches have been developed with purely CPU-centric consideration, i.e. with the concept of single processing mechanism in mind. According to Ikeda et al. (2001), it is difficult for current neural network approaches to implement actual associative memory principle, in which simple low-cost devices could be equipped with these algorithms for pattern recognition purposes.

In solving the scalability issue within pattern recognition applications, we intend to shift the recognition paradigm from being a sequential-based CPU-centric towards a parallel in-network approach. In-network processing paradigm concentrates on the delegation and distribution of processes over the body of a network, rather than utilising single-processing device or node. An important aspect in distributed approach for pattern recognition is the ability for the system to distribute data across a number of processors or nodes in the network. In this perspective, it is essential that pattern distribution technique be applied. In this thesis, two different pattern distribution techniques are described:

- i. Subpattern distribution - Each pattern is partitioned into subpatterns for recognition over the entire network. Each node within the network receives a chunk of subpattern for processing.
- ii. Set distribution - Distribution of pattern set that contains a number of patterns for recognition. Each pattern set will be executed by a specific processing node within the network.

Figure 2.7 shows a comparison between both techniques. These techniques will also be discussed in the following subsections.

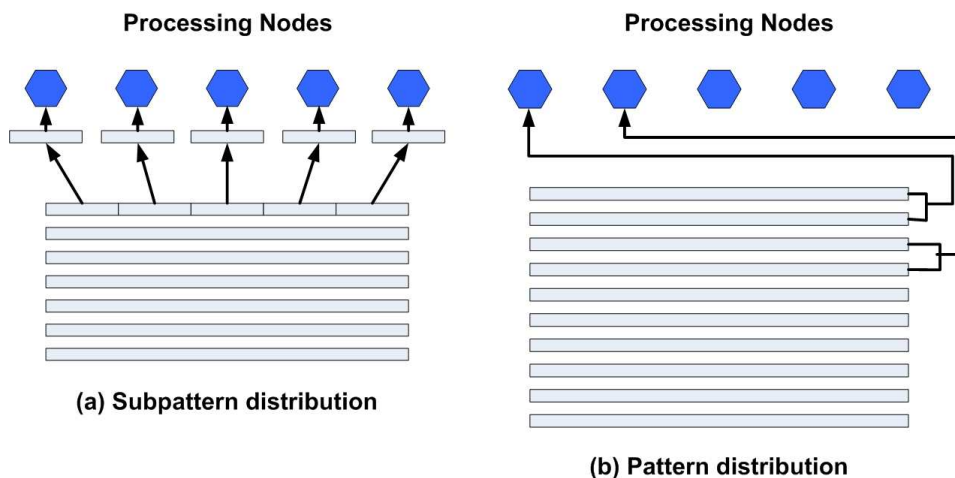


Figure 2.7: A comparison between subpattern and pattern subset distribution techniques.

2.4.1 Subpattern Distribution Technique

Subpattern distribution technique in distributed pattern recognition approach involves the process of dividing pattern into small-scale subpatterns. These subpatterns in return will be distributed across several processing nodes for recognition process. Examples of such implementation include the work of Garai and Chaudhuri (2007) in Distributed Hierarchical Genetic Algorithm (DHGA) for efficient optimisation and pattern matching. In this work, the entire search space is divided into subspaces and the search process is conducted at this level, allowing parallel genetic algorithm implementation on each subspace.

Ikeda et al. (2001) also have proposed a distributed approach for Hamming associative memory through its decoupled Hamming AM approach. Their work involves the process of partitioning input vector into a number of modules known as windows. Each window will be used in Hamming memory operation. The results of recognition obtained from each Hamming memory will be sent to a decision network for final output from the system. The decoupled Hamming AM approach was later being extended by Mu et al. (2007). This involved an introduction of voting mechanism for decision-making process.

Apart from division of input space into subspace, pattern distribution technique also includes a recognition process based on atomic pattern component that make up the entire pattern representation. For instance, Khan and Mihailescu (2004) have proposed parallel pattern recognition within wireless sensor network (WSN) environment using a Graph Neuron (GN) approach. In this work, each sensory data obtained from sensor node is considered as a component of the entire pattern represented by the network. Further works on this recognition approach has also been conducted by Baig et al. (2006).

Subpattern distribution technique allows recognition process to be performed in minimally, i.e. low complexity with regards to the size of the subpattern. Nevertheless, this technique is impossible to deploy in all deterministic approaches being discussed. Some algorithms are highly-cohesive in the sense that the whole input space must be included in its computations in order to obtain an optimum result.

2.4.2 Set Distribution Technique

Set distribution is a common approach in distributed pattern recognition. It involves the distribution of separate input datasets to each of the processing entities within the network.

Distribution of patterns is also being conducted at pre-processing stage within the classification/recognition process. For instance, Kokiopoulou and Frossard (2006) have proposed a distributed support vector machine (SVM) approach for classification of images within a sensor network. Their work involves the extraction of input signal into different feature subspaces. These feature subspaces will then be pre-processed and sent into final module that conducts actual classification process. This approach alleviates the need for large training datasets for SVM.

Some set distribution techniques also apply distribution of patterns to processing entities within a network. For instance, Lobo, Bandeira and Moura-Pires (1998) propose the use of distributed SOM for ship recognition process through acoustic signatures. This kind of technique requires global collection of results from each processing entity and further intensive post-processing mechanism. Similarly, Guoqing et al. (1992) also introduce multilayer distributed pattern recognition scheme using sparse RAM nets.

Set distribution technique does not minimise the computational complexity of the recognition algorithm. However, it reduces the execution time and allows parallel processing on patterns to be implemented. This technique is suitable for recognition schemes involving large number of patterns to be analysed. However, it does not fit well into system that caters for high-dimensional and large-scale data, such as high-intensity magnetic images including Magnetic Resonance Imaging (MRI) images.

Existing distributed pattern recognition approaches tend to employ the set distribution technique in their implementations. The main reason for this is that it alleviates the need for large number of training datasets, and consequently leads to fast learning speed. Nevertheless, complexity issue related to these approaches remain unsolved. Examples of distributed pattern recognition schemes using these approaches include (Kumar et al., 1994; Kokiopoulou and Frossard, 2006; Yang, Jafari, Kuryloski, Iyengar, Sastry and Bajcsy, 2007)

2.5 Graph Neuron for Scalable Recognition Scheme

Graph Neuron (GN) is a pattern recognition algorithm that implements a simple associative memory (AM) architecture, which provides a capability to recall patterns using similar or incomplete patterns. Associative memory architecture differs from conventional memory architecture in the sense that the store and recall operations on memory contents are based on the association with input value, rather than based on the address of the memory content. Hence, associative memory based pattern recognition algorithms are able to offer high recognition accuracy as compared to other algorithms which implement recognition using conventional memory architecture. Besides GN, other associative memory algorithms include Hopfield network, Kernel Associative Memory (KAM), Morphological Associative Memory (MAM) and Hamming Associative Memory.

In addition to its associative memory architecture, GN also follows some characteristics of graph-based pattern recognition algorithms as demonstrated in (Wilson, Hancock and Luo, 2005; Auwatanamongkol, 2007; Albiol, Monzo, Martin, Sastre and Albiol, 2008; Caetano, McAuley, Cheng, Le and Smola, 2009). However, GN implements in-network processing that solves the scalability issue (computationally prohibitive against an increase in the size and database of patterns) in other graph-based pattern recognition algorithms as described in (Garey and Johnson, 1990). According to Nasution (2007), the ability of GN to conduct its processes within a body of network offers a two-fold advantage. A) It eliminates the computational problems relating to large patterns and pattern databases. B) Its implementation is ideal for resource-constrained environment, such as in the event detection application within wireless sensor networks (WSNs).

In the following subsection, an overview of graph-based algorithms for pattern recognition, in which some of their characteristics were inherited by GN, will be presented.

2.5.1 Graph-based Pattern Recognition

A graph is composed of a set of vertices and edges. Hence, Graph G could be represented in the form of $G = (V, E)$, where V is the set of vertices (also commonly known as nodes or points) and E represents the edges (also known as lines or arcs), where $E \subset V \times V$. An edge $e \in E$ is said to connect two vertices $x, y \in V$ in the form of $e = (x, y)$. Graph vertices and edges can contain one or more pieces of information. If only a single piece of

information is available, the graph is known as a labelled graph, whereas for graph with more information on either vertices or edges, it is called an attributed graph. Figure 2.8 shows an example of labelled graphs.

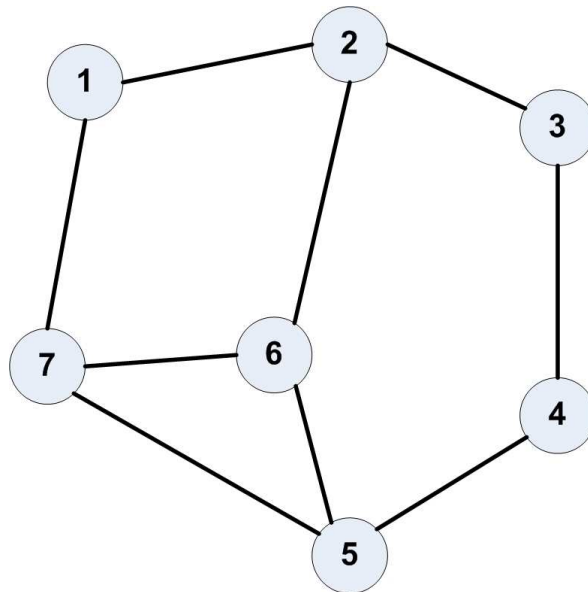


Figure 2.8: A labelled graph with a vertex set $V = \{1, 2, 3, 4, 5, 6, 7\}$ and edge set $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 7\}, \{2, 6\}, \{5, 7\}, \{1, 7\}\}$.

In graph-based pattern recognition, each pattern could be represented as a graph, as shown in Figure 2.8. Given two patterns P_{store} and P_{input} for modelled (or stored) pattern and input pattern respectively, each of these patterns could be represented by graphs G_{store} and G_{input} . Pattern recognition involving graph representation follows the graph matching problem as described in (Bengoetxea, 2003): Given $G_{store} = (V_s, E_s)$ and $G_{input} = (V_i, E_i)$, with $|V_s| = |V_i|$, there is a need to find a one-to-one mapping $f : V_i \rightarrow V_s$, such that $(x, y) \in E_i \iff (f(x), f(y)) \in E_s$. This mapping function is known as isomorphism and G_{input} is said to be isomorphic to G_{store} , and this kind of problems is known as exact graph matching. On the other hand, inexact graph matching follows the requirement in which isomorphism may not be occurring between two or more graphs, due to different number of vertices or edges, or having different sets of attributes.

In computer vision field, graphs could be used to represent images for recognition purposes. In graph-based image recognition, regions of image could be represented by vertices while edges are used to signify relationships between these regions. An important issue related to graph-based pattern recognition is such that an increase in either the size or quantity of the pattern stored will significantly affect the complexity of the algorithm.

According to Caetano et al. (2009), the number of possible matches between two graphs grows factorially with their size. Hence, scalability is an important issue to be solved.

Graph Neuron eliminates the scalability issue experienced by other graph-based pattern recognition algorithms through its in-network processing capability. GN scales up appropriately with an increase in both pattern size and database. This is achieved through distribution of recognition processes into a set of processing nodes that perform these processes in parallel. GN also capable of performing exact pattern matching as well as inexact pattern matching with different sets of attributes.

2.5.2 GN Architecture and Pattern Representation

GN network is built on the composition of interconnected processing nodes known as Graph Neuron (GN) that follows the size and dimension of given pattern. In its simplest form, GN network forms a two-dimensional array of GNs. Each GN is labelled with their own value and position (in terms of column and row positions). Figure 2.9 shows GN networks with 2-dimensional array formation. GN network receives an input, an able to store or process the input, following the instruction received. The emphasis of GN is such that it would be able to carry out parallel in-network processing as compared to the other recognition algorithms that mainly implementing CPU-sequential processing in their approaches. This allows the GN to perform fast recognition regardless of the size of input patterns. Furthermore, with this approach, the GN provides high storage capacity by disseminating patterns into pattern elements and distributes them across the network. According to Nasution (2007), GN algorithm is developed based on the hypothesis that a better associative memory resource can be created by changing the emphasis from high-speed sequential CPU processing to parallel network centric processing.

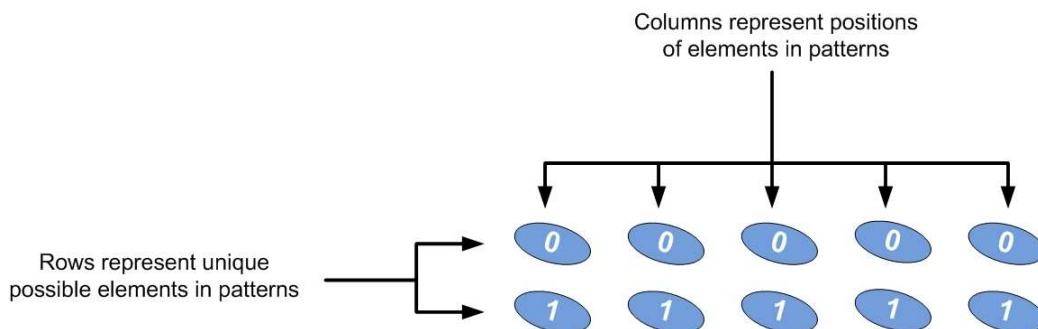


Figure 2.9: A two-dimensional GN network for binary pattern with 5-bit size.

GN pattern representation simply follows the representation of patterns in other graph-matching based algorithms as described in the previous subsection. Each GN in the network holds a *(value, position)* pair information of elements that constitutes the pattern. In correspondence towards graph-based structure, each GN acts as a vertex that holds pattern element information (in the form of value or identification (ID)) while the adjacency communication between two or more GNs is represented by the edge of a graph. Message communications in GN network are restricted only to the adjacent nodes (of the array), hence there is no increase in the communication overheads with corresponding increases in the number of nodes in the network as described in Khan et al. (2004). Figure 2.10 shows a 2-dimensional GN graph-based structure for a given input pattern. Note that only GN neurons with matched pattern element and position that will be activated and performed communication with its adjacent neurons. This self-organisation creates links between neurons and eventually built up pattern information within the network.

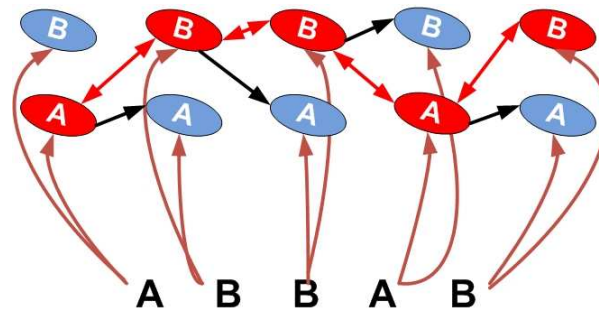


Figure 2.10: GN network activation from input pattern “ABBAB”.

Each GN within the same row holds similar ID or value, while differ in terms of their position (column position) within the network as shown in Figure 2.10. The purpose of having such value assignment is to uniquely mark the position of GN with its column number in the network. Nevertheless, this kind of arrangement is just one of the possible structural GN arrangements within the network. Further discussion on GN arrangement will be discussed in later chapter. Each GN within the network must be able to obtain the information on the size of the network, in order to determine other GNs’ positions. This is important, in order to identify its adjacent GNs for recognition processes.

An input pattern in GN network could be defined as a signal spike, or stimulus resulted from either user activation or information derived from executable program or sensory devices. In addition, it may also represent bit elements of an image (Khan, Muhamad Amin

and Raja Mahmood, 2010b) or a stimulus/signal spike produced within a network intrusion detection application (Baig et al., 2006). Each GN is capable of identifying value that corresponds to its ID from the pattern that has been introduced. For instance, in Figure 2.10, GN that holds the value 'B' will only respond to the pattern signal with element 'B' on the same position. In this context, GN network must be able to obtain prior knowledge of the pattern, in order to generate the network that matches the criteria of the patterns that will be used. This kind of network is considered to be a supervised GN network.

GN recognition process involves the memorisation of adjacency information obtained from the edges of the graph. Adjacency information for each GN is represented using the (*left, right*) formation. Each activated GN therefore records the information retrieved from its adjacent left or right nodes. In the GN terminology, this adjacency information is known as bias entry where each GN maintains an array of such entries. The entries for the entire stored pattern are collectively stored in the bias arrays. Each GN would hold a single bias array containing all the bias entries obtained in the recognition processes. In this context, GN offers low storage complexity in recognition process since each GN is only required to store a single array. Furthermore, each GN's bias array only stores the unique adjacency information derived from the input patterns.

In graph-matching representation, pattern recognition involving GN network implements graph comparison approach by treating each pattern as a graph with each element within a pattern as a vertex and the position between elements as an edge. Consider the following example: Given two patterns P_{in} and P_{st} , pattern P_{in} is said to match pattern P_{st} where the following conditions are met:

- i. Number of vertices, V_{in} is equivalent to the number of vertices V_{st} , i.e. $|V_{in}| = |V_{st}|$.
- ii. Number of edges, E_{in} is equivalent to the number of edges E_{st} , i.e. $|E_{in}| = |E_{st}|$.
- iii. Bias entry, $b \in B_{in}$ for each vertex $v \in V_{in}$ is a subset of bias array B_{st} for each vertex $v \in V_{st}$, i.e. $b \in B_{st}$.

The pattern recognition process initially takes place in the following phases:

Pattern Input phase: An input pattern, comprising $p(\text{value}, \text{position})$ pairs, is sequentially broadcast through the network. Each node based on its pre-defined position

and value setting responds to the relevant input pair; disregarding the remainder of the pattern. From Figure 2.11, $GN(X,1)$ with a pre-defined value = 'X' and position = 1, will respond to the first letter of pattern P1, i.e. "X"YXX, input as pair $p1(X,1)$. It will ignore the rest of the message. Similarly $GN(Y,2)$ will respond to the second pair $p2(Y,2)$, $GN(X,3)$ will respond to $p3(X,3)$, and $GN(X,4)$ will respond to $p4(X,4)$. All other GNs will remain inactive during this pattern input phase.

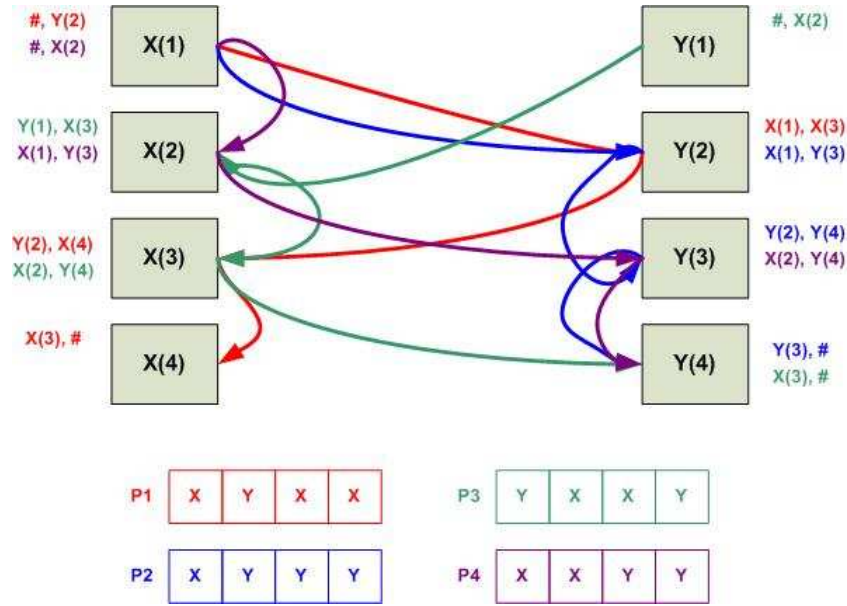


Figure 2.11: GN recognition process with bias array illustration for different input patterns.

Synchronisation phase: A broadcast signal is sent out marking the end of the incoming pattern to all the GNs.

Bias array update: During this phase, each activated GN contacts all the adjacent nodes to find out the ones which responded to the input. It can be seen from Figure 3.4. that for the input pattern P1 (XYXX), $GN(X,1)$ will update its local bias array with the entry $[GN(Y,2)]$. Similarly, $GN(Y,2)$ will update its bias array with the entry $[GN(X,1), GN(X,3)]$, $GN(X,3)$ will add $[GN(Y,2), GN(X,4)]$ to its bias array, and $GN(X,4)$ will add $[GN(X,3)]$. Thus each bias array entry records the adjacent nodes being activated within a particular pattern input phase. Thus a row of the bias array represents a part of the stored pattern. A new pair is defined, by a GN, as the one which has a different set of adjacent GNs to all existing rows of its bias array. A new pattern is found when at least one GN, within the list of activated GNs, cannot find a matching entry in its bias array. The new patterns are stored, and previously encountered patterns are recalled

in this stage. Table 2.1 shows the process where pattern “XXYX” is stored and then recalled. Note that when the pattern is being stored for the first time, the output from the GN network would be in the form of a null entry; represented by the ‘##’ pattern in the table. A null response indicates that no match has been found and that the segments of the pattern have been stored by the GN.

	Input Pattern	Output
First Input	XXYX	#### (Store)
Second Input	XXYX	XXYX (Recall)

Table 2.1: Store and recall responses of a GN Array.

Stages 1 and 2 of the GN learning phase take place in a completely parallel and decentralised manner. It may be seen from Figure 2.11 that the maximum bias array size of two occurs in $GN(Y,2)$ after the array has stored four patterns. The scalability tests, with up to 16,384 nodes, have shown that the computational complexity only increases nominally with the increases in the size of the network (Baquer et al., 2005).

In supervised GN approach, the size of the network relies on the size of the patterns and its number of unique value of elements for recognition or classification purposes. Given patterns $P = a$, an analysis involving one-dimensional GN network will require number of GN neuron, $N(a)$ as follows.

$$N(a) = s_a \times e_a \quad (2.4)$$

Where s_a represents the size, and e_a as the number of unique elements of pattern a .

Eventually, with an increase in the dimension of patterns, there will also be an increase in the number of GN neurons within the GN network. Hence, given a dimension of pattern a as d_a , the number of GN neuron could be determined as follows.

$$N(a) = s_a \times e_a \times d_a \quad (2.5)$$

The GN algorithm has initially been introduced by Khan (2002). Since then, GN has been used in a number of applications involving pattern recognition and classification. With lightweight and distributed features, GN has been applied in resource-constrained networks such as in the wireless sensor networks (WSNs). Khan and Mihailescu (2004) have proposed a GN implementation for pattern recognition within WSN network. By

simulating sensory reaction on artificial nervous system using WSN, GN has been shown to be capable of differentiating internal stress patterns within the network and patterns resulting from external loading conditions in a structural health monitoring (SHM) application. In addition, GN requires low data storage capacity and is therefore mostly suitable for WSN deployment. In another work, Baig et al. (2006) has proposed the use of GN pattern recognition algorithm for distributed denial of service (DDoS) attack detection within WSN network. GN algorithm has been able to detect DDoS attack patterns in WSN using analysis on internal traffic flow of the network. This implementation of GN has been tested on three different network topologies, and the results have shown that it produces high recognition accuracies for all topologies.

GN algorithm has also been proven to offer an energy-efficient mechanism for pattern recognition. This follows the work that has been carried out by Baqer and Khan (2007) on energy-efficient pattern recognition approach for WSN. In this work, the use of GN for event detection has been demonstrated. GN has shown to offer an energy-efficient mechanism for event detection within WSN by conducting the detection and analysis in situ, i.e. at the sensor node level. This is in contrary to existing approaches in which the analysis on the occurrences of event is conducted at the base station.

Apart from WSN deployment, GN has also been proposed for pattern recognition application within large-scale networks such as the Internet and peer-to-peer (P2P) networks. Nasution et al. (2005) have proposed a GN implementation as plan recogniser in the Trusted Transient Simple Network (TTSN) security control system architecture.

The ability of Graph Neuron (GN) algorithm to provide a fast, efficient and scalable solution for pattern recognition have made it capable to be deployed in a number of different network environment, ranging from resource-constrained networks such as WSN, to large-scale networks such as the Internet and peer-to-peer (P2P) networks. Nevertheless, GN implementation also has its own limitations, including high requirement for number of neurons in large-scale and multi-dimensional patterns, as well as inaccuracy from a phenomenon known as intersection or crosstalk problem. The first limitation is less significant, given that the structure of GN network could be abstracted in the form of either memory structure or actual processing nodes working together to form as a GN network. An important limitation of GN is the intersection problem. This problem is a result of GN's inability to obtain full information on the pattern. Rather, GN builds up pattern

information through links between adjacent neurons. In the following section, detailed discussions on this crosstalk issue in GN implementation will be presented.

2.5.3 GN Complexity Estimation

The following is the estimation of GN computational complexity in its recognition procedure. A Big-O analysis on GN procedure was performed, specifically on the bias array update phase. The justification for this selection is such that the core recognition function in GN is the bias array update within each GN neuron. In conducting this analysis, The bias array update procedure for each neuron is observed in the form of pseudocode as follows:

Algorithm 1 Bias Array Update Procedure for GN

```

1:  $\{input.l: \text{index from left-side GN, } leftGN\}$ 
2:  $\{input.r: \text{index from right-side GN, } rightGN\}$ 
3:  $\{b_{l,r}: \text{bias entry; } b_{array}: \text{bias array; } input_{l,r}: \text{combined index left and right}\}$ 
4:  $input.l \leftarrow leftGN$ 
5:  $input.r \leftarrow rightGN$ 
6: for all  $b_{l,r} \in b_{array}$  do
7:   if  $input_{l,r} \equiv b_{l,r}$  then
8:     return  $b$ 
9:   exit FOR
10: else
11:   if  $b_{l,r}$  is last entry then
12:      $b_{l,r} + 1 = input_{l,r}$ 
13:   else
14:     continue
15:   end if
16: end if
17: end for

```

Consider the bias array update as a function $f(B) = T_{f(B)}(N)$ in the previous pseudocode, it is clearly demonstrated that for each input pattern, the function implements a linear search mechanism, hence it's complexity could be derived as $O(N)$. Hence, we can deduce that $T_{f(B)}(N) = O(N)$. From this observation, it is evident that GN offers low complexity in its recognition process, by implementing a simple linear search technique for identifying recall or new patterns introduced into the network.

Another complexity estimation that has been carried out for GN implementation is storage capacity analysis. This analysis involves estimating the maximum size of bias array for each input pattern stored within the GN network. For 2-dimensional GN structure, the maximum number of bias entries is determined by the number of possible combinations

of entries (*left, right*), obtained from adjacent GNs. Hence, these possible combinations directly related to the number of rows (different number of pattern elements) within the composition. In addition, the maximum bias array size for each GN is also influenced by its position. Therefore, given number of rows n_r , there are two possible approaches to determine the maximum bias array size, B_{max} of GN:

- i. If GN is at the edge, $B_{max,e} = n_r$.
- ii. For non-edge GN, its maximum bias array size is, $B_{max,ne} = n_r^2$.

To determine the sum of maximum bias array capacity for all GNs within the network, n_{total} for patterns with size $S = a$, the following equation could be used:

$$\begin{aligned} n_{total} &= n_r (B_{max,ne} \times (a - 2) + 2B_{max,e}) \\ n_{total} &= n_r (n_r^2 (a - 2) + 2n_r) \\ n_{total} &= n_r^2 (n_r (a - 2) + 2) \end{aligned} \tag{2.6}$$

The total maximum bias array capacity of one-dimensional GN network is highly affected by the number of different elements in input patterns. However, an increase in the size of pattern moderately influenced its growth. In this context, large-scale patterns with minimum variation of elements will give lower impact on bias array capacity as compared to large-scale patterns with high variation of elements. Figure 2.12 shows the impact of both number of different pattern elements and pattern size towards the growth of total bias array size for GN networks.

It is best to note that the total maximum bias array size grows linearly with an increase in pattern size. Hence in this perspective, GN is proven to offer scalability for large-scale patterns. An increase in the dimension of the patterns would also affect the total size of bias array within the GN composition. This is due to an increase in the number of possible combination entries. For instance, in a 3-dimensional GN network, the bias entry of each GN could be in the form of (*left, right, top, bottom*), which is equivalent to number of rows n_r^4 .

2.5.4 Crosstalk Issue in GN

GN pattern recognition approach involves the process of obtaining subpattern information between two or more adjacent neurons. For instance, given a pattern “*abcdef*”, GN network

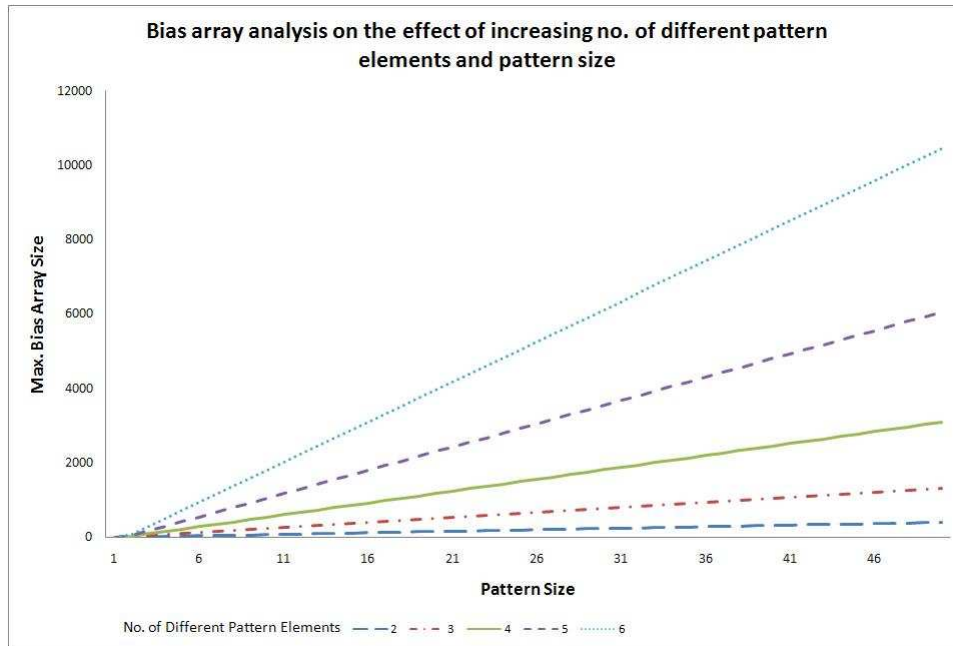


Figure 2.12: Maximum bias array analysis for GN implementation on increasing different number of pattern elements and pattern size.

will memorise this pattern in the form of subpatterns “*ab*”, “*abc*”, “*bcd*”, “*cde*”, “*def*”, and “*ef*”. Note that the number of subpattern compositions is equivalent to the number of active neurons.

GN’s limited perspective on overall pattern information would affect a significant inaccuracy in its recognition scheme. As the size of the pattern increases, it is more difficult for a GN network to obtain an overview of the pattern’s composition. This produces incomplete results, where different patterns having similar subpattern structure leads to false recall. Let us suppose that there is a GN network which can allocate 6 possible element values, e.g. u, v, w, x, y , and z , for a 5-element pattern. A pattern $uvwxy$, followed by $vwxyz$ is introduced. These two patterns would be stored by the GN array. Next, we introduce the pattern $vwxyz$, this will produce a recall. Clearly the recall is false since the last pattern does not match the previously stored patterns. The reason for this false recall is that a GN node only knows of its own value and its adjacent GN values. Hence, the input patterns in this case will be stored as segments uv, uvw, vwx, wxy, xy . The latest input pattern, though different from the two previous patterns, contain all the segments of the previously stored patterns. Figure 2.13 simplifies this example in graphical representation.

The example is extended by looking into the bias array analysis within the GN network for the example given previously. Figure 2.14 shows an illustration of bias array analysis on

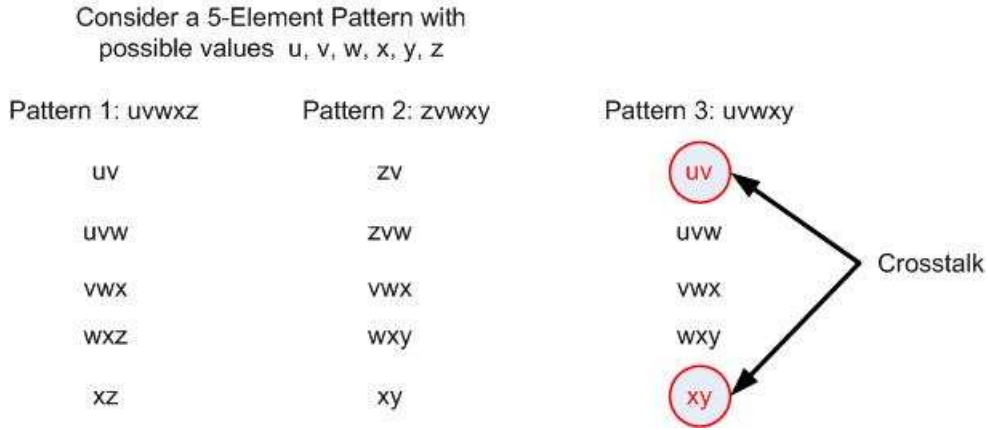


Figure 2.13: Crosstalk phenomenon illustration on input patterns in GN network.

the GN network for the crosstalk example. Note that the recall made for pattern $uvwxy$ is perceived to be true by all the activated GNs, since all the subpatterns are found. However, the actual recall is inaccurate, since the test pattern as a whole does not match the stored patterns. This phenomenon is known as intersection or crosstalk problem.

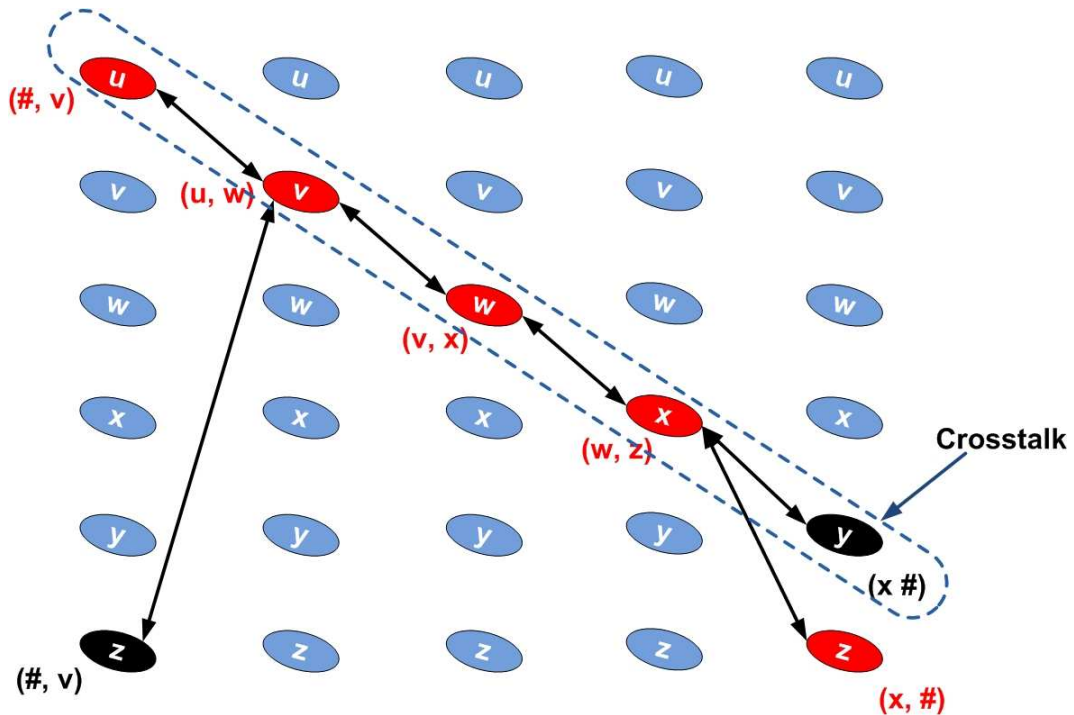


Figure 2.14: Crosstalk phenomenon in GN pattern recognition.

In this figure, the bias arrays for both patterns $uvwxyz$ and $zvwxy$ are stored (red and black respectively). With the introduction of pattern $uvwxy$, all the bias entries of these two patterns are recalled, hence creating a false recall.

In order to solve the crosstalk problem in GN pattern recognition algorithm, Nasution and Khan (2008) has proposed a hierarchical structure for GN, known as Hierarchical Graph Neuron (HGN). The underlying principle of HGN implementation is such that the capability of “perceiving neighbours” in each GN within the network must be expanded. This is achieved by having higher layers of GN neurons that oversee the entire pattern information. Hence, it will provide a bird’s eye view of the overall pattern. Figure 2.15 shows the hierarchical layout of HGN for binary pattern with size of 7 bits.

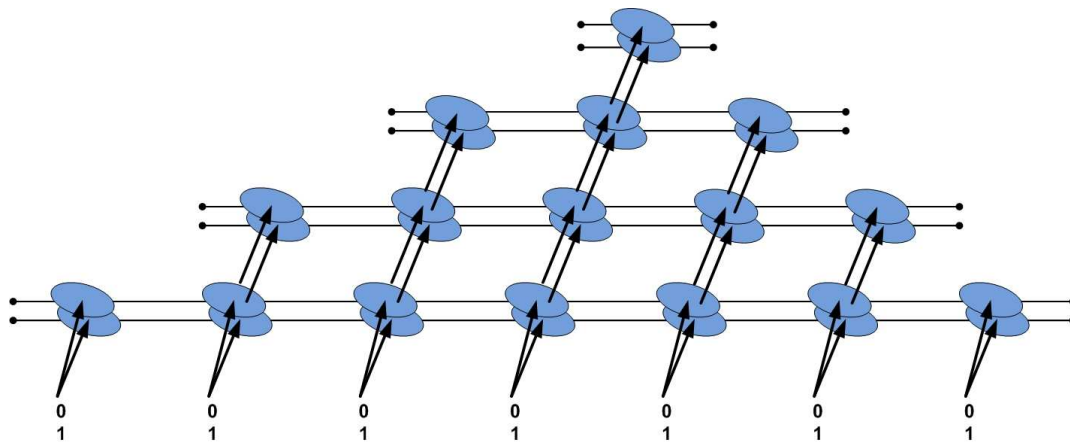


Figure 2.15: Hierarchical Graph Neuron (HGN) with binary pattern of size 7 bits.

Figure 2.15 demonstrates that HGN is composed of layers of GN networks arranged in a pyramid-like composition. This composition layout offers significant advantage for GN implementation, in the sense that it has a capability to contain all information related to the structure of the patterns stored in this network. Further discussion on HGN pattern recognition algorithm will be presented in the next section.

2.6 Hierarchical Graph Neuron (HGN)

The limitation of Graph Neuron (GN) on the crosstalk issue has brought forward the development of Hierarchical Graph Neuron (HGN). HGN extends the functionalities of GN algorithm for pattern recognition by providing a bird’s eye view of the overall pattern structure. It thus, eliminates the possibility of false recalls in the recognition process.

HGN network, as shown in Figure 2.15 is only used in pattern recognition applications involving one-dimensional patterns. However, HGN does not limit the dimensionality of patterns. HGN composition can also be expanded to two, three, or even multi-dimensional

hierarchies, for applications involving complex patterns. Figure 2.16 shows examples of HGN composition for two-dimensional pattern of size 49 (7×7) and three-dimensional pattern of size 147 ($7 \times 7 \times 3$). For simplicity, a number of pattern elements in this figure have been omitted.

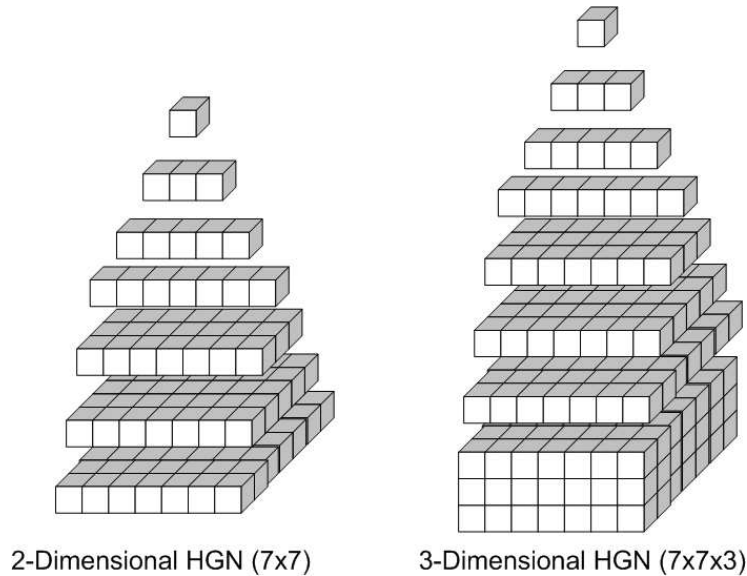


Figure 2.16: HGN composition of 2- and 3-dimension for pattern size 49 and 147.

With an increase in the dimension of HGN network, an interesting side effect has been discovered. According to Nasution (2007), by increasing the dimension of hierarchical composition in HGN, the number of GN within the hierarchy will be significantly reduced. Hence, this improves the efficiency of the network for recognition involving large-scale patterns. For example, given one-dimensional patterns of size 147, the total number of GN required is: $147 + 145 + 143 + \dots + 3 + 1 = 5476$. A two-dimensional ($21 \times 7 = 147$) GN composition requires: $21 \times 7 + 21 \times 5 + 21 \times 3 + 21 + 19 + \dots + 3 + 1 = 436$ GN. In this example, a 92% reduction in the number of GN within the composition has been made. In this context, having higher dimensional structure would lead to significant reduction in the size of the network.

As discussed in the previous section, pattern representation in GN network applies the graph-based format with *(value, position)* structure. HGN implementation also follows similar approach. In addition to this, HGN also has a requirement for the size of patterns. Patterns used in HGN recognition scheme must be in odd-size length format. This requirement is to cater for hierarchical structure of HGN network with the top neuron overseeing the overall pattern structure. In meeting this requirement, any pattern with

even-size length should add a 'dummy' value at the end of the pattern, as to form an odd-size pattern length.

2.6.1 Size of HGN Network

The HGN network is built up from a composition of GNs in a hierarchical structure. It is important to determine the size of the network, as it helps in constructing efficient composition within the network, based upon the availability and capacity of processing nodes within a physical network. In addition, as mentioned earlier, the patterns used in HGN recognition scheme should be in the form of odd-size patterns. Hence, the base layer of HGN network must also fulfil this requirement. To analyse the number of GNs required for a HGN network to conduct recognition on patterns of size, we use and extend the calculation methods described in (Nasution, 2007).

In HGN pattern recognition involving one-dimensional patterns of size $S = x$ with v different number of pattern element, the number of GN required, $N(x)$ could be derived from the following equation:

$$N(x) = vx + v(x-2) + v(x-4) + \dots + v = v \sum_{i=0}^{\left(\frac{x-1}{2}\right)} (x-2i) = v \left(\frac{x+1}{2}\right)^2 \quad (2.7)$$

For two-dimensional patterns with size dimension $S = x \times y$, the number of GN required, $N(x, y)$ is as the following:

$$\begin{aligned} N(x, y) &= xy + (x-2)y + (x-4)y + \dots + y + (y-2) + (y-4) + \dots + 3 + 1 \\ N(x, y) &= \left(\sum_{i=0}^{\left(\frac{x-1}{2}\right)} (x-2i) \right) y - y + \sum_{i=0}^{\left(\frac{y-1}{2}\right)} (y-2i) \\ N(x, y) &= \left(\left(\frac{x+1}{2}\right)^2 - 1 \right) y + \left(\frac{y+1}{2}\right)^2 \end{aligned} \quad (2.8)$$

Nevertheless, equation 2.8 does not take into account the number of different pattern element v . Therefore, we add up this value into this equation:

$$N(x, y) = v \left(\left(\left(\frac{x+1}{2} \right)^2 - 1 \right) y + \left(\frac{y+1}{2} \right)^2 \right) \quad (2.9)$$

To illustrate the effect of having higher-dimensional composition of patterns towards the number of GN required, equations 2.7 and 2.9 are plotted as a comparison between one-dimensional and two-dimensional composition for a series of binary pattern sizes. Patterns with quadratic-value sizes, i.e. $x = y$ for two-dimensional composition are used. Figure 2.17 shows the comparison.

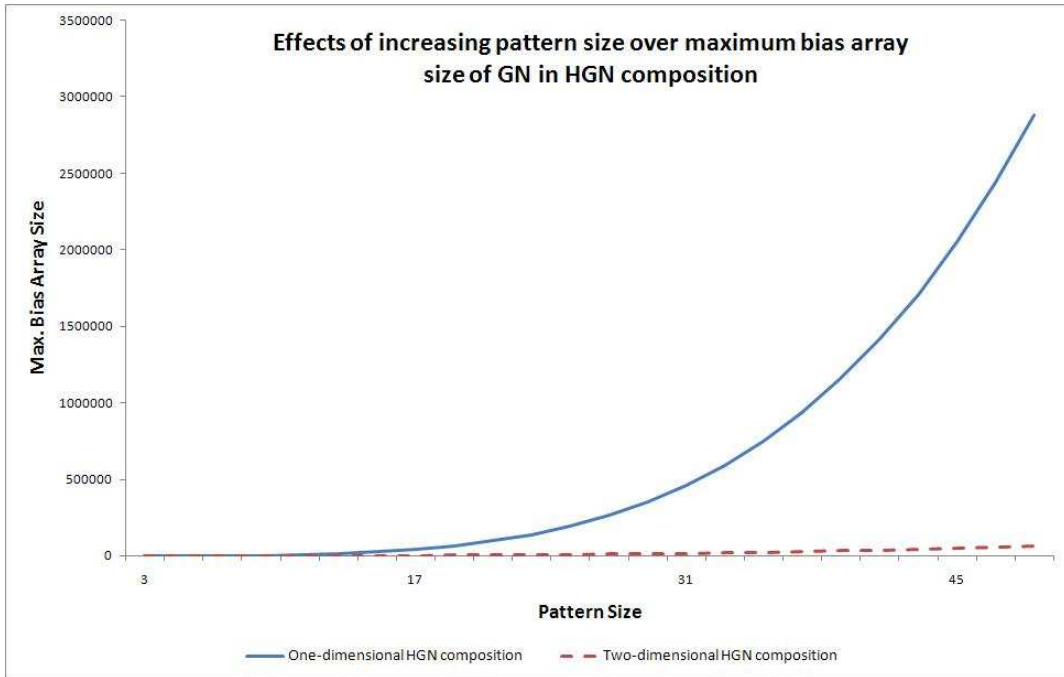


Figure 2.17: Analysis on the effects of increasing pattern size over total maximum bias array size within HGN for 1- and 2-dimensional compositions.

The graph shows significant reduction in the number of GN within a two-dimensional composition against one-dimensional structure. However, this reduction does not always guarantee that the complexity of HGN algorithm with higher-dimensional structure remain equivalent to one-dimensional composition. Further discussion on this aspect will be presented in Section 2.6.5

2.6.2 HGN Recognition Procedure

HGN pattern recognition procedure involves a number of stages that include recognition at every layer within the hierarchical structure. The communication paths within the

HGN layers are similar to the Simple GN. The HGN communications propagate from the base layer GNs to the top GN, and consequently, from the top GN to the base layer GNs.

The HGN communications occur in the following procedure. Each GN at the base layer receives an input pattern from an external entity, which we refer to as the Stimulator and Interpreter (SI) module after Nasution and Khan (2008). Each GN that receives an input is called an active GN. Active GN at the base layer would send its $p(\text{column}, \text{row})$ pair to all the adjacent GNs, acknowledging that it has been activated. The $p(\text{column}, \text{row})$ pairs make up the GN's bias array entry for the current input pattern for all GNs at the base layer. In the end, each neuron would have received two pairs from its adjacent neurons, with the exception of the neurons on the edges, which will receive a single pair. Each active GN must then calculate its bias index. If the incoming pair combination is found in its bias array, then the index of the entry would be noted. Otherwise a new index would be generated to store and reference the pattern. Each active GN would then send its index value to its corresponding higher layer GN within the same column, except for the GNs on the edges. This process continues until the top most layer has been reached. The top layer GNs decide whether the input is to be treated as a new pattern and stored or it is a previously known pattern which needs to be recalled. A new index value is propagated downwards for a stored pattern and an existing index value is propagated downwards for a recalled pattern.

In relation to the HGN recognition procedure, GN's bias array structure within the hierarchical composition also follows the bias array formation in GN network. Nevertheless, a modification has been made to cater the functionality of higher layer GNs to conduct recognition based upon the results of adjacency comparison made at lower layer GNs. The followings are bias entry conditions for GNs within any HGN network:

- i. For GNs at the base layer, their bias entry takes the form of $\{\text{left}, \text{right}\}$, where *left* and *right* represent the row number of left-adjacent and right-adjacent neuron respectively.
- ii. For GNs at the middle layer, their bias entry takes the form of $\{\text{left-index}, \text{lower-index}, \text{right-index}\}$, where *left-*, *lower-*, and *right-* indices represent indices obtained from its left, lower (within the same column), and right GNs respectively.

- iii. The bias entry structure of top layer GNs is in the form of $\{lower-index\}$, which is the index obtained from its lower layer GN (within the same column).

2.6.3 HGN Complexity Estimation

The following discussion focuses on the complexity analysis of HGN pattern recognition scheme. This chapter focuses on the bias array capacity analysis and Big-O estimation of HGN network. Similar analysis has been carried out on GN network as discussed in Section 2.5.3.

For Big-O estimation, HGN strictly follows GN recognition procedure, using adjacency comparison approach. The difference between HGN and GN implementations is on their execution process. HGN applies multiple-stage execution (based on hierarchical structure) while GN implements single-stage execution. Hence the complexity of HGN, in terms of Big-O estimation is still $O(N)$.

In regards to storage capacity analysis, this research considers the bias array capacity of each GN within the HGN composition. Detailed analysis of HGN storage capacity has been discussed in (Nasution and Khan, 2008). There is no intention to repeat the explanation in this thesis. However, a summary of the complexity estimation will be presented.

In this analysis, the size of the bias array is observed as different patterns are being stored. The number of possible pattern combinations increases exponentially with an increase in the pattern size. The impact of the pattern size on the bias array storage is an important factor in bias array complexity analysis. In this regard the analysis is conducted by segregating the bias arrays according to the layers within a particular HGN network. The following equations show the bias array size estimation for binary patterns. This bias array size is determined using the number of bias entries recorded for each GN.

At the base layer (0):

The size of bias array for base layer GN in HGN composition strictly follows the estimation that has been carried out for GN algorithm, as described in Section 2.5.3. The maximum size of GN's bias array could be derived from possible number of adjacency information combinations (from preceding and succeeding GNs. In this perspective, we consider the number of rows (number of different pattern elements) n_r , for each pattern set

used. Therefore, each non-edge GN neuron in HGN for one-dimensional patterns would accommodate maximum bias array size, $b_{ne,0}$ as the following:

$$b_{ne,0} = n_r^2 \quad (2.10)$$

On the other hand, each GN at the edge of the layer only receives adjacency information either from its preceding or succeeding GN. Hence, its maximum bias array size, $b_{e,0}$ could be determined as:

$$b_{e,0} = n_r \quad (2.11)$$

This is equivalent to the number of different pattern elements. Consequently, the total size of bias array of GNs at the base layer, $b_{total,0}$ for patterns with size $S = a$ could be derived using a similar approach as described in Section 2.5.3:

$$\begin{aligned} b_{total,0} &= n_r (b_{ne,0} \times (a - 2) + 2b_{e,0}) \\ b_{total,0} &= n_r (n_r^2 \times (a - 2) + 2n_r) \\ b_{total,0} &= n_r^2 (n_r \times (a - 2) + 2) \end{aligned} \quad (2.12)$$

At layer i :

GNs at middle layer receive indices from lower/base layer GNs and perform recognition procedure using these values. Hence, the maximum bias array size of GNs at lower/base layer affects the calculation of bias array estimation for these middle layer GNs. For non-edge GN in a middle layer i , the maximum size of its bias array could be derived as follows:

$$\begin{aligned} b_{ne,i} &= n_r^2 \times b_{ne,i-1} \\ b_{ne,i} &= n_r^2 \times n_r^{2i} \\ b_{ne,i} &= n_r^{2i+2} \end{aligned} \quad (2.13)$$

Similarly, for GNs at the edge:

$$\begin{aligned} b_{e,i} &= n_r \times b_{e,i-1} \\ b_{e,i} &= n_r \times n_r^{2i} \\ b_{e,i} &= n_r^{2i+1} \end{aligned} \quad (2.14)$$

The total size of maximum bias array for all GNs in each middle layer i , could be determined from the following equation:

$$\begin{aligned}
b_{total,i} &= n_r (b_{ne,i} (a - (2i + 2)) + 2b_{e,i}) \\
b_{total,i} &= n_r (n_r^{2i+2} (a - (2i + 2)) + 2n_r^{(2i+1)}) \\
b_{total,i} &= n_r^{2i+3} (n_r (a - (2i + 2)) + 2n_r^{(2i+2)})
\end{aligned} \tag{2.15}$$

GNs at the top layer:

At the top layer, the maximum size of the bias array could be derived from the previous level non-edge GN's maximum bias array size. Hence, the maximum size of the bias array of neuron and the sum of all possible bias arrays at the top level is:

$$\begin{aligned}
b_{top} &= n_r \times b_{ne,top-1} \\
b_{top} &= n_r \times n_r^{a-1} \\
b_{top} &= n_r^a
\end{aligned} \tag{2.16}$$

Given all the equations 2.12, 2.15, and 2.16, the sum of all bias arrays, b_{HGN} can now be constructed for all GNs in each HGN composition.

$$\begin{aligned}
b_{HGN} &= b_{total,0} + \sum_{i=1}^{\left(\frac{a+1}{2}\right)-2} b_{total,i} + b_{total,top} \\
b_{HGN} &= n_r^2 (n_r (a - 2) + 2) + \left(\sum_{i=1}^{\left(\frac{a+1}{2}\right)-2} n_r^{(2i+3)} (a - (2i + 2) + 2) + 2n_r^{(2i+2)} \right) + n_r^a
\end{aligned} \tag{2.17}$$

With regards to this bias array capacity estimation, maximum bias array size for complexity analysis of HGN has been considered. The results have shown that the size of bias array would be significantly affected, as the size of the network and the pattern size increased. However, this does not take into consideration where patterns being stored are having similar subpattern features or close resemblance to one another (i.e. not totally unique). In this context, uniform distribution could be used to estimate the average bias array size for a given pattern set being stored in the network. Hence, it can be assumed that the average size of bias array could be determined from the number of stored patterns divided by the maximum number of unique patterns (from combinations of different pattern elements for a given pattern size), i.e. $\frac{n_p}{n_r^a}$, where n_p represents the number of patterns stored in HGN composition.

2.6.4 HGN Solution to Crosstalk Problem

As already discussed in Section 2.5.4, the main problem with Graph Neuron implementation is on the intersection or crosstalk issue, due to its inability to oversee the entire pattern structure. This limitation has been overcome by HGN with its hierarchical network layout as shown in Figure 2.15. In this subsection, further analysis on this solution will be presented. The crosstalk example in Section 2.5.4 will be used for this explanation.

A one-dimensional HGN network for pattern size of 5 as shown in Figure 2.18 will be considered. The number of GN neurons required for this composition with 6 different pattern elements will be equivalent to $6 \times ((5 + 1) \div 2)^2 = 54$.

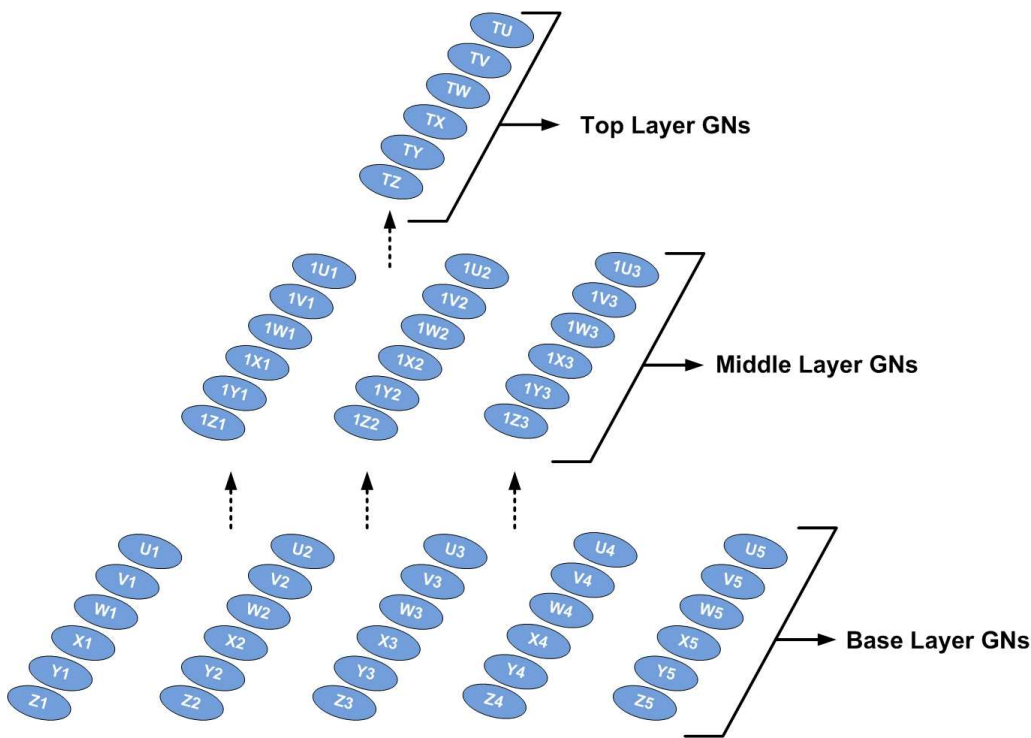


Figure 2.18: HGN composition for crosstalk example (Section 2.3.4).

When the pattern $uvwzx$ is first introduced into the HGN network, each GN with a matched *(value, position)* of elements within the pattern will be activated. Hence, GN $U1$, $V2$, $W3$, $X4$, and $Z5$ will be activated. Once the activation has been initiated, each active GN at the base layer will perform a recognition process by exchanging its value with the adjacent GNs. This activity will result in the formation of a bias array structure as shown in Table 2.2. All active GNs (with exception to GNs at the edges ($U1$ and $Z5$)) will send their bias index to corresponding higher layer GNs (in this case, $V2 \rightarrow 1V1$, $W3 \rightarrow 1W2$, and $X4 \rightarrow 1X3$). Once these layer-1 GNs receive the index, then it will

be activated. The recognition process at this level involves comparing indices from lower layer GNs obtained by adjacent GNs. Hence, the bias array contents of each GN at layer-1 are also shown in Table 2.2. This process follows by GN 1W3 sends its latest index to top layer GN, TW . At this stage, GN TW will check its bias array of any appearance of index retrieved from 1W2. If any, it will recall the index and propagate it back to all GNs within the network. Otherwise, new index will be generated and propagated to the network. Table 2.3 and 2.4 show the continuation of this process, in terms of bias arrays of all the GNs in the network, after patterns $zvwxy$ and $uvwxy$ has been introduced.

Layer	Active GN	Bias Array Entries
Base	U1	1(#, V2)
	V2	1(U1, W3)
	W3	1(V2, X4)
	X4	1(W3, Z5)
	Z5	1(X4, #)
Middle	1V1	1(#, 1, 1)
	1W2	1(1, 1, 1)
	1X3	1(1, 1, #)
Top	TW	1

Table 2.2: Bias array entries for all active GN in HGN composition when pattern $uvwzx$ is introduced.

Layer	Active GN	Bias Array Entries
Base	Z1	1(#, V2)
	V2	1(U1, W3) 2(Z1, W3)
	W3	1(V2, X4)
	X4	1(W3, Z5) 2(W3, Y5)
	Y5	1(X4, #)
Middle	1V1	1(#, 1, 1) 2(#, 2, 1)
	1W2	1(1, 1, 1) 2(2, 1, 2)
	1X3	1(1, 1, #) 2(1, 2, #)
Top	TW	1 2

Table 2.3: Bias array entries for all active GN in HGN composition when pattern $zvwxy$ is introduced.

Layer	Active GN	Bias Array Entries
Base	U1	1(#, V2)
	V2	1(U1, W3) 2(Z1, W3)
	W3	1(V2, X4)
	X4	1(W3, Z5) 2(W3, Y5)
	Y5	1(X4, #)
Middle	1V1	1(#, 1, 1) 2(#, 2, 1)
	1W2	1(1, 1, 1) 2(2, 1, 2) 3(1, 1, 2)
	1X3	1(1, 1, #) 2(1, 2, #)
Top	TW	1 2 3

Table 2.4: Bias array entries for all active GN in HGN composition when pattern $uvwxy$ is introduced.

With HGN implementation, pattern $uvwxy$ has found to be a different pattern from patterns $uvwzx$ and $zvwxy$. Hence, crosstalk issue has been solved by this hierarchical scheme. Nevertheless, with the advent of HGN pattern recognition scheme, an issue arising is the scalability of HGN for large and complex patterns. This issue will be discussed in the next subsection.

2.6.5 Scalability in HGN Approach

HGN pattern recognition scheme has the capability to perform highly-accurate analysis on patterns using in-network processing approach. This enabled collaboration of processing nodes for recognition involving large-scale patterns, instead of relying on other single-processing (or CPU-centric) recognition schemes. Nevertheless, an increase in the size of patterns leads to an overgrown network. As shown in Sections 2.6.1 and 2.6.3, as pattern size grows, it will significantly affect the number of GN required as well as the capacity of each GN's bias array.

The number of neurons generated in HGN implementation, increases quadratically with an increase in the size of the pattern. Figure 2.19 shows the graph representing a number of neurons in one-dimensional HGN recognition scheme for patterns with increasing number of different elements and pattern size.

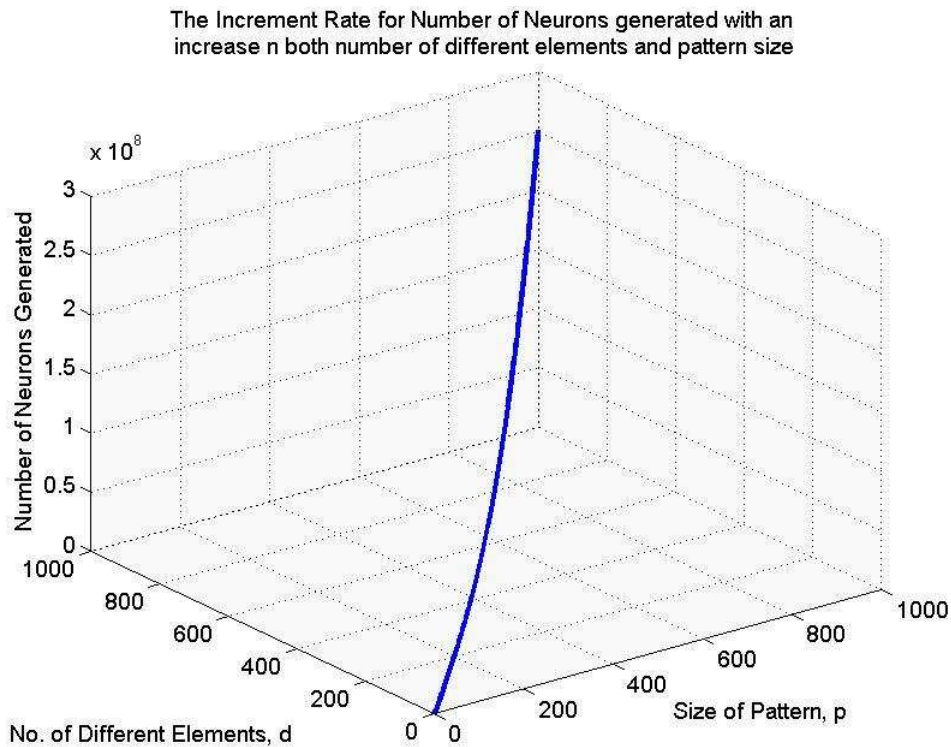


Figure 2.19: Growth rate of GNs in HGN composition with increasing number of different pattern elements and pattern size.

Distributed GN

In order to solve this overgrowing size of network, Nasution (2007) proposed a distributed approach for HGN implementation by distributing a complex HGN composition within a number of high performance computers. In this context, a HGN network could be decomposed into a number of sub-compositions, according to the number of hosts available within the physical network. Figure 2.20 shows a distribution of one-dimensional HGN composition for pattern size 13, into 4 different hosts. Each GN in the composition is treated as a memory block within a host that is communicated through an allocated terminal known as port. Port in a computer system is used to establish communication channel between processes.

Each GN in this network model is supplied with additional parameter known as port number. This port number identifies each GN and is used in inter-GN communication. The communication between hosts is achieved using physical communication such as Ethernet (using IP address). There are some limitations with this approach. These include:

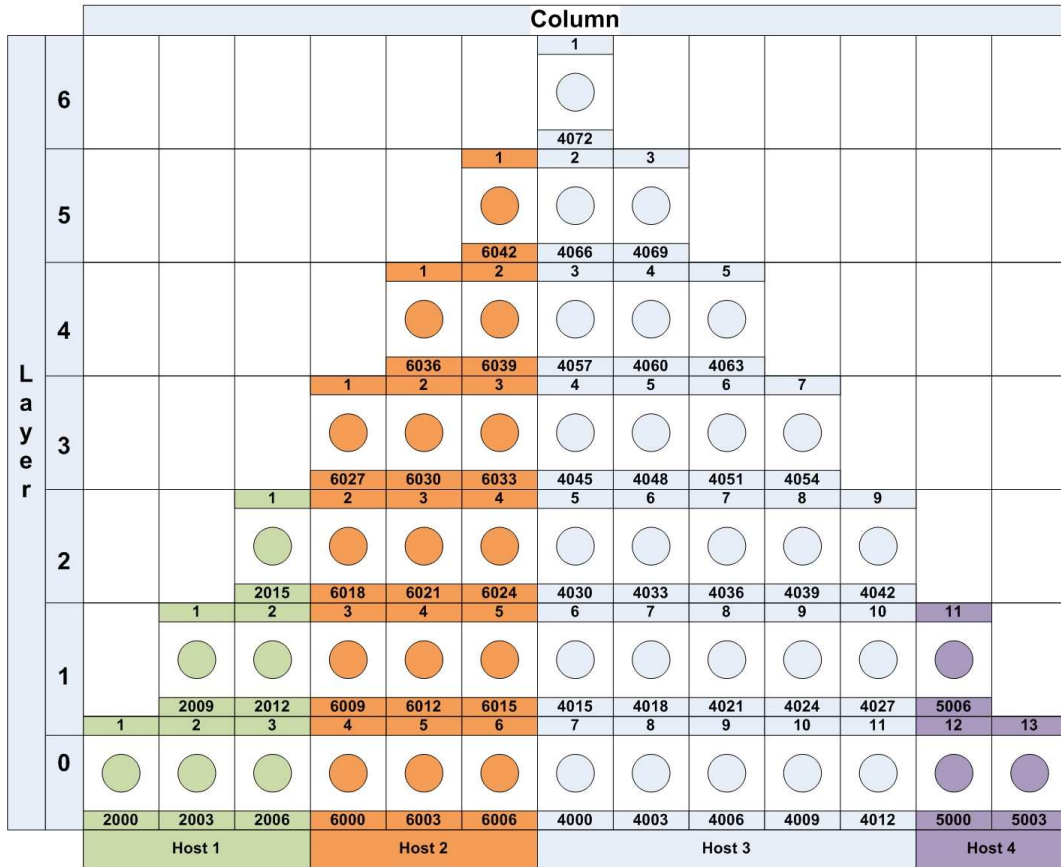


Figure 2.20: HGN decomposition into a number hosts within a physical network.

- i. Additional parameter and indices required. Each GN within the composition needs to acquire a unique port number, column index, row index, and ID. Layer index is also required to indicate the GN's level within the hierarchy. The effect of these additional indices and parameter leads to an increase in the complexity of processes involved in HGN recognition scheme. These values must be pre-assigned before actual recognition process being performed. Nevertheless, the change in its complexity is still minimal.
- ii. The port number assigned to each GN must be pre-determined beforehand. In addition, each GN must be able to calculate its adjacent GN's port number before any communication could take place. This pre-arrangement requires all GNs to evoke additional pre-processing step in order to identify and calculate the destination ports of its both preceding and succeeding GNs.
- iii. All the hosts involved must stay 'alive' for the duration of recognition process. This effect will create high-interdependency between hosts and it will be prone to a total

failure due to a failure of any single host. In addition, the costs of communication between hosts will increase due to massive and rapid message passing between GNs with different port numbers.

With these limitations in mind, a different arrangement of HGN composition is proposed, that could be distributed across physical network with low-interdependency among hosts and low requirement for the number of GN within its structure. Moreover, changes to the complexity level of HGN will be minimised, thereby retaining its overall structure.

2.7 Distributed Approach for HGN

This section describes an overview of the proposed distributed HGN scheme. A major portion of this section has been published as a case study in the book chapter by Khan et al. (2010b). HGN with distributed approach implements divide-and-distribute techniques by dividing pattern into subpatterns, and delegating these subpatterns to each available host that carries out recognition procedure using HGN sub-composition.

Distributed HGN essentially extends the original HGN infrastructure wherein its composition is decomposed into several sub-compositions. However, this is different from the previous approach, in which the whole HGN structure is decomposed and delegated to available hosts. Distributed HGN decomposes HGN network by creating smaller sub-networks, each acting as an actual HGN network that performs recognition on subpatterns. Instead of using the whole patterns as inputs, each pattern is segmented into smaller parts and each of the pattern segments acts as an input to the respective HGN sub-network composition. Figure 2.21 shows the logical illustration of the HGN decomposition into each HGN sub-compositions.

Each of the HGN subnet has the ability to process pattern segments independently from one another. Hence the compositions may be independently mapped onto the available nodes in the network without losing the HGN accuracy. Figure 2.22 shows a comparison between number of GNs required for original HGN formation and our proposed distributed HGN approach. The comparison is based on binary pattern segments with bit-size 7, corresponding to an overall pattern of 7-bit increments for the HGN. The distributed HGN scheme would require less than 1500 nodes for processing a 245-bit binary pattern.

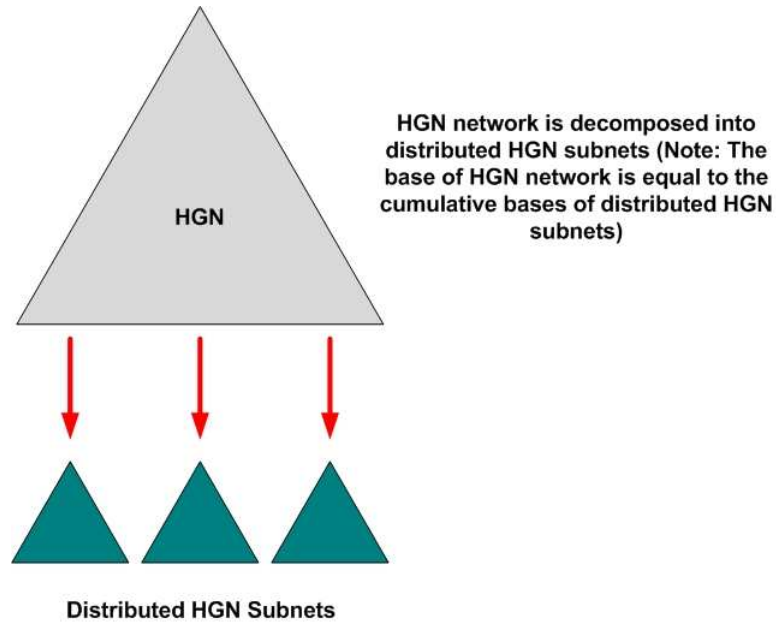


Figure 2.21: HGN Decomposition into distributed HGN sub-networks. The HGN network is decomposed into three HGN subnets.

On the other hand, original HGN structure would require about 30000 nodes for similar recognition process.

An important consideration in proposing this approach is that the distribution of large HGN network into smaller HGN subnets allows each subnet to be assigned to a specific host within a physical network. Having a smaller composition on each host will provide a two-fold advantage:

- i. Smaller capacity of memory space to be allocated for each HGN subnet, due to small HGN structure.
- ii. Reducing communication costs for inter-GN communications, while only maintaining inter-HGN communications.

Within each host, HGN subnet is structured as an executable code and each GN is represented as an associative data structure in a block of memory space for storing and recalling patterns. The communications between GNs could be achieved either using a sequential or parallel processing approach, via message passing infrastructure such as Message Passing Interface (MPI). Each GN could also be represented as a processing unit in a multi-core processor machine. Different configurations of this distributed HGN scheme will be discussed extensively in later chapters.

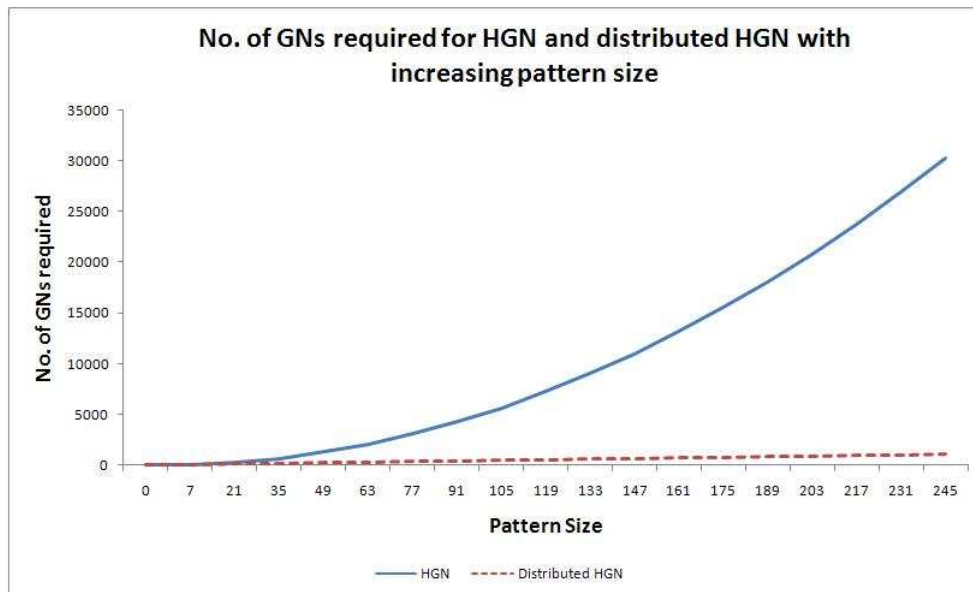


Figure 2.22: Comparison between HGN and distributed HGN for increasing size of patterns. 7-bit pattern segment is used for each HGN subnet in distributed HGN scheme.

In order to obtain an overall view of patterns, the distributed HGN scheme allows communications between HGN subnets that reside over different hosts. The communications involve message exchanges containing indices obtained from each of HGN subnets for every subpattern analysed by the network. Cumulatively, these indices represent the entire pattern structure.

To test the accuracy and scalability of distributed approach for HGN algorithm, two significant factors related to deployment of application within any distributed systems are considered, namely (1) the varying capabilities of the participating nodes and (2) the distribution of the computational load. Two different distributed schemes were simulated. The first test addresses varying processing capabilities within a distributed system through the non-uniform approach. The second test demonstrates the distributiveness of the approach through the uniform distributed HGN model.

2.7.1 Distributed Approach Design

The simulation application for pattern recognition application has been developed using C programming language with Message Passing Interface (MPI) support for GN communications. The test data for this simulation comprises a set of alphabet character patterns which can be visually distinguished. The letters 'A', 'I', 'J', 'S', 'X', and 'Z' were selected

in this regard. These letters were mapped onto 7x5 1-bit image representations, as shown in Figure 2.23.

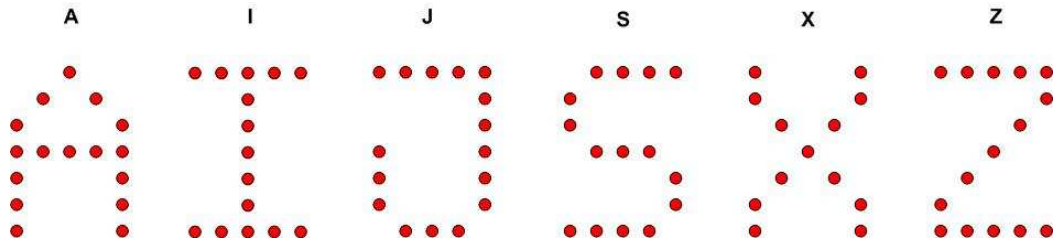


Figure 2.23: Test character representations in 7-by-5 1-bit format.

The letters would then be converted into sequences of 35-bit patterns, using a horizontal scanning approach. These patterns would then be fed into the simulation. Each pattern will be decomposed into several subpatterns, according to the quantity of hosts/sub-compositions available. Each of these subpatterns will then be introduced into each HGN subnet through its base layer of GNs. In bitmap representations, the dot pixel on the image could be represented by 1, while the blank pixel could be represented by 0; as demonstrated in Figure 2.24.

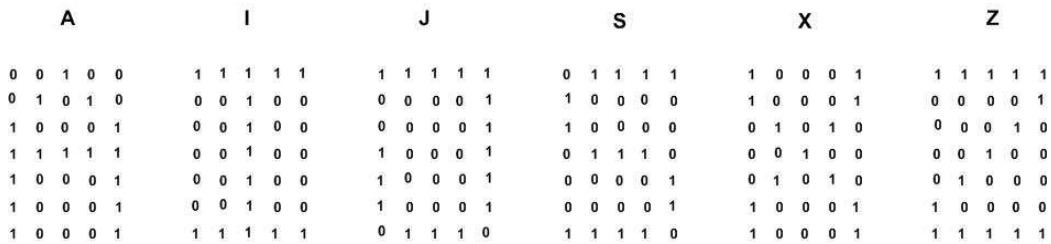


Figure 2.24: Character bitmap image representations.

The command “I10” would hence initialise the base-layer GNs with pattern element 1 and 0 respectively. The ‘STORE’ command is used to store the original patterns. An example of the command to store a particular subpattern would be “S1111111”, where the pattern after the letter ‘S’ will be stored. In response to the “STORE” command, each subnet will generate a new index if the subpattern store operation was successful. Otherwise, the index under which the subpattern was previously stored would be recalled. The “RECALL” command is used to input distorted or test subpatterns to the distributed HGN network. HGN subnet would respond with the output 0, if a close match is not found. The subpattern index will be returned if a match was found. The “RECALL” command takes the form of “R0000000”, where the letter ‘R’ is used to signify a recall

operation. This operation informs the subnet that the subpattern is either a new one or it is a distorted version of a previously stored subpattern.

2.7.2 Non-Uniform Approach

As mentioned in the previous section, the distributed approach in HGN takes the form of multiple HGN compositions. These compositions may then be distributed across the networks. In non-uniform model, the compositions may vary in size. For this simulation, a 7-21-7 composition has been chosen, where there are three sub-structures of HGN subnet, comprising two 7-element HGNs, and one 21-element HGN. Figure 2.25 illustrates these compositions. Note that in this diagram, the middle host/network has been named substantially larger than the other two hosts/networks.

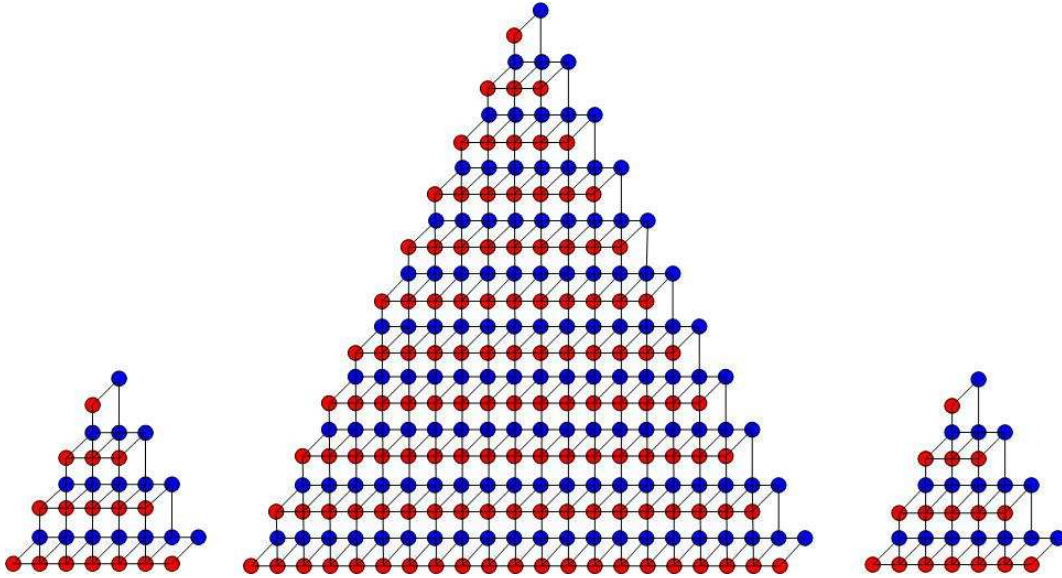


Figure 2.25: Non-uniform distributed HGN approach with 7-21-7 compositions for 35-element patterns with two possible values.

The non-uniform distribution takes into consideration an environment where some parts of the network would have lower power resources, and hence would only be able to provide limited processing capability as compared to other parts of the network. With this scenario in mind, the effect of an unbalanced composition on the pattern recognition accuracy of the distributed approach is analysed. The results of this simulation have shown that the non-uniform model offers almost equivalent level of accuracy to the HGN. Furthermore, it requires less number of GNs in its composition. The number of GN required for a single HGN composition could be derived from Equation 2.7 as described in

Section 2.6.1. On the other hand, the number of GN neurons required, $N(P)$ for s subnets in distributed HGN composition for a given pattern with size $P = a$ is determined using the following equation:

$$N(P) = v \left(\left(\frac{a_1+1}{2} \right)^2 + \left(\frac{a_2+1}{2} \right)^2 + \left(\frac{a_3+1}{2} \right)^2 + \dots + \left(\frac{a_n+1}{2} \right)^2 \right); \quad \sum_{i=1}^n a_i \quad (2.18)$$

$$N(P) = v \sum_{i=1}^s \left(\frac{a_i+1}{2} \right)^2$$

It may be readily noted that the squared term in Equation 2.18 would be substantially smaller than the one in Equation 2.7 for the same sized problem, resulting in lesser number of GNs being required.

The mapping process within our simulation begins with the input of the patterns. Each of the patterns, as shown in Table 2.5, are segmented and then loaded into the HGN subnets, by the SI module. In this regard, Figure 2.26 shows the bitmap of character ‘*T*’ being analysed by the distributed HGN. In this case, the character ‘*T*’ is stored after the character ‘*A*’, which has the index value of 1. Hence the results show the character ‘*T*’ as a new pattern with the index value of 2. Note that for this simulation, each segment is input sequentially. However in an actual implementation, the processing of these pattern segments would occur in parallel; vastly improving the execution time.

Character	35 bits Representation
A	00100010101000111111100011000110001
I	11111000010000110001100011000101110
J	01111100001000001110000010000111110
S	10001100010101000100010101000110001
X	10001100010101000100010101000110001
Z	111110000100010001000100010001000011111

Table 2.5: Character representations of 35-bit patterns using a horizontal scanning approach.

Pattern recognition process

The overall store or recall decision depends upon on the decisions reached by the individual HGN subnets. The top-layer GNs of each subnet would decide whether the subpattern would produce a recall or a store. If the pattern segment has not been identified then the active top GN would output the index value 0. Otherwise, the recalled index value of

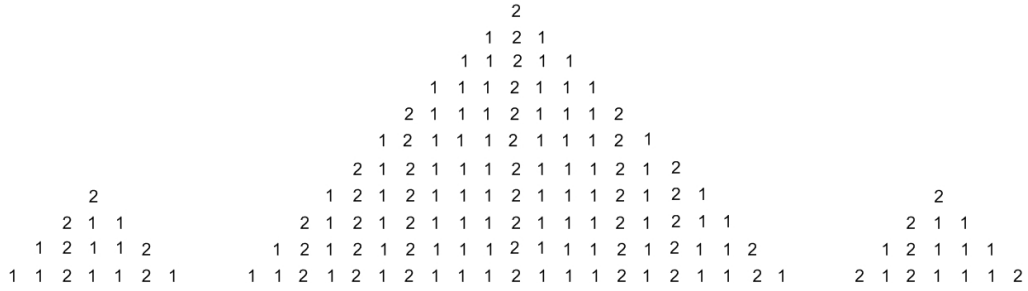


Figure 2.26: The HGN subnets successfully store the bitmap pattern for character ‘I’ at index value of ‘2’ after having stored the bit map pattern for character ‘A’ at index value of ‘1’.

the subpattern will be displayed. Figure 2.27 shows the result of 1-bit distorted character pattern ‘A’ being introduced to the network after the character patterns ‘A’, ‘I’, ‘J’, ‘S’, ‘X’, and ‘Z’ have been stored.

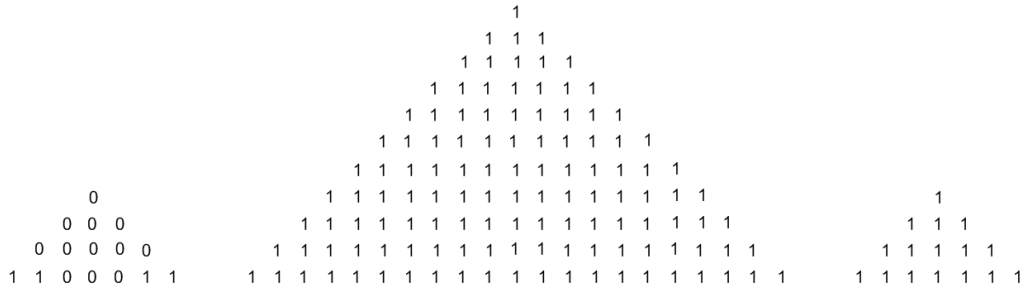


Figure 2.27: Results for introducing 1-bit distortion pattern of character ‘A’. The first HGN subnet shows that a new subpattern has been found (with assigned index 0) while other compositions correctly recall this as the pattern associated with index 1 (bitmap pattern of ‘A’).

Figure 2.27 shows that only one of the subnets records the subpattern as a new pattern. Other subnets recall the index value of 1, which is the index for the stored character pattern ‘A’. The decision whether the pattern is a recall or store is made using the cumulative decision among the distributed HGN subnets, through the determination of recall value. Equation 2.19 shows the calculation of recall R_c of distributed HGN scheme for s subnets. Note that $n_{r,i}$ represents GNs that produce index that is similar to the index of targeted pattern class r while $n_{t,i}$ represents each GN in a subnet i .

$$R_c = \frac{n_{r,1} + n_{r,2} + n_{r,3} + \dots + n_{r,s}}{n_{t,1} + n_{t,2} + n_{t,3} + \dots + n_{t,s}} \tag{2.19}$$

$$R_c = \frac{\sum_{i=1}^s n_{r,i}}{\sum_{i=1}^s n_{t,i}}$$

Using an example from Figure 2.28, the recall value for 1-bit distortion pattern of character 'A' is $(4 + 121 + 16) \div (16 + 121 + 16) = 141 \div 153 = 0.9216$. Therefore, its recall percentage would be 92.16%.

The non-uniform distributed scheme was tested by applying six levels of distortion to the stored character patterns and then measuring the recall accuracy for these distorted patterns. Figure 2.28 shows the original character patterns and the distorted patterns.

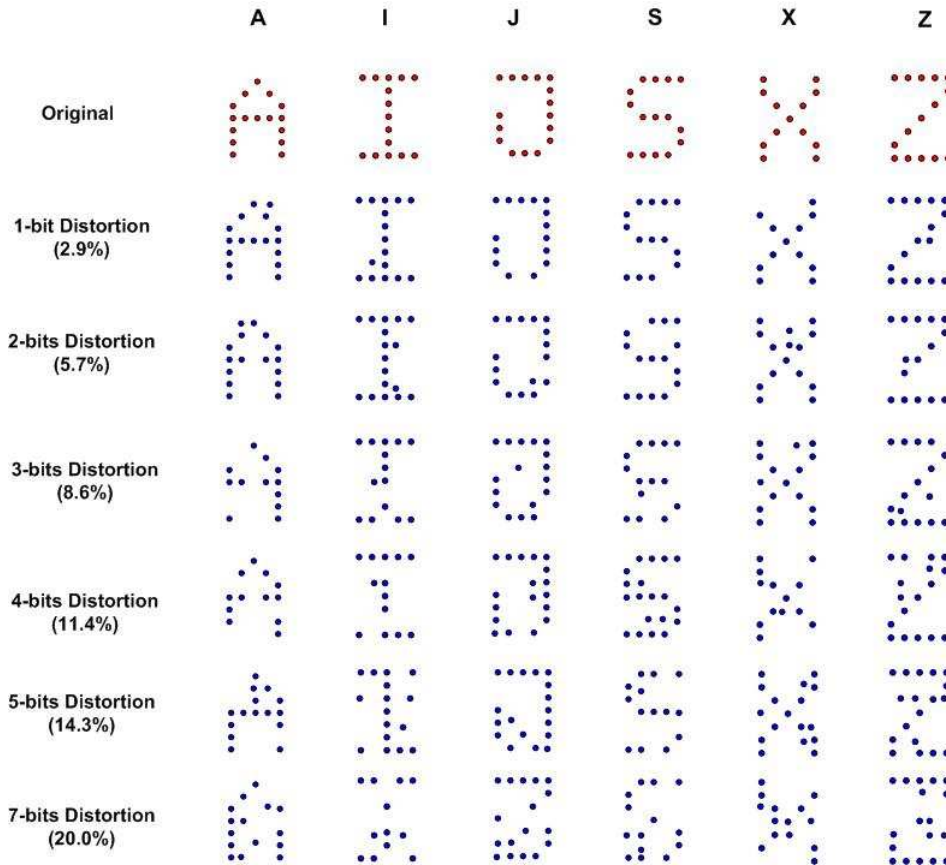
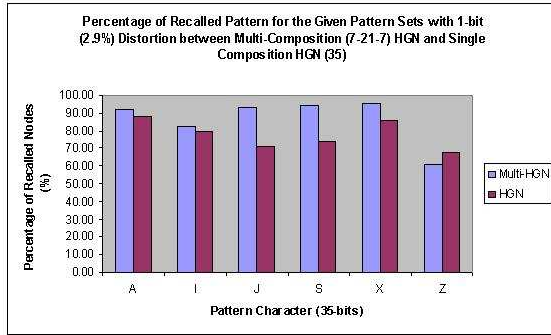


Figure 2.28: Character set used in distributed HGN simulation on pattern recognition.

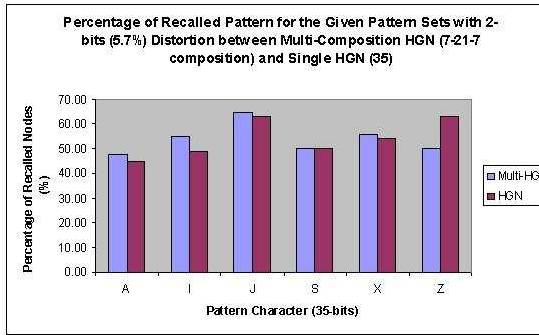
The original character patterns were first stored in the HGN network, and then the distorted patterns were introduced. The simulation was carried out using a Monash University's SUN Grid system where parts of the network were assigned to separate grid nodes for processing.

Results and discussion

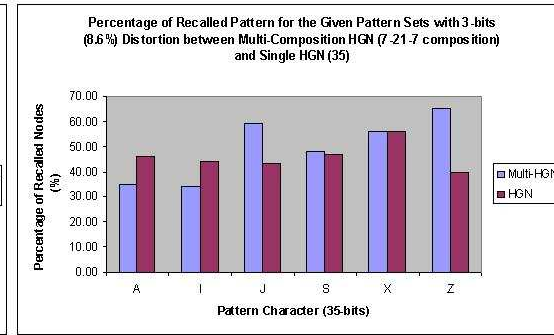
The pattern recognition recall rates for distributed HGN and HGN, with respect to identifying distorted patterns correctly, are shown in Figure 2.29.



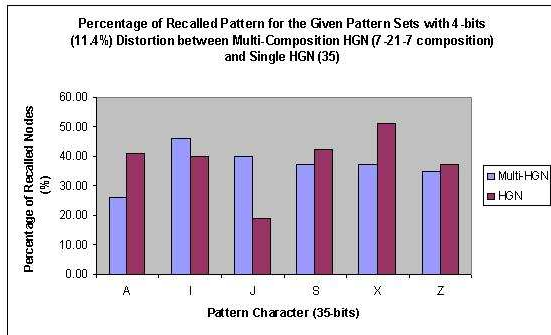
(a) 1-bit (2.9%) distortion.



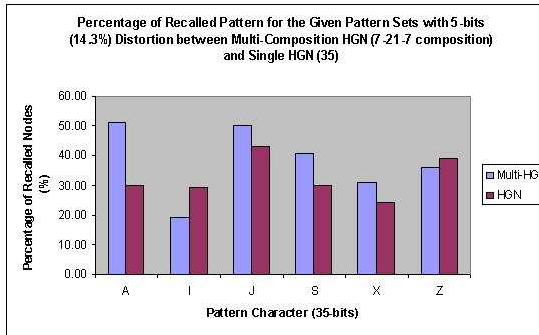
(b) 2-bit (5.7%) distortion.



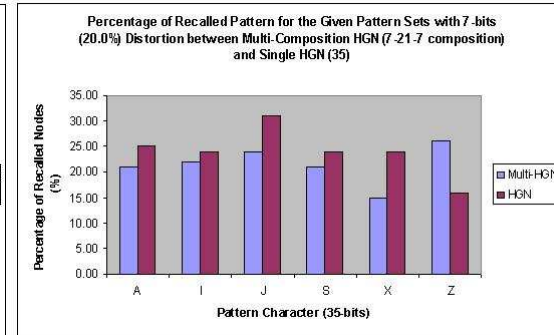
(c) 3-bit (8.6%) distortion.



(d) 4-bit (11.4%) distortion.



(e) 5-bit (14.3%) distortion.



(f) 7-bit (20.0%) distortion.

Figure 2.29: Comparison on recall accuracy between non-uniform distributed HGN and HGN pattern recognition schemes on different distortion levels applied to character set $A, I, J, S, X,$ and Z .

It may be seen from the figures that the recall values for distributed approach are very similar to the results of the non-distributed approach. In fact, some of the values obtained are higher than those of the HGN. For example, the 1-bit distorted patterns show a significant increase in the recall as compared to the HGN. This is owing to the encapsulation effect of distributed HGN where the effects of the distortion occurring within a particular subnet do not affect the other subnets. Figure 2.30 shows the encapsulation effect. It also shows the internal state of the subnets from the 1-bit distorted pattern of character 'A'. The effects of the distortion are limited to the first subnet, where a part of the distorted pattern is analysed, the remaining subnets are not affected by the distortion.

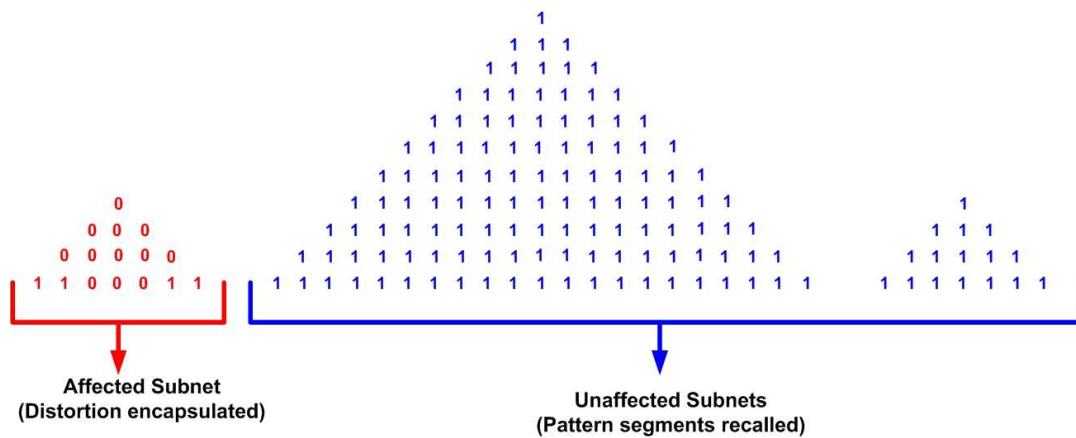


Figure 2.30: A one-bit distortion occurring within the overall input pattern 'A' stays encapsulated within the left composition.

The HGN subnets are able to provide higher recall accuracy owing to this encapsulation effect. The downside to this effect is that if the distortion occurs within the larger subnet, then the recall accuracy may be adversely affected. This problem can be easily resolved if all the compositions are of similar sizes. The uniform approach therefore implements equal-sized compositions.

2.7.3 Uniform Approach

The uniform distributed HGN is introduced as a measure to delimit the effects of distortion location experienced in non-uniform model. For the purpose of this pattern recognition simulation, five HGN subnets for 7-bit subpattern were implemented to analyse 35-bit binary character patterns. Figure 2.31 shows the structure of this composition.

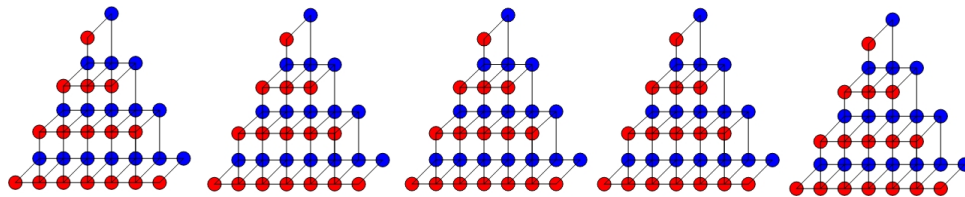


Figure 2.31: Uniform distributed model composition for analysing 35-bit binary patterns.

The uniform approach has been developed to test the distribution of the HGN algorithm for networks comprising small devices and/or limited processing and storage capabilities. With the relatively smaller sized subnets, each processing node is able to store smaller pattern segments and thus requires lesser processing capability for the pattern recognition process. Having similar sized compositions also removes the problem of a single composition affecting the accuracy of the results.

The simulation of uniform approach, using distorted character patterns from Figure 2.28, shows that uniform model produces greater efficiency in terms of pattern recognition as compared to the non-uniform approach and the HGN. In this regard, Figure 2.32 shows the recall rates for the uniform distributed HGN and the HGN.

It may be seen from these graphs that the uniform model's recall values are significantly higher than those of the HGN. The increase in the recall accuracy is owing to the encapsulation effect where the distortions are generally compartmentalized within specific composition(s) and thus do not affect the findings of other compositions. The added benefit of the uniform approach is that all the compositions are of similar sizes, hence the problem of an over-sized composition affecting the accuracy of the results is alleviated. Figure 2.33 shows the encapsulation effect within the uniform distributed approach for pattern of character 'A' with 2-bit distortion.

Figure 2.33 demonstrates that the distorted pattern segments are encapsulated within the first and the third compositions from the left. The rest of the pattern segments are recalled as pattern of character 'A' (represented by the bias index entry of '1').

A comparison of recall accuracy for the uniform and non-uniform models are shown in Figure 2.34.

It is apparent from these figures that the uniform approach generally produces higher recall accuracy values for the distorted patterns as compared to the non-uniform ones.

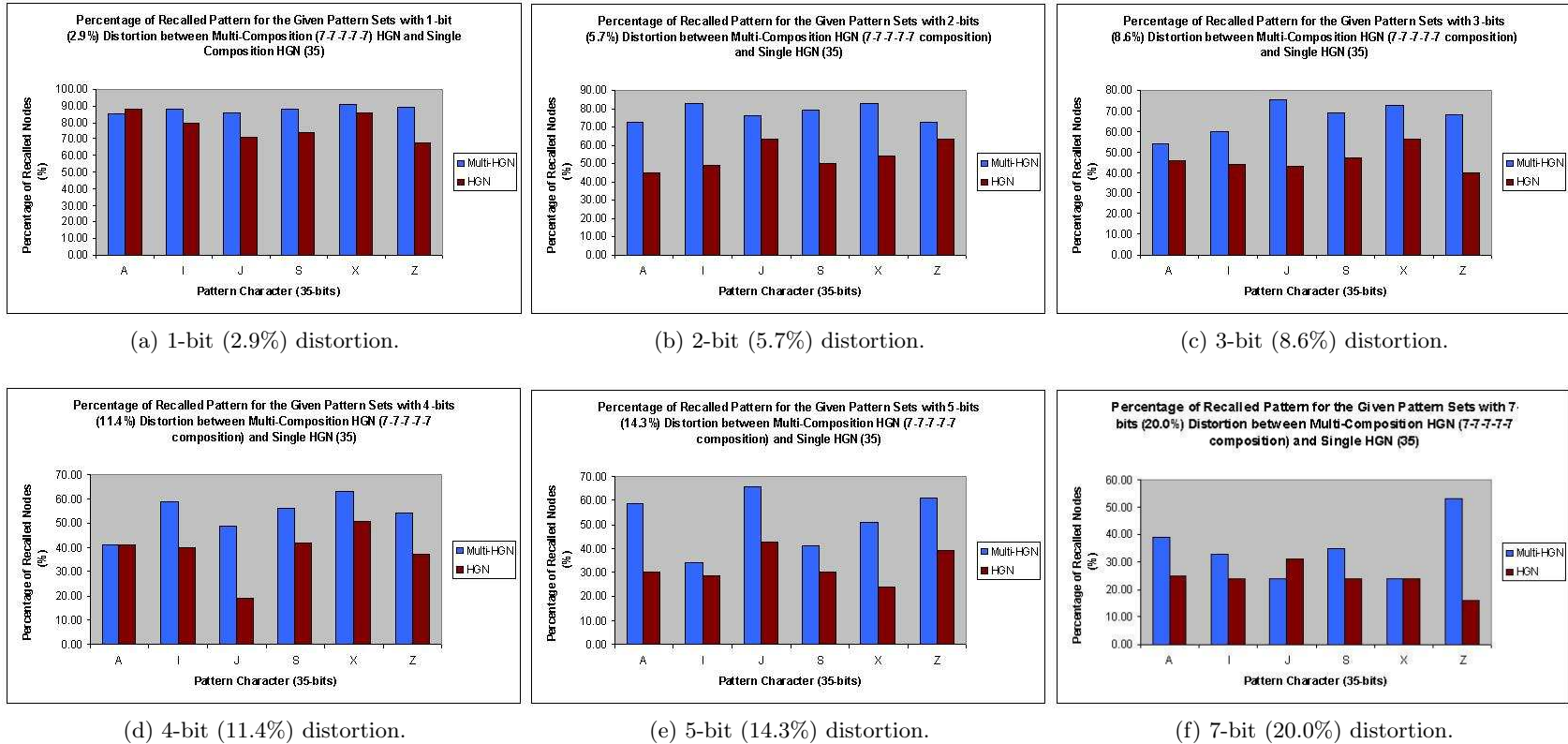


Figure 2.32: Comparison on recall accuracy between uniform distributed HGN and HGN pattern recognition schemes on different distortion levels applied to character set $A, I, J, S, X,$ and Z .

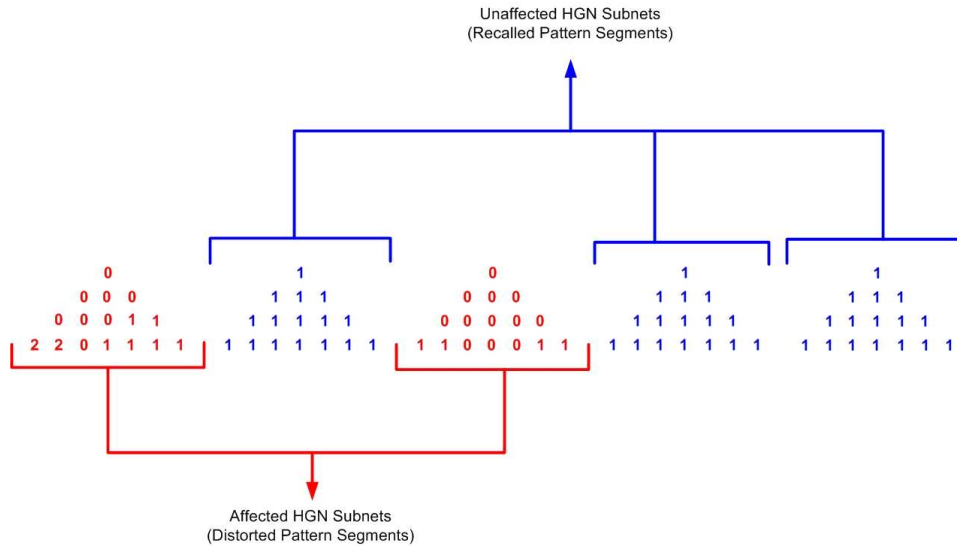


Figure 2.33: Encapsulation effect within uniform model when processing a pattern of character ‘A’ with 2-bit distortion. The effects of the distortions are localised within the two compositions on the left and do not influence the findings of the remaining compositions.

Better performance of the uniform distributed HGN is due to the standard-size encapsulation of the local distortions. A close-up view of the difference in handling of the distortion by the uniform approach may be observed in Figure 2.35. It shows the results for 1-bit distortion using three 7-bit HGN subnets and one 21-bit HGN composition for pattern recognition.

The distortion effect within the HGN composition cannot be localised and it propagates along the right hand side of the composition (Figure 2.35), leading to a null recall. It is evident that the smaller and similar sized distributed compositions have a better chance of discovering the distorted pattern as compared to a single HGN composition.

It can be concluded from the information presented and discussed above that the distributed approach in HGN provides a completely decentralised solution for pattern recognition within distributed systems. It also retains single-cycle recognition characteristic of HGN. This approach could be deployed in both micro and macro configurations. In micro configuration, each GN in HGN subnet could be assigned to a specific computing node within a physical network. Consequently, each HGN subnet is represented by a network of computing nodes. In macro configuration, each HGN subnet is assigned to a specific host within a network, and these hosts in return, formed a computational network of HGN distributed structure. Further discussion on distributed HGN configurations will be presented in the next chapter.

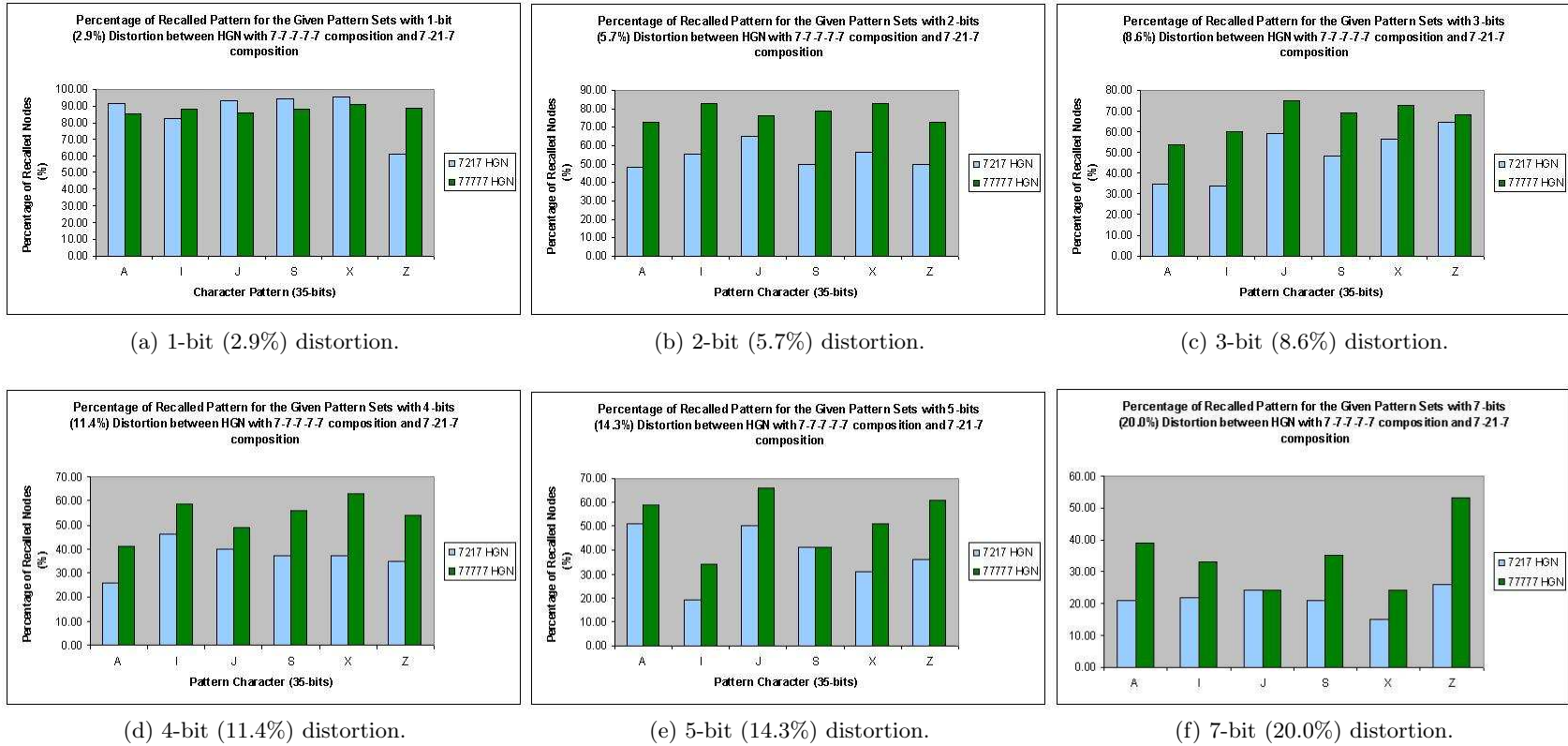


Figure 2.34: Comparison on recall accuracy between uniform and non-uniform distributed HGN pattern recognition schemes on different distortion levels applied to character set $A, I, J, S, X,$ and Z .

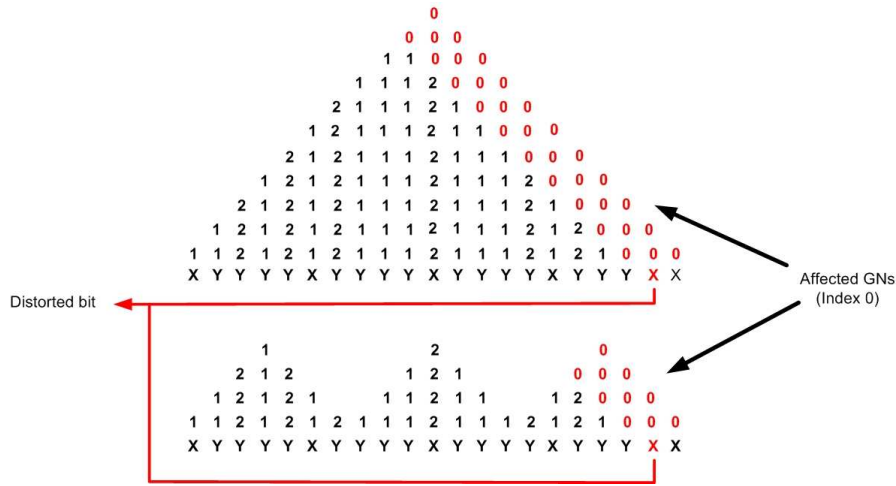


Figure 2.35: The effects of a 1-bit distortion in the pattern get localised within the uniform distributed compositions (lower) whereas the effects of the distorted pattern are propagated along the right side of the entire HGN composition, leading to a false conclusion.

2.8 Conclusions

In this chapter, a discussion on the pattern recognition concept and theories has been presented and different kinds of approaches to pattern recognition have been explored. The advantages and limitations of each approach in relation to scalability issue within pattern recognition has also been discussed. In addition, based on the two scalability factors being discussed, it can be derived that existing neural networks are far from providing a scalable scheme for recognition purposes. However, initiatives are currently being undertaken in creating more effective approaches using existing algorithms. One of these initiatives is the development of Graph Neuron (GN) pattern recognition algorithm that implements an in-network and parallel pattern recognition scheme.

This chapter has also presented a detailed analysis and discussion on GN-based pattern recognition algorithms, namely Graph Neuron (GN) and Hierarchical Graph Neuron (HGN). In summary, GN is a graph-based scheme that employs graph matching concepts and performs an in-network processing within its recognition procedure. GN has been proposed and implemented in different areas of applications, ranging from pattern recognition to event detection in wireless sensor networks (WSNs) and mobile ad hoc networks (MANETs). An extensive analysis on GN has also been presented. Common limitation of GN, i.e. crosstalk or intersection problem has also been discussed in details with examples.

Moving from GN perspective, this chapter also discussed the evolution of HGN as a solution towards crosstalk issue in GN pattern recognition scheme. HGN algorithm for pattern recognition has been extensively described in (Nasution, 2007; Nasution and Khan, 2008). In this chapter, an overview of HGN was provided, with its complexity estimation. HGN has been found to be a highly accurate recognition scheme that is able to perform recognition procedure within a distributed system. However, scalability is an issue in HGN implementation. This is due to overgrowing network size with an increase in size of patterns used. A number of possible solutions have been proposed, including the use of high-dimensional HGN structure. Nevertheless, this increases the complexity of the algorithm.

The main contribution of this chapter is the proposed distributed scheme for HGN implementation. This distributed approach has been presented as a solution towards the scalability issue in HGN pattern recognition algorithm. An overview of the proposed algorithm has been provided, including a case study on different forms of its implementations for binary character recognition. The results of the simulation that has been carried out indicate that distributed approach produces higher recall accuracy for pattern recognition as compared to existing HGN scheme. This is mainly due to the error encapsulation within the distributed HGN subnet that controls error propagation within HGN network. Distributed approach for HGN also requires significantly less number of GNs as compared to HGN implementation.

This chapter has established the core direction in this research. The next chapter intend to explore this distributed HGN algorithm and its capabilities as a distributed pattern recognition algorithm. In achieving this, a number of case studies and analyses have been carried out. Distributed HGN as a GN-based algorithm also can be considered as an associative memory (AM) algorithm and has a capability to perform parallel recognition process. From this point onwards, the proposed distributed approach for HGN algorithm will be known as Distributed Hierarchical Graph Neuron (DHGN). Further details on DHGN algorithm will be presented and discussed in the next chapter.

Chapter 3

Distributed Hierarchical Graph Neuron

Graph Neuron (GN)-based algorithms have been developed based upon two different concepts known as graph-matching and associative memory. These two concepts have given an added advantage in terms of scalability for GN-based algorithm implementations. GN has the ability to perform pattern recognition processes on distributed systems due to its simple recognition procedure and lightweight algorithm. Furthermore, GN incurs low computational and communication costs when deployed in a distributed system. Previous chapters have analysed both GN and HGN, and introduced distributed version of HGN known as Distributed Hierarchical Graph Neuron (DHGN).

An important aspect in the development of pattern recognition scheme is its algorithmic design. A proper design will lead to high efficiency, and have the ability to generate a more accurate classification strategy. In this chapter, the algorithmic design and perspective of the proposed DHGN algorithm for distributed pattern recognition scheme for large-scale data sets is extensively discussed. The proposed algorithm extends the scalability of the existing Hierarchical Graph Neuron (HGN) implementation by reducing its computational requirement in terms of the number of neurons for recognition processes. It provides comparable recognition accuracy as the HGN implementation that has been presented in the previous chapter. DHGN provides a capability for recognition process to be deployed as a composition of sub-processes that are being executed in parallel across a distributed network. Each sub-process is conducted independently from each other, making it less cohesive as compared to other pattern recognition approaches.

The objectives of this chapter are as follows:

- i. To describe features and characteristics of DHGN as a distributed pattern recogniser.
- ii. To consider different pre-processing requirements for pattern recognition.
- iii. To perform extensive evaluation and analysis on pattern recognition scheme using DHGN algorithm.
- iv. To present simulation results of DHGN distributed scheme for optical character and image recognition.
- v. To discuss possible extension for DHGN implementation using multiple values.

Some parts of this chapter have been published as a series of conference and journal papers and book chapter including (Khan and Muhamad Amin, 2007; Muhamad Amin and Khan, 2008b; Raja Mahmood, Muhamad Amin and Khan, 2008; Muhamad Amin and Khan, 2009; Khan et al., 2010b).

The composition of this chapter is as follows: Section 3.1 contains an overview of DHGN algorithm for pattern recognition. In this section, further description of the algorithm together with its associated components will be presented. Section 3.2 describes a number of dimensionality reduction techniques for patterns pre-processing that are related to DHGN implementation. Section 3.3 provides an extensive review of the analyses that have been carried out on the DHGN algorithm. These analyses focus on both complexity and scalability of the algorithm. A number of comparative analyses between DHGN and other existing pattern recognition algorithms are also conducted, and Section 3.4 presents a set of pattern recognition simulation studies that have been carried out using DHGN algorithm for optical characters and binary images. A multi-value DHGN model is proposed in Section 3.6. This model extends existing DHGN capabilities by enabling it to efficiently accept multiple different values of elements within patterns. Finally, Section 3.6 concludes this chapter.

3.1 DHGN for Distributed Pattern Recognition

In this chapter, a detailed explanation on DHGN algorithm for distributed pattern recognition is provided. In chapter 2, an analysis of distributed HGN approach has been presented. It will be shown that having smaller sub-compositions in HGN networks will lead

to better error control in distorted pattern recognition. DHGN follows this distributed approach and implements a scalable pattern recognition scheme, suitable for large-scale recognition deployment.

This section will extend our discussion on DHGN algorithm. This begins with an explanation on associative memory concept that is being adopted by DHGN and other GN-based algorithms. A discussion on AM concept will be presented in Subsection 3.1.1. This section will also describe some important components of DHGN algorithm including its architecture (Subsection 3.1.2), recognition procedure (Subsection 3.1.3), pattern storage (Subsection 3.1.4) and learning mechanism (Subsection 3.1.5).

3.1.1 GN Associative Memory Concept

From a pattern recognition perspective, AM refers to a set of functions (or a learning network) that has the ability to make an association between input and output. According to Román-Godínez, López-Yáñez and Yáñez-Márquez (2009), Associative memory M is a system that provides an input-output relationship as follows: $a \rightarrow M \rightarrow b$ where a and b are input and output respectively. In this perspective, each input vector is associated with an output vector. This association can be represented in the form of a fundamental set of associations: $\{(a^\mu, b^\mu) \mid \mu = 1, 2, \dots, p\}$. This set is a priori knowledge that must be known by the AM system.

There are two types of AM for pattern recognition, namely auto-associative memory (auto-AM) and hetero-associative memory (hetero-AM). In auto-AM, the system recognises an input pattern that was presented and produces its associated output pattern. Hence, for a given set of associations (a^μ, b^μ) , the auto-AM rule is true with the following condition: $a^\mu = b^\mu, \forall \mu \in \{1, 2, \dots, p\}$. It enables the system (either neural network or learning system) to pass through input patterns towards output pattern without any changes. This is due to the fact that input patterns and output patterns have similar characteristics. An example of auto-AM algorithm is the Hopfield network.

Alternatively, hetero-AM pattern recognition follows the rule of association, such that incomplete input patterns may also possibly lead to complete output patterns. Therefore, in terms of association set (a^μ, b^μ) , the following rule applies where $a^\mu \neq b^\mu$, for $\exists \mu \in \{1, 2, \dots, p\}$. In this case, given a distorted pattern \bar{a}^x of original pattern a^x , the hetero-AM system will be able to gain full recall of pattern a^x . Bidirectional associative

memory (BAM) is one of the neural network approaches that adopt this hetero-AM concept. Hetero-AM also offers an ability to conduct a recognition based on patterns with different sizes, such as being demonstrated in the work of Kosko (1988).

The AM approaches including Hopfield network, Fuzzy Associative Memory (FAM) (Kosko, 1992), generally tend to be computationally intensive and iterative. Morphological Associative Memory (MAM) (Ritter et al., 1998) on the other hand, provides solution within a single iteration and thus implements single-cycle learning. MAM is however a tightly coupled scheme, which relies on global maximum/minimum computations and is not readily distributed.

Graph Neuron (GN) based algorithms including HGN and DHGN implements an autoassociative memory approach in their recognition procedure. GN has the ability to recall patterns that have been memorised by the network. The memorisation could occur either in pre-execution stage or instantaneously during the recognition process. The former means that GN performs a supervised recognition, while the latter represents an unsupervised mechanism. Furthermore, GN performs recognition on patterns with equivalent size. Hence, the features of auto-AM have been fulfilled.

The scalability of DHGN and other GN-based algorithms has also been contributed by the adoption of this associative memory approach. DHGN is an associative memory system that capable of recognising patterns (either original or noisy), and it is able to match multiple streams of input with historical data within the network in real-time (Khan et al., 2010b). DHGN also performs internal association in the sense that for a given pattern, an association between elements within a pattern is also being considered. For example, given a pattern P with 5 elements $\{p_1, p_2, p_3, p_4, p_5\}$, DHGN also take into account the associations set $\{(p_1, p_2), (p_2, p_3), (p_3, p_4), (p_4, p_5)\}$. The following subsection will further discuss the architecture of DHGN, in line with its pattern recognition process.

3.1.2 System Architecture

DHGN formalises the distributed HGN approach that has been described in Chapter 2. DHGN adds a clustering mechanism in pattern recognition, by dividing and distributing patterns into subpatterns. Each of the subpatterns undergoes a one-shot recognition procedure. The results of sub-recognition will cumulatively add up to obtain the actual recognition result.

DHGN network constitutes a number of DHGN subnets (HGN sub-composition) and a Stimulator/Interpreter Module (SI Module) node, as described in Muhamad Amin and Khan (2008b). Figure 3.1 shows a complete architecture of DHGN network. In this figure, a decomposition of binary image pattern ‘K’ into subpatterns is illustrated. This decomposition is performed by the SI Module node. The input activates the GN nodes corresponding to the bits of the input pattern. In doing so each pattern element within a subpattern is mapped to relevant GNs in the respective subnet. Each subnet integrates its responses and sends the results to the SI Module to form an overall response.

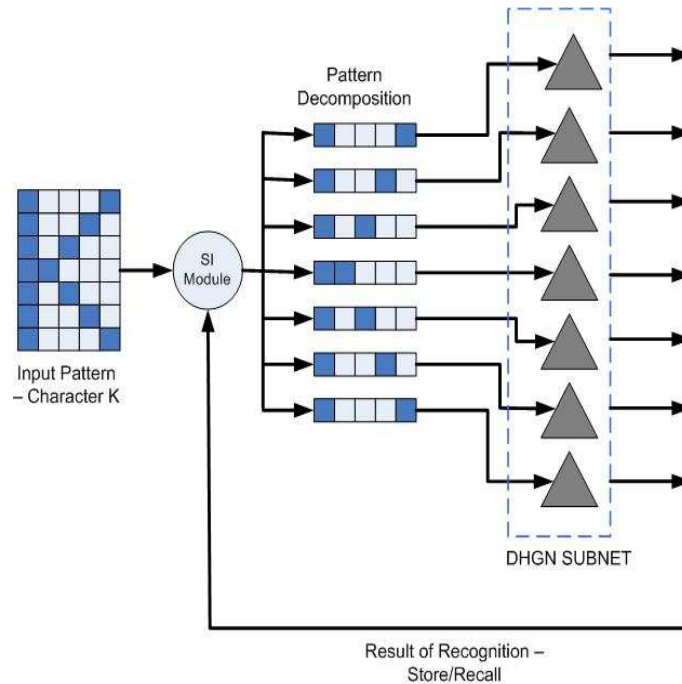


Figure 3.1: DHGN framework for distributed pattern recognition.

Figure 3.1 also shows that communications within DHGN network occur in a single-cycle environment, in which each pattern is passed through the network only once. Recognition result, in terms of recall (pattern is known) or store (pattern is memorised). Within each DHGN subnet, the recognition process involving communication between GNs also happening once for each subpattern. In this point of view, DHGN offers fast recognition procedure by eliminating the needs for iterative mechanism to recall or store patterns.

Each DHGN subnet is derived from a composition of interconnected GNs. The size of subnet depends on the size of the subpattern used in the system and the number of different elements in the subpattern. Therefore, to define the size of each subnet, we

consider the number of GNs n_{gn} required for subpattern with size s_{sub} and v different element as shown in the following equation:

$$n_{gn} = v \left(\frac{s_{sub} + 1}{2} \right)^2 \quad (3.1)$$

Network Generation

In order for the DHGN scheme to perform recognition on patterns, it must first be generated. Network generation involves the construction of SI module node and a collection of DHGN subnets. SI module node is a control node, responsible for managing the inputs and outputs among the DHGN subnets. The distribution of DHGN subnets within the network depends on the pattern decomposition by the SI module. Given a pattern vector $P = \{p_1, p_2, p_3, \dots, p_m\}$ of size m , and subpattern length s_{sub} . The number of DHGN subnets n_{sub} that needs to be generated is determined by Equation 3.2:

$$n_{sub} = \frac{m}{s_{sub}}, \quad s_{sub} \leq m \quad (3.2)$$

The GN nodes within a DHGN subnet are structured in hierarchical manner, similar to the HGN. Each GN layer within the DHGN subnet is populated with GN nodes. The number of GN layers, l_{gn} required within a DHGN subnet is given by the following equation:

$$l_{gn} = \frac{s_{sub} + 1}{2} \quad (3.3)$$

Note that the number of GN layers could be directly determined from the calculation of size of the network as shown in Equation 3.1. The conditions for GN node generation within a particular layer are as follows:

- i. At base layer l_{base} , the number of GNs generated $n_{gn}^{l_{base}}$ is equivalent to the size of subpattern multiplied by the number of different elements v , i.e. $n_{gn}^{l_{base}} \times v$.
- ii. At a middle layer l_i , the number of nodes $n_{gn}^{l_i}$ varies according to the level of the layer i in the hierarchy, except for the top layer. Therefore, $n_{gn}^{l_i} = v(s_{sub} - 2i)$.
- iii. At the top layer l_{top} , the number of processing nodes required is equivalent to the number of different elements v . Hence, $n_{gn}^{l_{top}} = v$.

In the network generation stage, SI module is also responsible for initialising DHGN subnets. The initialisation involves communication of possible input values to the base layer GN nodes before the actual store/recall operations can start. The message communication between SI module and base layer GN nodes (within each DHGN subnets) is conducted using a specific message communication protocol that has been developed for bitmap patterns. SI module sends the possible input values to each DHGN subnet using the instruction, message format. For example, if binary values are to be communicated then the message would be *initialize, (0,1)*.

Each initialisation message received by the base layer GN nodes is used to coordinate the GN nodes within the base layer. Each node within the base layer represents a specific position. The following pseudo code shows the formation of the base layer GN nodes for binary pattern recognition:

Algorithm 2 Base Layer GN Formation

```

1: for  $GN \in l_{base}$  do
2:   if  $GN_{id} \leq p_{size}$  then
3:      $GN_{val} = 1$ 
4:   else
5:      $GN_{val} = 0$ 
6:   end if
7: end for

```

Note that the initialisation process involves uploading distinct *(value, position)* pairs into the respective GNs for later use in the store/recall operations.

GN Communications

Communications in the DHGN recognition scheme involve a message-passing mechanism, in which a single processing node communicates with other nodes in the network for exchanging messages. It composed of two different types, namely macro- and micro-communication. In macro-communication, communication costs at system level are taken into account, i.e. communications incurred between SI Module and DHGN subnets. On the other hand, micro-communication deals with GN communications within a particular subnet for each pattern introduced into the system.

Macro-communication in DHGN implementations happened between SI Module node and either base layer GNs or top GNs in each subnet. It occurs at three different phases:

- i. Network generation phase: SI module is responsible for communicating possible input values of the patterns, which will be used in the recognition process, to all base layer GNs within DHGN subnets. Equation 3.4 shows the number of messages that needs to be communicated by SI module to these GN nodes, $n_{SI \rightarrow sub}^{msg}$:

$$n_{SI \rightarrow sub}^{msg} = n_{sub} \times s_{sub} \times v \quad (3.4)$$

In this equation, n_{sub} represents the number of available subnets. This equation is based on the assumption that all DHGN subnets are of the same size. The messages communicated from SI module to each GN is in the form of *instruction*, *message* format as described earlier.

- ii. Pattern input phase: After all DHGN subnets have been generated, SI module will perform a divide-and-distribute process on input pattern that has been introduced into the system. This process decomposes pattern into a number of subpatterns according to the number of subnets available. Consequently, these subpatterns will be sent to each subnet within the network. However, in the actual format, SI module will communicate directly with each GN at the base layer of each DHGN subnet. Hence, the number of messages communicated is similar to the number of messages in network generation phase (as in Equation 3.4).
- iii. Result communication phase: After recognition process in each DHGN subnet is completed, the results (in terms of recall or store) will be communicated back to SI module for further analysis. In this communication, messages in the form of *subnet_{id}*, *status*, *index* will be sent to SI module by all the top-layer GN of each subnet. In regards to the communication cost, the total number of messages communicated from subnets to SI module, $n_{sub \rightarrow SI}^{msg}$ is equivalent to the number of subnets available, n_{sub} . Hence, $n_{sub \rightarrow SI}^{msg} = n_{sub}$.

The following relations describe the micro-communications involved between GNs within each DHGN subnet:

Base Layer. For each GNs in the base layer, the amount of message communications incurred could be derived from the number of messages communicated between adjacent neurons for each input subpattern. For GNs at the edge of base layer, the number of

communication exchange is equivalent to the number of different elements within the subpattern. For non-edge GNs, the communication is required between adjacent neurons in both the preceding and the succeeding columns, as well as the communication of bias indices to the GNs at the next higher layer. In this context, the amount of message exchange is v^2+1 . The cumulative communication costs involved for each input recognition process for all GNs in the base layer of a single DHGN subnet is derived from the following equation:

$$n_{l_{base}}^{msg} = ((v^2 + 1)(s_{sub} - 2) + 2v) \quad (3.5)$$

Middle layers. The communication costs for GNs in the middle layers are similar to that at the base layer. However, the difference would be in the number of nodes available within each layer. For each middle layer i , where $1 \leq i \leq top - 1$, the number of message exchanges occurred for a single input subpattern recognition could be derived as the following:

$$n_{l_i}^{msg} = ((v^2 + 1)(s_{sub} - (2i + 2)) + 2v) \quad (3.6)$$

Equation 3.7 presents cumulative communication costs for all GNs in the middle layers:

$$n_{l_i}^{msg} = \sum_{i=1}^{top-1} ((v^2 + 1)(s_{sub} - (2i + 2)) + 2v) \quad (3.7)$$

Top layer. These GN nodes are only responsible for communicating the final index for each subpattern stored/recalled to the SI module. The costs for communicating these indices have been included in the macro-communication evaluation.

This subsection has presented a detailed description of DHGN architecture for distributed pattern recognition. This architecture represents an abstract formation of the network. In reality, this architecture could be deployed in a coarsely-distributed or finely-distributed networked environment.

3.1.3 Dual-Phase Recognition Procedure

DHGN architecture that has been described in the previous subsection comprises two important entities; SI Module and DHGN subnets. Recognition of patterns mainly occurred

within each DHGN subnet. However, at this instance, all that is known to each subnet is only a sub-composition of the overall pattern. This means that there is a need for DHGN system to restructure the overall information of the pattern, and produces result for the entire pattern, i.e. whether the input pattern is known to the system or not. In this regard, there is a need for another phase of recognition involving the results of the recognition process executed within each of the subnet.

Recognition procedure for DHGN implementation can be analogically represented as a distributed analysis procedure as shown in Figure 3.2. Imagine if there is a large block of data needs to be analysed. Given a set of analysts, this large block of data could be decomposed into sub-structure of data, and each analyst would work on it. In the end, the results of the analysis must be recompiled to form the overall results on the analysis of the overall large block of data.

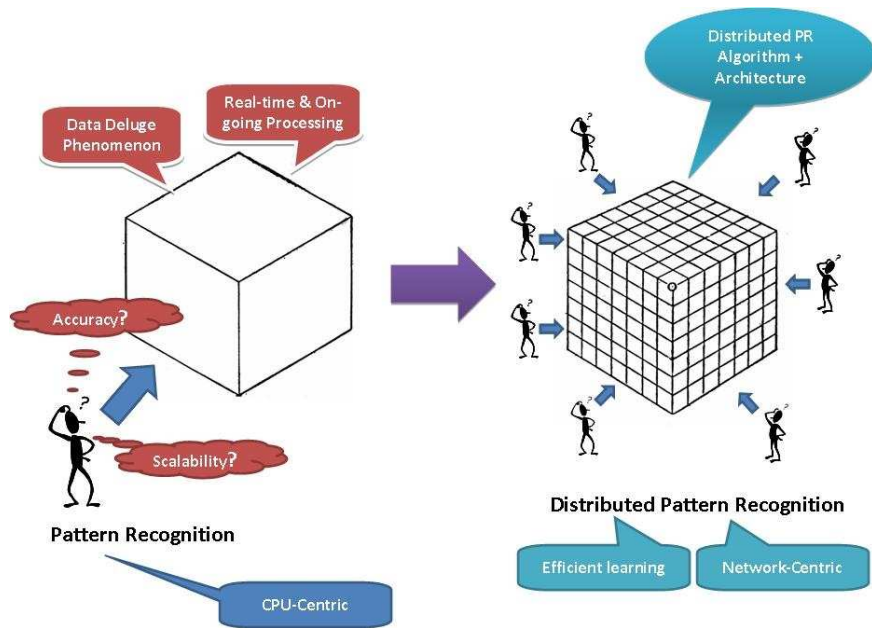


Figure 3.2: Analogical representation of DHGN distributed pattern recognition scheme.

DHGN distributed pattern recognition performs pattern analysis on two different phases: (1) subpattern recognition; and (2) pattern reconstruction and recognition. It is best to note that these two phases occurred consequently and within a single-cycle recognition mechanism.

Phase 1 - Subpattern Recognition

In DHGN implementation, the core recognition process is conducted at the subpattern level. There are four stages involved in this process:

Stage 1. After receiving an input from the SI module, each activated GNs at the base layer will send a signal message to other nodes in the adjacent columns containing the row number/address of the activated node. Those activated nodes that are at an edge of the layer will only send the activation signal messages to the GNs in the penultimate columns. The activated GNs that receive the signal messages from their adjacent neighbours will respond by updating their bias array noting the activation signals. All other GNs will remain inactive.

Stage 2. All active GNs at the base layer will then update their bias arrays. If the bias entry value, $b_{ent}(left, right)$ received from both the activated nodes in preceding and succeeding columns have been recorded, the index of the entry will be sent to the respective GN in the same position at the higher layer. If the $b_{ent}(left, right)$ value is not found within the bias array, then a new index will be created and sent to the GN node in the higher layer. Note that active nodes at the edges of the base layer will not be communicating with higher layer nodes since there is no node present at the edges of the higher layer owing to the pyramid-like structure of the DHGN subnets.

Stage 3. GN nodes at a layer above the base that receive a signal message, containing the index of the bias entry that has been created or recalled from stage 2, will be activated. Similar process as in stage 1 and 2 will occur. However, the contents of the signal messages from preceding and succeeding columns would be in the form of $b_{ent}(left, middle, right)$ for non-edge nodes and either $b_{ent}(left, middle)$ or $b_{ent}(middle, right)$ for the edge nodes. The values for left, middle, and right are derived from the indices retrieved from the lower layer nodes. For instance, left is for the preceding GN node's index received from its lower layer counterpart. After the message communication between adjacent nodes has completed, the active GNs will update their bias arrays and send the stored/recalled index/indices to the node at the same position in the higher layer (except for the GNs at the edges). This stage will be repeated for each layer above the base layer, until it reaches the top layer GN nodes.

Stage 4. One of the top layer GNs will receive a bias index from a GN in the layer underneath it. This top layer activated node will search it bias array for this index. If the

index is found, then this node will trigger a recall flag with the recalled index. Otherwise, it will trigger a store flag and store the new index in its bias array. It will then send a signal message to SI module with the message format $\{subnet_id, status, index\}$, where status is either recall or store. The signal message sent by top layer active GN marks the completion of the recognition at subpattern level. In a DHGN implementation, lower bias arrays are updated whenever a new entry is found. Note that bias index for lower layer nodes may not be the same for a given pattern index.

Figure 3.3 shows the process workflow of the proposed recognition algorithm.

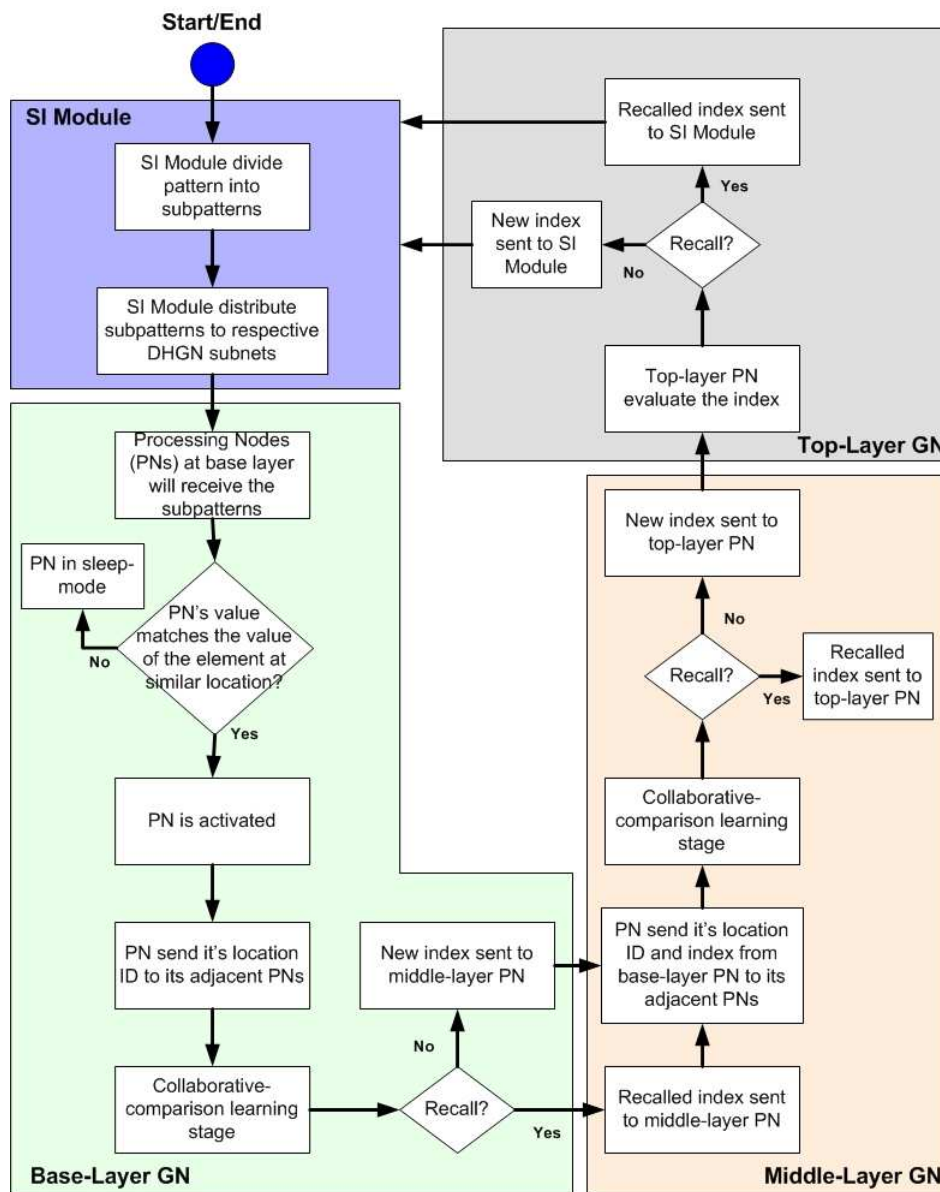


Figure 3.3: DHGN pattern recognition process workflow. This diagram represents DHGN network with 3-layer subnets.

Phase 2 - Pattern Reconstruction and Recognition

Recognition results obtained by SI module from all subnets within a DHGN network require further analysis to derive an overall recall of respective input subpattern. In accommodating this analysis, there are two different methods that have been considered. These are recall-percentage and voting methods. These two methods differ in terms of the mechanism being adopted. This research intends to compare and contrast these two approaches from an accuracy perspective.

Recall-Percentage Method. The recall-percentage method underlines the use of bias indices obtained from all GNs within each subnet. The main principle of this approach is that the recall/store decision is mainly based upon the cumulative decisions of all GNs within the network.

This method requires additional procedure conducted by each DHGN subnet for index collection before final recognition result being submitted to SI module. For each subpattern introduced into the subnet and after all the recognition process have been completed, the activated top GN will collect all the index information from all GNs underneath it. These indices will then be compiled and structured with the format $\{index : count\}$. These outputs will then be sent to SI module using message format $\{subnet_{id}; (index_1 : count_1), (index_2 : count_1), (index_3 : count_1), \dots, (index_n : count_n)\}$ for all n indices recalled or stored.

Some of the advantages of recall-percentage implementation for recognition at pattern level include its high recall value precision, in terms of the percentages of pattern indices being recalled. In this context, for a given input pattern, DHGN is capable of presenting its precise recall value. DHGN also has a capability to analyse pattern composition, based on the previous input patterns that have been stored within the network.

Apart from its advantages, the recall-percentage method also comes with a number of limitations. These include its effect on DHGN recognition accuracy, as described in Section 2.7.2. A nature of DHGN recognition process implies that a slight change in the structure of the subpattern will affect the index calculation of the entire subnet.

The recall-percentage method also raises the issue on the level of confidence of the outputs of the system. For instance, assuming a recognition output of a pattern obtained from a DHGN network consists of three patterns previously stored: $P1\ 0.4; P2\ 0.3; P3\ 0.3$; the result of this recognition will favour $P1$ as the recalled pattern. However, criteria

of $P2$ and $P3$ have also been detected in the pattern. Therefore, there is a need to establish a level of confidence towards this kind of results in recognition perspective.

Voting method. Most of the existing pattern recognition schemes apply rejection technique to remove highly-distorted patterns in its classification procedure. This technique adopts rejection/accuracy rate as a parameter to indicate levels of similarity of patterns. The technique offers a precise mean to obtain good classification measurement. However, it is mostly suitable for deployment within a single-decision system, in which the classification is conducted using a single classifier/recogniser. With an introduction towards distributed pattern recognition and/or classification, an important decision-making mechanism is needed in order to combine all the decisions (in terms of accuracy/rejection) made by each of the classifier.

One of the possible methods for combining decisions on classification is by using voting method. There are several forms of voting available in literature. These include majority, common-consent, unison, and unanimity voting (Battiti and Colla, 1994; Kuncheva, 2004). In DHGN implementation, majority voting is used as a mean to obtain a combined decision on the recalls made by each of the subnets within a recognition network.

For each recognition process, a decision whether the input pattern has been recognised (i.e. recall) or new to the network (i.e. store) is determined by obtaining majority consent from all the DHGN subnets. In this perspective, for a pattern to be recalled, the network should confirm that most of the subpatterns belong to the respective input pattern. The majority voting concept that has been adopted follows the work by Cruz, Sossa and Barrón (2007), and has been described in (Muhamad Amin and Khan, 2008b).

In this pattern reconstruction and recognition process, SI module will initially receive all the results of the recognition at subpattern level in the form of signal messages from all the DHGN subnets. After all these messages have been received, the actual recognition process is carried out. There are two stages involved at this level.

- i. All the indices received from the DHGN subnets for original patterns are stored in a 2-dimensional vector matrix $S = \{s_{11}, s_{12}, \dots, s_{mn}\}$. The width of the matrix is equivalent to the size of the pattern, i.e. m , while the height corresponds to the number of stored patterns, n .

- ii. Calculate the frequency of the indices for each test pattern. All the indices for the test pattern are stored in a vector $R = \{r_1, r_2, \dots, r_m\}$. The width of the matrix is also equivalent to the size of the pattern. If an entry in vector R gives the list of indices as $\{1, 2, 2, 2, 1\}$, then this indicates that three subnets have given a recall result of pattern 2 while two subnets have given a recall result of pattern 1. Therefore, by using the voting approach, the pattern will be recalled as pattern 2.

To describe the voting mechanism used, a simple pattern recognition problem is shown as follows. Figure 3.4 shows four binary character images: A , E , U , and a distorted version of A .

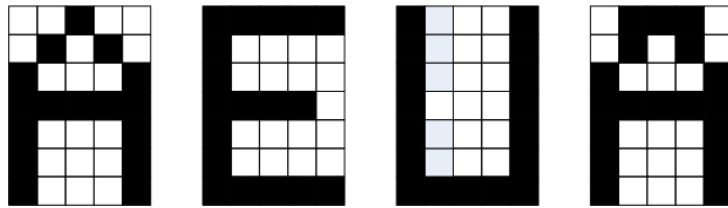


Figure 3.4: Samples of binary character images.

Consider P_A , P_E , P_U , and P_A^d represent the binary patterns for character A , E , U , and the distorted image A , as shown below:

$$P_A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad P_E = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$P_U = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad P_A^d = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Each pattern will then be decomposed into subpatterns and will be sent to DHGN subnets for the first level recognition process. In this example, each character pattern is decomposed into seven subpatterns where each subpattern represents a row of binary values, as shown below:

$$P_A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{aligned} P_A^1 &= (00100) \\ P_A^2 &= (01010) \\ P_A^3 &= (10001) \\ P_A^4 &= (11111) \\ P_A^5 &= (10001) \\ P_A^6 &= (10001) \\ P_A^7 &= (10001) \end{aligned}$$

The results of the recognition process at subpattern level will be sent back to SI module node. These results in the form of recalled/new indices for each subnet, k will be received by SI module node and represented as a voting matrix V as shown in Table 3.1.

		DHGN Subnets						
		k_1	k_2	k_3	k_4	k_5	k_6	k_7
Patterns	P_A	1	1	1	1	1	1	1
	P_E	2	2	2	2	2	2	2
	P_U	3	3	3	3	1	1	2
	P_A^d	4	1	1	1	1	1	1

Table 3.1: Recalled indices retrieved from all DHGN subnets after each pattern input.

The results of the recognition processes, show that when character pattern A is introduced, all subnets response with index 1. This shows that all subnets agreed that this is a newly stored pattern. Similarly, when pattern E is being introduced, all subnets give

feedback with an increase in the index value, i.e. index 2. Consequently, pattern U obtain various results from DHGN subnets. Four out of seven subnets produce a new index, while two subnets recalled index of pattern A and one gives index of pattern E. In this case, the maximum number of recalled/new indices will be chosen as the recalled/new pattern. Similarly, for distorted pattern A, the highest number of index being recalled is index 1 that correlates with pattern A. Therefore, pattern A is recalled.

Consider that P is an array of stored patterns $P = \{p_1, p_2, p_3, \dots, p_m\}$, where m represents the number of patterns being stored. For any pattern p_x to be recalled, maximum vote $V_{max}^{p_x}$, should be obtained using the following equation:

$$V_{max}^{p_x} = \arg \max(w_x), \quad x \in m \quad (3.8)$$

Where w_x represents the voting element of pattern p_x in voting vector W_P . It may be noted from the steps involved in pattern recognition using DHGN approach, the recognition process for each pattern occurs in a single-cycle containing a fixed number of steps. Also, DHGN adopts an unsupervised learning approach where no prior training on pattern data is required. A discussion on extended analysis of voting mechanism used in DHGN and how it affects the accuracy of DHGN recognition scheme can be referred to in Appendix A.

3.1.4 Bias Array Design

In DHGN implementation, patterns are stored in the form of association between its elements. This is somewhat different from other neural network approaches, in which patterns are stored as composition of values. Pattern storage mechanism adopted by DHGN is in the form of bias array, similar to the techniques used by GN and HGN approaches as described in Chapter 2. However, the bias array capacity for DHGN might be different from HGN and GN. Further discussion on this aspect will be presented in Section 3.3. Figure 3.5 shows an abstract representation of GN node with its storage structure.

DHGN minimises the storage requirement for input patterns, in the sense that the bias array design limits the growth of storage element within each GN, through the use of *Index{left, right}* format of bias entry for one-dimensional input patterns. Consider a

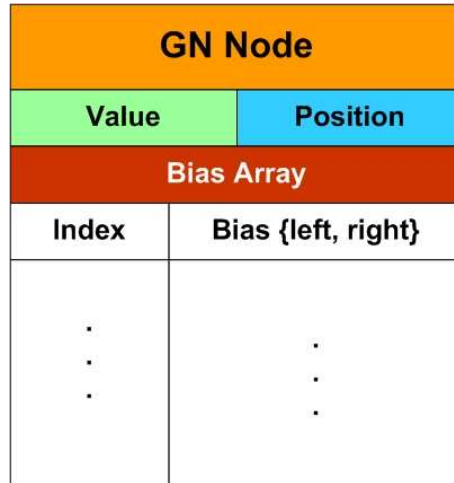


Figure 3.5: GN node abstract representation showing its storage framework.

comparison between DHGN bias entry and feed-forward neural network storage requirement capacities for each neuron, given different binary pattern sizes used in the networks.

In feed-forward network, each neuron requires input from all the elements within a particular pattern. Given a pattern P with n input elements (i.e. size) and d dimension, each neuron must be able to memorise d^n combinations of patterns. Conversely, DHGN only requires $2d$ storage capacity for each neuron for memorisation. In this perspective, DHGN offers significantly higher storage efficiency as compared to feed-forward neural network. Further discussion and evaluation of the storage capacity of DHGN will be discussed in later sections.

3.1.5 Collaborative-Comparison Learning (CCL)

DHGN implements a single-cycle learning approach in its recognition procedure. This learning approach differs from other learning approaches such as Hebbian and incremental learning in the sense that it implies that learning occurred through collaborative learning between nodes, rather than independent learning by each of the processing nodes in the network. The term we used for this collaborative learning is known as Collaborative-Comparison Learning (CCL). The content of this subsection has been included in (Muhamad Amin and Khan, 2009).

In DHGN implementation, an adjacency comparison approach is employed in the learning scheme using simple signal/data comparisons. Each GN node holds a segment of the overall subpattern. Collectively, these neurons will have an ability to represent the entire

subpattern. Consider the following base-level DHGN subnet structure as shown in Figure 3.6. The five GNs, where each is responsible to capture its adjacent neurons' values, will be able to store the entire pattern "ABCDE". If we link up these neurons in a one-dimensional structure, we are able to determine collaborative GNs that contain a memory of pattern "ABCDE".

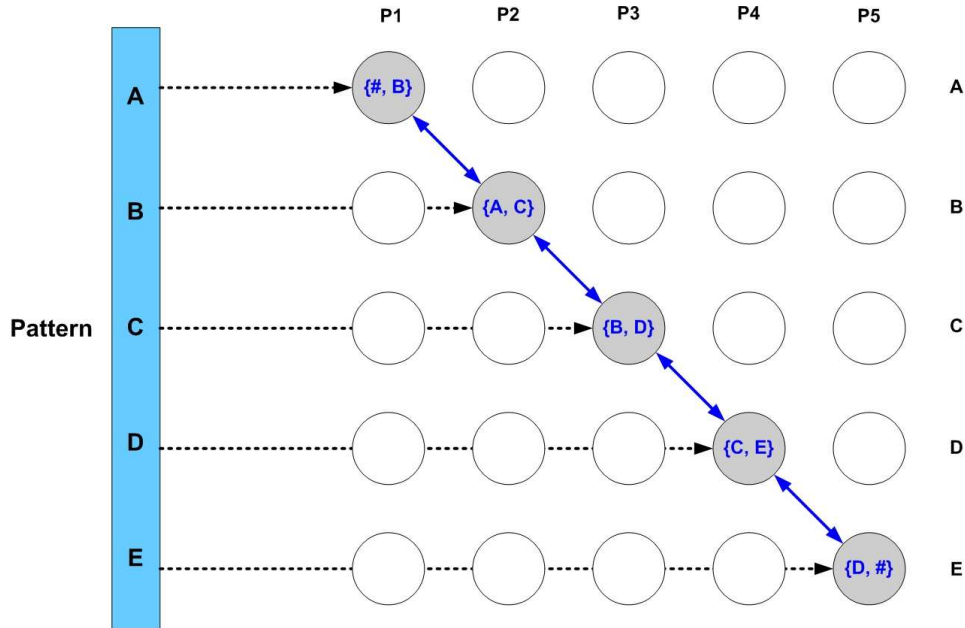


Figure 3.6: Collaborative-comparison learning approach for one-dimensional pattern "ABCDE". Each activated graph neuron (GN) stores the signals received by its adjacent neurons.

The collaborative-comparison learning approach compares external input pattern with the stored entries within each GN's bias array, which is a local data structure containing history of adjacent node activation. In this perspective, each GN learns through comparisons among the signals from its adjacent neighbours and the recorded entries within its memory i.e. the bias array. Consider a bias array $M = (s_1, s_2, \dots, s_x)$ which consists of signal entries s_i for $i \in x$. If external signal s_{ext} matches any of the stored entries, i.e. $s_{ext} \in M$, then the respective bias index i of the matched s_i entry will be recalled. Otherwise, the signal will be added into the memory as s_{x+1} . There are two-fold advantages using this approach. Firstly, it minimises data storage requirement, using the bias array design for pattern storage. Secondly, the proposed approach accepts all kinds of data. For instance, the signal could be in the form of data vectors or frequency signals, allowing spatial and temporal data to be accommodated. In addition, the proposed learning technique

does not require synaptic plasticity rule used by other learning mechanisms, such as the Hebbian learning (Hebb, 1988) and incremental learning (Schlimmer and Granger, 1986) approaches. Thus new patterns can be learnt without affecting previously stored information.

3.2 Dimensionality Reduction in Pattern Pre-Processing

Pre-processing is an important task that needs to be carried out before any recognition procedure. It is considered as a pre-requisite for some pattern recognition systems, due to its critical influence in ensuring that pattern data is in the specific form that suits the algorithm or implementation. Moreover, raw pattern data might need to be normalised beforehand, to ensure that the data is well-distributed and does not contain any outlier values.

When dealing with complex data, such as images, environmental sensory readings, biomedical and biochemical structural data, the dimensions of data involve usually are at higher dimensions (more than 1). In this perspective, there are two different approaches that could be carried out to reduce the complexity of data, in terms of its dimensionality:

- i. Structural reduction - In this approach, the structure of data will be reduced into lower dimension.
- ii. Content reduction - Data in higher dimension is reduced into its equivalent low-dimension form using a specific data dimensionality reduction technique.

In this section, these two approaches are discussed in relation to DHGN implementation.

3.2.1 Structural Reduction

Structural reduction in DHGN pre-processing involves the reduction of structural composition of patterns from high-dimensional structure, into its corresponding low-dimensional representation. In this approach, pattern data undergoes structural deformation, while the contents or elements within the pattern remain intact. Furthermore, structural reduction works on the basis that the structure of data is unlikely to be significant in determining the characteristics of pattern.

Consider two-dimensional binary images with the size of 7-by-5 bits, i.e. 35-bit image as shown in Figure 3.7. In the structural reduction approach, this image will be rearranged in the form of one-dimensional bit-string. This rearrangement enables the algorithm to work on patterns in low structural dimension. Hence, in the DHGN's implementation perspective, this approach enables each subnet to conduct recognition process using a simple one-dimensional DHGN subnet structure. Therefore, it reduces the structural complexity of DHGN subnets within the network. An advantage of using this structural reduction approach is such that it reduces the structural complexity of patterns, while maintaining the integrity of the contents or elements within these patterns. Hence, the content information in each pattern is preserved.

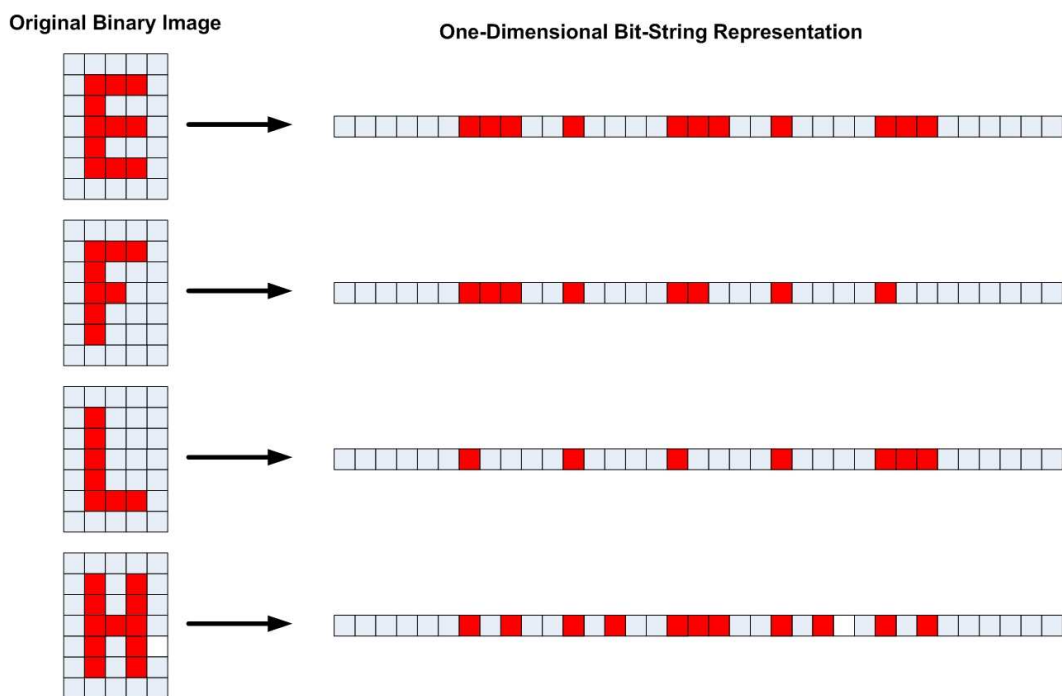


Figure 3.7: Structural reduction on binary character images into one-dimensional bit-string representation.

A limitation with this approach is such that it loses the structural information related to the pattern. In this context, the structure of the pattern or data is unknown to the system. Consider the same images as in Figure 3.7. DHGN pattern recogniser does not have the knowledge that the image represents a character 'E'. Rather, it acknowledges the bit information and its association between neighbouring pixels in one-dimensional formation.

A test was conducted to examine the effectiveness of structural reduction approach for DHGN pre-processing scheme for both randomly and structurally distorted patterns. The results of this analysis have been published in (Khan and Muhamad Amin, 2007). Figure 3.8 shows the datasets that have been used and Figure 3.9 shows the results obtained from the recognition procedure conducted using DHGN on one-dimensional binary-bit string representation of the images.

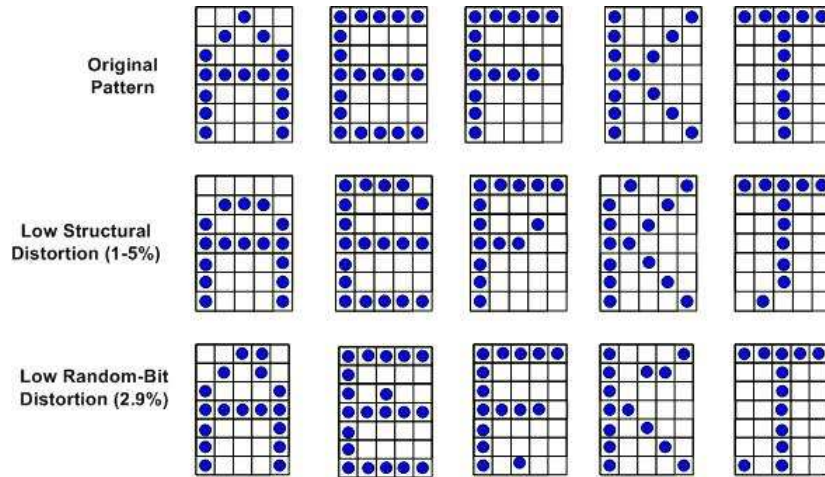


Figure 3.8: Character patterns with structural and random distortions.

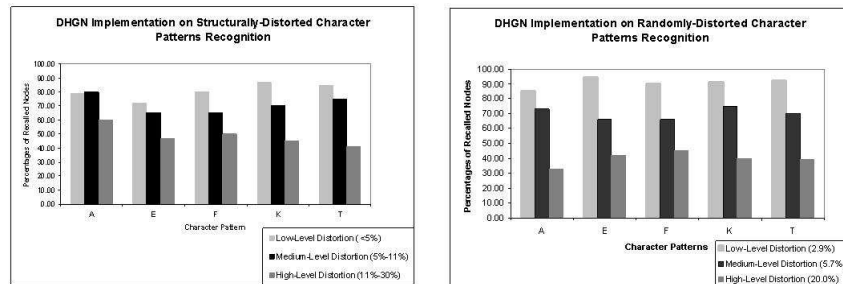


Figure 3.9: Results of DHGN pattern recognition on structural and random distorted character patterns.

Note that for recognition involving structural reduction technique, the results show that structurally-distorted patterns are less likely to be recognised by the system than the randomly-distorted patterns. This is owing to the fact that structural distortion affects the shape of the character. Whilst with random-bit distortions, the structure of the characters often stays preserved. In this perspective, DHGN with one-dimensional pattern representation was unable to detect the changes in the structure of the character, due to its

one-dimensional representation. Nevertheless, DHGN shows significantly accurate results with average recall accuracy of 80% for structurally distorted patterns.

3.2.2 Content Reduction

Content reduction, more generally known as dimensionality reduction approach, involves the process of selection or extraction of features from data, to be used in pattern recognition system. It also transforms the data from high-dimensional space into its equivalent low dimension format. Some examples of dimensionality reduction techniques include Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA), Local Linear Embedding (LLE) and Kohonen maps.

Dimensionality reduction approach allows the recognition system to obtain the best and cost-efficient data representation that has been extracted from the original raw data obtained from sensory devices or from surroundings. However, some dimensionality reduction techniques require such an expensive computational cost for feature processing, selection, and extraction.

Apart from these techniques, there are other techniques that have been proposed for dimensionality reduction, which incur significantly low-level of computational complexity. For instance, in content-based image retrieval (CBIR), the use of histograms and signatures are the common approaches towards dimensionality reduction for image retrieval using colour feature. This research specifically looks into signature scheme for data dimensionality reduction, based upon the works that have been carried out in CBIR using signature approach by Nascimento and Chitkara (2002).

In the next subsection, a case study on the DHGN implementation using a binary signature scheme for image recognition is presented.

3.2.3 Case Study: DHGN Image Recognition based on Binary Signature

Existing DHGN implementation has been focusing on the recognition of spatio-structural representation of an image via the pixel-by-pixel analysis. This approach recognises the integrity of the contents of an image against any occurrence of random-bit distortion. However, it is insufficient for the recognition of image with multi-dimensional colour representation, including grayscale images. The changes in the colour of an image may influence accuracy of the recognition system. Our proposed approach adopts the binary signature

scheme for content-based image retrieval (CBIR) in the colour recognition process, while maintaining the binary analysis of image for its spatio-structural recognition.

Binary Signature for Colour Abstraction

In image recognition, colour is an important element that usually being considered. In recognition perspective, colour enables us to differentiate two different objects. For instance, imagine two balls coloured blue and orange respectively. An addition of colour dimension in recognition enhances the accuracy of the proposed recognition scheme. The content of an image could abstractly being represented by its colour distribution. Thus, image recognition should also involve colour distribution analysis of a particular image.

Common approach in representing colour distribution within an image is to use a Global colour Histogram (GCH). Given an n -colour model, a GCH will be developed with an n -dimensional feature vector (p_1, p_2, \dots, p_n) , where p_i represents the normalized percentage of colour pixels that corresponds to each colour element c_i within an image. Chitkara, Nascimento and Mastaller (2001) has proposed an alternative towards colour distribution representation by using a binary signature scheme. Binary signature is a compact form of existing GCH that uses binary bit-string as a signature that provides an abstract representation of the image's colour distribution. The bit-strings used in this format are of a pre-determined size. The following section describes the image colour distribution abstraction using binary signature.

The use of GCH in colour distribution representation has been directed towards finding colour within an image having significant pixel dominance. However, this differs from the binary signature approach that puts great emphasize on less dominant colour. The justification for applying this concept is that less dominant colours usually differ from one image to another. For example, an image of a person with a given background might change, with similar background. The following scheme shows the image abstraction used in this approach:

- i. Image is quantized into a fixed number of n colours, $C = (c_1, c_2, \dots, c_n)$.
- ii. Each colour element c_i is then discretised into j binary bins $B^i = b_1^i b_2^i \dots b_j^i$ of equal or varying capacities. This is referred to as bin-size. Fixed-size bin arrangement is known as Constant-Bin Allocation (CBA), otherwise it is known as Variable-Bin

Allocation (VBA). The binary signature of an image is derived from both bins and colour values, and is represented in a bit-string format. For instance, consider an image comprising of n colours and j bins. The binary signature of this image, S could be represented as $S = b_1^1 b_2^1 \dots b_1^2 b_2^2 \dots b_j^n$, where, b_k^i represents the k -th bin relative to the colour element c_i .

- iii. Each bin would correspond to the normalised percentage values of colour within an image. For example, in CBA approach $j = 5$ bins will each corresponds to percentage values as shown in Table 3.2:

b_j	Colour Composition Value
1	0-20%
2	21-40%
3	41-60%
4	61-80%
5	81-100%

Table 3.2: Colour composition values correspond to each bin b_j .

In this section, we present an example of image abstraction. Figure 3.10 shows an image with 4 different colours.

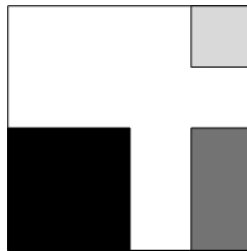


Figure 3.10: Block image with four different colours.

In this example, $n = 4$, hence, $C = (c_1, c_2, c_3, c_4) = (\text{white}, \text{black}, \text{grey}, \text{light grey})$. The normalised colour densities of this image could then be represented by the vector $H_n = (h_1, h_2, h_3, h_4) = (0.56, 0.25, 0.13, 0.06)$, where h_j represents the percentage pixel dominance of colour c_i . Assume that the colour distribution is discretised into $j = 5$ bins of equal capacities as shown in Table 3.2. Thus, the image could be represented by the following signature $S = 00100010001000010000$. The details of the signature are shown in Table 3.3.

Colour	Density	Binary Signatures				
		b_1	b_2	b_3	b_4	b_5
c_1	56%	0	0	1	0	0
c_2	25%	0	1	0	0	0
c_3	13%	1	0	0	0	0
c_4	6%	1	0	0	0	0

Table 3.3: Detailed binary signatures for image in Figure 3.10.

Binary Signature within DHGN implementation

In DHGN implementation for image recognition, we can use the binary signature scheme to detect and recognize the colour distribution of an image. Each sub-signature (each signature that represents each colour c_i) can be entered into a single DHGN subnet. Cumulatively, this approach will lead to colour recognition within an image. Figure 3.11 shows the processes involved within this colour recognition process.

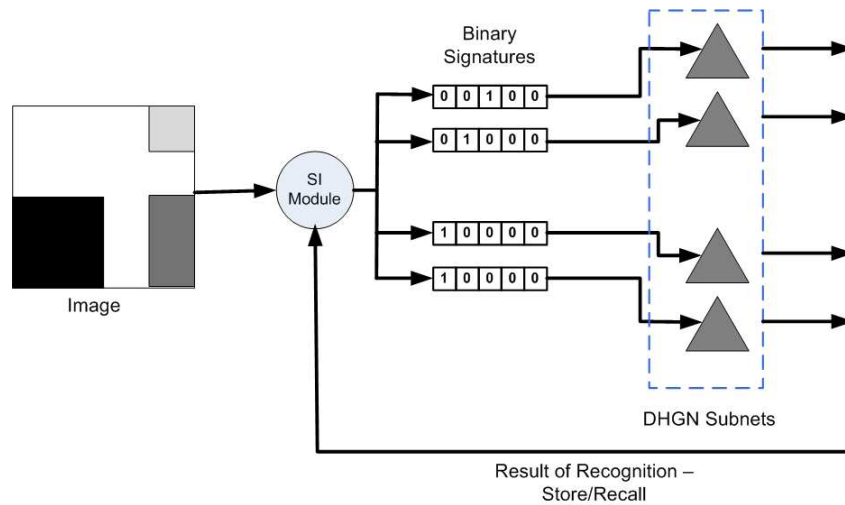


Figure 3.11: DHGN implementation for colour recognition using binary signature.

The example given previously has shown the method of translating colour distribution of an image into a single binary signature representation. However, there is a limitation in this approach, where the coverage area for the colour distribution is significantly large, and does not able to represent a slight or minimal changes within a section of an image. Thus, this will affect the result of the recognition. Local binary signature approach could be used, in which each image will be divided into grids and each grid will have its own signature, as shown in Figure 3.12.

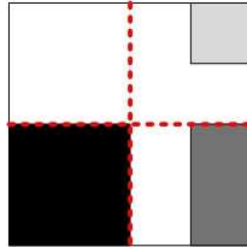


Figure 3.12: Block image is divided into grids with equivalent sizes.

Image in Figure 3.12 will have four different signatures, representing each quadrant of the image (each quadrant is numbered clockwise, starting from top-left quadrant).

Table 3.4 shows the detailed signatures of the block image:

Quadrant	Colour	Density	Binary Signatures				
			b_1	b_2	b_3	b_4	b_5
1	c_1	100%	0	0	0	0	1
	c_2	0%	0	0	0	0	0
	c_3	0%	0	0	0	0	0
	c_4	0%	0	0	0	0	0
2	c_1	75%	0	0	0	1	0
	c_2	0%	0	0	0	0	0
	c_3	0%	0	0	0	0	0
	c_4	25%	0	1	0	0	0
3	c_1	50%	0	0	1	0	0
	c_2	0%	0	0	0	0	0
	c_3	50%	0	0	1	0	0
	c_4	0%	0	0	0	0	0
4	c_1	0%	0	0	0	0	0
	c_2	100%	0	0	0	0	1
	c_3	0%	0	0	0	0	0
	c_4	0%	0	0	0	0	0

Table 3.4: Detailed binary signatures of block image in Figure 3.12.

With localised signatures, the colour distribution representation of an image will be further optimized and able to provide higher possible recall precision for a given set of images.

Results and Discussion. A greyscale image recognition test has been conducted using DHGN on binary signature scheme for 40 images with a dimension of 512 x 512 pixels. In doing this, the results of this scheme are compared with existing support vector machine (SVM) implementation. The results from this test show that DHGN with binary signature scheme produces comparable results to SVM implementation. Figure 3.13 shows a sample

dataset, comprising of image Lena that has been used, together with their respective global color histograms. The rest of the datasets are included as Appendix A.

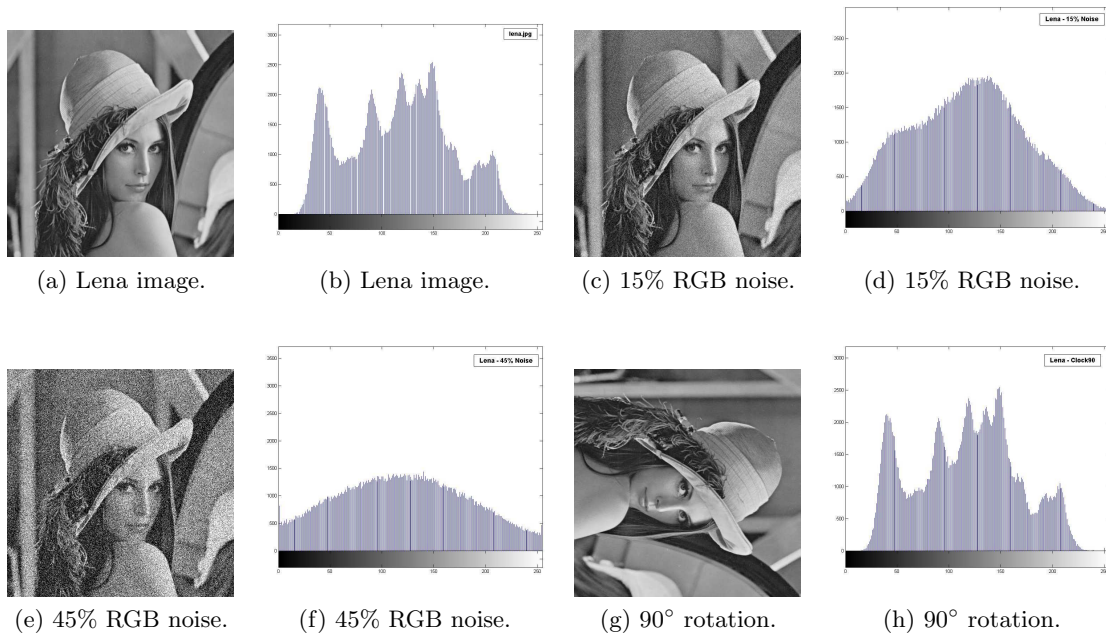


Figure 3.13: Different levels of noise and rotational distortion for greyscale image of Lena with respective colour histograms, used in the image recognition test.

The images used in this test are classified into five images classes namely image Baboon, Camera, Goldhill, Lena, and Peppers. Each class consists of an original image, five noisy images (RGB noise has been selected for the noise effect), and two rotationally-distorted images. The recognition test is used to recognize and classify these images into their respective classes. This therefore tests these images using both DHGN and SVM pattern recognition schemes.

With the added RGB noise to original image, the colour distribution was also being affected. RGB noise tends to distribute the colour composition of an image, in which median colour has higher pixel ratio over extreme colours. This effect will somehow influence the image pattern, as each colour composition values change.

In this recognition test, variable color compositions were used according to the quantization level used on the images. Figure 3.14 shows the total recall and error rates for the recognition test that were conducted on 40 images with 5 original images for training. These rates have been derived from the number of images that have been classified correctly, according to the classes specified previously.

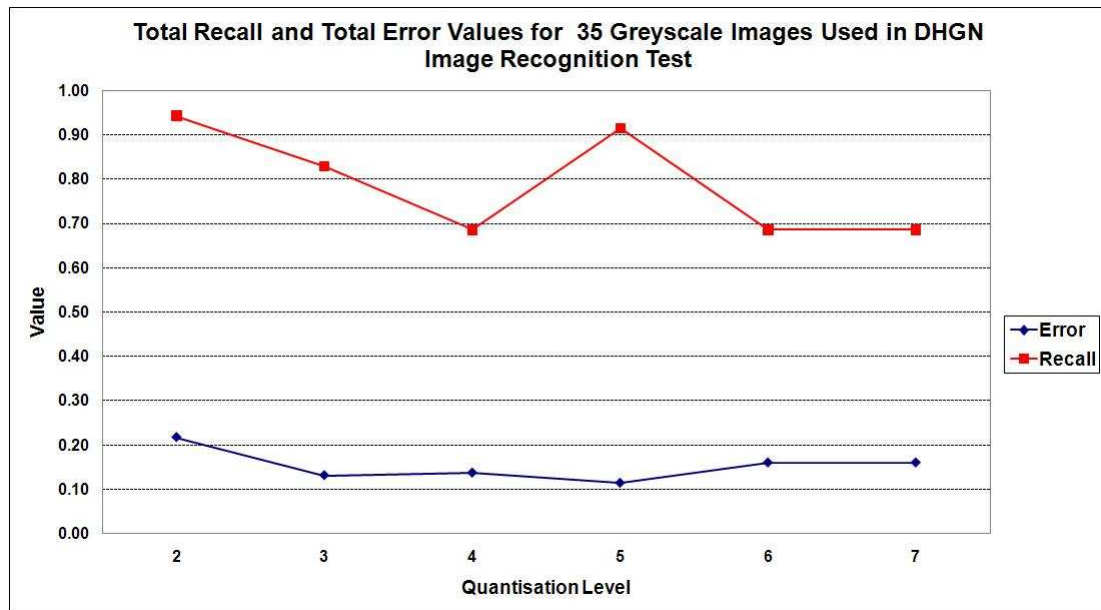


Figure 3.14: Total recall and error rates for DHGN greyscale image recognition on 40 images using colour feature analysis.

From this result, it is evident that the quantisation level affects the recognition accuracy of the scheme. At level 2 (binary level), colour composition produces highest recall value. DHGN was able to fully detect positive images from the whole test set. Nevertheless, error value recorded was 0.22, which is higher than the values obtained using other quantisation levels. This may be due to the effect of low quantisation level that produces closely-similar set of binary signatures for different images. Alternatively, high quantisation level would also give an insignificant effect, as it tends to distribute colour frequency to high number of colour classes and thus, reducing the possibilities of colours to be grouped into similar classes. Therefore, color recognition of images tends to be difficult as each colour composition has a small range of colours. Figure 3.15 shows the transformation of global color histogram for image Baboon from original image to 3-quantisation level image.

From the results of this recognition test, for the existing dataset, the recognition scheme works best with quantisation level 3. In order to measure the efficiency of the proposed scheme, a comparative test between DHGN with global binary signature scheme and SVM pattern recognition scheme has been made. The recall values for both schemes, with given quantisation level, are shown in Figure 3.16. A radial-based kernel SVM was used for this test.

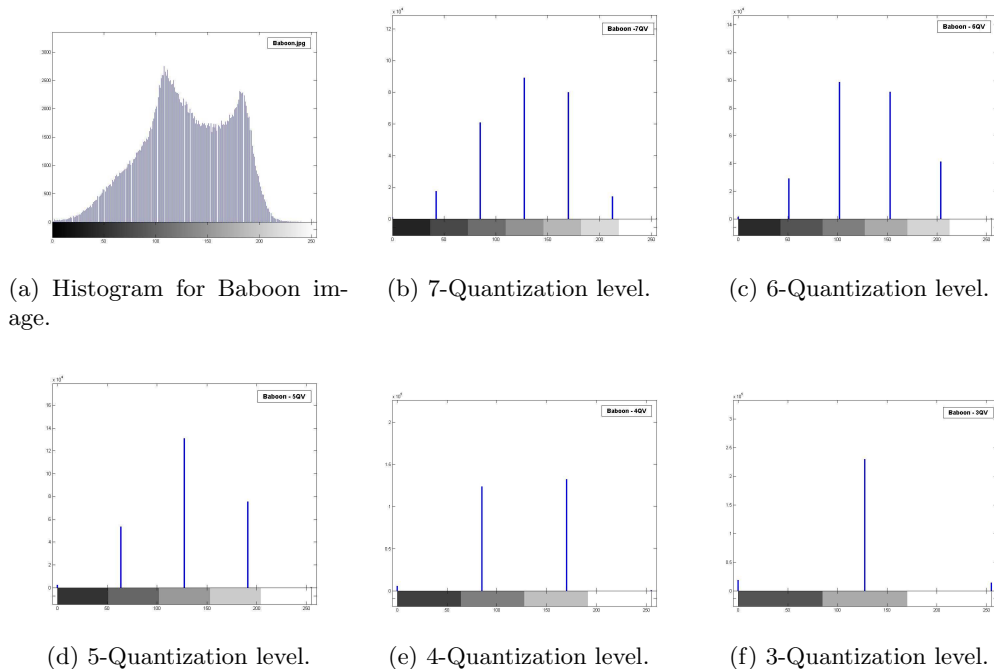


Figure 3.15: Transformation of global colour histogram of image Baboon from original image to 3-quantisation level image.

The recognition test conducted has shown that DHGN produces higher recall values, in comparison with SVM classifier. This is mainly due to the feature of DHGN recognition which implies simple deterministic approach by using differences between the properties of images. Furthermore, DHGN achieves this recall accuracy via a single-cycle learning and one-shot recognition process.

The considerably-low recognition accuracy influenced by the global colour composition within the proposed DHGN image recognition scheme could be further enhanced using a localised approach. Further research will be carried out to analyse the effect of localisation of colour composition on greyscale image recognition.

Another important aspect that has been observed with this binary signature scheme is that it eliminates the need for pixel-by-pixel image recognition. Each image will have the same signature size, regardless of the size of the image.

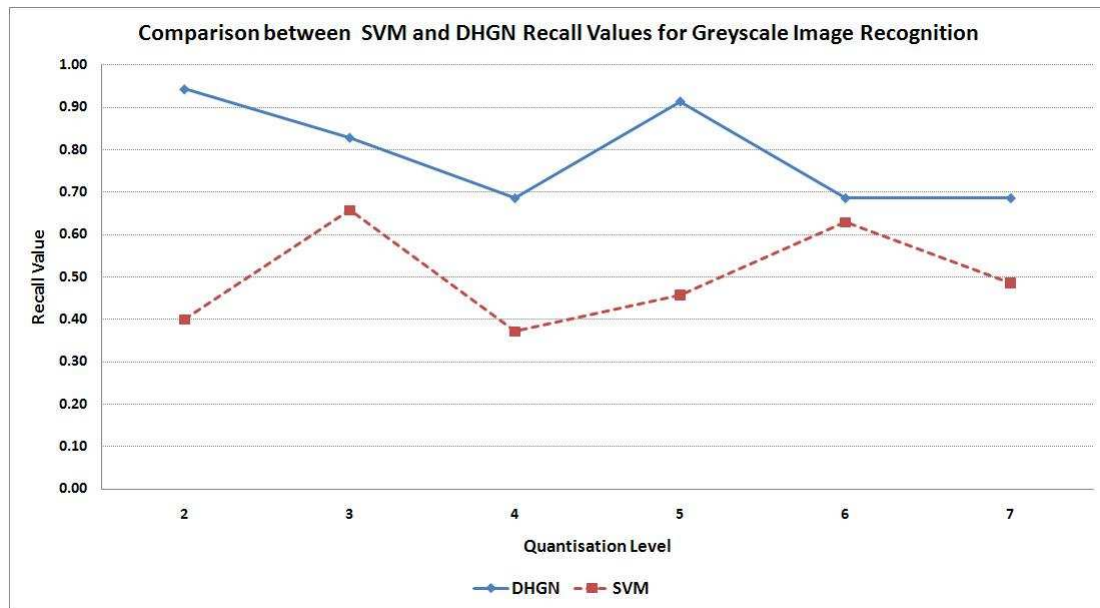


Figure 3.16: Comparison between SVM and DHGN total recall rates for greyscale image recognition.

3.3 Analysis and Evaluation

A series of analyses and evaluations for DHGN implementation were conducted. These evaluations focus on both the complexity and scalability of the proposed algorithm. The following subsections described in details the analyses that have been carried out.

3.3.1 Complexity Evaluation

In regards to the complexity of DHGN algorithm, Big-O analysis has been considered as the computational complexity indicator. It is mostly used as the computational complexity measurement tool to describe how the input data affects an algorithm's usage of computational resources. According to Black (2008), Big-O analysis is defined as a theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n , which is usually the number of items. Informally, saying some equation $f(n) = O(g(n))$ means it is less than some constant multiple of $g(n)$. Comparisons have been made with three different algorithms, namely Hopfield network, Kohonen Self-Organising Map (SOM), and instantaneously-trained neural networks (ITNNs). It is best to note however, that the comparative study that has been carried out does not intend to outweigh the capabilities of these algorithms. Rather, indicates that DHGN has a capacity to acquire significantly low computational complexity for its operations. A

number of papers have been published, showing the results of the evaluations, including (Muhamad Amin, Raja Mahmood and Khan, 2008; Raja Mahmood et al., 2008).

DHGN vs. Hopfield Network

In determining the complexity of the algorithms, a 2-stage process was implemented; network generation and recognition stages. The notations for each stage within the implementation have been derived.

Network Generation Stage. This stage involves the formation of a network that comprises computing elements known as neurons. The number of neurons generated depends on the algorithm being implemented.

In Hopfield network implementation, the number of neurons generated, R_{Hopf} is equivalent to the size of the pattern P_{size} :

$$R_{Hopf} = P_{size} \quad (3.9)$$

On the other hand, the number of neurons generated, R_{DHGN} in DHGN implementation with number of different elements within the pattern v and a number of DHGN subnets, N_{DHGN} is given in this equation:

$$R_{DHGN} = v \left(\frac{\frac{P_{size}}{N_{DHGN}} + 1}{2} \right)^2 \times N_{DHGN} \quad (3.10)$$

However, during network generation stage, only the base layer neurons of each DHGN subnet, R_{DHGN}^{init} will be initialised as shown in Equation 3.11.

$$R_{DHGN}^{init} = v \times P_{size} \times N_{DHGN} \quad (3.11)$$

Table 3.5 shows the details of the Big-O notation derived for the Hopfield network and DHGN implementations.. The estimated time derived is based on the assumption that the instruction speed used is 1 microsecond (μs) per instruction.

Algorithm	Big-O	Efficiency	Iterations (n)	Estimated Time (in seconds)
Hopfield	$O(n)$	Linear	R_{Hopf}	$R_{Hopf} \times 0.00001$
DHGN	$O(n)$	Linear	R_{DHGN}^{init}	$R_{DHGN}^{init} \times 0.00001$

Table 3.5: Big-O notations for Hopfield network and DHGN implementation in network generation stage.

The results show that both DHGN and Hopfield network acquires comparable computational complexity. However, in regards to the number of neurons generated at this stage, DHGN incurred higher complexity since $R_{DHGN}^{init} > R_{Hopf}$. However, if parallelism is taken into account, for each DHGN subnet, the number of neurons generated for each subnet is less than the overall neuron initialisation within the network. Therefore, the estimated time for network generation in DHGN is lower than in the Hopfield network implementation.

Recognition Stage. Recognition stage is the core process within the pattern recognition application. Each algorithm shows different approach in handling this process. In the Hopfield network, the recognition stage involves three sub-processes, namely weight accumulation, weight determination for the whole network, and network propagation to derive optimum solution. On the other hand, DHGN algorithm only implies a single-cycle process of recognition within this recognition stage. This process of recognition involves either store or recall process. Table 3.6 shows the Big-O notations derived from the analysis on the Hopfield network recognition process. Similarly this is based on the assumption that the instruction speed used is 1 microsecond (μs) per instruction.

Process	Big-O	Efficiency	Iterations (n)	Estimated Time (in seconds)
Weight Accumulation	$O(n)$	Linear	P_{size}	$P_{size} \times 0.00001$
Weight Determination	$O(n^2)$	Quadratic	R_{Hopf}^2	In minutes
Network Propagation	$O(n^k)$	Polynomial	R_{Hopf}^k	In hours

Table 3.6: Big-O notations for Hopfield network in recognition stage.

The Hopfield network incurs a considerably high computational complexity, as indicated in Table 3.6 with respect to its weight determination and network propagation processes. Figure 3.17 shows the computational complexity for the three processes in recognition stage using the Hopfield network implementation. Note that for network propagation process, the value $k = 3$ was used for polynomial representation.

The recognition stage for pattern recognition using DHGN algorithm involves a single-cycle process in which each input pattern will be passed through the DHGN subnets once and the store or recall process will be activated according to the instruction given. Table 3.7 shows the Big-O notation for recognition stage using DHGN algorithm for each DHGN array being used.

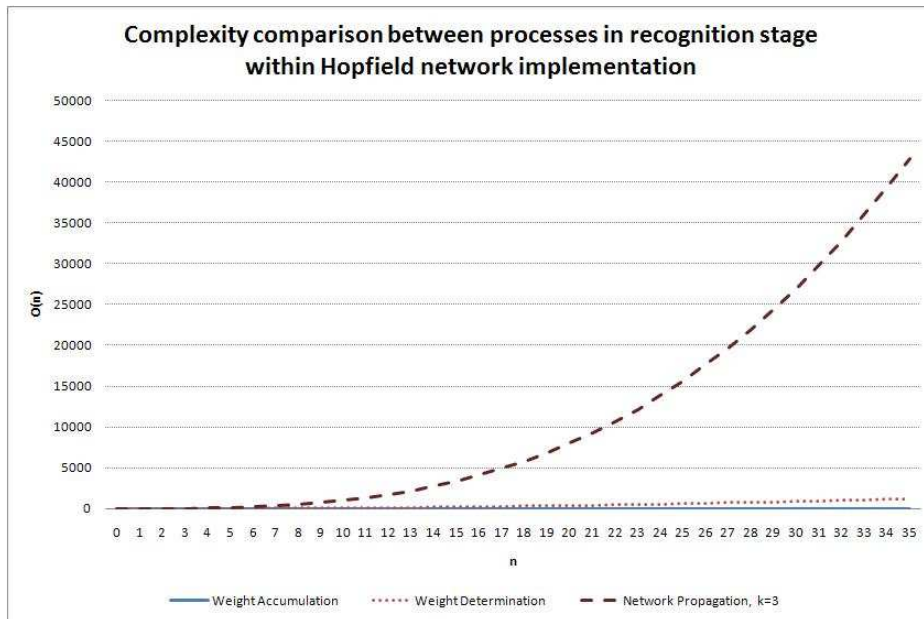


Figure 3.17: Big-O notation comparisons for processes within Hopfield network recognition stage.

Big-O	Efficiency	Iterations (n)	Estimated Time (in seconds)
$O(n)$	Linear	R_{DHGN}	$R_{DHGN} \times 0.00001$

Table 3.7: Big-O notation for DHGN implementation in recognition stage.

From the Big-O notations derived from the analysis, it is best to conclude that DHGN incurs less computational complexity in pattern recognition processes as compared to the Hopfield network implementation. Specifically, DHGN employs simple linear function, whereas the Hopfield network employs expensive polynomial and quadratic functions.

DHGN vs. Kohonen SOM

The Big-O notations for both SOM and DHGN have been estimated to study their complexity levels. The supervised SOM consists of three important stages: (i) weight initialisation, (ii) BMU calculation, and (iii) weight adjustment. In the weight initialisation stage, nodes are created with random assigned weight. At this stage, the computational complexity depends heavily on the number of created nodes. Hence, for a given weight initialisation process w , the complexity of n nodes can be simplified as $f(w) = O(n^3)$. Figure 3.18 shows the estimated time taken to initialise up to 100,000 nodes. The estimated time derived is based on the assumption that the instruction speed used is 1 microsecond (μs) per instruction.

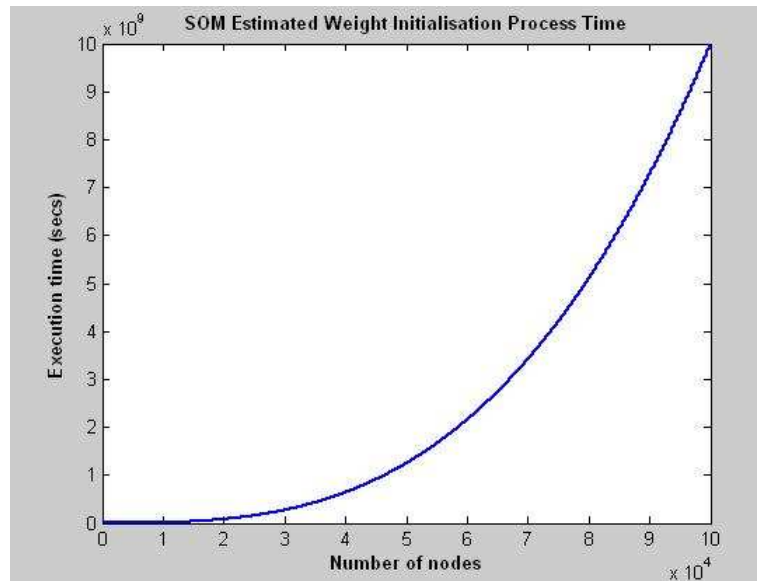


Figure 3.18: Complexity measurement of SOM's weight initialisation process.

In the BMU calculation stage, the complexity depends heavily on the number of iterations during training as well as the number of the input vector. Hence, for a given BMU calculation process m , the complexity of n training iterations can be simplified as $f(m) = O(n^4)$. The estimated time taken to perform up to 100,000 iterations of calculating the Euclidean distance between the input values and all neurons in this stage is given in Figure 3.19.

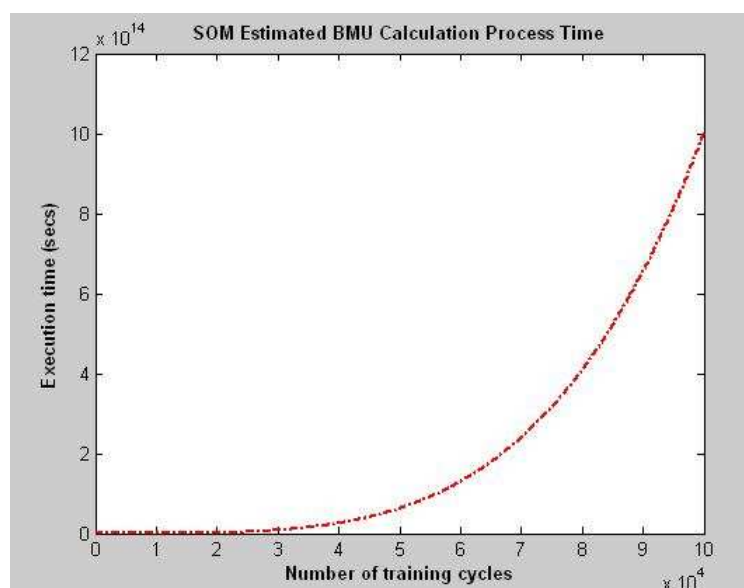


Figure 3.19: Complexity measurement of SOM's BMU calculation process.

In the last stage, the weight adjustments are provided not only for the winning neuron but also for its neighbours in a certain neighbourhood. The degree of adjustment depends on the degree of similarity between the neuron and the input. As a result of weight adjustment, a group of neurons are obtained forming a cluster. The Big-O for the weight adjustment is similar to the BMU calculation (refers to Figure 3.19) and hence not provided.

DHGN's initialisation stage, as discussed previously is a low-computational process, and hence acquires less computational time in comparison to SOM's weight initialisation process. Figure 3.20 shows the estimated time for this process. Similar speed assumption of 1 microsecond (μs) per instruction is applied in this analysis. It can be seen that the time taken in the DHGN initialisation process is far less than SOM. For instance, DHGN takes only 0.2 seconds while SOM takes about 1.0×10^{10} seconds (see Figure 3.18) to initialise 20,000 nodes.

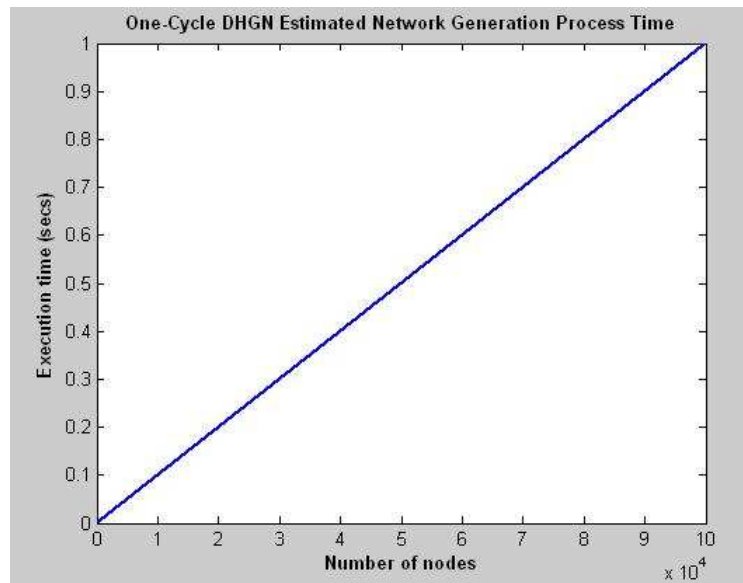


Figure 3.20: Complexity measurement of DHGN's network generation process.

In the classification process, only few comparisons are made for each subpattern, i.e. comparing the input subpattern with the subpatterns of the respective bias index. The computational complexity for the classification process is somewhat similar to the network generation process. DHGN's classification process requires less computational complexity in comparison to SOM's BMU calculation and weight adjustment activities. For instance, the time taken for classification by DHGN in a network of 50,000 nodes is less than 3 seconds as shown in Figure 3.21, while SOM's BMU calculation process alone takes about

5×10^{13} seconds to complete (see Figure 3.19), and with similar time needed to perform weight adjustment.

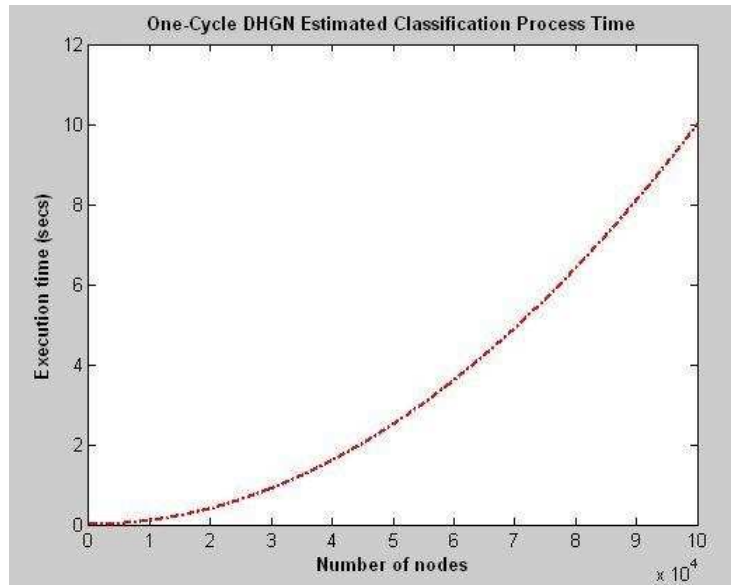


Figure 3.21: Complexity measurement of of DHGN’s classification process.

In summary, the estimated time graph of the DHGN algorithm is linear, while the corresponding graph of SOM algorithm is exponential. This proves that the DHGN provides an efficient, lightweight, and fast algorithm, comparable to SOM implementation.

DHGN vs. Instantaneously-Trained Neural Networks (ITNNs)

Instantaneously-trained neural network (ITNN) is a class of artificial neural networks that allow rapid learning for classification and generalisation problems. These networks have been developed based upon the motivation to model short-term memory in computational intelligence applications. This motivation has led to the development of ITNN-based networks, such as corner classification (CC) and fast classification (FC) neural networks (Kak, 2002). In principle, ITNN implements non-intensive computations in its learning mechanism, and involves linear mapping between patterns and networks. In terms of network architecture, ITNN follows a basic feed-forward network structure, consisting of neurons within input, hidden, and output layers.

The learning phase of FC neural network (an example of ITNN) involves a two-stage process:

- i. Synaptic weights assignments for input and output layers.

- ii. Radius generalisation for each training sample.

The complexity of learning mechanism applied in FC neural network is highly-dependable on the radius generalisation process. This process takes place in order to find the maximum class boundary for each training sample. In FC network, the number of neurons within hidden layer is equivalent to the number of training samples. This is a price that has to be paid, in order for the networks to perform fast learning mechanism.

There are two steps involved in radius generalisation process:

- i. Determine the Euclidean distances between each training sample.
- ii. Calculate radius associated with each training sample.

The computational complexity of finding Euclidean distances involves a non-linear process for each training sample, as shown in the following pseudocode:

Algorithm 3 Finding Euclidean Distances in FC Neural Networks

```

1: for  $i = 1$  to MaxNeuron do
2:   for  $j = 1$  to MaxNeuron do
3:     if  $i = j$  then
4:       break()
5:     else
6:       for  $k = 1$  to MaxInputWeight do
7:          $dist_{i \rightarrow j} = \sqrt{\sum (i.k - j.k)^2}$ 
8:       end for
9:     end if
10:  end for
11: end for

```

Using Big-O notation for computational complexity, the complexity of this Euclidean distance function is equivalent to $O(n^3)$, where n represents a single executable instruction within the function. Hence, $f(EucDist) = O(n^3)$.

When all the distances for each training sample have been determined, the training process for FC network will initiate the radius calculation step. In soft generalisation process, the radius is calculated as one-half of the minimum distance between a sample pattern, with other stored patterns. This step can be represented as a quadratic function $f(RadCal)$ shown in Algorithm 4.

In this case, the complexity of radius calculation function is $f(RadCal) = O(n^2)$.

DHGN, as shown in the previous subsections, entails low computational complexity in its collaborative-comparison learning (CCL) technique. This is comparable with ITNN

Algorithm 4 Radius Calculation Function in FC Neural Networks

```

1:  $minDist = 999$ 
2: for  $i = 1$  to MaxNeuron do
3:   for  $j = 1$  to MaxNeuron do
4:     if  $i = j$  then
5:       break()
6:     else
7:       if  $dist_{i \rightarrow j} \leq minDist$  then
8:          $minDist = dist_{i \rightarrow j}$ 
9:       end if
10:       $rad_i = minDist \div 2$ 
11:    end if
12:  end for
13: end for

```

networks such as FC neural network, with DHGN's complexity is lower than that of FC networks. Both DHGN and FC networks however, share the same one-shot learning characteristic, in which each pattern passes through the learning procedure only once. Nevertheless, FC network requires additional computations in determining both Euclidean distances and radius for each training patterns. A major drawback of this approach is such that, with large number of training patterns used, it will severely affecting the performance of the network.

The complexity analysis using Big-O notation for DHGN algorithmic implementation has demonstrated the capability of this proposed approach in providing fast and low-complexity scheme for large-scale data analysis. An important aspect in this kind of implementation is the ability of DHGN to perform recognition procedure within a single-cycle pass, without having to conduct iterative training procedure to train the network for adaptation purposes. Rather, DHGN performs in situ recognition, in which the training set could be memorised within a single pass (or cycle). This gives an edge to DHGN as a solution for large-scale pattern recognition.

3.3.2 Scalability Analysis

Scalability factor for DHGN distributed pattern recognition scheme could be determined from two different aspects - the storage capacity and the communication efficiency. A high requirement for storage capacity would affect the scalability of the algorithm. For efficient pattern recognition scheme, the storage requirement should not be heavily affected by an

increase in the number of stored patterns and the communication should stay relatively contention free.

Table 3.8 represents the terms that we will use to estimate the computational complexity of DHGN algorithm.

Symbol	Explanations
s_{size}	Size of Subpattern.
m_{SI}	Number of messages communicated between SI module and GN nodes in network generation stage.
n_r	Number of rows of GN nodes within a layer.
n_{sub}	Number of DHGN subnets within a network.
$m_{ne}^{l_0}$	Number of messages communicated from non-edge GN nodes in base layer.
$m_e^{l_0}$	Number of messages communicated from GN nodes at the edge of base layer.
$m_{total}^{l_0}$	Total number of messages communicated from GN nodes at base layer.
m_{ne}^i	Number of messages communicated from non-edge GN nodes at middle layer i .
m_e^i	Number of messages communicated from GN nodes at the edge of middle layer i .
m_{total}^i	Total number of messages communicated from GN nodes at middle layer i .
m_{total}^{DHGN}	Total number of messages communicated between all GN nodes in DHGN subnet.
$bs_{ne}^{l_0}$	Maximum size of bias array for each non-edge GN node at base layer.
$bs_e^{l_0}$	Maximum size of bias array for each GN node at the edge in base layer.
$bs_{total}^{l_0}$	Total maximum bias array size for all GN nodes in base layer.
bs_{ne}^i	Maximum size of bias array for each non-edge GN node at middle layer i .
bs_e^i	Maximum size of bias array for each GN node at the edge in middle layer i .
bs_{total}^i	Total maximum bias array size for all GN nodes in middle layer i .
bs_{all}^{top}	Maximum bias array size for all GN nodes in top layer.
bs_{total}^{DHGN}	Total maximum bias array size for all GN nodes in DHGN subnet.
l_{SI}	Size of the messages in network generation stage.
l_{SI}^{total}	Total size of the messages communicated in network generation stage.
p_{size}	Pattern size.

Table 3.8: DHGN computational complexity terms.

Analyses have been conducted on the computational complexity of DHGN algorithm for pattern recognition. In doing this, the two computational factors mentioned previously: storage capacity and communication efficiency have also been considered. The computational complexity of DHGN with HGN algorithms for binary pattern recognition have also been compared. The following subsection outlines the analysis.

Storage Capacity Analysis

Storage capacity estimation for DHGN algorithm involves the analysis of bias array capacity for all the GN nodes within the distributed architecture, as well as the storage capacity of the SI module node. In analysing the capacity of the bias array, the size of the bias arrays observed, as different patterns are being stored. The number of possible pattern combinations increases exponentially with an increase in the pattern size. The impact of the pattern size on the bias array storage is an important factor in bias array scalability analysis. In this regard the analysis is conducted by segregating the bias arrays according to the layers within a particular DHGN subnet.

The following equations show the bias array size estimation for binary patterns. This bias array size is determined using the number of bias entries recorded for each GN node. In this analysis, a DHGN implementation for one-dimensional binary patterns has been considered; wherein a two-dimensional pattern is represented as a string of bits.

i. Base Layer.

For each non-edge GN node the maximum size of the bias array:

$$bs_{ne}^{l_0} = n_r^2 \quad (3.12)$$

For each GN node at the edge of the layer:

$$bs_e^{l_0} = n_r \quad (3.13)$$

The cumulative maximum size of bias arrays at the base layer in each DHGN subnet could be derived as shown in Equation 3.14:

$$bs_{total}^{l_0} = n_r \left(bs_{ne}^{l_0} = n_r^2 (s_{size} - 2) + 2bs_e^{l_0} \right) \quad (3.14)$$

The maximum size of bias array, i.e. the total number of bias entries at the base layer is mostly determined by the number of possible combinations of values within a pattern.

ii. Middle Layers.

The maximum size of the bias array at a middle layer depends on the maximum size of the bias array at the layer below it. For non-edge GN node in a middle layer, the maximum size of its bias array may be derived as follows:

$$bs_{ne}^{l_i} = bs_{ne}^{l_i-1} \times n_r^2 \quad (3.15)$$

For each GN node at the edge, the maximum size of its bias array could be derived as the following:

$$bs_e^{l_i} = bs_{ne}^{l_i-1} \times n_r \quad (3.16)$$

Therefore, the cumulative maximum size of bias arrays in a middle layer (of a subnet) could be estimated using the following equation:

$$bs_{total}^{l_i} = n_r \left(bs_{ne}^{l_i} (s_{size} - (2i + 2)) + 2bs_e^{l_i} \right), \quad for 1 \leq i \leq l_{top} - 1 \quad (3.17)$$

iii. Top Layer.

At the top layer, the maximum size of the bias array could be derived from the preceding level non-edge GN node's maximum bias array size. Hence, the maximum size of the bias array of GN node at the top level is:

$$bs_{all}^{l_{top}} = bs_{ne}^{l_{top}-1} \times n_r \quad (3.18)$$

From these equations, the total maximum size of all the bias arrays within a single DHGN subnet could be deduced as shown in Equation 3.19:

$$bs_{total}^{DHGN} = bs_{total}^{l_0} + \sum_{i=1}^{l_{top}-1} bs_{total}^{l_i} + bs_{all}^{l_{top}} \quad (3.19)$$

Communication Complexity Analysis

DHGN is a distributed pattern recognition algorithm. In any distributed algorithm, communication plays an important role in ensuring the efficiency of the algorithm. High communication costs will incur additional overhead for the network to support the core functions of the algorithm. Hence, the intention is to minimise the communication costs within DHGN. In conducting an analysis of the communication costs, all the four steps in the distributed pattern recognition scheme have been considered. This subsection estimates the communication costs for the implementation.

i. Network Generation Step:

Network generation in DHGN implementation involves the initialisation of DHGN subnets for recognition processes. Within this step, SI module is responsible for communicating possible input values of the patterns, which will be used in the recognition process, to all the base layer GN nodes within DHGN subnets. Equation 3.20 shows the number of messages that needs to be communicated by SI module to these GN nodes.

$$m_{SI} = n_r \times n_{sub} \times s_{size} \quad (3.20)$$

This equation is based on the assumption that all DHGN subnets are of the same size. In addition, the cumulative size of all the messages that will be transmitted is shown in Equation 3.21:

$$l_{SI}^{total} = l_{SI} (n_r \times n_{sub} \times s_{size}) \quad (3.21)$$

ii. Pattern Input Step:

Within this step, SI module is required to decompose the pattern into subpatterns and distribute these subpatterns to all available DHGN subnets. The distribution of subpatterns to all DHGN subnets requires communication between SI module and all the base layer GN nodes within the subnets. The communication costs incurred during this step of recognition is similar to the previous step.

iii. Recognition at Subpattern Level:

The following relations show the communication complexity of DHGN algorithm at the subpattern recognition level.

(a) Base Layer.

For each GN nodes in the base layer, the communication costs could be derived from the number of messages communicated between adjacent nodes for each input subpattern.

For GN nodes at the edge of base layer:

$$m_e^{l_0} = n_r \quad (3.22)$$

For non-edge GN nodes:

$$m_{ne}^{l_0} = n_r^2 + 1 \quad (3.23)$$

Note that for non-edge GN nodes, the communication is required between adjacent nodes in both the preceding and the succeeding columns, as well as the communication of bias indices to the GN nodes at the next higher layer.

The cumulative communication costs for all GN nodes in the base layer could be derived as the following:

$$m_{total}^{l_0} = n_r \left(m_{ne}^{l_0} (s_{size} - 2) + 2m_e^{l_0} \right) \quad (3.24)$$

(b) Middle layers.

The communication costs for GN nodes in the middle layers are similar to the GN nodes at the base layer. Hence Equation 3.22 and Equation 3.23 apply. However, the difference would be in the number of nodes available within each layer.

The cumulative communication costs for all GN nodes in each middle layer:

$$m_{total}^{l_i} = n_r \left(m_{ne}^{l_i} (s_{size} - (2i + 2)) + 2m_e^{l_i} \right), \quad for 1 \leq i \leq l_{top} - 1 \quad (3.25)$$

(c) Top layer.

These GN nodes are only responsible for communicating the final index for each subpattern stored/recalled to the SI module. Therefore, there is only one message that needs to be passed to the SI module for each input subpattern.

The total cumulative number of communications required for each subpattern stored/recalled in a DHGN subnet could be derived from Equation 3.26.

$$m_{total}^{DHGN} = m_{total}^{l_0} + \sum_{i=1}^{l_{top}-1} m_{total}^{l_i} + 1 \quad (3.26)$$

iv. Recognition at Pattern Level:

The recognition at pattern level does not require any communication since recognition takes place within the SI module.

Comparative Analysis

The research has conducted a comparative analysis between HGN and DHGN pattern recognition algorithms with regards to their storage capacity and communication complexity.

i. Storage Capacity:

Both DHGN and HGN implementations share the common storage entity known as bias array. However, the total maximum number of possible bias entries is significantly higher in HGN implementation as compared to DHGN implementation. An increase in the size of the pattern can therefore affect the storage capacity in HGN to a higher degree. This is primarily due to the larger hierarchical shape factor of the HGN. Table 3.9 shows the total maximum size of bias arrays at each layer within a HGN network for binary pattern with size 55.

The total maximum size of bias arrays vs. number of layers within the HGN network and the maximum size of bias array for each GN node vs. number of layers within the network can be plotted. Figure 3.22 shows the results of the analysis for 55-bits binary pattern.

Layer (i)	$\log bs_{total}^{l_i}$	Layer (i)	$\log bs_{total}^{l_i}$
0	2.64	14	10.75
1	3.22	15	11.31
2	3.81	16	11.88
3	4.39	17	12.44
4	4.97	18	13.00
5	5.56	19	13.55
6	6.14	20	14.09
7	6.72	21	14.63
8	7.30	22	15.15
9	7.88	23	15.65
10	8.46	24	16.13
11	9.03	25	16.56
12	9.60	26	16.86
13	10.18	27	16.56

Table 3.9: Total possible bias entries for each layer within a HGN network for 55-bits binary patterns.

The top chart in Figure 3.22 shows a slight drop in the total maximum possible bias entries after layer 26. This is due to the number of the top layer GNs being substantially lower than the middle layers. An increase in the number of layers within HGN network would result in an increase in the possible maximum size/node and the total possible maximum size of bias arrays. Thus, the GN processing nodes must have large storage capacity in order to fulfill all different combinations of input elements within a pattern. Therefore, HGN could be applied in large processing nodes such as grid node that is capable of storing large amount of data. It is important to note that the possible maximum sizes do not imply that the actual storage requirements will follow these theoretical maxima. The actual storage will be highly influenced by the variations in the input patterns. With DHGN implementation, the storage capacity requirement for each GN processing node has been reduced significantly, to cater for lightweight processing nodes such as sensor nodes, to participate in the recognition process.

The decomposition of HGN hierarchical structure into small-scale DHGN subnets reduces the levels of the hierarchical network significantly, thereby improving its storage capacity requirement. Table 3.10 shows the comparison between HGN and DHGN implementation in terms of the total possible maximum size of bias arrays

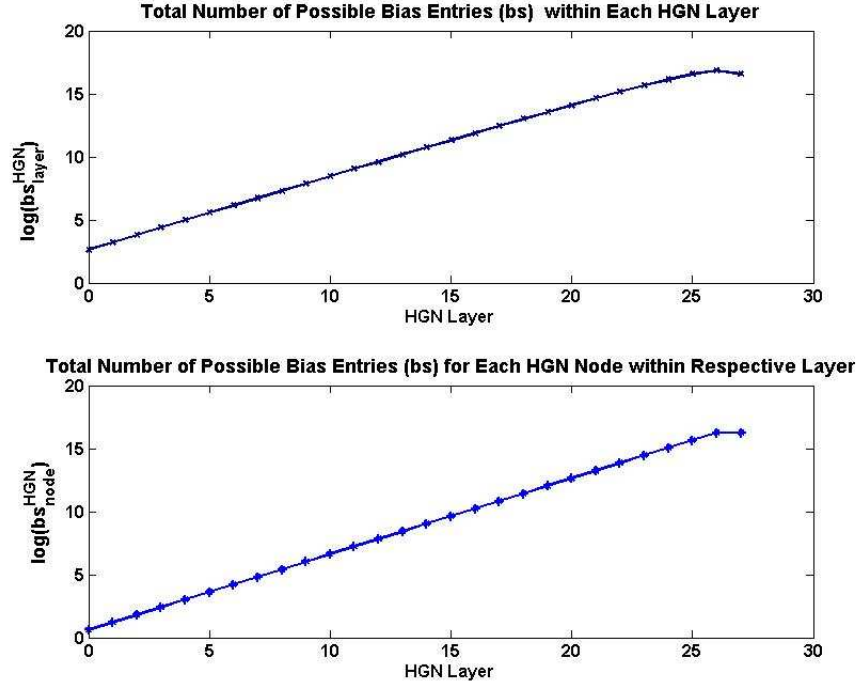


Figure 3.22: The charts showing the total maximum possible bias entries for each layer and within individual nodes for 55-bit binary patterns using the HGN approach.

and the size of the stored patterns. Note that for the equivalent DHGN implementation, the subnet with subpattern size of 5 has been chosen. On average, DHGN reduces the size of total memory requirement for about 68.9% from equivalent HGN implementation. Figure 3.23 shows this comparison.

p_{size}	$\log bs_{\text{total}}^{\text{HGN}}$	$\log bs_{\text{total}}^{\text{DHGN}}$
5	2.11	2.11
15	5.11	2.58
25	8.18	2.81
35	11.19	2.95
45	14.20	3.06
55	17.22	3.15

Table 3.10: Comparison between the total possible maximum bias array size for HGN and DHGN implementations for binary pattern recognition with different pattern sizes.

DHGN offers lower storage capacity requirement than HGN. Furthermore, DHGN offers higher scalability with respect to the increase in the pattern size. Neither HGN nor DHGN algorithm are affected by an increase in the number of patterns stored. This is due to the fact that both DHGN and HGN bias array concept has

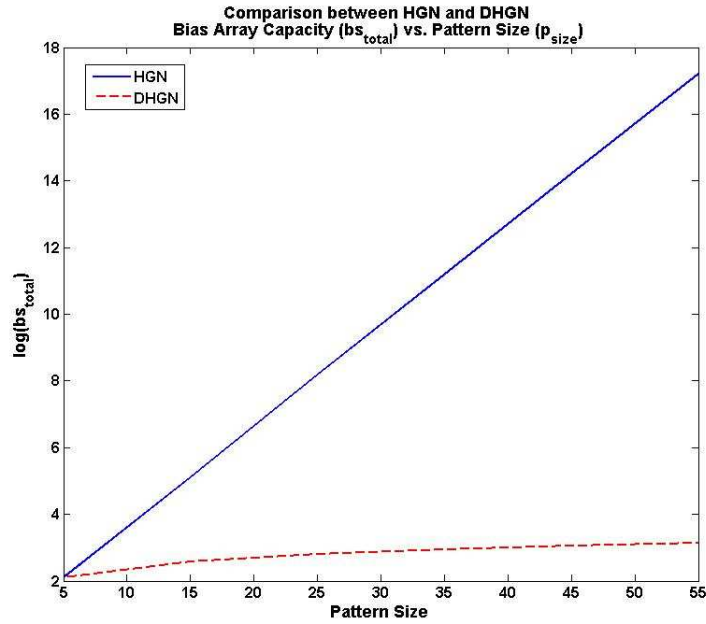


Figure 3.23: Comparison between the total cumulative bias entries in DHGN and HGN implementations. The subnet for DHGN is for handling subpattern with size 5 in this comparison.

been specifically designed to rely on the adjacency information thereby avoiding the bulk of global pattern recognition computations. Some patterns might share similar adjacency information amongst each other. Therefore the bias array size estimation is conservatively done, by assuming the worst case where the input patterns do not share any adjacency information.

ii. Communication Complexity:

DHGN requires fewer hierarchical layers in comparison to HGN and thus reduces the overall communication cost. Table 3.11 and Figure 3.24 show the comparison of communication cost for HGN and DHGN.

p_{size}	m_{total}^{HGN}	m_{total}^{DHGN}
5	58	58
15	548	174
25	1538	290
35	3028	406
45	5018	522
55	7508	638

Table 3.11: Comparison between HGN and DHGN implementations with regards to the number of messages communicated per pattern recognition.

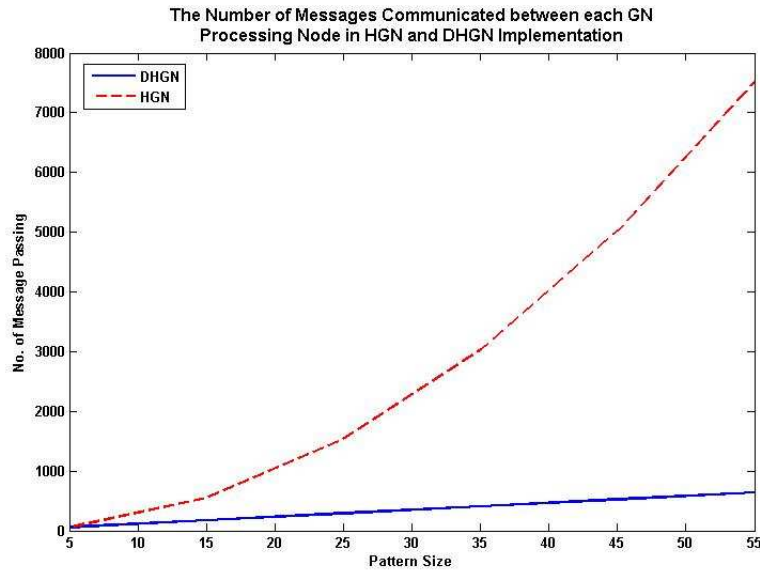


Figure 3.24: A comparison of communication costs, for each of the GN node, within HGN and DHGN.

From these results, it can be derived that on average, 83.4% of message passing savings have been obtained through DHGN implementation in comparison to HGN. It can be further deduced that the distribution factor in DHGN algorithm helps with improving the efficiency of this GN-based approach for pattern recognition.

The analyses that have been carried out indicate that DHGN provides high scalability towards increasing size and dimension of patterns through the use of divide-and-distribute approach within a single-cycle learning. Furthermore, DHGN has been proven to reduce the complexity of HGN algorithm, in regards to its processing requirement.

3.4 Pattern Recognition Simulation and Results

A series of recognition tests using DHGN's distributed pattern recognition approach has been conducted. The results of these tests are also included in Appendix A. In this chapter, two different sets of patterns have been tested and discussed. The first set of patterns is binary character patterns as shown in Figure 3.25, while the second set of patterns is 16KB binary images. These sets were used as the base for generating noisy patterns in their respective categories. Equal sized DHGN subnets, capable of storing either 5-bit or 9-bit binary patterns, were adopted for these tests. The subnet size may be

calculated by dividing the largest input pattern size with the number of available nodes within the computer grid. Inter-nodes communications and SI-to-subnet communications were implemented using MPICH-2 library for the message passing interface (MPI) (Gropp, Thakur and Lusk, 1999). This simulation has been written in C/C++ language. The full pseudocode for this simulation is included in Appendix B.

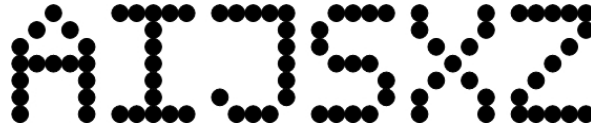


Figure 3.25: Original binary character patterns used in the recognition tests.

3.4.1 Binary Character Pattern Recognition

The character patterns used in this recognition test have been grouped into three different representations. These are 5-by-7 bit representation, 8-by-8 representation, and 16-by-16 representation. Figure 3.26 shows the formation of these representations.

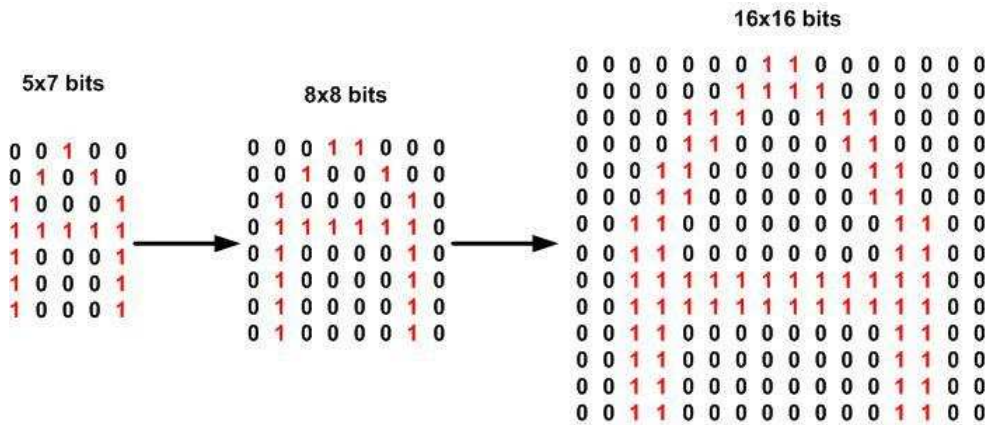


Figure 3.26: Bit representation for binary character patterns used in the recognition test.

For this test, Each DHGN subnet is used to store/recall 5-bit binary subpatterns. Each character image used has been decomposed into 5-bit subpatterns of equal size. This decomposition is conducted by SI module node. Recall rate using precision and recall technique with voting mechanism has been used as a classification parameter in these tests. Recall rates R , for the tests were obtained using the following equation:

$$R = \frac{n_{DHGN}^T}{n_{DHGN}^T + n_{DHGN}^F} \quad (3.27)$$

n_{DHGN}^T and n_{DHGN}^F both represent the number of DHGN subnets with correct and incorrect recall respectively.

Figure 3.27 shows the comparison of recall rates among the three character patterns “A” of different sizes as shown in Figure 3.26, which has been used in this test. The recall rates are similar for all these different representations indicating that DHGN recall accuracy is not sensitive to change in the pattern size.

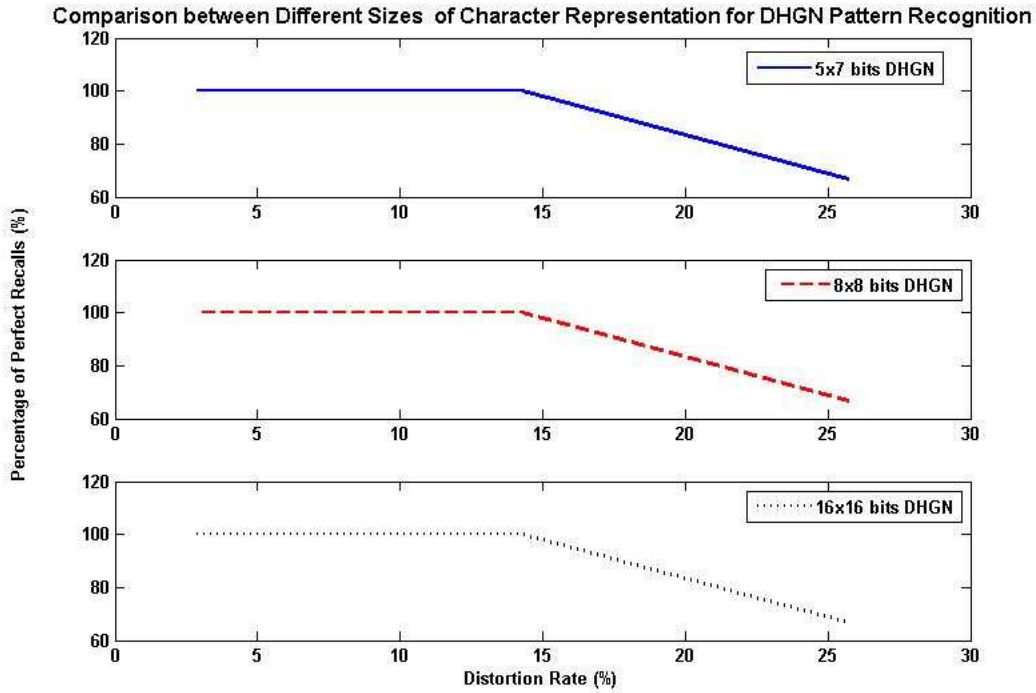


Figure 3.27: Comparison among different character representation for DHGN pattern recognition.

The recognition test has also been conducted on a set of binary character patterns with random distortion. The random distortion applied to the character patterns vary according to the level of distortion. Eight different levels of distortion were used as shown in Figure 3.28.

For this test, Each DHGN subnet is able to store/recall 9-bit binary subpatterns. Therefore, the SI module is responsible for decomposing each character pattern into 9-bit subpatterns.

Figure 3.29 shows the results of the test. DHGN is able to recognise distorted character patterns up to 25.70% distortion. This shows that DHGN offers a reasonably high level of

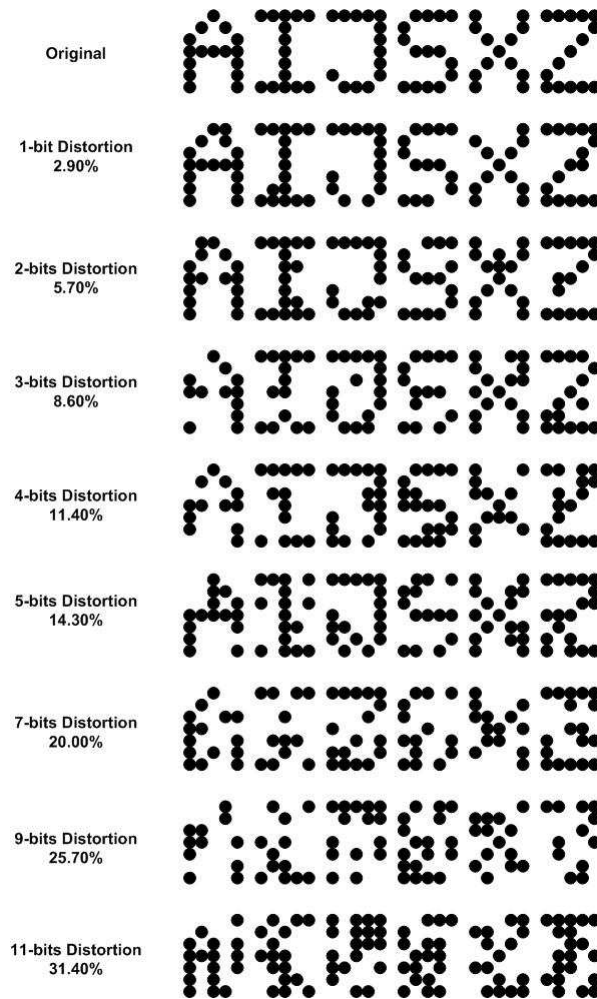


Figure 3.28: Eight different levels of random distortion applied to binary character patterns.

recall accuracy as a single-cycle learning algorithm. It may be noted that patterns with random distortion higher than 25% are difficult to be identified, even with the human eye.

In analysing the effectiveness of our proposed two-level recognition scheme, a comparative test was conducted between one-level DHGN (DHGN at subpattern level only) as described in (Khan et al., 2010b) and the newly-proposed two-level DHGN recognition schemes. Similar dataset has been used in this study. Figure 3.30 shows the results of the test.

Two-Level DHGN recognition scheme provides higher recall percentage as compared to the previous implementation of One-Level DHGN. This is due to the additional recognition feature of SI module that enables the results of the recognition process at subpattern level to be reanalysed to produce higher recall accuracy through voting mechanism as described in Section 3.1.

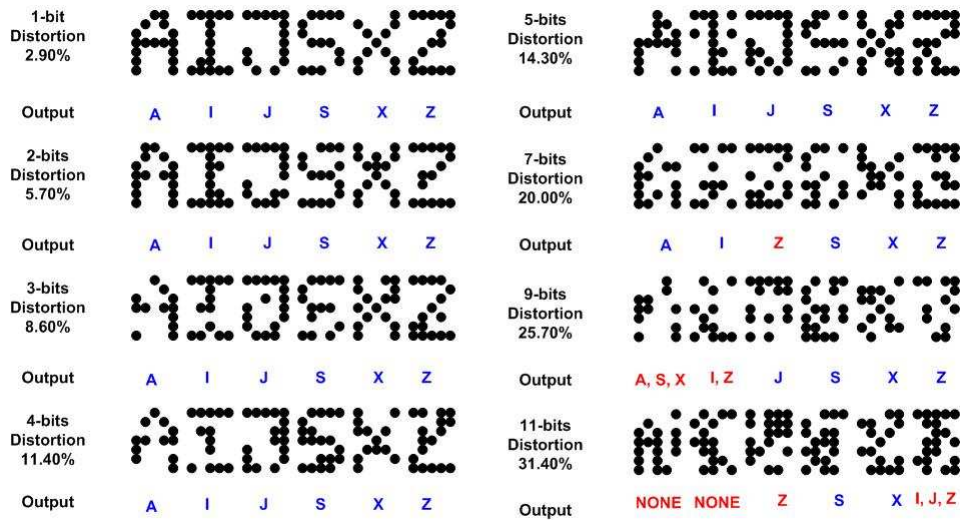


Figure 3.29: Results for binary character pattern recognition with DHGN.

3.4.2 Recognition Test on Binary Images

The second recognition test in our study involves the recognition of noisy 128-by-128 bit binary images. In this test, we have added different levels of Gaussian noise to the image “Lena” and test these images over a set of different heterogeneous images. The Gaussian noise used in this test has been determined using different percentages of noise with Gaussian distribution over the size of the original image. With Gaussian distributed noise, the original Lena image pixel value distribution has been changed using Gaussian distribution. To test the recall accuracy of DHGN algorithm, we have used distorted image “Lena” against a series of heterogeneous images as shown in Figure 3.31. We have applied different levels of noise using Gaussian distribution to the “Lena” image, as shown in Figure 3.32.

The pixel value distributions for noisy images are significantly different from the original image, making it quite difficult to be recognised as the original image. Furthermore, the number of pixels corresponds to the pure black and pure white values (0 and 255) are deteriorating with an increase in the percentage of noise added to the image. The Gaussian-distribution noise levels down the pure black and white pixels into different levels of either blackness or whiteness (within range of 0-255).

This recognition test was divided into two parts. The first part involves the recognition of Gaussian distributed noisy images with 10 heterogeneous binary images previously stored within the DHGN network, while the second part implies similar configuration

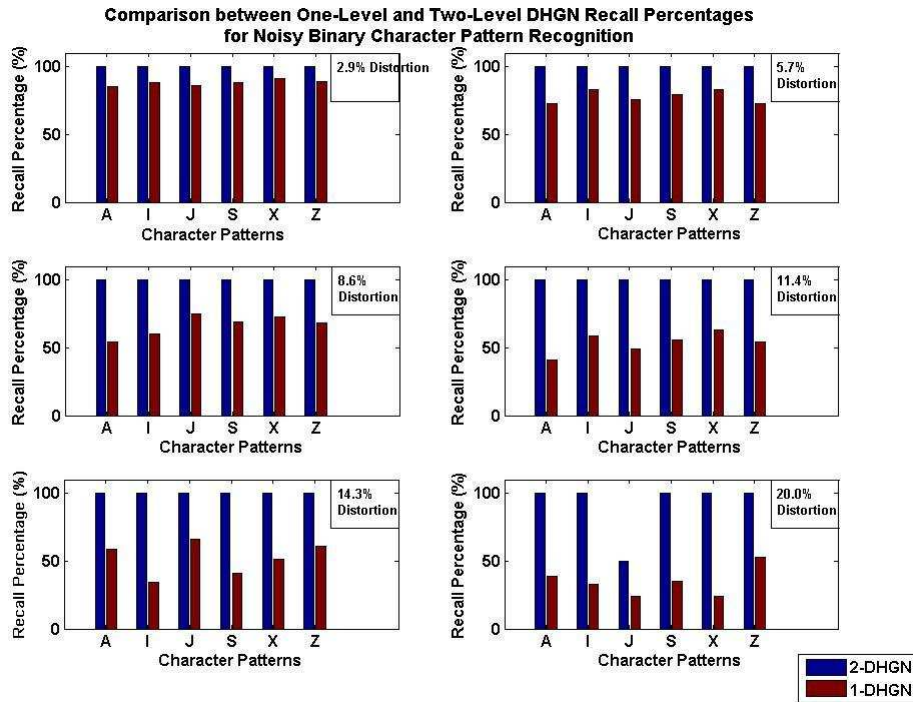


Figure 3.30: Comparison between One- and Two-Level DHGN pattern recognition on binary characters.

with 20 heterogeneous binary images. Figure 3.33 and Figure 3.34 show the results of recognition tests conducted.

The DHGN pattern recogniser is capable of providing high recall accuracy for distorted heterogeneous binary image recognition. The dimensionality reduction from greyscale images to binary images does not affect the recall accuracy of DHGN. In addition, DHGN is able to store all the 20 heterogeneous images' data within a single-cycle learning process without any reductions in its recall accuracy.

3.4.3 Performance Test on Binary Images

A study has been conducted on DHGN's performance with respect to its recognition time taken for each subnet, with different number of subpatterns stored/recalled. These subpatterns were derived from similar set of binary images as shown in Figure 3.31. Figure 3.35 and Table 3.12 show the results of the test that has been carried out.

The total recognition time for each DHGN subnet shows a stable performance of less than 1.5 seconds for different number of patterns stored/recalled. However, the average recall/store time for each image within a DHGN subnet decreases, with an increase in the number of images stored/recalled. This reduction in recall/store time is caused by the

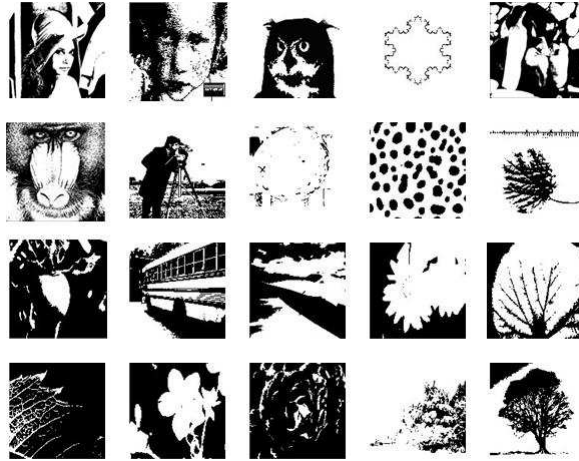


Figure 3.31: Heterogeneous binary images used in the image recognition test.

No. of subpatterns Stored/Recalled	Average Recognition Time (in seconds)
10926	0.099683
12747	0.116156
14568	0.078546
16389	0.082982

Table 3.12: Average recognition time for each subnet in DHGN network for a given 16kb binary image.

load-balancing feature of MPI communications used in our implementation. MPI communications enable the message-passing process to be conducted at higher speed, with an increase in the number of subpatterns introduced to the network. This load-balancing feature of MPI communications within DHGN network enables the overall system to perform at a stable state, with an increase in the number of subpatterns introduced. As a result, this performance test has shown that an increase in the number of subpatterns stored within the network does not have any impact on the recall/store time of DHGN implementation. It can therefore be concluded that the scalability of DHGN algorithm is not affected by the number of stored pattern within the DHGN network. Further analysis on DHGN message-passing model will be described in Chapter 5.

3.5 Multi-Value DHGN Model

One possible approach to reduce the cost of communications in DHGN implementation is to reduce the number of processing nodes participated in the recognition process. This could be achieved by implementing a distributed pattern recognition scheme using a single

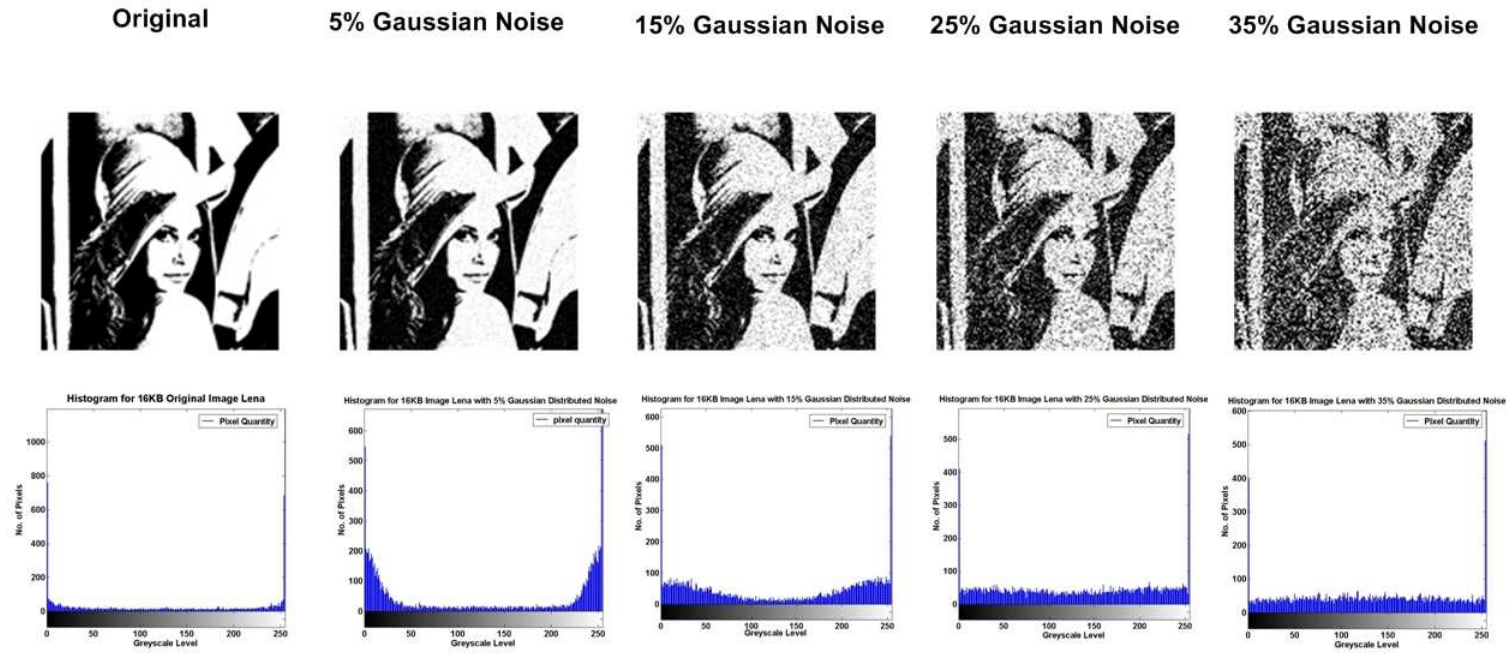


Figure 3.32: Four different levels of Gaussian noise added to image Lena.

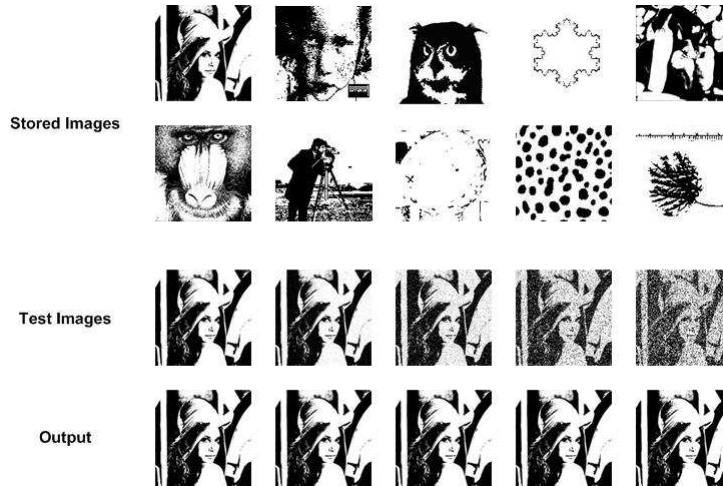


Figure 3.33: Results of the image recognition test with ten binary images stored.

dimension DHGN structure, as shown in Figure 3.36. Note that each possible value within the pattern is implemented as an internal action, rather than involving message communication between different nodes. This scheme is known as Multi-Value DHGN (MV-DHGN).

With this compressed scheme, the cost for communications is significantly reduced. Table 3.13 shows the formulas to derive the number of communications occurring by implementing this approach, given n number of GNs.

Communication Type	Representation
Input	n
Communication	$\sum_{i=0}^{\frac{n+1}{2}-1} 2v + (n - (2i + 2)) (3)$
Output	1

Table 3.13: Representations for the amount for each different types of communication performed in MV-DHGN message-passing model.

Note that for MV-DHGN approach, the dimension of patterns have been eliminated in the calculation, since each node is capable of handling all different element values. However, it requires the process of redesigning the bias array scheme for pattern storage in DHGN implementation, in which each bias entry at the base level should include the value, matched with the position of the node in a given pattern, in the form of $idx(v_l, v_r)$, where idx, v_l, v_r represent bias index, value from left-adjacent node, and value obtained from the right-adjacent node respectively.

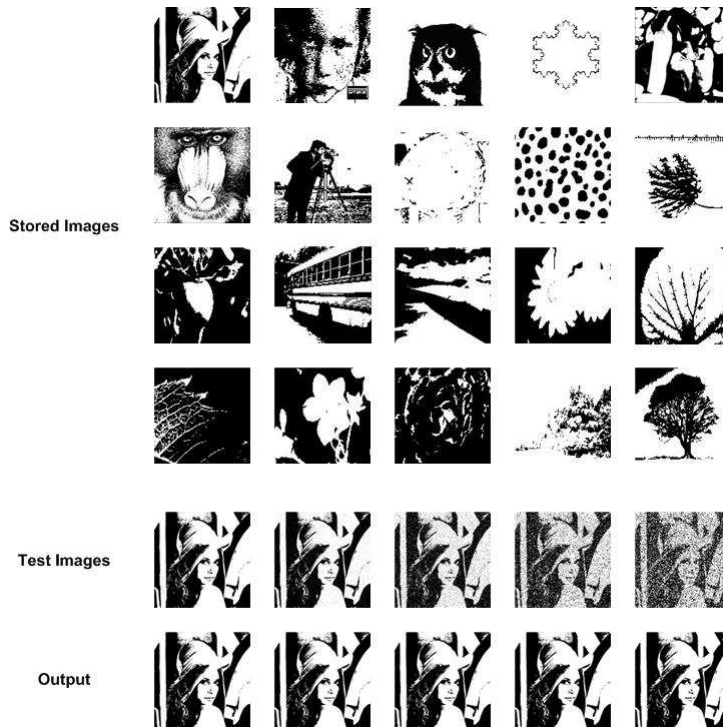


Figure 3.34: Results of the image recognition test with twenty binary images stored.

An analytical observation on the effect of reducing the number of processing nodes for communication savings has been conducted. Figure 3.37 shows the result of the comparison between non-reduced and reduced approaches for DHGN message-passing model on binary patterns with size up to 27-bits.

From the observation, It can be deduced that on average, 42.07% savings on the cost of communications have been obtained through the use of MV-DHGN, in comparison to original binary DHGN scheme. Therefore, the reduced approach seems to offer more efficient communication strategy, in regards to the process action estimation for DHGN message-passing model.

Besides an effective communication aspect, MV-DHGN has also been designed as a mechanism for fault tolerance implementation, specifically in the event where processing nodes failed to perform in any subnet during DHGN recognition process. This will be further discussed in Chapter 5. Moreover, MV-DHGN implementation can also be included in existing DHGN architecture. Figure 3.38 shows the architecture of DHGN with enjoined MV-DHGN subnets.

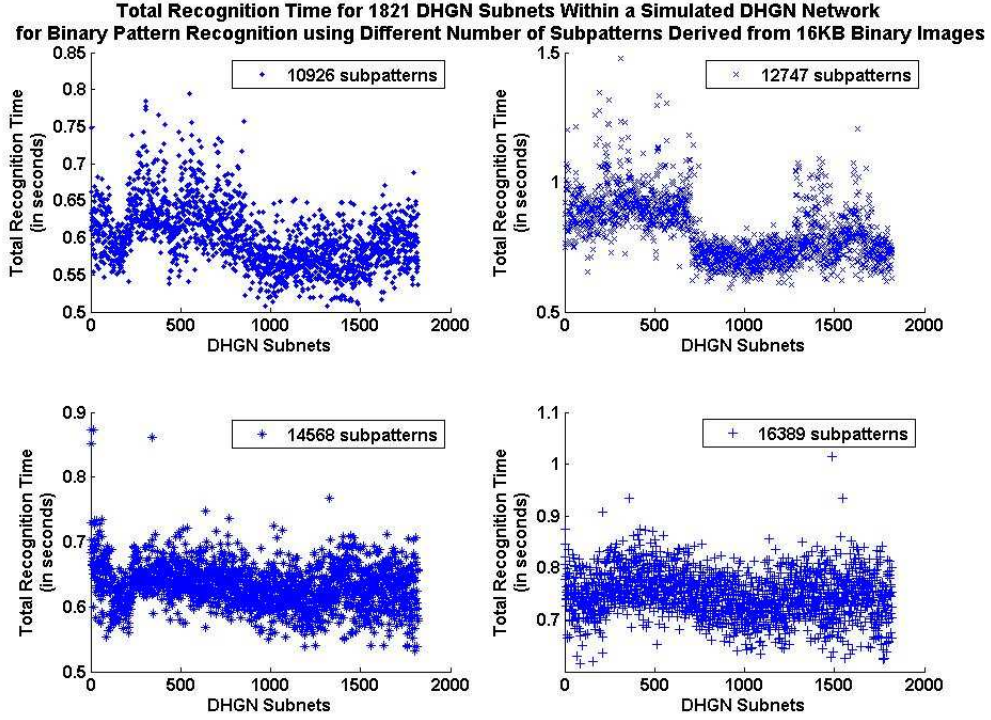


Figure 3.35: Total recognition time for each DHGN subnet in binary pattern recognition with different number of subpatterns derived from 16KB binary images.

3.5.1 Complexity Estimation

Two different aspects of complexity are considered in our proposed MV-DHGN scheme. These include processing capacity and bias array design. Comparisons were also made in terms of these aspects with existing DHGN scheme.

Processing Capacity

Processing capacity for MV-DHGN implies that its structure only accommodates each position of elements within a given subpattern. In this perspective, each processing node is capable of receiving different kinds of pattern elements. Thus, it reduces the structural capacity of existing DHGN subnet composition. Given a DHGN composition for binary patterns with subpattern length of l bits, the size of subnet, i.e. the number of GN nodes, n_{GN}^V required in MV-DHGN structure can be deduced from the following equation:

$$n_{GN}^V = \left(\frac{l+1}{2} \right)^2 \quad (3.28)$$

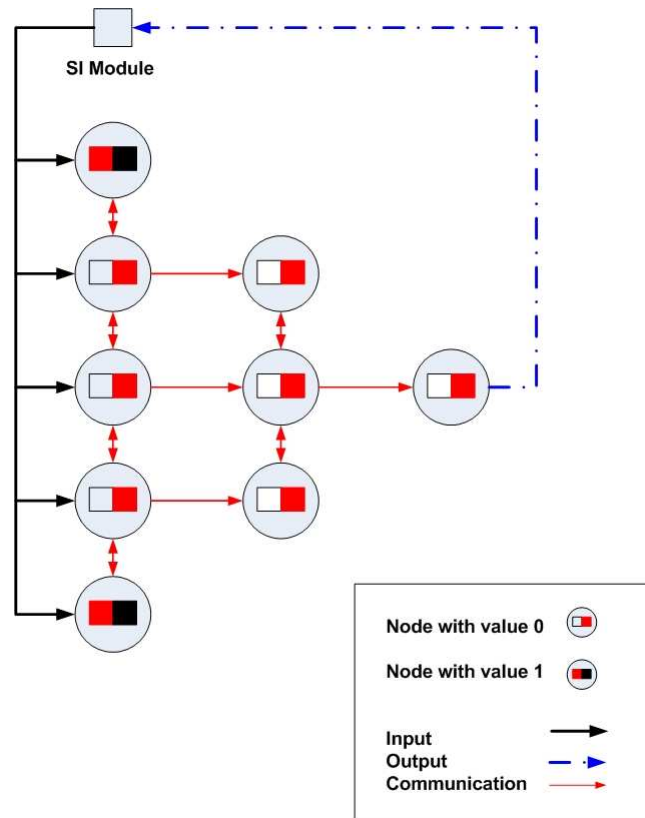


Figure 3.36: Multi-Value DHGN for binary pattern recognition on 5-bit patterns.

MV-DHGN eliminates the dimension factor of the estimation involving the number of processing nodes required. Hence, the size of each subnet is smaller than existing DHGN implementation. Figure 3.39 illustrates the comparative estimation that has been carried out on the size of subnet in both binary DHGN and MV-DHGN schemes for increasing pattern sizes.

Based on the analysis that has been carried out, 50% savings in terms of the number of processing nodes required are achieved using MV-DHGN scheme.

This research also examined a comparative estimation between MV-DHGN and DHGN implementation on the impact of increasing number of different pattern elements, on the number of GN nodes required for recognition involving patterns with length $l = 25$. Figure 3.40 shows the result of this estimation.

MV-DHGN implementation implies that each GN node is capable of storing different values. Hence, a requirement for node's value initialisation may be discarded in the proposed scheme. Therefore, this increases its flexibility to adapt to different data values, without having a need for reinitialisation.

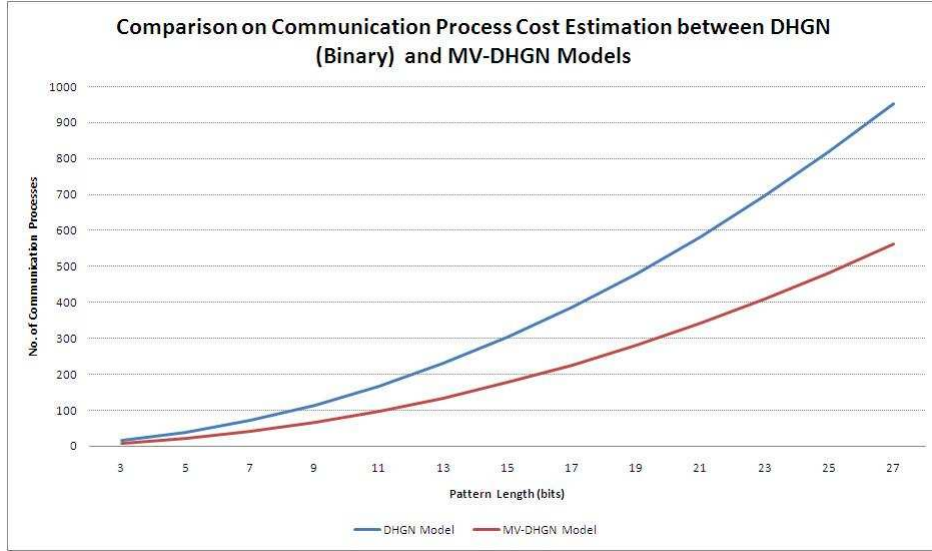


Figure 3.37: Comparison between DHGN (with binary values) and MV-DHGN distributed pattern recognition models on cost of communications.

Bias Array Design

The bias array composition in MV-DHGN model is exactly similar to existing DHGN scheme. It implements $index\{left, right\}$ composure for GN nodes at the base layer and $index\{left, middle, right\}$ for nodes at the middle layers. However, the value $left$ or $right$ in MV-DHGN at the base layer does not reflect the row number of DHGN subnet composition. It represents the value obtained from adjacent nodes. For GNs at the middle layers, each item in the entry represents the index obtained by its adjacent nodes from their respective lower-layer GNs at the same position.

The size and total cumulative size of bias array is considered at each layer. Table 3.14 represent the terms that will be used in our bias array estimation.

Base Layer. For each GN at the edge, the maximum bias array size is equivalent to the number of different elements within a pattern. Therefore, $\max b_{base}^e = v$.

For non-edge GNs, the maximum bias array size may be determined using the following equation:

$$\max b_{base}^{ne} = v^2 \quad (3.29)$$

The cumulative maximum bias array size at the base-layer in MV-DHGN subnet could be deduced as follows:

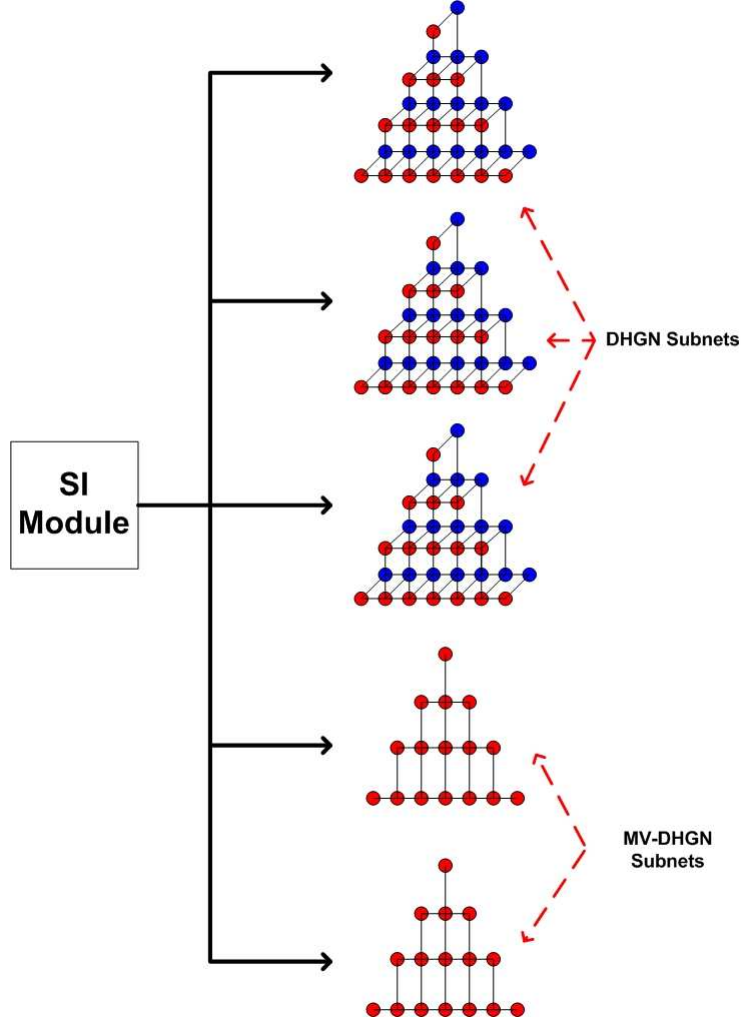


Figure 3.38: DHGN compositions with DHGN and MV-DHGN subnets.

$$Cb_{base}^{\max} = \max b_{base}^{ne} (S - 2) + 2 \max b_{base}^e \quad (3.30)$$

Middle layers. Similar to DHGN implementation that has been described in Section 3.1.4, the maximum bias array size for each GN at the middle layers are highly-dependent on the size of bias array of GNs at the lower layer. Hence, for middle-layer GNs at the edge, the maximum bias array size can be derived from Equation 3.31:

$$\max b_{l_i}^e = \max b_{l_{i-1}}^{ne} \times v^2 \quad (3.31)$$

Consequently, the maximum bias array size for non-edge middle-layer GNs can be estimated as following:

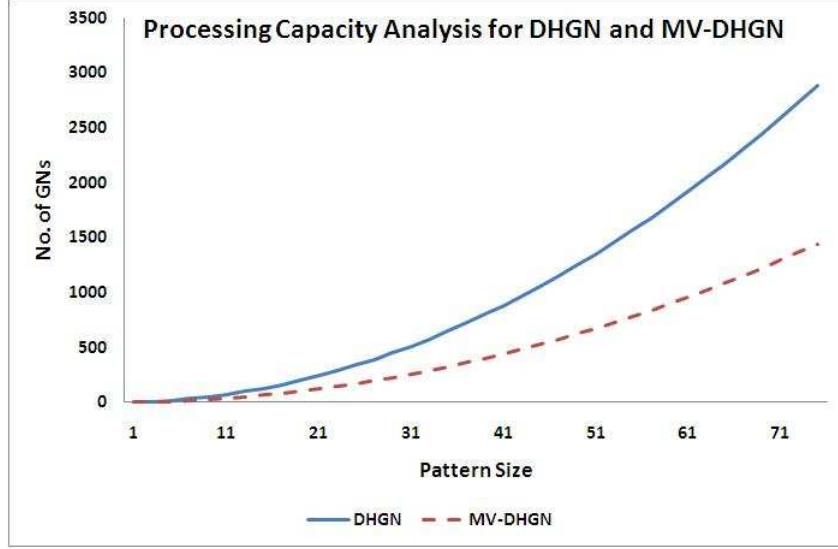


Figure 3.39: Processing capacity estimation for both DHGN and MV-DHGN implementations.

$$\max b_i^{ne} = \max b_i^e \times v \quad (3.32)$$

The cumulative bias array size for GNs at the middle layer could therefore be represented using the following equation:

$$Cb_i^{\max} = \max b_i^{ne} (S - (2i + 2)) + 2 \max b_i^e \quad 1 < i < \frac{S+1}{2} \quad (3.33)$$

Top Layer. The maximum bias array size of GN at the top layer is exactly equal to the maximum bias array size of the non-edge GN in the layer below it. Hence, $Cb_{top}^{\max} = \max b_{top-1}^{ne}$.

Based on all the equations obtained, the overall size of bias arrays for all GNs in the MV-DHGN subnet can be estimated using Equation 3.34:

$$Tb_{C-DHGN} = Cb_{base}^{\max} + \sum_{i=2}^{\frac{S+1}{2}-1} Cb_i^{\max} + Cb_{top}^{\max} \quad (3.34)$$

A comparison between the maximum size of bias array for each layer in a subnet between MV-DHGN and DHGN schemes has been carried out, identifying a binary sub-pattern with 55-bits. Figure 3.41 shows the result of this comparison.

The outcome of the analysis has shown that MV-DHGN implementation incurs equal loads of bias entries for the entire subnet. However, it is best to note that the similar

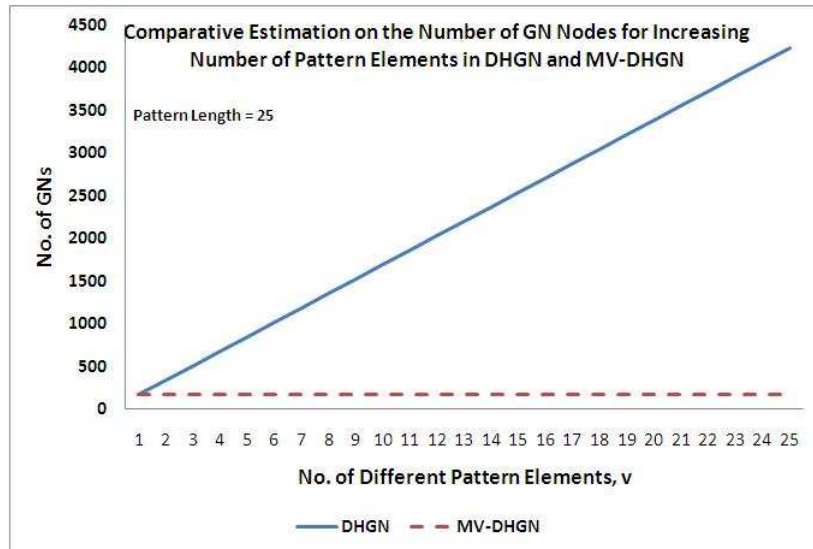


Figure 3.40: Comparative processing capacity estimation for DHGN and MV-DHGN on the impact of increasing number of different pattern elements.

quantity of entries is distributed across less number of GNs within the subnet. Hence, the computational load of each GN may increase significantly with this scheme. Therefore, the use of this approach in critically resource-constrained network may not be appropriate for long duration of time.

3.5.2 Recognition Accuracy

Pattern recognition simulation has been conducted to determine the accuracy of the proposed C-DHGN. Similar character pattern dataset as in Section 3.4.2 has been used in this test. The results of this simulation have shown that MV-DHGN performs equal-level of accuracy as in DHGN implementation. Figure 3.42 and 3.43 show the results of the recognition tests performed on MV-DHGN approach.

The results have indicated that MV-DHGN has a capability to recall all the patterns, given different range of random distortions. However, the majority votes obtained, based upon the character given shows that at higher distortion rates, some characters are dominant than the other. For instance, at 8.60% and 11.40% distortion rates, character patterns J and S obtained higher majority votes as compared to the other characters.

The recognition scheme has been simulated on a distributed environment using a high-performance computing (HPC) architecture. A scalability analysis on MV-DHGN was also conducted, demonstrating its capability to perform recognition on different large number

Symbol	Explanation
l	Layer in C-DHGN subnet
$\max b_{base}^e$	Maximum bias array size for base-layer GN at the edge.
$\max b_{base}^{ne}$	Maximum bias array size for non-edge base-layer GN.
Cb_{base}^{\max}	Cumulative bias array size for all base-layer GNs.
$\max b_{l_i}^e$	Maximum bias array size for edged GNs at the middle-layer l_i , where $1 \leq i \leq \frac{S+1}{2}$.
$\max b_{l_i}^{ne}$	Maximum bias array size for non-edged GNs at the middle-layer l_i , where $1 \leq i \leq \frac{S+1}{2}$.
$Cb_{l_i}^{\max}$	Cumulative bias array size for all GNs in the middle layer l_i , where $1 \leq i \leq \frac{S+1}{2}$.
Cb_{top}^{\max}	Maximum bias array size for GN in the top layer.
v	Number of different pattern elements.
S	Length of subpattern.
Tb_{C-DHGN}	Total maximum bias array size for MV-DHGN subnet.

Table 3.14: MV-DHGN bias array estimation terms.

of patterns. Random bit subpatterns were used in this test. Table 3.15 describes this dataset.

Dataset	Subpattern Size (bits)	No. of Stored Subpatterns
1	5	$2^5 = 32$
2	7	$2^7 = 128$
3	9	$2^9 = 512$
4	11	$2^{11} = 2048$

Table 3.15: Dataset collection comprising of random binary subpatterns.

For each dataset, 20000, 40000, and 60000 random patterns were generated, and used to test the recognition accuracy of MV-DHGN. The results of this test indicated that all random subpatterns were able to be classified accordingly to each of the respective stored subpatterns, i.e. no additional index has been created (for example, in 5-bits subpatterns dataset, all the random patterns are classified according to the 32 subpatterns that have been stored).

A performance test was also conducted, in regards to the recognition time incurred for each subpattern in all datasets. The simulation was performed using a supercomputer facility at Victorian Partnership for Advanced Computing (VPAC), Australia. The machines that were used comprise of high-performance cluster with AMD Opteron system, where it consists of 95 Compute Nodes, 760 CPUs (or more correctly, cores). Each node

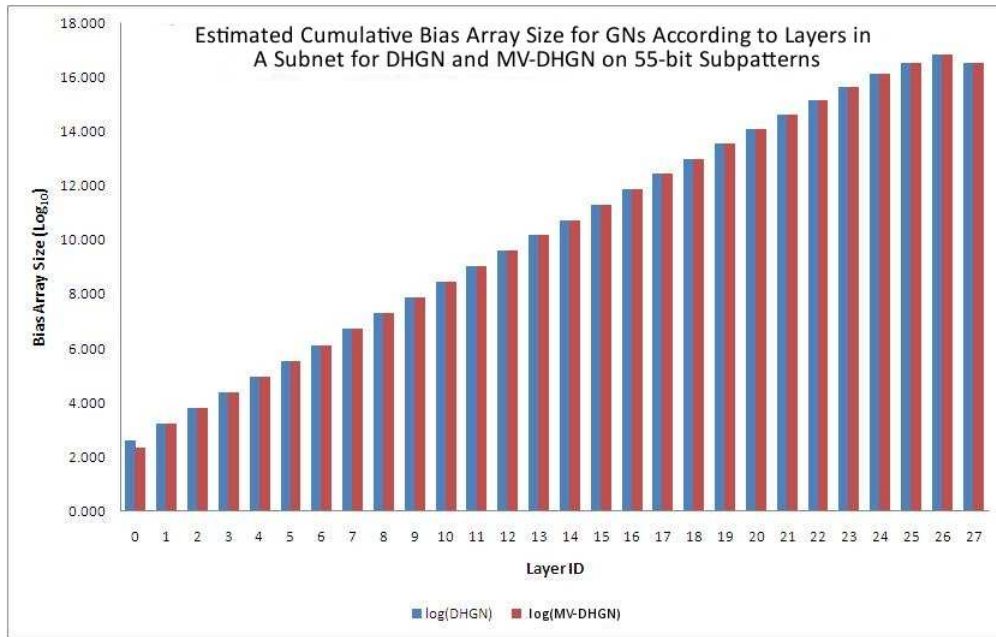


Figure 3.41: Estimated cumulative bias array at all levels within a subnet for MV-DHGN and DHGN implementations on binary 55-bits subpatterns.

has 2 AMD 2356 Quad Core Opterons, 32GB RAM and $4 \times 320\text{GB}$ disks (1.2TB scratch space).

Figure 3.44 shows the results of recognition time per subpattern obtained in the MV-DHGN simulation using three different datasets as in Table 3.15.

From this graph, it shows that recognition process for each subpattern only requires maximum of less than one millisecond for 20000 or more random patterns. In this regard, MV-DHGN is capable of performing fast recognition and is not significantly affected by an increase in the amount of subpatterns used.

3.5.3 Summary

This section has presented a new DHGN model known as MV-DHGN. It reduces the amount of processing nodes required while maintaining the recognition accuracy that is comparable with DHGN scheme. A significant drawback of this scheme is such that the bias array size for each GN may increase as a result of its capability to handle different types or dimension of elements within a subpattern. Therefore, the proposed compression model may only be utilised in coarse-grained highly-resourceful networks such as computational grids.

Results of Noisy Pattern Recognition Tests using MV-DHGN algorithm































Original:						
1-bit Distortion 2.90%						
Recalled Pattern:	A	I	J	S	X	Z
2-bits Distortion 5.70%						
Recalled Pattern:	A	I	J	S	X	Z
3-bits Distortion 8.60%						
Recalled Pattern:	A	I	J	S	X	Z
4-bits Distortion 11.40%						
Recalled Pattern:	A	I	J	S	X	Z

Figure 3.42: Recognition test results obtained from the simulation using binary character patterns A, I, J, S, X, and Z.

3.6 Conclusions

This chapter has presented the proposed approach for distributed pattern recognition, known as Distributed Hierarchical Graph Neuron (DHGN). DHGN implements a divide-and-distribute approach to HGN networks. Owing to its single-cycle learning and in-network processing features of GN-based algorithms, DHGN is able to offer efficient recognition scheme with high recall accuracy. Furthermore, DHGN's distributed pattern recognition scheme exerts low and stable recognition time, due to its ability to distribute the recognition processes across a computational network.

DHGN is able to lower the storage and communication complexities for pattern recognition process. In addition, the two-level recognition in DHGN algorithm offers both recognition at pattern and subpattern levels, which contributed towards higher recall accuracy for both the binary character patterns and the heterogeneous binary images. Moreover, the use of dimensionality reduction schemes such as binary signature implies low computational requirements for DHGN deployment.

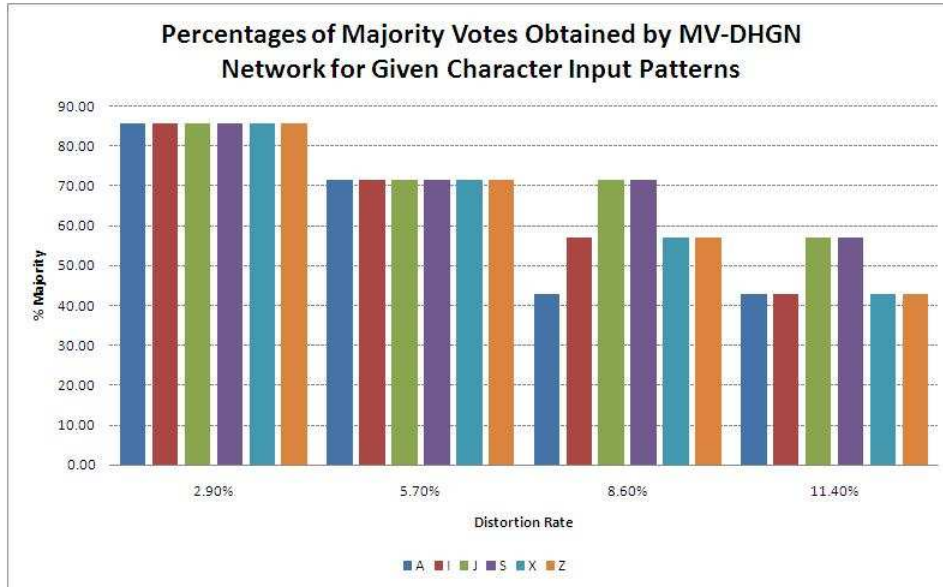


Figure 3.43: Percentage of majority votes for each character for different sets of distortion rates.

In regards to the algorithmic complexity, DHGN has been proven to adapt low-level complexity in its computation. This allows for fast recognition procedure with single-cycle learning capability and divide-and-distribute approach towards patterns. Comparisons with other pattern recognition algorithms including Hopfield network and Kohonen SOM have also been conducted for the purpose of complexity measurement.

In terms of scalability, a detailed comparison has been made between the Hierarchical Graph Neuron (HGN) approach and our proposed distributed version of the algorithm for one-dimensional binary patterns. There is an 80.5% reduction in the number of GN processing nodes in DHGN implementation in comparison to HGN. This reduction however does not affect the recall accuracy and scalability of DHGN algorithm for binary pattern recognition. In addition, there is an improvement in memory saving from an average reduction of 68.9% in DHGN in comparison to equivalent HGN implementation. Furthermore, we have also provided evidence of message passing savings of 83.4% on average in DHGN implementation.

A series of recognition tests were conducted on relatively small binary images. However, it is best noted that there is no restriction on the overall size of the images being used. Furthermore, the results of the tests have shown that the number of stored images does not negatively influence the accuracy and resource requirements for the proposed approach.

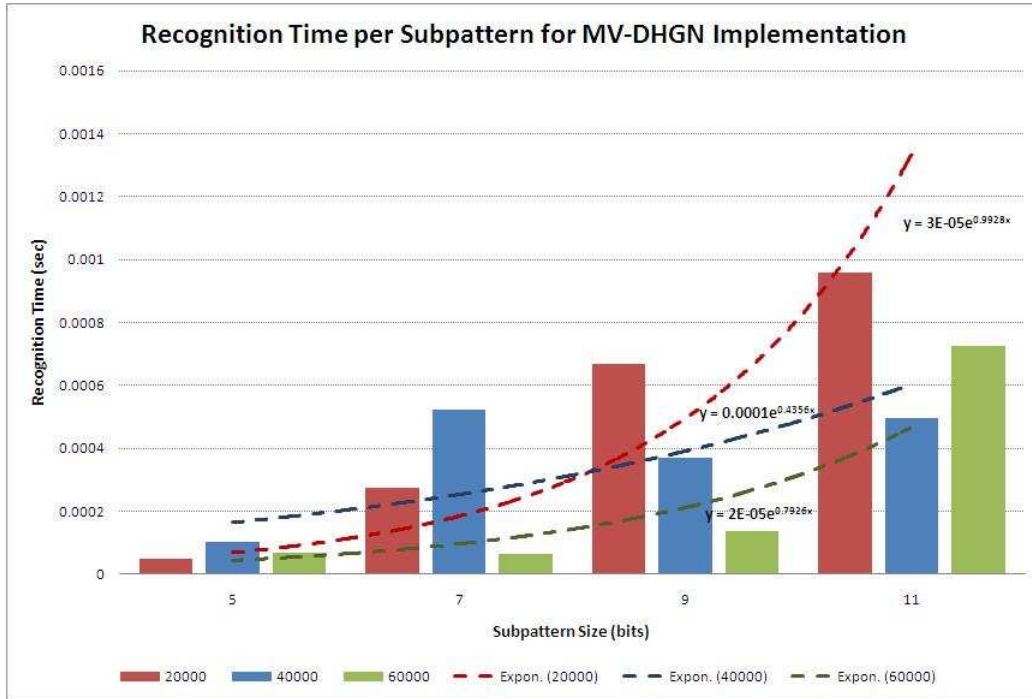


Figure 3.44: Recognition time per subpattern for different subpattern size and number of random subpatterns.

The divide-and-distribute approach, by working on relatively small segments of an image, also allows adjustment of subnets according to the feature variation within an image. Thus monotonous regions would be handled by fewer subnets than the ones with high degree of variations. Our approach may also provide means to tackle the curse of dimensionality since it radically reduces the computational complexity and resource requirements within our hierarchical associative memory network.

This chapter also presented a work on multi-value DHGN for increased scalability in handling patterns with large number of different elements. This multi-value model reduces the number of communications between adjacent GNs while maintaining the similar level of recognition accuracy, in comparison with existing DHGN scheme. However, there is a limitation in this multi-value DHGN implementation that is related to the bias array design. An increase in the size of bias array is expected with an increase in the number of different elements within the pattern. Further research on accommodating multiple value within DHGN is possible, by observing the design of bias array for pattern storage mechanism. An application of MV-DHGN will be further discussed in Chapter 4.

Chapter 4

Multi-Feature Pattern Recognition: A Distributed Approach

Pattern recognition involves a set of processes to define similarities and/or differences between two or more patterns. In this perspective, patterns or data need to be evaluated or measured in order to find such distinctive characteristics of those patterns. In achieving this, the first step in any pattern recognition schemes is to identify measurable quantities or characteristics of patterns that matched with a specific class of data. These measurable quantities are known as features. According to Theodoridis and Koutroumbas (2003), features can be defined as a set of measurements used for recognition and classification. These measurements in turn, form a feature vector that is used for recognition purposes. In image recognition, examples of features may include colours, edges, and spectrum frequencies.

Pattern recognition as described in previous chapters consists of a series of processes including data acquisition, data pre-processing, and classification (Bow, 2002). For each data presented for recognition, it has a number of related features that may be used to classify it to a respective data class with closest matched characteristics. These features are first to be extracted, before any classification/recognition process can take place. This process runs within the pre-processing stage of pattern recognition. A major problem with existing pattern recognition schemes is such that the number of features used is very large.

As a result, a phenomenon known as curse of dimensionality occurs. This is a result of adding more features as dimensions to computational space.

This chapter focuses on pattern recognition scheme involving multiple features. Multiple-feature implementation enables a holistic approach towards pattern recognition procedure that takes into consideration all significant features, which represent a particular set of patterns, such as images and sensor readings. This intends to reduce the bias effect of selecting only a single feature for classification/recognition purposes. Current approaches in pattern recognition require significant amount of effort to analyse different forms of features, due to the curse of dimensionality problem. This limits their ability to seamlessly and effectively perform recognition and classification involving complex data sets. Furthermore, most of the existing schemes incur high computational complexity that inhibits their capability to scale up for increasing number of features.

In solving this issue, the use of distributed approach in implementing pattern recognition on multiple features is envisioned. It is argued that by having a set of distributed computational networks working together, forming a distributed recognition network will alleviate the issue of scalability of pattern recognition scheme against increasing number of features to be considered. In addition, the use of single-cycle learning distributed pattern recognition algorithm such as DHGN, will further improve the performance of this multi-feature scheme. In contrast with contemporary machine learning approaches, our approach allows induction of new patterns in a fixed number of steps. Whilst doing so it exhibits a high level of scalability i.e. the performance and accuracy do not degrade as the number of stored pattern increases over time. The pattern recognition capability remains comparable with contemporary approaches such as the support vector machine (SVM), self-organising map (SOM), and artificial neural network (ANN). Furthermore all computations complete within the pre-defined number of steps and as such the approach implements one shot, i.e. single-cycle or single-pass, learning. The one shot learning within this method is achieved by side-stepping the commonly used error/energy minimisation and random walk approaches. The network functions as a matrix that holds all possible solutions for the problem domain. The network after applying our algorithm can find the solution in a single-cycle (i.e. fixed number of steps). The DHGN approach finds and refines the initial solution by passing the results through a pyramidal hierarchy of similar arrays. In doing so it eliminates/resolves pattern defects, with distortions up to 20% being

tolerated (Khan et al., 2010b). Previously encountered patterns are revealed whilst new patterns are memorised without loss of stored information. In fact the pattern recognition accuracy continues to improve as the network processes more sensory inputs (Nasution and Khan, 2008). To achieve this goal, the work on DHGN distributed pattern recognition algorithm is extended for multi-feature recognition and analysis of complex data.

The objectives of this chapter are as followings:

- i. To discuss all significant factors in implementing multi-feature recognition. This will include review of related approaches.
- ii. To extend the work on DHGN distributed approach for one-shot multi-feature pattern recognition scheme.
- iii. To evaluate the complexity and scalability of the proposed multi-feature scheme.
- iv. To present studies related to multi-feature recognition implementation on complex data, such as face image and handwritten character recognition.

This chapter is structured as follows. Section 4.1 describes basic concepts of data features and some common approaches in pattern recognition based on multiple data features. Section 4.2 provides a detailed description of our proposed multi-feature pattern recognition using DHGN. This section will also covers complexity estimation on the proposed scheme. A study on the implementation of multi-feature scheme on face image recognition will be presented in Section 4.3. In Section 4.4, further analysis on DHGN multi-feature implementation in complex handwritten numeral character classification will also be discussed. Finally, Section 4.5 concludes the chapter.

4.1 Data Features for Pattern Recognition

Consider a data representation shown in Figure 4.1. With mean pixel value as a feature for a set of images, we are considering a one-dimensional problem of determining which class does a particular image belongs to. However, as another feature is added as shown in Figure 4.1b, additional computation is required to determine the correlation between features that produces distinctive classes of images. With more features added, the computational costs of determining such correlations become progressively higher. According

to Theodoridis and Koutroumbas (2003), although two features may carry good classification if treated separately, there is small gain if these features are combined in the feature vector, due to high mutual correlation. In this regard, complexity increases with fewer benefits to the recognition process.

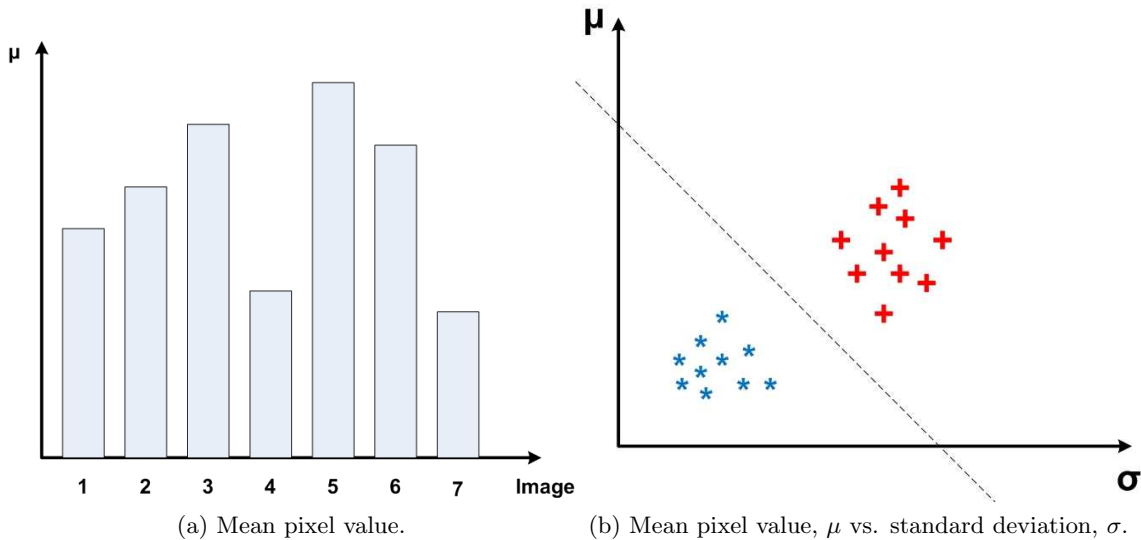


Figure 4.1: Data feature representation for a set of images.

In classification design of existing pattern recognition schemes, the high number of features used is directly translated to the number of classifier parameters adopted in the schemes. Therefore, this increases the complexity of the algorithm to determine synaptic weight and make adjustments to it during the recognition process, as demonstrated in the work of Hongtao, Feng and Rong-chun (2002) in multi-feature recognition using radial-basis function networks (RBFNs). It is imperative that the number of features shall be kept at minimum to ensure the efficiency of the recognition scheme. Nevertheless, selecting features for recognition is a complex process that needs to be performed objectively.

4.1.1 Common Approaches to Feature-based Pattern Recognition

In dealing with the curse of dimensionality problem, current approaches extend the recognition process by having a feature selection mechanism to select the best features that represent the whole dataset that is being used. However, this process adds to the complexity of recognition processes by having to perform such dimensionality reduction on features using costly algorithm including principal component analysis (PCA). Furthermore, the selection of features may affect the accuracy of the recognition scheme through

possible erroneous feature selection. According to Bow (2002), much work has been done in finding the dependences of the probability of misclassification on the dimensionality of the feature vector, the number of training samples, and the true parameters of the class-conditional densities. In this perspective, there is a need for a simpler recognition scheme that allows more than one features to be analysed, without having prior feature selection mechanism in dealing with selection of best features for data representation.

4.1.2 Pattern Recognition (PR) using Multiple Features

The concept of pattern recognition using multiple features has been introduced in a number of publications including (Cao et al., 1995; Zhu, Zhang, Sun and Xiao, 2008; Yu, Ma and Lu, 2007; Hongtao et al., 2002). Most of the work on multi-feature recognition has been dealing with images and optical characters. This is probably due to large number of features that could be extracted from images, including lines, colour, curves, and texture regions.

Existing literature on multi-feature recognition has been able to produce high recognition accuracy due to increased number of features to be considered. Nevertheless, the complexity of overall recognition scheme also increases due to the large number of features to be analysed. For instance, the multi-feature face detection scheme proposed by Zhu et al. (2008) implements probabilistic-based AdaBoost to train different features for recognition purposes. This significantly affects the scalability of the scheme due to iterative probabilistic interpretation process to converge these features to global minimum output. In other research, Cao et al. (1995) have proposed multiple-expert system for handwritten recognition using neural network. They introduced incremental learning in recognising handwritten patterns. In addition, iterative clustering algorithm has been adopted in this implementation. The combination of complex and highly-iterative algorithms makes this approach less scalable. Furthermore, accuracy of this scheme is highly-dependable on the outputs produced by a single rejection neuron.

Apart from these studies, Duin and Tax (2000) have also considering the use of combined classifiers for pattern recognition involving multiple features. In their research, a number of classifiers, including statistical, machine learning and neural networks have been applied for classification process. Nevertheless, the proposed scheme incurs significantly-high computational costs in determining accurate classification output. As a result, the

scalability of the scheme deteriorates as the number of training and testing data sets increases.

In this chapter, a new approach to multi-feature recognition is proposed by including the distributiveness that occurs in natural schemes. The DHGN algorithm is adapted, which is fully-distributed for recognition using multiple data features. The following sections will describe our proposed scheme together with a number of implementation examples.

4.2 DHGN Multi-Feature Recognition

The proposed multi-feature scheme conducts distributed pattern recognition using multiple features obtained from pattern data through a feature extraction method. It provides a scalable approach, in which the number of features required for the recognition purposes may always be extended, as long as sufficient computational resources are available. In this context, the number of features used f , is directly proportional to the computational resources available for the recognition scheme to be conducted, c . Hence, $f \propto c$. These resources are in the form of distributed computational networks, that provide greater scalability for recognition purposes.

4.2.1 System Architecture

The architecture for multi-feature recognition comprises a collection of DHGN networks. Each network performs a distributed recognition scheme for a single feature. Figure 4.2 represents DHGN multi-feature recognition system architecture.

In this configuration, a coordinator node that is used for data acquisition and networks coordination is introduced. This node will communicate the patterns received to all SI module nodes (see Section 3.1.2) on different DHGN network. Each SI module will have a copy of pattern set for the recognition process. This process starts with SI module generating a single feature obtained from the input patterns. The feature data will then be used as pattern for recognition purposes. The rest of the recognition procedures within each network are similar to the original DHGN scheme as described in Chapter 3. The results for each recognition process conducted by each DHGN network will then be sent to its respective SI module. Each SI module will produce a result of the recognition/classification

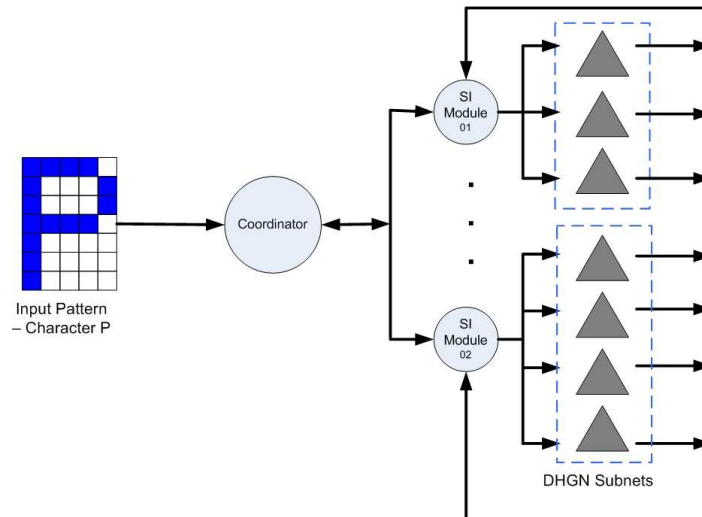


Figure 4.2: DHGN multi-feature recognition scheme that is made up of a collection of DHGN networks for analysing patterns using multiple sets of features.

for each pattern, in context with the operator-specified accuracy parameter(s). These parameters may include recall, precision, and error values. These results will then be passed to coordinator node for error calculation. Each error value, per test object, P_{err} for a given number of test objects o_{test} is calculated using the following equation:

$$P_{err} = \frac{F_{+ve} + F_{-ve}}{o_{test}} \quad (4.1)$$

Where F_{+ve} and F_{-ve} represent false positive and negative values respectively.

4.2.2 Recognition Analysis

The results of recognition on each feature will be sent to coordinator for determining the best feature(s) using a selected set of accuracy parameters. For instance, a recognition procedure may require a selection of features that produces lowest error for any given patterns. With this kind of condition, a minimum/maximum voting scheme is proposed to determine the feature with best parameter values for a given pattern class.

Consider the following scenario for recognition accuracy based on error values. Given a series of patterns P with n classes $P = \{p_1, p_2, \dots, p_n\}$, and a set of features $F = \{f_1, f_2, \dots, f_m\}$. For each pattern class p_x , $x = 1, 2, \dots, n$, select feature f_y , $y = 1, 2, \dots, m$ that produces the minimum recognition error value, err_{f_x} for all test patterns. Therefore, the recall accuracy r_{p_x} for each pattern class can be derived using the following equation:

$$r_{p_x} = \arg \min\{err_{f_1} : err_{f_m}\}, \quad x = 1, 2, \dots, n \quad (4.2)$$

It is important to note that minimum error value is not the only parameter for determining the most effective recall accuracy for multi-feature pattern recognition. Other possible means include normal mean, median and standard deviation and other statistical estimations, such as Bayes and maximum-likelihood estimators.

4.2.3 Complexity Estimation

In multi-feature recognition using DHGN distributed pattern recognition (DPR) scheme, we apply a similar approach towards recognising features for each pattern, as the original DHGN implementation described in previous chapter. Hence, the complexity of basic recognition function (for recognition at subpattern level) remain as low as the originally-proposed scheme. However, in multi-feature scheme, the voting mechanism is applied at two levels, i.e. at the SI module and the coordinator nodes.

At the SI module node, voting determines the matched pattern class for a given pattern, while at the coordinator node; voting is used to select the best selected feature that reflects optimum value for the specified accuracy parameter.

Voting Scheme at SI Module

As described earlier, voting scheme applied at SI module involves the process of classifying test pattern into a specific pattern class, based upon similar characteristic or feature value acquired. Inputs to this voting process are the indices retrieved from all DHGN subnets. In regards to multi-feature recognition, Each SI module will be handling specific feature for a particular dataset.

The maximum voting scheme in this DHGN implementation involves a process of finding the maximum number of similar indices obtained from all the subnets. There are two stages involved in our voting scheme, namely vote counting and maximum vote search.

In vote counting process, the SI module will have to perform index-matching process to compare index obtained from the test pattern with the index of pattern stored for each pattern class. The following pseudocode illustrates this process:

Algorithm 5 SI Module Voting Scheme

```

1: for  $i = 1$  to MaxTestPatternNo do
2:   for  $j = 1$  to MaxSubnetNo do
3:     for  $k = 1$  to MaxStoredPatternNo do
4:       if  $i.index \equiv k.index$  then
5:          $k.vote ++$ 
6:       end if
7:     end for
8:   end for
9: end for

```

The complexity of this process may be further analysed using a Big-O analysis. In this context, we can deduce that the complexity for vote-counting process is of n -polynomial, where $n = 3$. Given a vote-counting function $f(v_{count})$, its complexity in Big-O notation:

$$f(v_{count}) = O(n^3) \quad (4.3)$$

Where n represents a single executable instruction within the function. After the number of votes has been counted, the SI module performs a search function to identify the highest voted pattern class for the tested pattern. In this regard, this function will execute a linear search to find the maximum number of votes.

A copy of the pseudocode for SI module's voting mechanism is included in Appendix B.

Voting Scheme at Coordinator

Voting scheme at coordinator is used to select the best accuracy parameter of a feature from a collection of available features that have been used in the earlier recognition scheme using multiple DHGN networks.

In multi-feature recognition, each SI module will communicate the results of the recognition of features as patterns to coordinator node for further analysis. In this regard, the coordinator will have to store all the received accuracy parameters from SI modules. Table 4.1 shows a sample of error values obtained from two SI module nodes for each feature, on 5 different pattern classes.

Based on the values obtained from Table 4.1, we may conclude that Feature 1 is best to represent pattern classes 1, 3 and 4, since its error values are lower than Feature 2. Meanwhile, pattern classes 2 and 5 are likely to be represented by Feature 2.

Feature	Pattern Class				
	1	2	3	4	5
1	5.26	4.25	1.78	0.85	3.99
2	21.03	3.25	9.36	10.05	2.01

Table 4.1: Example of data obtained from SI modules, in the form of error values for each feature.

With regards to the complexity of the voting function within the coordinator node, it simply represents linear search complexity. The following pseudocode outlines the coordinator's voting function:

Algorithm 6 Coordinator Voting Scheme

```

1: MinFeature = 99.99
2: for  $i = 1$  to MaxPatternClass do
3:   for  $j = 1$  to MaxFeatureNo do
4:     if  $i.j.FeatErr \leq MinFeature$  then
5:       MinFeature =  $i.j.FeatErr$ 
6:     end if
7:   end for
8: end for

```

This function is used to find the minimum error value for each feature, obtained from the recognition process. Similar to vote counting function in SI module nodes, we can derive a big-O notation for the coordinator's voting function, $f(v_{\min})$ as a n -polynomial function with executable instructions, $n = 2$. Hence the following big-O notation applies:

$$f(v_{\min}) = O(n^2) \quad (4.4)$$

Figure 4.3 shows the estimated execution time for the voting function with increasing number of features used, n_{feat} for 10000 pattern classes involved in the recognition process, assuming that an instruction takes up $1 \mu s$ of the computation time.

Note that the minimum voting function takes only one second to select the lowest error value from 100 features on 10000 pattern classes (trained patterns). This exhibits higher scalability for recognition approach using our proposed multi-feature scheme, through distribution of recognition procedure on a group of collaborative DHGN networks.

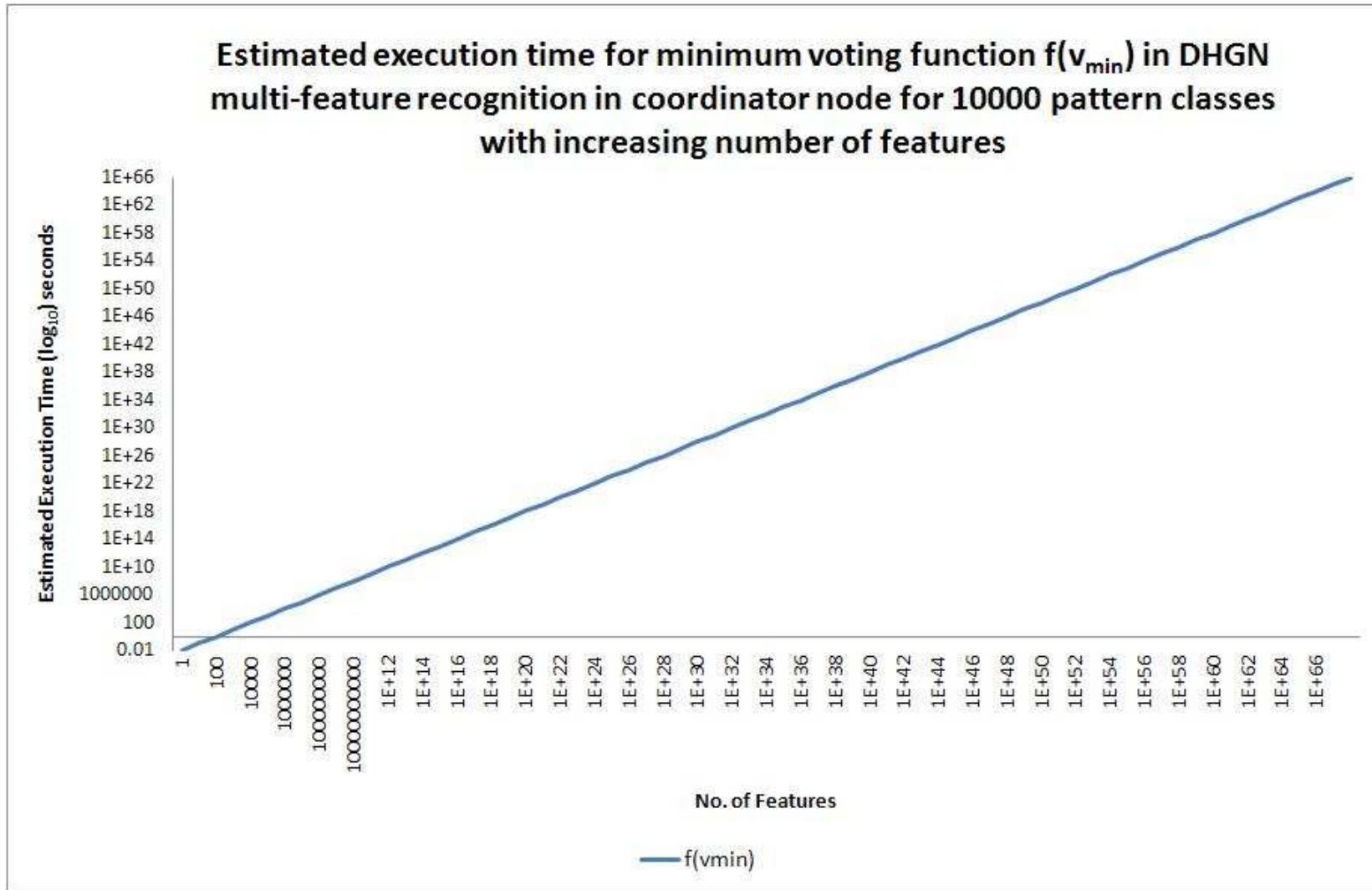


Figure 4.3: Estimated execution time for minimum voting function within coordinator node for 10000 pattern classes with increasing number of features.

Based upon the complexity analysis presented, we can deduce that DHGN multi-feature recognition scheme allows recognition to be performed in a scalable manner, extending a capability to use large number of features for patterns in its recognition procedure. By having a distributed architecture in recognition scheme, DHGN provides an avenue for recognition/classification to be executed in a highly-scalable fashion, while maintaining its low computational complexity.

A point to note however, that the proposed pre- and post-processing mechanisms described in this section do not entail a rigid framework. Different data analysis and feature extraction may be accommodated in this scheme. In this context, the proposed multi-feature scheme may be considered as a commodity application that can be used in different kinds of application domains. Further discussions on this matter will be presented in Chapter 5.

In regards to the applications of DHGN multi-feature recognition, both face and hand-written image recognition have been tested. The following two sections provide descriptions of this current work on image recognition using distributed multi-feature approach.

4.3 Greyscale Image Recognition using Multi-Feature Approach

In this section, we intend to demonstrate the capabilities of our proposed approach to achieve multi-feature pattern recognition using collaborative-comparison single-cycle learning in DHGN within a computational network. This chapter will demonstrate that our distributed pattern recognition scheme is able to include multiple image features as inputs within the recognition process. It is also capable of providing accurate classification within the bounds of single-cycle learning. The proposed multi-feature DHGN is readily deployable within various network environments, ranging from coarse-grained computational networks such as computational grid network to fine-grained networks such as the WSN (See Chapter 5).

Accuracy in image recognition is generally the benchmark against which most contemporary schemes are measured against. Scalability and learning flexibility are other key aspects that must also be taken into consideration if the effectiveness of a scheme is to be fully assessed. Existing DHGN distributed pattern recognition scheme has been able to

provide high scalability for the recognition scheme, as explained in both Chapter 2 and 3. Furthermore, our proposed collaborative-comparison learning enables memorisation of patterns to be conducted with low level of complexity.

In this section, we extend our DHGN implementation to include a multi-feature approach for facial image recognition. Our approach looks into two distinctive features for facial images, namely colour intensity and edge. The following subsections described the work that has been carried out.

4.3.1 Multi-Feature DHGN for Facial Image Recognition

The proposed scheme takes a holistic approach towards incorporating both the colour and spatio-structural features into the image recognition process simultaneously. A binary signature scheme has been adopted for content-based image retrieval (CBIR) proposed by Nascimento and Chitkara (2002) within a pattern recognition procedure. This scheme integrates this global binary signature with Sobel's edge detection (Kimmel et al., 2005) for DHGN single-cycle image recognition. This approach simply follows the design of multi-feature DHGN scheme as described in Section 4.2.

A number of networks were selected, each comprises a number of DHGN subnets for each feature within an image. Thus any number of features can be included for pattern analysis by incorporating a sufficient number of DHGN networks.

Global Binary Signature Scheme for Colour Recognition

A common approach in representing colour distribution within an image is to use a Global Colour Histogram (GCH). Given an n -colour model, a GCH is developed with an n -dimensional feature vector $\{p_1, p_2, \dots, p_n\}$, where p_i represents the normalised percentage of colour pixels that corresponds to each colour element within an image. Nascimento and Chitkara (2002) have proposed an alternative approach for colour distribution representation by using a global binary signature scheme. It is a compact form of existing GCH that uses binary bit-strings as a signature. This signature is an abstract representation of the image's colour distribution. The bit-strings are of a pre-determined size, which makes it ideal for use within DHGN binary pattern representations. Further explanations of the proposed binary signature scheme can be referred to in Section 3.2.3.

Sobel's Edge Recognition for Structural Information

Edges provide important spatio-structural information for image recognition. The proposed scheme of this thesis therefore includes the edge detection into its colour-based recognition process. Sobel's edge detection mechanism has been adopted, such that outputs from the edge detection process are represented as an edge map. Figure 4.4 shows the transformation of a greyscale image into the corresponding edge map. In our implementation, we have used a common detection threshold value of 70, to generate the edge maps.

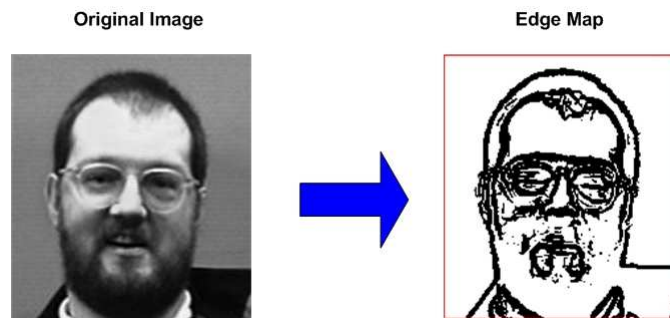


Figure 4.4: Edge map generation using Sobel's edge detection technique when applied to an original greyscale facial image.

With the ability to capture and convert the two main features of an image, namely colours and edges into binary patterns, the researcher in this thesis was able to apply a highly scalable single-cycle learning technique for binary patterns within computational networks towards multi-feature pattern recognition. Any number of features may be included in the scheme as long as a separate network is available for each feature (see Figure 4.2).

4.3.2 Recognition Accuracy Analysis

The following tests have been conducted to investigate the relative usefulness of our multi-feature approach. There are two important factors that need to be investigated, namely recognition accuracy and recognition speed. For recognition accuracy, the simulation results were compared with a standard back-propagation neural network (BPNN). This simulation involves multi-feature detection within greyscale facial images using the DHGN network. In this regard, 80 DHGN subnetworks, of 18 nodes each were used to handle horizontal image segments of 5 bits.

Facial images were chosen that include similar background condition and different structural representations as the test data set. The simulation program was executed within a parallel distributed environment to represent natural characteristics of a computational network. A set of 1000 facial images of 50 different individuals were used in this study, retrieved from Face Recognition Data, University of Essex, United Kingdom. These images were of the size 180 x 200 pixels as shown in Figure 4.5.



Figure 4.5: Fifty different individuals in the face image dataset obtained from the Face Recognition Data.

For colour recognition, all the greyscale images were quantised into 4 grey-levels ($n = 4$). Ten different range of values was used (termed as bins (See Section 3.2.2)) ($j = 10$) for signature representation. It was determined that these values were able to represent distinctive colour feature for all the analysed images. 40-bit signatures were created from this process. Each signature was then decomposed into sub-signatures of size 5-bits for use with the DHGN setup.

For edge detection, small-scale edge maps were developed for all the facial images. Each edge map was a binary representation of the image with the size of 18 x 20 pixels, and it was developed using 3x3 Sobel's matrix kernel with both horizontal and vertical scanning procedures. These edge maps were also decomposed into 5-bit subpatterns for the DHGN recognition process. The multi-feature recognition scheme was implemented using a message-passing DHGN model with fully-distributed approach which will be later described (Chapter 5).

4.3.3 Results and Discussion

The recognition tests involved classification of 1000 facial images into 50 classes, corresponding to 50 distinct individuals. Each class consists of 20 images of the same individual with different facial expressions. These images were converted into greyscale images, to standardise the background representation for these images. For both multi-feature DHGN and BPNN schemes, 50 randomly selected individual images were used as the training set, while the rest of the images were used as the testing set.

Recognition Accuracy

In our simulation, two separate studies were conducted to assess on the accuracy of multi-feature DHGN using single-feature recognition and dual-feature recognition.

Single-feature recognition. This study relates to facial recognition using only image colour for binary signature generation. The results from this test show that our single-cycle learning DHGN scheme has been able to produce an error value of 0.0173 (Equivalent to 1.73%). 42% of the 50 image classes produced 100.00% recall, i.e. perfect recognition. In this implementation, recall error was defined as the number of wrongly classified images from the overall 950 images used in the test. Figure 4.6 shows the error values for all the 50 facial image classes obtained using DHGN scheme.

To improve this result, larger quantisation value may be used in order to provide better greyscale value representation of the facial images.

A comparison between our single-feature DHGN scheme and iterative Hebbian-based learning back propagation neural network (BPNN) however showed that the approach generally produced a lower recall error as compared to the BPNN for all the image classes. Table 4.2 shows the parameters for BPNN implementation. From the results shown in Figure 4.7, BPNN incurred an overall of 59.85% error for this test which is on the average substantially higher than that of DHGN. High error values for BPNN may be due to the low number of training images introduced to the network. Hence, optimum outputs that mostly reflect the original trained images were not achieved.

The results also indicate that colour recognition alone does not provide overall information on an image. So the next set of tests incorporates recognition of multiple features by including edge information within the greyscale image analysis.

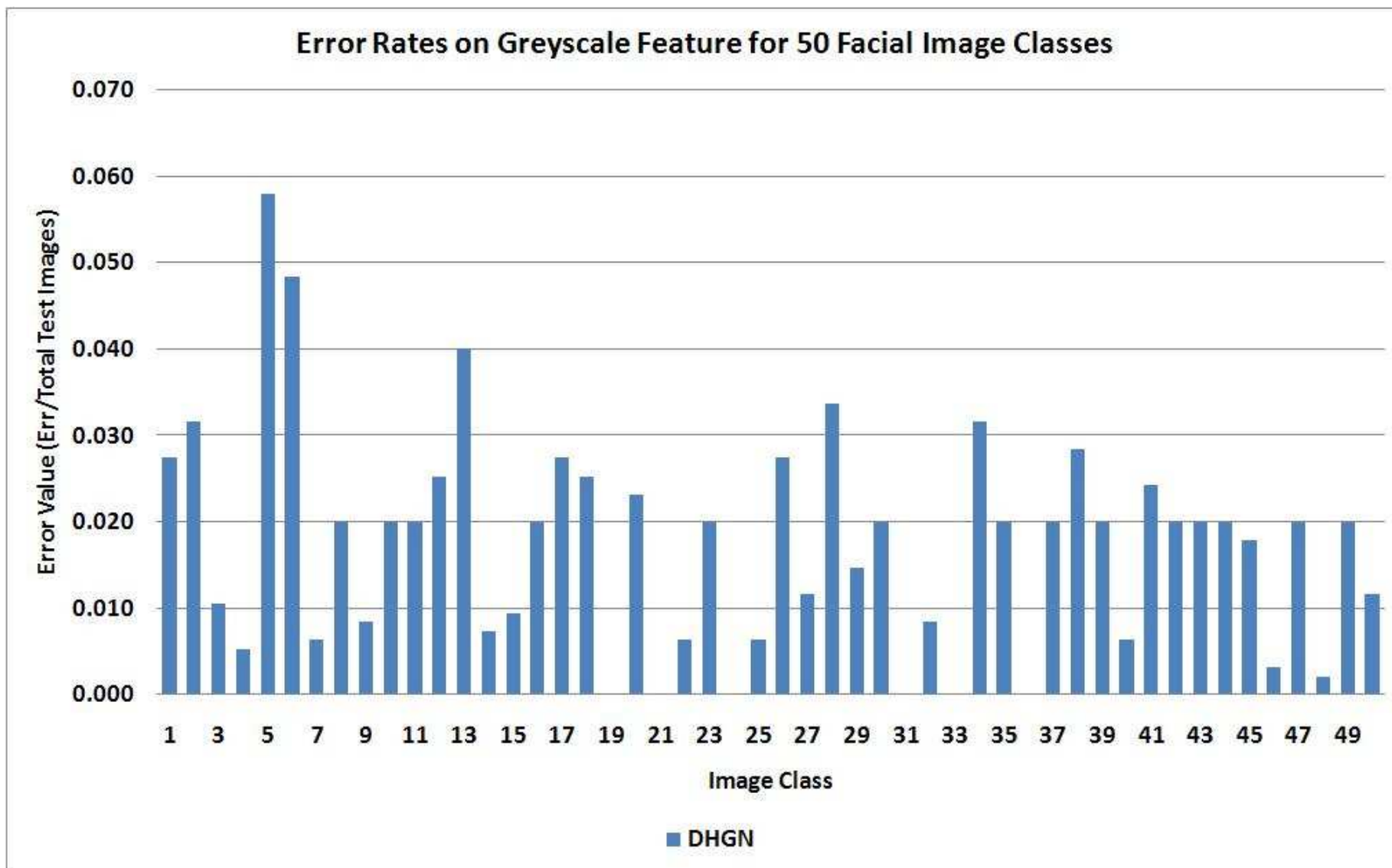


Figure 4.6: Error values obtained for 50 facial image classes from original DHGN implementation using only colour feature on 1000 test images.

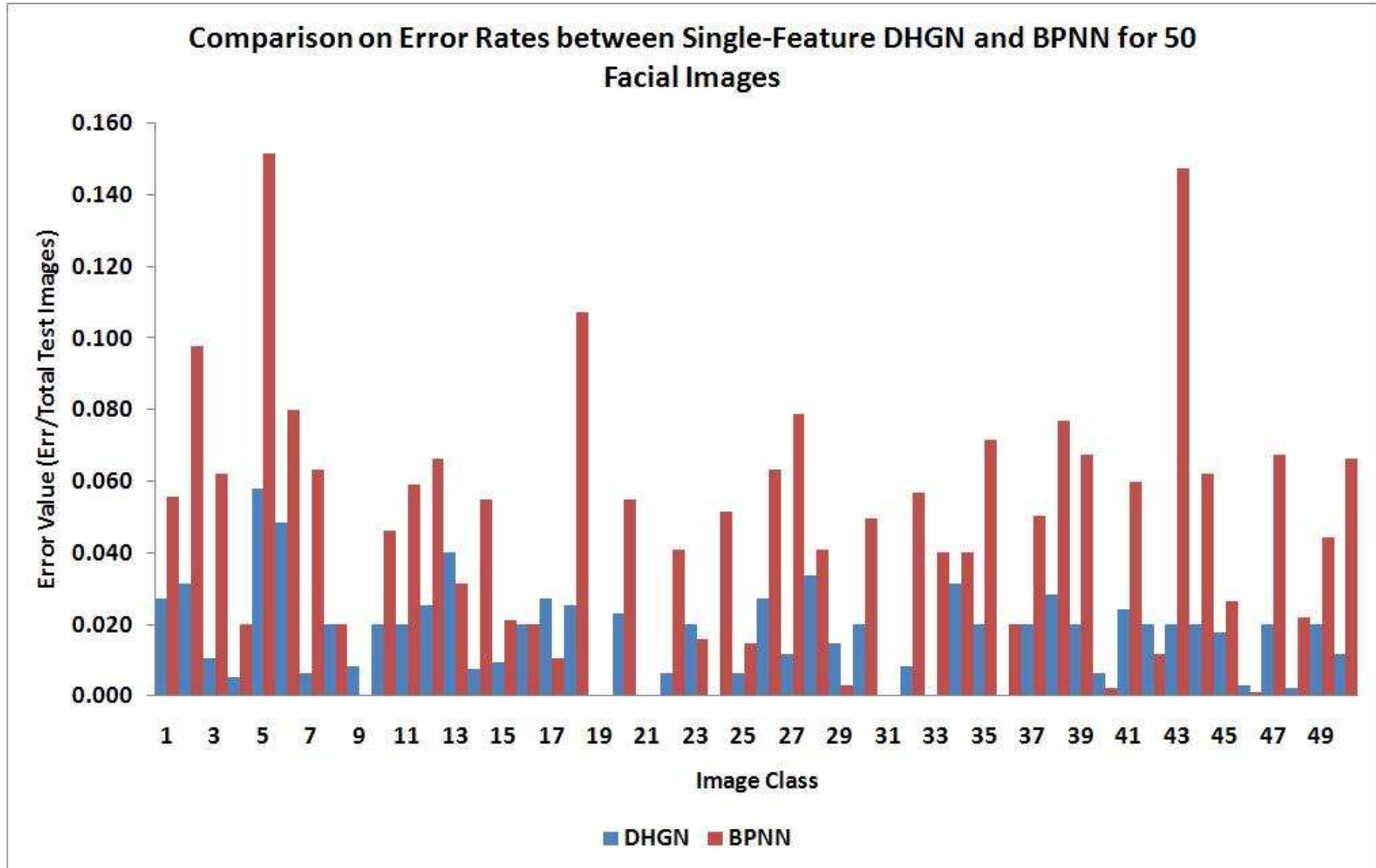


Figure 4.7: Analysis on error values obtained from recognition simulation on greyscale value feature using DHGN and BPNN.

Parameters	Values	
Epoch	1000	
Layer	3	
No. of Neurons	Input Layer	4
	Hidden Layer	8
	Output Layer	1

Table 4.2: BPNN execution parameters.

Dual-feature recognition. Dual-feature recognition on facial images integrates both greyscale recognition and edge detection into a single process. The results from both recognition processes was analysed by the coordinator. In this simulation, minimum voting within the coordinator node is performed to obtain the minimum error value for each of the facial images used. Figure 4.8 shows all the error values retrieved by DHGN multi-feature scheme.

The recall/store decision was made using the voting mechanism as previously done by Muhamad Amin et al. (2008). Figure 4.9 shows the dual-feature recognition results, in terms of the error values produced and its comparison with the BPNN approach.

It is evident from the result shown in Figure 4.9 that the recognition using both edge and greyscale features vastly improved the image recognition accuracy. The overall recall error value was reduced from of 0.0173 (Equivalent to 1.73%) to of 0.0081 (Equivalent to 0.81%), while 54% of the images recorded 100% recall. These results show that the multi-feature DHGN scheme can integrate multiple features simultaneously to increase the image recognition accuracy.

Recognition Efficiency

The execution times taken for store/recall within each DHGN subnet for edge feature recognition are shown in Figure 4.10. The recognition approach used in this study took on the average less than 50 milliseconds for each store/recall process per image. Furthermore, this time limit remained consistent throughout the test.

The results of the store/recall time has once again shown the significant contribution of our proposed scheme, in providing fast recall time for pattern recognition using a distributed approach. Similar results can also be referred to in Chapter 5.

Figure 4.11 shows the overall store/recall time for each DHGN subnet for processing the input patterns. Each DHGN subnet recorded a total execution time of less than 30

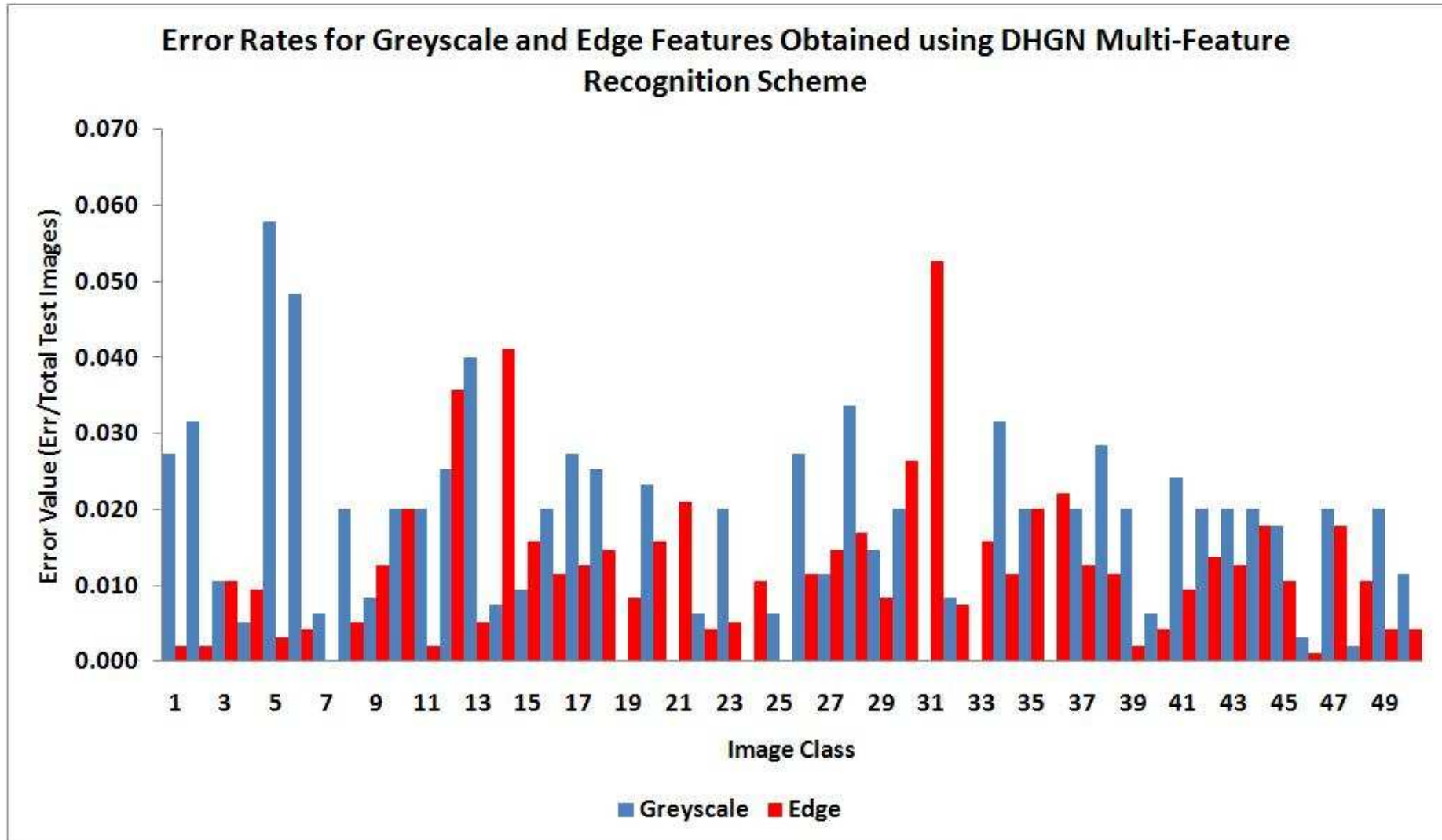


Figure 4.8: Comparison on error values between greyscale value and edge feature on 50 facial images obtained using DHGN multi-feature scheme.

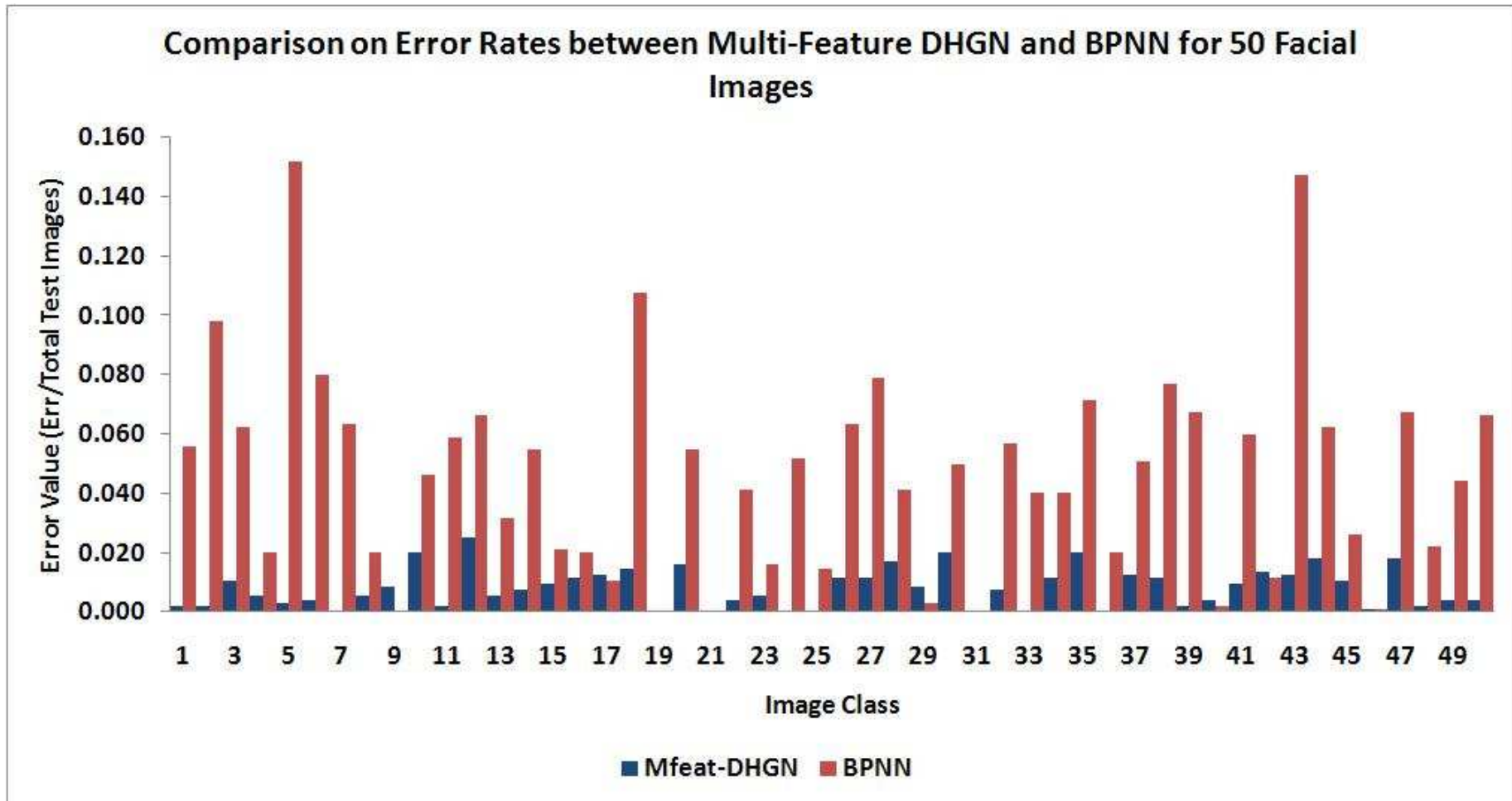


Figure 4.9: Analysis on error values obtained from recognition simulation on greyscale and edge features using Multi-Feature DHGN and BPNN.

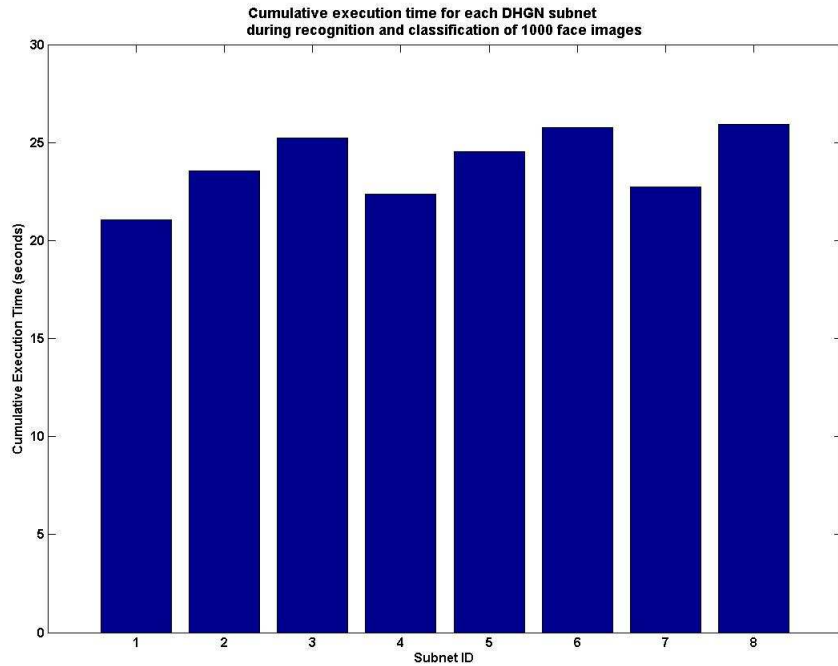


Figure 4.10: Total execution time for each subnet in DHGN network for edge recognition process.

seconds for processing all the 1000 images within a simulated computational network. The processing times will substantially reduce for a real computational network with parallel processing resources, making it possible to process live image data streams and large data sets in real-time.

4.3.4 Concluding Remarks

This section has shown that our multi-feature DHGN recognition scheme displays superior single-cycle learning and improved accuracy to the existing Hebbian learning techniques such as the back-propagation neural network (BPNN). Nevertheless, the comparison made solely based upon the objective of demonstrating DHGN's capability for multi-feature recognition. DHGN multi-feature scheme provides a highly efficient and scalable mechanism for undertaking multi-feature pattern recognition using computational networks. This multi-feature recognition approach represents a holistic process where more features can be taken into consideration without any changes to the approach. The scheme is shown to be highly scalable where the processing time and recognition accuracy are not

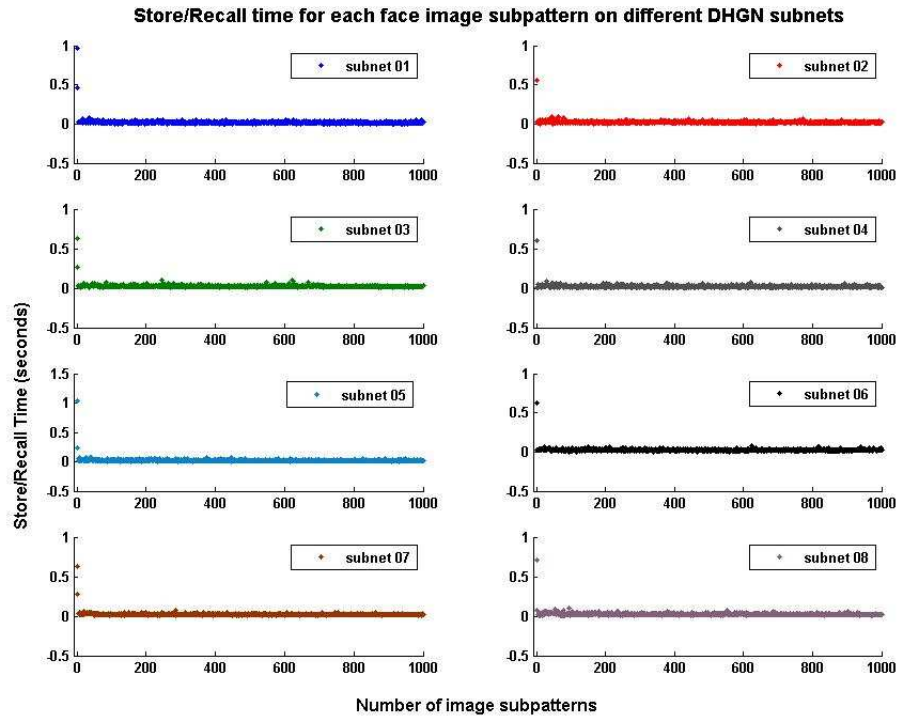


Figure 4.11: Store/recall time for each subpattern within each DHGN subnet for edge recognition process.

adversely affected with the increase in number of processed patterns. The approach discussed in this chapter works well on greyscale images and it can thus be applied across a number of areas requiring greyscale image analysis. The flexibility to include any image feature at any point creates a 'plug-and-play' capability for dynamic image analysis. This scheme opens up the possibility of real time image recognition to be performed on large data sets in biomedical imaging and video streaming. Furthermore, through distribution of features, DHGN also capable of performing recognition process, with increasing size and dimension of patterns. It is best to note however, that the use of facial images in the recognition simulation conducted does not imply DHGN as a face recognition application with promising high level of accuracy. Rather, the simulation indicates the capability of DHGN to perform multi-feature recognition on complex patterns such as greyscale images.

4.4 Handwritten Object Recognition with Multiple Features

Apart from greyscale facial image recognition, a series of tests were also conducted using DHGN multi-feature scheme on handwritten character recognition, using feature data

obtained from Frank and Asuncion (2010). In this section, the capabilities of DHGN distributed scheme as a single classifier for combined multi-feature pattern recognition will be demonstrated. A comparative evaluation with the previous works done by Duin and Tax (2000) will also be discussed. It is best to note however, that this work on DHGN multi-feature scheme is not intended for showcasing an optimum solution with high accuracy for complex pattern recognition. Rather this study has been carried out to provide an alternative approach for pattern recognition scheme involving multiple features.

4.4.1 The Data Set

The tests have been carried out on a data set containing four out of six different feature sets obtained from a similar set of objects. It contains 2000 handwritten numeral characters extracted from a set of Dutch utility maps. This data set comprises of ten classes of characters ranging from numerals “0” to “9”, in which each class holds 200 objects. This data set is also publicly available from Machine Learning Repository by Frank and Asuncion (2010). Each object character in this data set was converted to a 30 x 48 binary image.

For the purpose of recognition tests, the data set was divided into training and testing data sets. The training data contains 10 objects for each class, while the remaining 190 objects are used for testing. The four features that have been used in this analysis include:

- i. Fourier: 76 Fourier coefficients of the character shapes.
- ii. Pixel: 240 pixel averages in 2 x 3 windows.
- iii. Zernike: 47 Zernike moments.
- iv. Morph: 6 morphological features.

4.4.2 Classification Procedures

The classification process in our proposed DHGN multi-feature scheme involves a series of single-cycle stages that have been applied to the feature data set of numeral characters described earlier. A 3-stage process was implemented in the recognition scheme; feature pre-processing, recognition, and results evaluation. It should be noted that our proposed scheme implements a single-classifier for multi-feature recognition. The following subsection will detail out these implementation stages.

Feature Pre-Processing

In the pre-processing stage, all selected features will undergo a discretisation process, in which continuous feature value is transformed into a discrete format. This process is a pre-requisite for existing DHGN scheme that implements recognition procedure using discrete-format data values.

A discretisation was performed on the feature set using a binning approach. For each feature set, with an exception on pixel average, five bins (thresholds) were defined with a different range of values. These bins were created based on the maximum, minimum, and mean values obtained from the whole feature set. Table 4.3 shows instances of the bin range values for all features used. For pixel average, the exact values were used, since the data provided are of discrete values.

Feature	min	max	μ	Bins				
				1	2	3	4	5
Zernike	0.0011	777.86	88.64	≤ 25	26-50	51-90	91-400	401-800
Fourier	0.0002	0.7965	0.1320	≤ 0.001	0.002-0.05	0.06-0.14	0.15-0.50	0.51-0.80
Morph	1.1431	17572.2	2104.4	≤ 50	51-500	501-2500	2501-10000	10001-18000

Table 4.3: Discretisation on feature data values using variable-binning methods.

This discretisation process allows a reduction in feature data composition by transforming the feature set from continuous data space to a discrete one. This reduces the complexity of the data set to be used in the recognition procedure, however, it also exposes the data set to inaccurate data representation, as some of the actual values are lost during the conversion.

The output of this discretisation process is a set of patterns for each feature. These patterns correspond to all the test objects used in the tests. Table 4.4 shows a sample of patterns for Zernike moment features obtained from the discretisation process. The size of patterns reflects the number of values/coefficients for each feature, while the dimension of patterns corresponds to the number of bins used, i.e. 5.

Multi-Feature Recognition

The multi-feature recognition for multiple features of numeral character objects have been carried out using our proposed DHGN scheme, as shown in Figure 4.2. Four different DHGN networks were implemented to process feature patterns obtained from the pre-processing stage. Each network performs recognition on a specific feature set. In this

Object ID	Feature Pattern
1	12343000132001234000030000400120033010033011400
2	01243000222001134001320022300220131012033011400
3	01243000121001234001200022400100130011034011400
4	12243000112001233001120023300120131001034002400
5	01143000122001234000120022300120132001033012400

Table 4.4: Sample of Zernike moment's feature patterns obtained using discretisation.

regard, the composition for each network is different from one another, in terms of its size. Table 4.5 shows details of the DHGN architecture that have been used in this multi-feature recognition. This simulation adopts the recognition scheme using multi-value format for DHGN implementation (See Chapter 3).

Parameters	Values	
No. of DHGN networks	4	
Subpattern size	9	
No. of GNs per subnet	25	
No. of subnets	Zernike	5
	Fourier	9
	Morph	1
	Pixel Avg.	27

Table 4.5: Multi-feature DHGN architecture parameter details.

The recognition process starts with the coordinator node communicating the entire feature patterns to SI module node on each network according to specific feature that has been assigned. The communications of patterns in this scheme follow a message-passing model that will be further described in Chapter 5.

SI module node in each network then divides and distributes the received patterns to all the available subnets within the network. Each DHGN subnet then initiates a recognition process at subpattern level. The results of this recognition process is sent back to the SI module node for maximum voting process to identify the best match pattern class for each respective pattern. After a completion of this process, SI module will then determine the values of accuracy parameters used in the scheme. These parameters may include precision rate, recall rate, accuracy level, and error value. These values will then be communicated to coordinator node for results evaluation stage.

Results Evaluation

The results evaluation stage involves a process of determining the best or optimum feature to be selected as the best representative for each pattern class in the recognition scheme. This process occurred within the coordinator node. The values obtained from SI module node were compared to accuracy parameter(s) used. For the purpose of this recognition test, error value, precision, recall and accuracy value were selected as our parameters for evaluation.

The results evaluation stage in DHGN multi-feature recognition applies rather a generic approach, in which different sets of recognition accuracy parameters may be used in the classification process. In this perspective, it allows decision on classification to be implemented in a flexible fashion, in which different accuracy factors may be observed and analysed.

4.4.3 Results and Discussion

As mentioned in Section 4.2, the recognition tests have been carried out on the feature data sets of 2000 numeral character objects with 100 training and 1900 test objects inclusively. A number of recognition accuracy parameters were selected for use in classification decision, including error value, precision and recall rates, and accuracy value. Table 4.6 shows detailed descriptions on each of these parameters, given values of total number of test objects c_{test} , true positive t_{+ve} , false positive f_{+ve} , true negative t_{-ve} , and false negative f_{-ve} .

Recognition Parameters	Representations
Precision	$\frac{t_{+ve}}{t_{+ve} + f_{+ve}}$
Recall	$\frac{t_{+ve}}{t_{+ve} + f_{-ve}}$
Accuracy	$\frac{t_{+ve} + t_{-ve}}{t_{+ve} + t_{-ve} + f_{+ve} + f_{-ve}}$
Error Value	$\frac{f_{+ve} + f_{-ve}}{c_{test}}$

Table 4.6: Recognition accuracy parameters and their respective representations.

Recognition Accuracy Analysis

Table 4.7 shows the results of classification decision obtained from the outputs of the proposed DHGN multi-feature recognition scheme. The value of the best result (selected feature) for each object class is underlined.

Precision											
Feature	Object Class										Average
	0	1	2	3	4	5	6	7	8	9	
Morph	<u>0.8863</u>	0.2538	<u>0.6550</u>	0.3744	0.4524	0.0000	0.4973	0.0000	<u>0.9734</u>	0.0000	0.4092
Zernike	0.3557	0.3945	0.3079	0.2845	0.1831	0.1754	0.3118	0.5625	<u>0.4225</u>	0.3059	0.3304
Fourier	0.3944	0.2071	0.3514	0.3333	0.2028	<u>0.4700</u>	0.3492	<u>0.5672</u>	0.6338	0.3107	0.3820
Pixel	0.3023	<u>0.4212</u>	0.5685	<u>0.4061</u>	0.6994	0.3750	<u>0.5188</u>	0.5560	0.5577	<u>0.4811</u>	<u>0.4886</u>
Recall											
Feature	Object Class										Average
	0	1	2	3	4	5	6	7	8	9	
Morph	<u>0.9842</u>	<u>0.9684</u>	<u>0.5895</u>	0.3842	0.1000	0.0000	<u>0.9632</u>	0.0000	<u>0.9632</u>	0.0000	<u>0.4953</u>
Zernike	0.6684	0.6000	0.5526	0.3579	0.1368	0.1579	0.1526	0.3316	0.1579	0.1368	0.3253
Fourier	0.9632	0.3684	0.4789	0.2842	0.1526	<u>0.2474</u>	0.2316	0.4000	0.2368	0.1684	0.3532
Pixel	0.4789	0.6895	0.5895	<u>0.4895</u>	<u>0.6368</u>	0.2211	0.4368	<u>0.7579</u>	0.1526	<u>0.2684</u>	0.4721
Accuracy											
Feature	Object Class										Average
	0	1	2	3	4	5	6	7	8	9	
Morph	<u>0.9858</u>	0.7121	<u>0.9279</u>	0.8742	0.8979	<u>0.9000</u>	0.8989	0.9000	<u>0.9937</u>	<u>0.9000</u>	<u>0.8991</u>
Zernike	0.8458	0.8679	0.8311	0.8458	0.8526	0.8416	0.8816	0.9074	0.8942	0.8826	0.8651
Fourier	0.8484	0.7958	0.8595	0.8716	0.8553	0.8968	0.8800	0.9095	0.9100	0.8795	0.8706
Pixel	0.8374	<u>0.8742</u>	0.9142	<u>0.8774</u>	<u>0.9363</u>	0.8853	<u>0.9032</u>	<u>0.9153</u>	0.9032	0.8979	0.8944
Error Value											
Feature	Object Class										Average
	0	1	2	3	4	5	6	7	8	9	
Morph	<u>0.0142</u>	0.2879	<u>0.0721</u>	0.1258	0.1021	<u>0.1000</u>	0.1011	0.1000	0.0063	<u>0.1000</u>	<u>0.1009</u>
Zernike	0.1542	0.1321	0.1689	0.1542	0.1474	0.1584	0.1184	0.0926	0.1058	0.1174	0.1349
Fourier	0.1516	0.2042	0.1405	0.1284	0.1447	0.1032	0.1200	0.0905	0.0900	0.1205	0.1294
Pixel	0.1626	<u>0.1258</u>	0.0858	<u>0.1226</u>	<u>0.0637</u>	0.1147	<u>0.0968</u>	<u>0.0847</u>	0.0968	0.1021	0.1056

Table 4.7: Results of the classification decision on 4 different features of numeral character objects.

The results of the analysis have shown that morphological and pixel average features produce best and optimum recognition rates for most of the object classes. On average, morphological feature indicates low error value and high recall and accuracy rates. On the other hand, pixel average feature gives a high precision rate. These two features are closely-related to the structural information of the numeral character objects. It is not surprising that these two features effected a high level of recognition accuracy, in comparison with other features, since structural information is essential in determining the differences and similarities of character objects, including handwritten numerals.

With a focus on error value, all the estimated error values are obtained from the same data set and thereby not independent. It was observed that on average there were 192

objects that have been erroneously classified by the scheme from the total of 1900 test objects. In this regard, an error estimate for this finite set has a standard deviation of 0.007, which is insignificant.

Comparative Analysis

In comparison with other classifiers, described in Duin and Tax (2000), DHGN multi-feature recognition implements a comparable level of recognition accuracy for the same feature sets. Table 4.8 shows some results of comparative analysis that have been carried out on the error values obtained from all the classifiers.

Classifiers	Features			
	Morph	Zernike	Fourier	Pixel
Bayes Normal-2	0.3100	0.2120	0.2520	0.0620
Bayes Normal-1	0.2910	0.1800	0.2130	0.0990
Nearest Mean	0.5400	0.2780	0.2240	0.0960
1-NN	0.5700	0.1970	0.1920	<u>0.0370</u>
k-NN	0.5100	0.1930	0.1890	<u>0.0370</u>
Parzen	0.5210	0.1850	0.1710	<u>0.0370</u>
Fisher	0.2820	0.2100	0.2480	0.1530
Decision Tree	0.3290	0.5980	0.4540	0.5490
ANN-20	0.3280	0.9000	0.9000	0.8520
ANN-50	0.7170	0.2650	0.2450	0.8100
SVC-1	0.8480	0.2940	0.2460	0.0770
SVC-2	0.8110	0.1930	0.2120	0.0600
DHGN	<u>0.1011</u>	<u>0.1347</u>	<u>0.1295</u>	0.1058

Table 4.8: Comparative analysis on error values between DHGN and other classifiers for similar data set with respective features.

Based on the outcome of comparative analysis that has been performed, DHGN imposes lowest error values for all the tested features with an exception on pixel average. Nevertheless, DHGN performs sufficiently accurate on pixel average with a difference of 0.0688. The results also reveal that artificial neural networks (ANN) perform least accurate, in the sense that they incur a high rate of misclassification in the recognition process. On the other hand, other statistical approaches produce comparatively high level of accuracy. However, their complexity in regards to its parameter estimation process is significantly high.

Another important point to be discussed is that the number of objects in training and testing sets of the features in DHGN and other classifiers is different. In DHGN, we implemented 1:19 (100 to 1900 objects) ratio of training to testing data, while in the

research of Duin and Tax (2000), it is 1:1 (1000 to 1000 objects). In this perspective, this study only used 10% of the training data used in their research. For an efficient pattern recognition scheme, it is essential that the recognition scheme to use minimum number of training data, while producing high level of recognition accuracy. DHGN seems to acquire this characteristic.

4.4.4 Concluding Remarks

In this section, the study on the multi-feature recognition involving handwritten numeral characters using the proposed multi-feature DHGN scheme has been presented. An analysis of the accuracy of our proposed scheme has been made using four different sets of features derived from the objects. The results have demonstrated DHGN's ability to cater for multi-feature (combined) pattern recognition using small number of training data, within a simple computational scheme using in-network single-cycle processing approach, and have identified morphological and pixel average features as the best representation for this kind of character recognition, due to their ability to project the object's structural information. The benefits of distributed pattern recognition scheme have also been demonstrated. These include:

- i. Distributed approach in DHGN allows an extension to the number of features used within the recognition process, by allocating additional DHGN network for each feature recognition.
- ii. DHGN allows a single-classifier mechanism to be used over different number of features. Unlike existing multi-feature schemes that implement combined-classifiers for classification.

Comparisons with some other classifiers have shown that DHGN produces comparable level of accuracy, in terms of the overall error value. However, the main intention of this study is not to provide an optimum solution for multi-feature recognition with high-level of accuracy. Rather, this study intends to demonstrate DHGN's capability to perform multi-feature recognition as a scalable classifier with low level of complexity. This has been shown in our complexity estimation analysis for DHGN distributed scheme (see Sections 3.3.1 and 4.2.3)

4.5 Conclusions

This chapter has presented a distributed approach for multi-feature pattern recognition. We propose the use of single-cycle learning DHGN algorithm for distributed feature analysis on a collaborative computational network. The proposed approach implements a single classifier scheme for different feature sets. This is achieved using a divide-and-distribute approach on the available features for each data set. The study has proven that the proposed approach is not affected by the curse of dimensionality problem, which is mainly due to increasing number of features used. DHGN approach implements a scalable recognition scheme, by allowing features to be added in the analysis, using available computational networks.

This study implemented a multi-feature recognition scheme that allows multiple features to be included in the recognition and classification analysis. This provides more effective scheme that is capable of taking into account all features within the data analysis process. This is essential in data mining applications involving complex data, such as biomedical images that contain hundreds or thousands of features to be considered. We have also exhibit DHGN's ability to perform recognition and classification using small training data set, and at the same time producing comparably high-level of recognition accuracy. This characteristic is beneficial for effective pattern recognition that imposes minimum complexity while producing high accuracy.

The proposed scheme works equally well with two arbitrarily chosen pattern recognition problems i.e. greyscale images and handwritten expressions. Each of the two problems has quite different features sets and pattern recognition requirements. The approach is thus not only highly scalable and supports single-cycle learning, but it is also generic where large and complex data sets from a variety of sources and representing diverse pattern recognition requirements can be analysed in real-time. The scheme thus demonstrates that large-scale pattern recognition is possible through the distributed processing. Generic commodity based systems such as computer clusters and grid when enabled with a distributed learning scheme such as DHGN can effectively deal with data deluge of complex data comprising geometric relationships and structural properties, as previously described in Chapter 1.

Despite its robust and effective approach towards multi-feature recognition, there are a number of possible limitations in existing DHGN implementation. These include network availability. DHGN distributed scheme requires a distinct amount of available networks to perform its recognition procedure. By having this, DHGN scalability is again, highly-correlated with network availability. Further works on the analysis of network availability and resource-awareness in DHGN implementation will be presented in Chapter 5.

Chapter 5

Resource Considerations for Distributed PR

A distinctive difference in conventional and distributed pattern recognition is on the resource consideration. In a distributed approach, the system must be capable of utilising the available resources effectively and efficiently. In acquiring this, a communication model needs to be considered to ensure proper utilisation and communication of resources between processing nodes.

Distributed pattern recognition (DPR) has the capability to scale up the process, with an increase in the size of the problem. However, it should be noted that scalability depends on the resource availability within a particular computational network. This resource availability may be influenced by the capacity and stability of the computational network.

The network capacity aspect in distributed applications such as DPR, may be observed in terms of the granularity of the network. Commonly, computational network may either be in the form of coarse-grained network such as Grid computing or in the form of fine-grained network as in wireless sensor network (WSN). These networks may differ in terms of their processing capacity and capability. Existing DPR schemes tend to be non-adaptive to different range of network granularity, due to specific application deployment that focuses on a single problem domain. DHGN distributed pattern recognition scheme as described in Chapter 3, has been developed with adaptive network granularity consideration (Muhamad Amin and Khan, 2008b), that enables the algorithm to be deployed in both coarse- and fine-grained networks.

Apart from capacity, the stability of a computational network also plays an important role in determining its resource availability. A stable network may be defined as a network with minimum or no resource interference due to fault or error occurrences. In order for a particular application such as DPR to smoothly performed its function with minimum or no interruptions, fault tolerance mechanism needs to be considered.

In this chapter, the discussion on DHGN message-passing as a communication model is extended, and an analysis of resource-awareness in DHGN implementation is presented. Existing distributed pattern recognition schemes may suffer from performance degradation due to fault or lack of processing and storage resources available within a given network. For instance, classification schemes such as artificial neural networks (ANNs) that have been commonly applied as classification techniques for event detection schemes in WSN, are unable to achieve optimum performance due to lack of computational resources. These schemes include the research by Catterall, Van Laerhoven and Strohbach (2003) through an implementation of Kohonen Self-Organising Map (SOM) in sensory data clustering for distributed event classification within WSN. SOM incurs extensive computations in its classification scheme, hence requires substantial processing resources.

In considering resource-awareness of DHGN implementation, two important aspects of computational networks have been identified. These are network granularity and fault tolerance mechanism. These aspects are highly-correlated with the aforementioned resource availability factors (capacity and stability)of computational networks. To evaluate the proposed scheme against these two aspects, this chapter will aim to:

- i. evaluate DHGN message-passing model for inter-process communication in distributed pattern recognition;
- ii. analyse the performance and adaptability aspects of DHGN against the granularity of network; and
- iii. consider fault tolerance as important implementation factor for DHGN as a DPR system.

The composition of this chapter is as follows: Section 5.1 provides an extensive discussion on DHGN message-passing model, that has been briefly discussed in Chapter 3. Message-passing model represents an implementation-based model of DHGN execution

within the body of network. Section 5.2 presents a discussion on network granularity aspect of DHGN implementation. This section intends to examine the robustness and scalability of DHGN DPR scheme against varying granularity of networks. In Section 5.3, we present a discussion on the fault tolerance aspect and the fault tolerance mechanisms adopted by DHGN recognition scheme. Finally, Section 5.4 concludes this chapter.

5.1 Message-Passing Model for DHGN

Process communication plays an important role in any distributed systems. It determines the efficiency of a system, in terms of how it deals with different network configurations and characteristics. Communication of processes within a particular network may happen through message exchanges between processing nodes. In any distributed system, each processing node may require data exchange with other nodes in order to complete a specific task or process. In this regard, a thorough analysis of inter-process communication should be carried out, to ensure that the proposed system is capable of handling different network conditions. Similarly, distributed algorithms being proposed require significant consideration on this process communication aspect.

A number of inter-process communication models have been developed in recent years. These include message-passing, shared memory, and mobile agent (Ghosh, 2006). These models provide assistance towards better understanding of the communication procedure occurring within a computational network. In DHGN implementation, we consider message-passing model for inter-process communication within a distributed network. This section briefly discusses this model.

Message-passing is one of the inter-process communication models that have been developed as a guideline for process cooperation between computing nodes in a parallel environment. This model is based on a set of fundamental principles including:

- i. Each process has its own local memory.
- ii. Processes communicate their data using message exchange structure (sending and receiving messages).
- iii. The transfer of data requires cooperative operations by each process involved, i.e. each send operation must have its corresponding receive operation.

The cooperative operations in message-passing model deal basically with how the communication is being conducted between processing nodes within a particular network. These operations formed the components of a message-passing library, which in return being used in the implementation of message-passing communication. Examples of message-passing libraries include Message Passing Library (MPL) that has been introduced for IBM SP2, Parallel Virtual Machine (PVM), and Message Passing Interface (MPI).

MPI library provides extensive portability and capable to be deployed on different kinds of platforms. MPI has been established as a standard specification for message-passing routines. An enhancement of MPI standard, known as MPI-2, offers dynamic task control, as well as the functional parallel I/O capability (Gropp et al., 1999).

Message-passing model in DHGN implementation has been designed to cater for different network formation, ranging from coarse-grained to fine-grained computational networks. DHGN has been developed with important principles of in-network and parallel processing capabilities. With that in mind, it has been developed using message-passing model. Two important components in this model include process actions and system synchronisation.

5.1.1 Process Actions

DHGN distributed pattern recognition scheme is being conducted over a computational network that comprises a collection of processing nodes. With this understanding, the message-passing model has been designed with an assumption that each neuron within the DHGN logical network is assigned to each processing node within a physical network. Hence, the followings apply:

- i. Each neuron has its own local memory.
- ii. Communications between neuron involve message-passing procedure.
- iii. Cooperative cooperation is being held for each message transfer.

As discussed in Chapter 3, DHGN recognition procedure involves a two-stage process, i.e. recognition at both subpattern and pattern level. In communication perspective, this procedure is conducted using different process actions between participating networks/processing nodes. According to Ghosh (2006), there are four process actions available within the message-passing model. These include:

- i. Internal action - action involving internal memory structure processes of particular processing nodes.
- ii. Input action - communication from external input to the system.
- iii. Output action - communication involving message send from a network to external entity.
- iv. Communication action - action involving message exchange between processing nodes in a network.

Figure 5.1 illustrates the process actions involved in a DHGN network with a single subnet, when pattern “01110” is introduced. Note that this network is capable of recognising binary patterns with 5-bits length, and takes the assumption that the SI module node is external to the network.

Given a DHGN network with equal-sized subnets for subpatterns with v different elements and size n , the process actions involved (input, output, and communication) may be derived using the representation as shown in Table 5.1.

Process Action	Representation
Input	n
Communication	$\sum_{i=0}^{\frac{n+1}{2}-1} 2v + (n - (2i + 2)) (v^2 + 1)$
Output	1

Table 5.1: Representations for different process actions in DHGN message-passing model.

The input action corresponds to the length of subpatterns used, while the communication action relates to the number of messages communicated within each subnet during each recognition process, as previously discussed in Chapter 3. The value 1 in output action is derived from the number of messages communicated between top node and SI module. This message contains an index retrieved from the recognition of a particular subpattern.

Collectively, the total process actions, T_{pa}^{DHGN} occurring in a DHGN network with S subnets may be derived using the following equation:

$$\begin{aligned}
 T_{pa}^{DHGN} &= snv + s \sum_{i=0}^{\frac{n+1}{2}-1} 2v + (n - (2i + 2)) (v^2 + 1) + s \\
 T_{pa}^{DHGN} &= s \left(nv + \sum_{i=0}^{\frac{n+1}{2}-1} 2v + (n - (2i + 2)) (v^2 + 1) + 1 \right)
 \end{aligned} \tag{5.1}$$

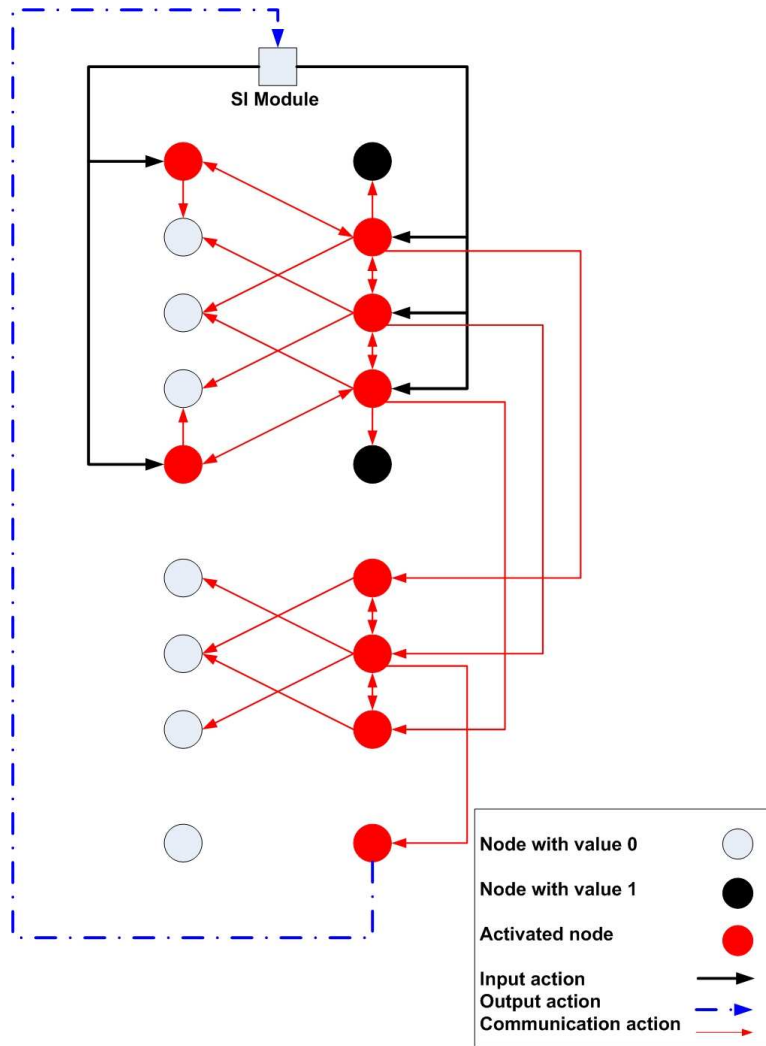


Figure 5.1: DHGN process actions within a message-passing model for binary pattern recognition on 5-bit patterns.

The complexity of DHGN in message-passing model in regards to the cost of process actions depend upon two different variables, namely pattern size and dimension. Size of patterns does not significantly increase the action cost for each processing node, in comparison with the dimension of pattern (number of different possible elements). An increase in the dimension of patterns requires the node to communicate the value retrieved from either SI module or node from lower layer within a subnet to all adjacent nodes. The number of adjacent nodes n_{adj} for each corresponding node in a DHGN subnet is directly proportional to the number of different elements r for a given pattern, in the form of $n_{adj} = 2r$ for one-dimensional DHGN structure. On the other hand, a pattern length increment will only affect the number of processing nodes required to perform a

recognition process. In this context, DHGN is scalable against an increase in the length of patterns but slightly susceptible to an increase in the dimension of patterns.

In summary, this section has considered all the process actions involved in the message-passing model of DHGN implementation. An exception is given to the internal action, since it mainly deals with the bias array access and search as being described in Chapter 2 and 3.

5.1.2 System Synchronisation

DHGN message-passing model implementation can either be performed in synchronous or asynchronous communication environment. This is mainly due to the flexibility of the algorithm, in which the recognition process at each node only require a linear search function on the bias array for matched (*value, position*) entries from adjacent nodes. Therefore, the urgency for the message to receive from one node to the other is not an important consideration in the implementation. However, in pattern recognition applications involving real-time response, this issue might need to be taken into account.

A significant difference between synchronous and asynchronous communications in DHGN implementation is that an increase in latency will be experienced in asynchronous mode. This latency occurs from the waiting time for the processing node to receive its neighbouring/adjacent indices or values in the form of communicating messages. Figure 5.2 shows the timeline diagrams for synchronous and asynchronous DHGN communication for a single processing node n with two adjacent nodes $n - 1$ and $n + 1$.

Note that an uncertainty, in terms of waiting period may occur in asynchronous communication. Moreover, the bias entry search can only be instantiated when all the indices/values have been obtained from all adjacent nodes. In an implementation using message-passing programming model such as MPICH-2 (Gropp et al., 1999), the use of a wait function, such as *MPI_WAIT()* is required in asynchronous (non-blocking) mode, to ensure that all the messages have been received at a particular node.

In synchronous communication, each processing node will ensure that each message arrived at the destination, before other process continues. This helps in the coordination of processes. However, in an unsecure and unreliable network environment, this may cause infinite waiting time, in the case of the lost of message transmission or erroneous messages retrieved.

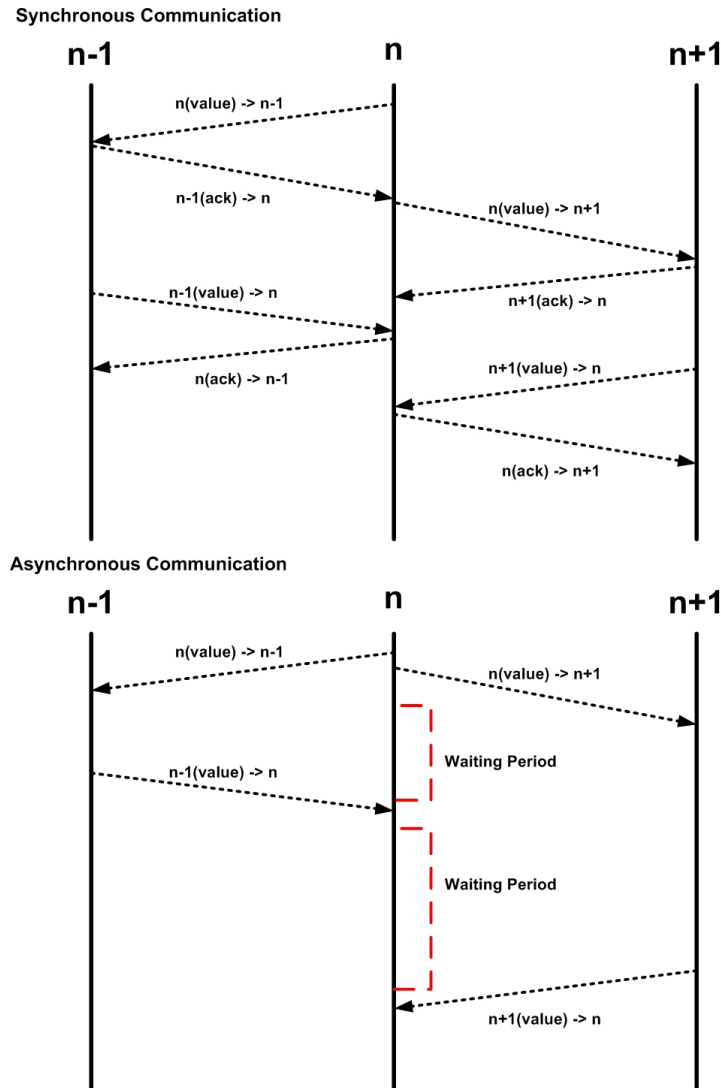


Figure 5.2: State chart for DHGN implementation in synchronous and asynchronous communications.

5.2 Network Granularity Analysis and Evaluation

Granularity of a computational network refers to the levels of its composition. Coarse-grained network mainly consists of a few large processing entities, which are capable of handling significantly high computational loads. An example of this kind of network is computational grid network. Conversely, fine-grained network can be defined as a network that comprises many small processing nodes that perform simple and lightweight tasks, such as the wireless sensor network (WSN). Table 5.2 shows some of the differences between coarse-grained and fine-grained computational networks.

DHGN implementation for distributed pattern recognition takes into account these two levels of granularity within any computational network. This is essential in providing

	Coarse-Grained	Fine-Grained
No. of Processing Nodes	Low to High	High
Processing Capacity	High	Low
Storage Capacity	High	Low
Energy Supply	High	Low
Example	Computational Grid	WSN

Table 5.2: Comparison between fine-grained and coarse-grained networks.

a scalable and robust scheme that could be used in different network conditions. Furthermore, with this network granularity consideration, the proposed DHGN algorithm may be made aware of the resource availability for a given computational network in which a recognition process is to be performed. This section will describe some of the DHGN concepts and implementations that have been applied in both coarse- and fine-grained networks.

5.2.1 DHGN Configurations for Adaptive Granularity

In considering different network capabilities and conditions, two different configurations for DHGN implementation were proposed; fully-distributed and clustered configurations.

Fully-distributed Configuration

The original configuration for DHGN algorithm, as described in Chapter 3 involves distributing all the GNs within DHGN subnet among the processing nodes. This implies that each node is responsible for a single GN within a DHGN subnet. This configuration eliminates the requirement for high processing capability and storage capacity since the recognition process only involves a single atomic element within an overall input sub-pattern to be processed by the computing node. However, the communication costs for each node are in need of considerable attention - each node is required to communicate with other neighbouring nodes frequently as to update its bias array. Figure 5.3 shows the fully-distributed configuration of DHGN algorithm for WSN. Note that each GN is mapped to a processing node. The processing nodes which are close together are grouped into individual DHGN subnets.

This fully-distributed DHGN configuration may be deployed in a fine-grained network such as WSN, in which each sensor node has restricted computing resources. A major issue in this implementation is such that it requires rapid inter-node communications for message

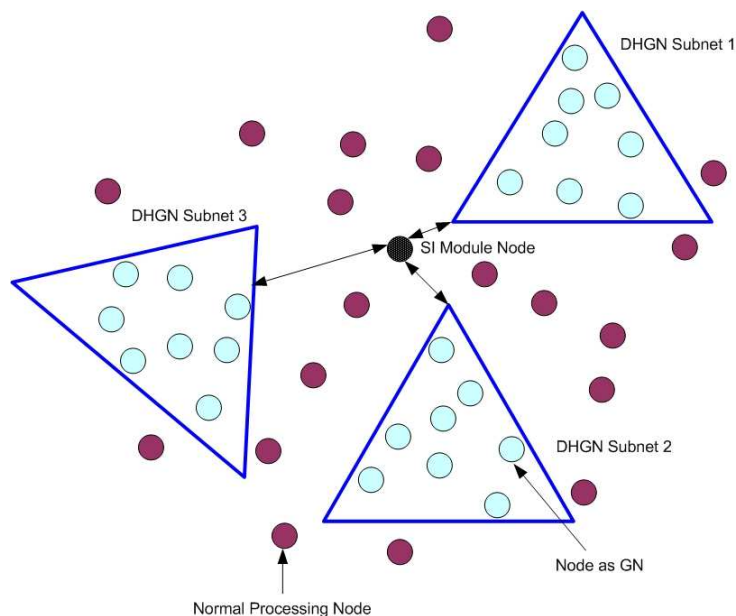


Figure 5.3: Fully-distributed DHGN configuration for fine-grained network.

exchange during recognition process. In this aspect, DHGN deployment must be able to perform single-hop communication between adjacent nodes for message exchange process. Hence, physical distance factor should be taken into account when implementing DHGN distributed recognition application in WSN, to ensure that efficient energy-communication utilisation is achieved during the recognition process.

Clustered Configuration

Clustered configuration involves mapping of each DHGN subnet over a single processing node. Each node is capable of conducting recognition process based on the input subpatterns obtained from the SI module node. In this configuration, processing node should acquire high processing capability and storage capacity, since the recognition process involves the entire input subpattern. However the communication costs between sensor nodes are minimised, since the communication only involves message communication from SI module node to each of the processing nodes. Figure 5.4 shows the clustered configuration of DHGN algorithm for WSN.

Each processing node in clustered DHGN configuration may performed recognition on each subpattern independently from other processing nodes. In this context, the node should be able to provide sufficient processing and storage capacity in order to conduct the recognition process. This configuration is intended to be used on coarse-grained networks

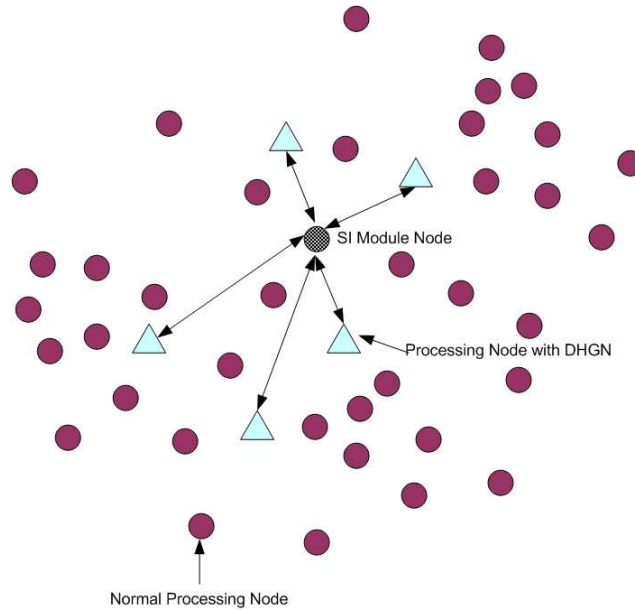


Figure 5.4: Clustered DHGN Configuration for coarse-grained network where each DHGN node is capable of performing the entire subpattern recognition processes.

such as grid and cloud computing, in which additional processing and storage capacity made available to be used. Nevertheless, this research intends to look into its implementation on scalable WSN networks.

An important benefit of having this DHGN cluster performed on a single processing node is such that it eliminates all the communication actions involved in DHGN message-passing model for distributed systems. For each subpattern recognition process, each node only communicate the corresponding index generated to the SI module. Therefore reducing the chances of recognition failures due to transmission or communication errors.

DHGN implementation using clustered configuration involves the formation of DHGN subnet using node's internal memory structure. An associative array structure for each DHGN subnet was adopted as shown in Table 5.3, for DHGN subnet with 5-bit binary subpatterns.

Communications between each GN memory structure in DHGN subnet are conducted using conventional value store/retrieve process in which internal bias array values are updated using value assignment as shown in the code snippet below:

```
GN[x+1].entry[LEFT] = GN[x].row;
GN[x+4].entry[LEFT] = GN[x].row;
```

GN ID	Row	Layer	Value	Bias Array	
				Index	Entry
1	1	0	0	1	#,1
2	1	0	0	1	1,1
3	1	0	0	1	1,1
4	1	0	0	1	1,2
5	1	0	0		
6	2	0	1	1	#,2
7	2	0	1	1	2,2
8	2	0	1	1	2,2
9	2	0	1	1	2,2
10	2	0	1	1	1,#
				2	2,#
11	1	1	#	1	#,1,1
12	1	1	#	1	1,1,1
13	1	1	#	1	1,1,#
14	2	1	#	1	#,1,1
15	2	1	#	1	1,1,1
16	2	1	#	1	1,1,#
17	1	2	#	1	1
18	2	2	#	1	2

Table 5.3: DHGN subnet associative array structure after subpatterns 00001 and 11111 have been memorised.

This code shows the process of assigning row number of activated GN (at the edge) x to its adjacent GNs.

5.2.2 Performance Analysis

Apart from recognition accuracy, the performance of DHGN distributed pattern recognition scheme with different configurations can also be observed from the recall and total execution times taken during each subpattern recognition process undertaken by each DHGN subnet. A series of recognition performance tests using DHGN message-passing model (with MPICH-2 implementation) have been conducted on both fully-distributed and clustered configurations.

Clustered DHGN

A series of recognition performance tests on DHGN recognition scheme on binary patterns have been conducted, and a 35-bit pattern recognition involving 7 DHGN subnets (each capable of storing/recalling 5-bit subpatterns) has been implemented. These tests have been performed on a HPC machines provided by Victoria Partnership for Advanced

Computing (VPAC). Each compute node in this architecture has 2 AMD 2356 Quad Core OpteronsTM, 32GB ram and 4 x 320GB disks (1.2TB scratch space). Inter-node communication is conducted using Infiniband Interconnect (2μ measured MPI latency) on 95 nodes.

This research examined total and average recall times for each recognition procedure performed on subpattern within each DHGN subnet. In clustered configuration, each DHGN subnet is assigned to a single compute node. The study implemented these tests on 8 compute nodes in which one of these nodes is assigned as the SI module node.

Figure 5.5 shows the total execution time for each DHGN subnet with increasing number of patterns used.

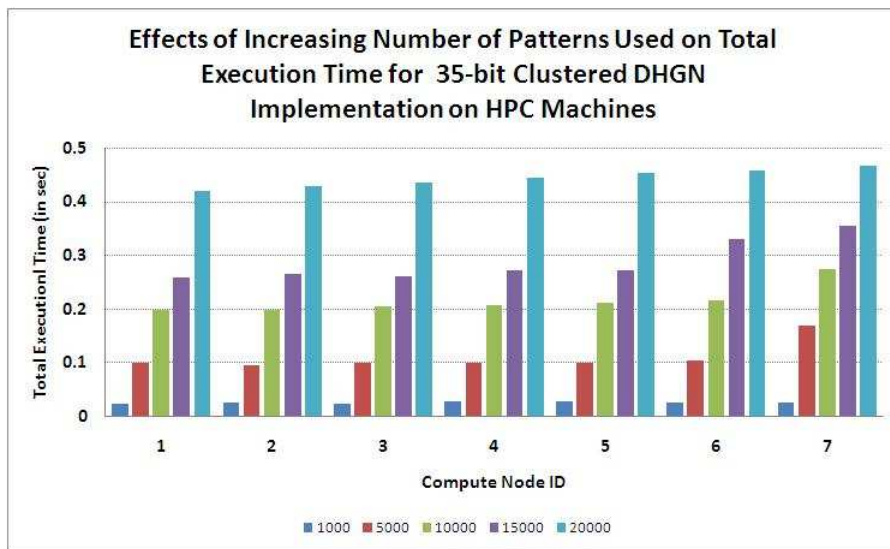


Figure 5.5: Effects of increasing binary pattern length on the total execution time for clustered DHGN implementation on HPC machines.

Throughout the tests, total execution times for all compute nodes (DHGN subnets) consistently increasing with an increase in the number of patterns used. Nevertheless, the execution times remain less than 0.5 seconds even for 20000 binary patterns with 5-bit length. This significant performance of DHGN demonstrates its enhanced scalability and robustness for coarse-grained network deployment. Furthermore, with abundant resources available for each compute node, DHGN has a greater ability to produce fast recognition procedure.

The average recall time for each subpattern in clustered DHGN implementation was evaluated, and is shown in Figure 5.6.

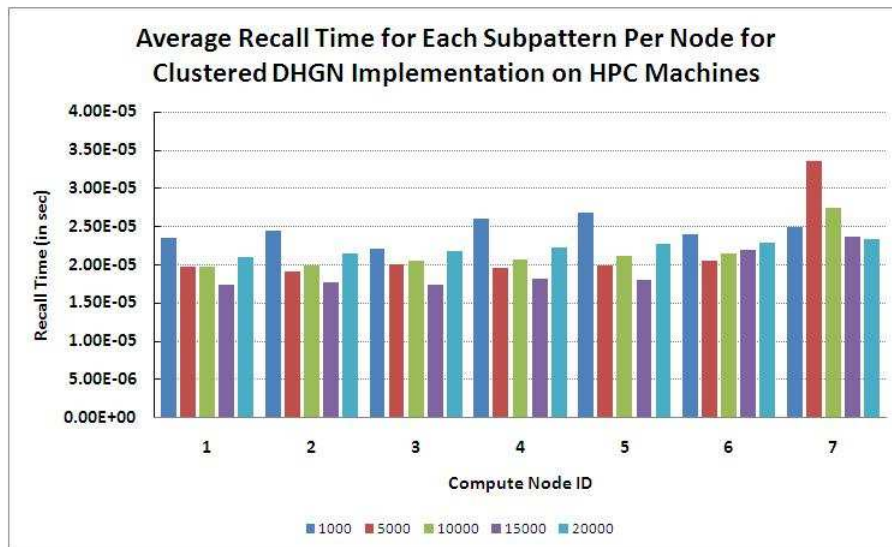


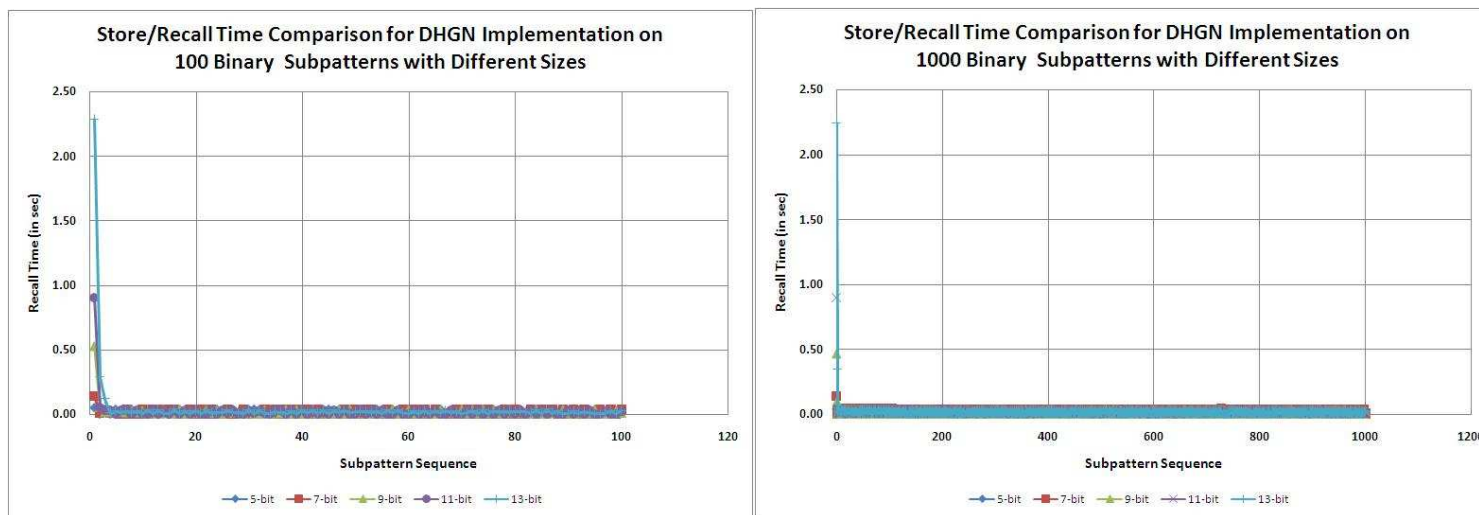
Figure 5.6: Average recall time for each subpattern in clustered DHGN implementation on HPC machines.

The results exhibit fast recall time in comparison with the outcomes of the performance tests that have been described in Section 5.2.1. Results indicate that clustered DHGN configuration does not affect the recognition accuracy of DHGN scheme, as described in Chapter 3. Rather, the configuration improves the performance of recognition process when it is being conducted in the resource-abundant coarse-grained networks.

Fully-Distributed DHGN

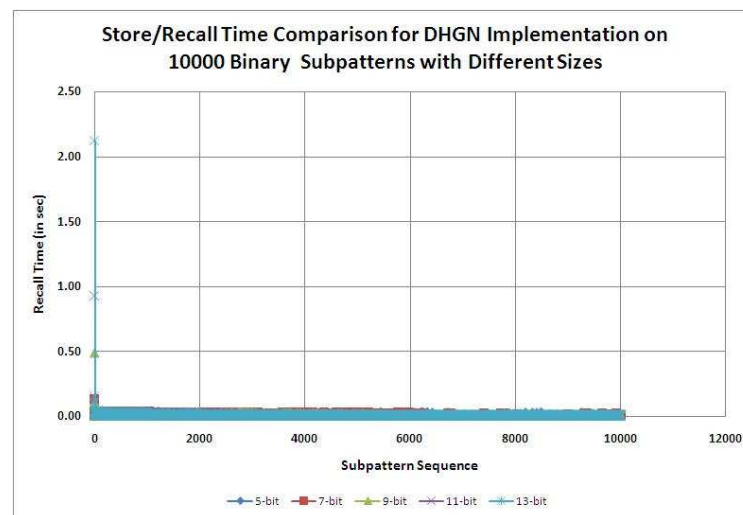
In fully-distributed configuration using message-passing model, a binary pattern recognition was implemented on 5-, 7-, 9-, 11-, and 13-bits subpattern size with different number of subpatterns used. Each node containing GN communicates with its adjacent nodes using message-passing scheme that is handled by MPICH-2 functions such as *mpi_send()* and *mpi_recv()*. This implementation has been executed using Unix GCC C programming platform with MPICH-2 support on a Pentium PC with processing speed of 2.80GHz and 1GB RAM. Each GN in DHGN subnet is assigned to a virtual processing node created by MPI program.

Figure 5.7 shows the recall time for each subpattern with different size and quantities used for a single DHGN subnet.



(a) 100 binary subpatterns.

(b) 1000 binary subpatterns.



(c) 10000 binary subpatterns.

Figure 5.7: Recall time for a single binary subpattern in DHGN fully-distributed configuration implementation over different subpattern sizes and quantities.

Note that for different number of subpatterns used, the recall times remain consistent and at minimum (less than 0.1 seconds), with an exception of the first subpattern introduced. This may be caused by the node's initiation process that leads up to an increase in the recall time for the first subpattern. Based on the results obtained, it can also be deduced that DHGN imposed fast recognition, in which on average, it requires less than 100 milliseconds for each subpattern to be stored or recalled. Furthermore, this recall time does not being significantly affected by an increase in the number or size of the subpatterns.

An interesting outcome of the recognition tests that have been carried out is that the average recall time reduces significantly with an increase in the number of subpatterns stored or used in the recognition tests, as shown in Figure 5.8. This may be caused by the recall/store mechanism in DHGN implementation, in which all the possible entries in bias array has been achieved and hence, no update is required for each store/recall process. Figure 5.9 shows the effect of increasing subpattern length on average recall time for different number of subpatterns used.

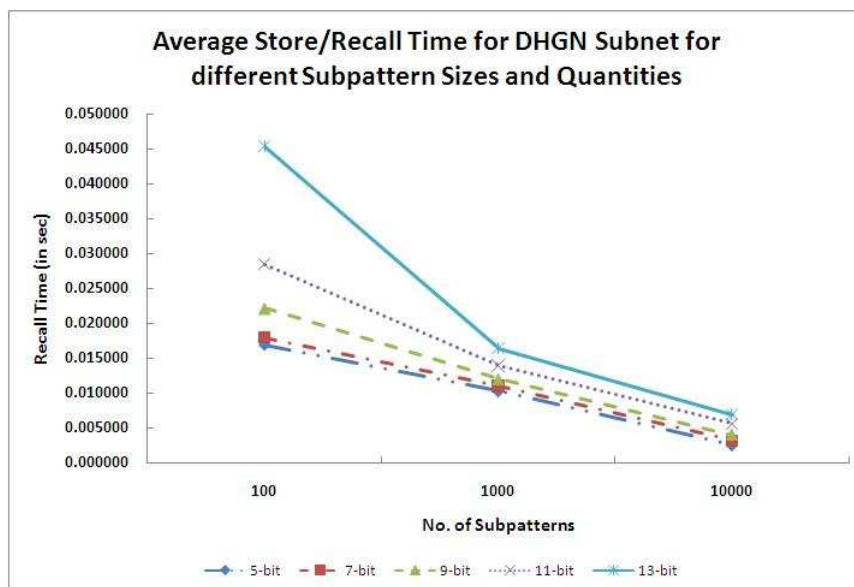


Figure 5.8: Average store/recall time for DHGN implementation on binary subpattern with different length and quantities.

This research has implemented a series of case studies involving DHGN in different network arrangements ranging from coarse-grained to fine-grained networks. These include the works on Structural Health Monitoring (SHM) using WSN and DHGN intrusion detection scheme in mobile ad hoc networks (MANETs). The contents of these case studies have

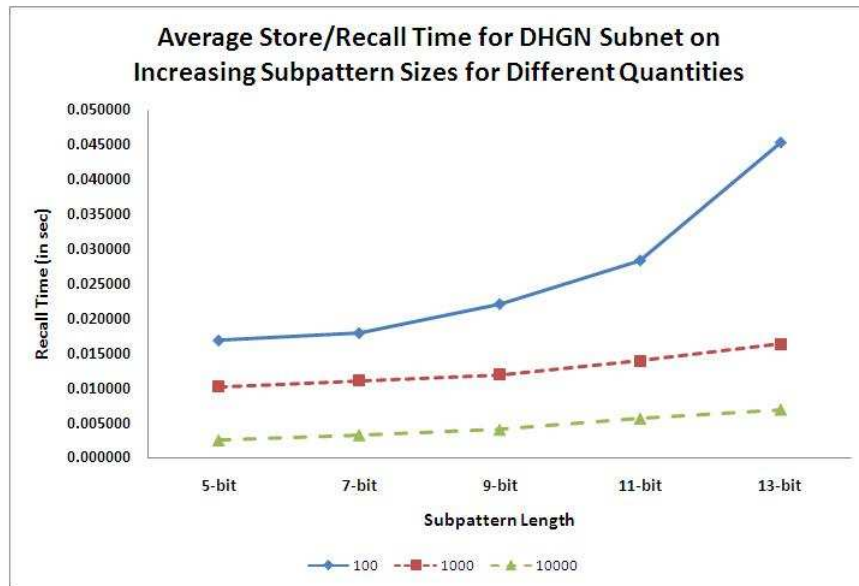


Figure 5.9: Effects on the average store/recall time for DHGN implementation with increasing binary subpattern length.

also been published as conference papers and book chapters, including (Muhamad Amin and Khan, 2009; Khan and Muhamad Amin, 2009; Khan, Muhamad Amin and Raja Mahmood, 2010a).

A discussion on DHGN adaptation on different levels of network granularity has brought forward an initiative to examine in greater details on the different DHGN configurations that have been proposed. In this study, two different implementation perspectives were assessed, namely DHGN implementations on coarse-grained and fine-grained networks. This study intended to demonstrate DHGN's ability to perform under different network distributions, and hence, exhibits DHGN's resource-awareness capability. The following subsections detailed out some of the works that have been carried out in determining the adaptiveness of DHGN against varying network granularity.

5.2.3 Clustered DHGN for Coarse-Grained Networks

The rapid development of high-performance computing (HPC) and grid architecture has enabled a variety of applications to be deployed over a fast distributed network. With increasing processing speed and storage space, HPC architecture such as grid and cloud computing promises a scalable mean towards applications involving large-scale domains. These computing architectures represent a coarse-grained network infrastructure, in which each processing node has extensive processing and storage capabilities.

In this subsection, DHGN's performance in HPC environment is examined using the proposed clustered configuration. An analysis on DHGN recall time for distributed pattern recognition involving binary patterns over grid architecture will be presented. A proposed framework for DHGN implementation in a commodity-grid infrastructure will then be presented and explained. This infrastructure offers significant extension for DHGN implementation for DPR across different application domains. Some of the contents of this subsection have been published as a conference paper in (Muhamad Amin and Khan, 2008a).

DHGN Commodity Grid Framework for DPR

Distributed pattern recognition provides an avenue for achieving large-scale pattern recognition by using a state-of-the-art data classifier for fast tracking large-scale data analyses. A framework has been proposed, that employs a grid-enabled DHGN distributed pattern recognition scheme. The framework comprises commodity-grid network (von Laszewski, Foster and Gawor, 2000) for pattern recognition processing using the DHGN single-cycle learning approach. The commodity-grid can provide an easy-to-use front-end for accessing a distributed system supporting complex operations.

The proposed framework for our distributed pattern recognition is a combination of commodity-grid based architecture with the single-cycle learning DHGN associative memory approach for pattern recognition. Having commodity grid, as the infrastructure, enables us to offer the pattern recognition service to multiple users from different expertise domains and application areas. For instance, the climatic change research may use the proposed system for long term climate pattern discovery while bioinformatics field may use this resource for protein structure recognition and classification. This extends the scalability of DHGN DPR scheme across different application domains.

Distributed Pattern Recognition Architecture. With regards to the distributed pattern recognition framework, the architecture for the pattern recognition application would directly follows the DHGN architecture with clustered configuration described earlier. Figure 5.10 shows the grid network outlay for the proposed framework.

The communication between the DHGN sub-networks and SI module is done using the existing file transfer or resource allocation services such as GridFTP or GRAM. Each DHGN sub-network may be hosted by a single computing node, or group of nodes within a sub-network. Within each of these sub-networks, the communications among the nodes

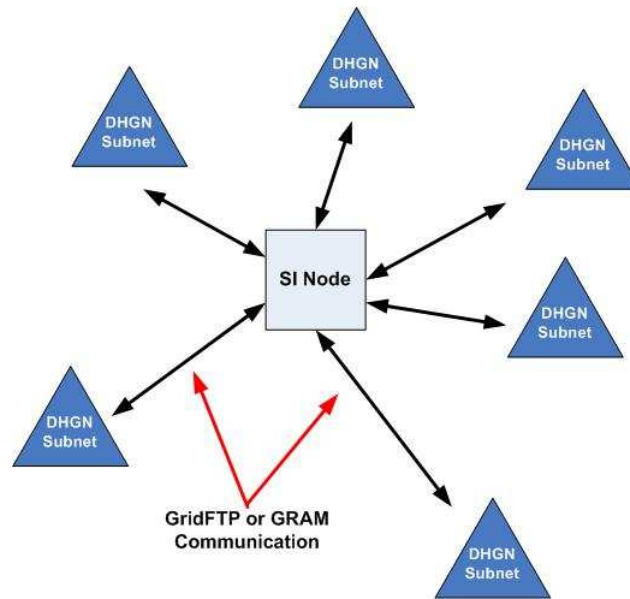


Figure 5.10: Proposed commodity grid-based distributed pattern recognition framework.

are handled by Message Passing Interface (MPI). MPI also facilitates the parallel DHGN computations.

The proposed distributed pattern recognition is a real-time application that is able to produce the results in a single cycle of computations. Furthermore, each of the DHGN sub-networks executes independently thus providing a high level of scalability and efficiency by removing the need for inter-sub-network communications. The SI role and a DHGN sub-network role could be easily interchanged, where any node within the grid could take over the SI role for the framework.

Distributed Pattern Recognition Workflow. Work flow support is the key to diversifying this application as a generic resource for E-Research. Figure 5.11 illustrates the workflow for the proposed distributed pattern recognition framework.

The proposed framework utilises both the commodity-grid processes, as well as the core pattern recognition service. Also to note is that the front-end of the system is handled by the CoG portal.

The Proposed Framework. Figure 5.12 shows the framework for implementing the distributed pattern recognition system.

The framework is designed to cater for different types of users/applications that need to access a large-scale low latency pattern recognition resource in a flexible manner. The CoG Portal and Engine also offers the authentication and security services for the users.

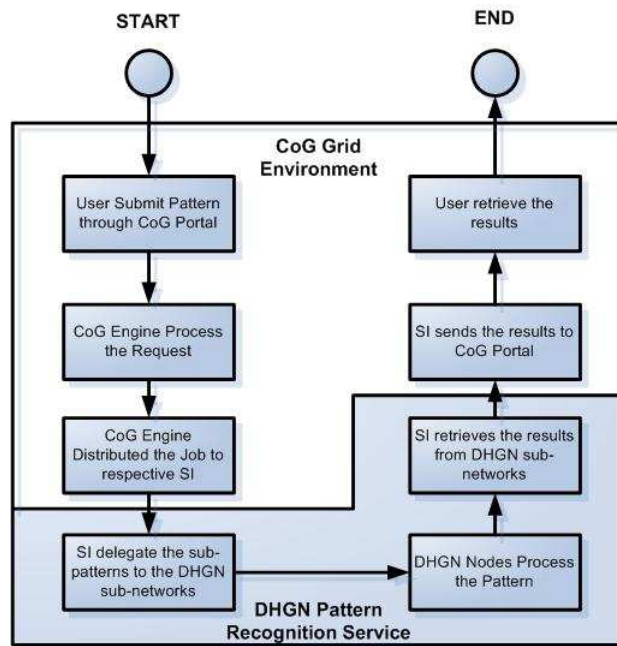


Figure 5.11: DPR-Commodity Grid workflow.

In addition, Java CoG Kit offers a security infrastructure compatible with the Globus Toolkit Grid Security (von Laszewski, Gawor, Peña and Foster, 2002). More information on the security implementation of Java CoG Kit can be found in (Laszewski and Sosonkin 2005).

The framework used in this study implemented the Karajan CoG grid engine (von Laszewski, Hategan and Kodeboyina, 2006). Figure 5.13 shows the Karajan architecture adopted from von Laszewski et al. (2006).

The Karajan architecture offers additional libraries for the front-end design through its HTML and forms libraries. It uses the task library for grid integration, which is based on the Java CoG Kit abstractions.

The CoG grid architecture provides its own method of task creation for handling user defined workflows. For this framework, the initial task would be to execute the distributed pattern recognition program. This could be done in the CoG environment using the following pseudo code:

```

Task prTask = new Task();
JobSpecification prSpec = new JobSpecificationImpl();
prSpec.setExecutable("usr/bin/mpixec");
prSpec.addArguments("-n 30 /home/anangh/dhgn/dhgnMain");
  
```

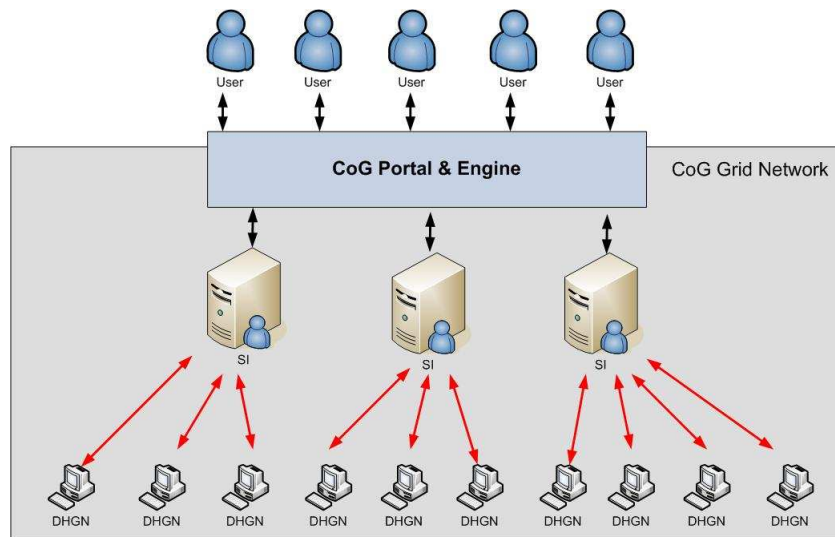


Figure 5.12: Framework for commodity-grid based pattern recognition.

```
prSpec.setStdOutput("results.txt");
prTask.setSpecification(prSpec);
```

CoG Grid Karajan Engine also offers a specification language (usually in XML format) for tasks creation and delegation. The following code shows specification language used in our framework for remote execution of the pattern recognition application.

```
< projectname = "DistributedPRExecution" >< includefile = "cogkit.xml" / >
< task : authenticateprovider = "GSI" / >
< setname = "host" value = "gngrid.infotech.monash.edu.au" / >
< setname = "path" value = "/usr/bin" / >
< task : transferdesthost = host" provider = "gridftp" srcfile = "dhgnMain" / >
< task : Executehost = "host" provider = "gt4" executable = "path/mpiexec"
arguments = " -n30dhgnMain" / >
< /project >
```

It may be noted that the proposed framework also offers user authentication through the Grid Security Infrastructure (GSI).

Summary. In this subsection, a framework for distributed pattern recognition resource utilisation is presented. The framework is a combination of commodity-grid environment and single-cycle learning approach. The framework can effectively harnesses the advantages of distributed systems within a grid to form a generic online data classification resource for a range of E-Research applications.

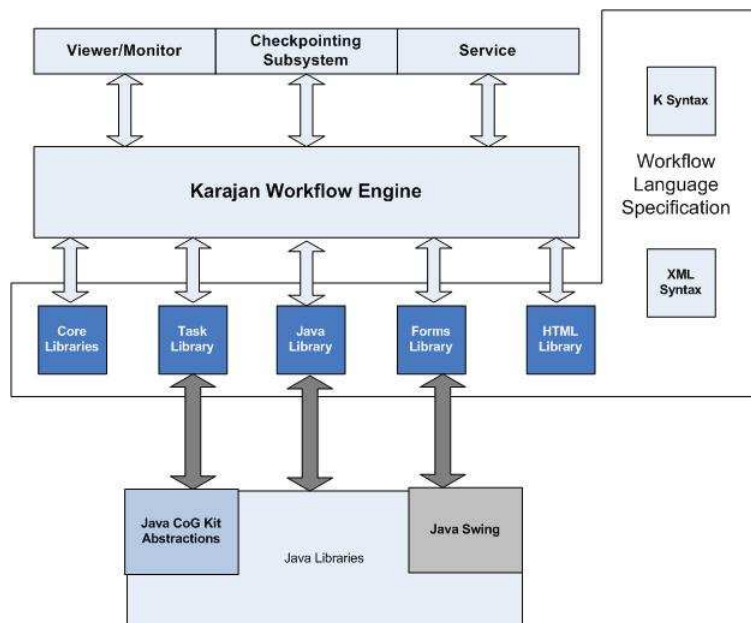


Figure 5.13: The Karajan grid engine architecture.

5.2.4 Fully-Distributed DHGN for Fine-Grained Networks

The intensive development of wireless technologies and increasing miniaturisation of RF devices and micro electro-mechanical systems (MEMS) have been a driving force in the advancement of small and tiny computing devices such as in the WSN technology (Hafez, Haroun and Lambadaris, 2005). These devices are inter-connected, forming a computational network that capable of providing a frontline processing schemes for specific applications such as event detection. This kind of networks could be referred as fine-grained networks, due to their characteristics of having large number of computing nodes with limited power, storage, and processing capabilities.

DHGN implementation as extensively described in Chapter 3, has been configured in a fully-distributed manner in which each GN is assigned to a single compute node and the collaborations of inter-connected compute nodes will then formed a DHGN subnet. Having simple bias array search computations involved for each node, this configuration is well-suited for fine-grained networks such as WSN in which simple computations are possible with limited processing and storage capacities. This research demonstrates the robustness and scalability of DHGN for distributed recognition process over such fine-grained networks. In achieving this, the research investigates a distributed pattern recognition scheme for event detection within a WSN environment. Some of the contents of this subsection

has been published as conference papers in (Muhamad Amin and Khan, 2008a; Amin and Khan, 2008; Muhamad Amin and Khan, 2009).

A Distributed Event Detection Scheme for Wireless Sensor Networks

Highly-complex computations, iterative learning, and large training set requirements are some of the weaknesses of the commonly employed event detection schemes in Wireless Sensor Network (WSN). These schemes mainly apply conventional neural networks or machine learning algorithms that require extensive retraining as well as a huge number of training datasets for effective generalisation. Furthermore, centralised processing or single-processing approach in existing schemes create some major problems including high communication overheads due to the constant flow of sensory data, re-routing procedures, and relocation activity of sensor nodes that often occurs in real-time applications and significantly long delays in detecting critical events with the presence of computational bottleneck. These problems limit the schemes scalability for massive sensory data processing.

Artificial neural networks (ANNs) and other machine learning techniques are the most commonly applied classification techniques for event detection schemes in WSN. Catterall et al. (2003) have proposed an implementation of Kohonen Self-Organising Map (SOM) in sensory data clustering for distributed event classification within WSN. Radial-Basis Function (RBF) neural network has been proposed for dynamic energy management within WSN network for particle filter prediction by Wang, Ma, Wang and Bi (2007). RBF has been proven to offer fast learning scheme for neural networks. However, its learning complexity and accuracy are highly affected by the training and network generation method being used, e.g. K-means clustering or evolutionary algorithms. Kulakov and Davcev (2005a) proposed the implementation of Adaptive Resonance Theory (ART) neural network for event classification and tracking. The scheme reduces the communication overhead by allowing only cluster labels information is sent to the base station. However, the implementation incurs excessive learning cycles to obtain optimum cluster/class matches.

The proposed event detection scheme within WSN implements a fully-distributed DHGN configuration in which a collection of sensor nodes collaborate and formed a DHGN subnet and perform an event detection based upon the sensory readings obtained from their environment as shown in Figure 5.14.

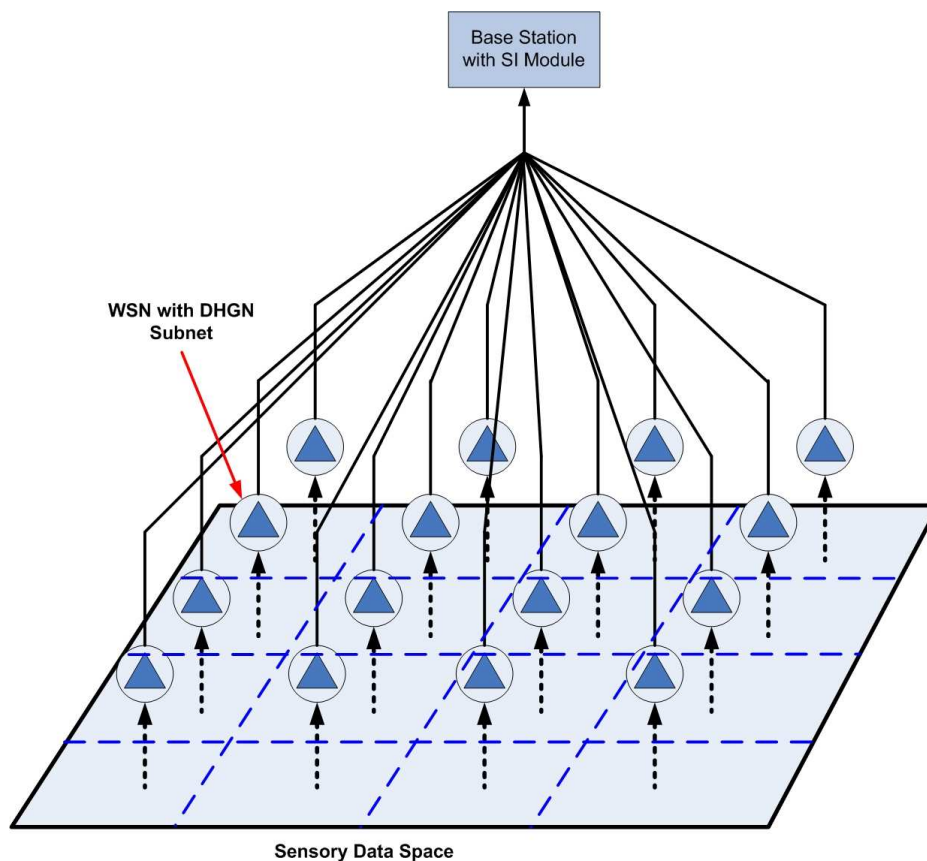


Figure 5.14: DHGN distributed event detection framework.

Note that SI Module is intended to be deployed in a controlling node, such as the base station. On the other hand, DHGN subnet module is located within each WSN subnet that is located within a specified sensory region.

Event Detection using Classification Approach. The classification process within DHGN event detection scheme is a dual-layer process. The first layer focuses on the subpattern recognition at DHGN subnet, while the second layer involves pattern classification using a voting scheme that is conducted by the SI Module. Subpattern recognition is a process of determining the recall/store status of an input subpattern. This process is conducted within DHGN subnets. The output of this process is either a recalled index of the stored subpattern or a new index for the respective input subpattern. It will then be sent out to the SI module for pattern classification. Figure 5.15 shows the proposed workflow for the distributed event detection scheme within WSN.

Classification process within DHGN implements voting mechanism as described by Muhamad Amin and Khan (2008b). It is performed by the SI Module. Each subpattern index received from each DHGN subnet is analysed and the result of this process are

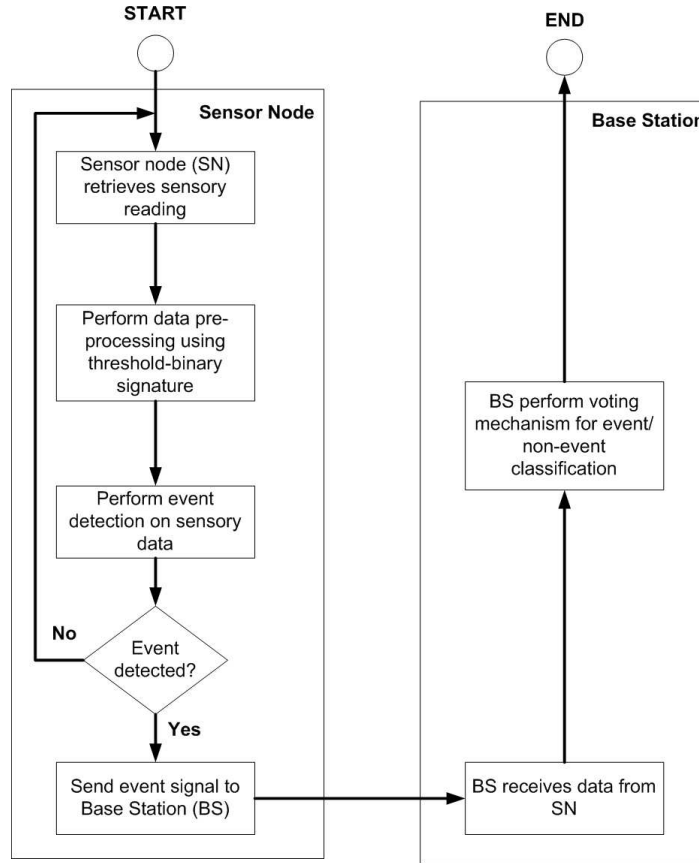


Figure 5.15: Process workflow for the proposed event detection scheme using fully-distributed DHGN algorithm

recorded in the form of class labels. For supervised classification, the number of class label is fixed, while in unsupervised classification, it can be incremented.

The proposed scheme only requires binary input patterns, and consider multiple sensory readings that are used to detect the occurrence of critical events. Given a set of x sensory readings $\{s_1, s_2, \dots, s_x\}$ where $s_i \in \{\mathbb{R}\}$ and $i = 1, 2, \dots, x$, we perform a dimensionality-reduction technique known as threshold-signature that converts each reading value to its respective binary signature. The threshold-signature technique utilises the threshold classes to represent a single data range into a binary format. Given a sensory reading s_i where $i = 1, 2, \dots, x$ and with K -threshold class, the equivalent binary signature that implies $b_i \rightarrow s_i$ is in the form of $b_i \in \{0, 1\}^K$. Therefore, for x -set sensory readings $\{s_1, s_2, \dots, s_x\}$ will be converted into a set of binary signatures $\{b_1, b_2, \dots, b_x\}$.

If the output index from DHGN subnet matches the stored pattern for the critical event, then a signal is transmitted to the base station in the form of data packet $(node_id, timestamp, class_id)$. The $class_id$ parameter is the identification for class of event that

	Threshold Range	Class
Noise Level	≥ 25	Event
Light Exposure Level	≥ 100	
Noise Level	< 25	Non-event
Light Exposure Level	< 100	

Table 5.4: Threshold classes with respective value range used in the tests.

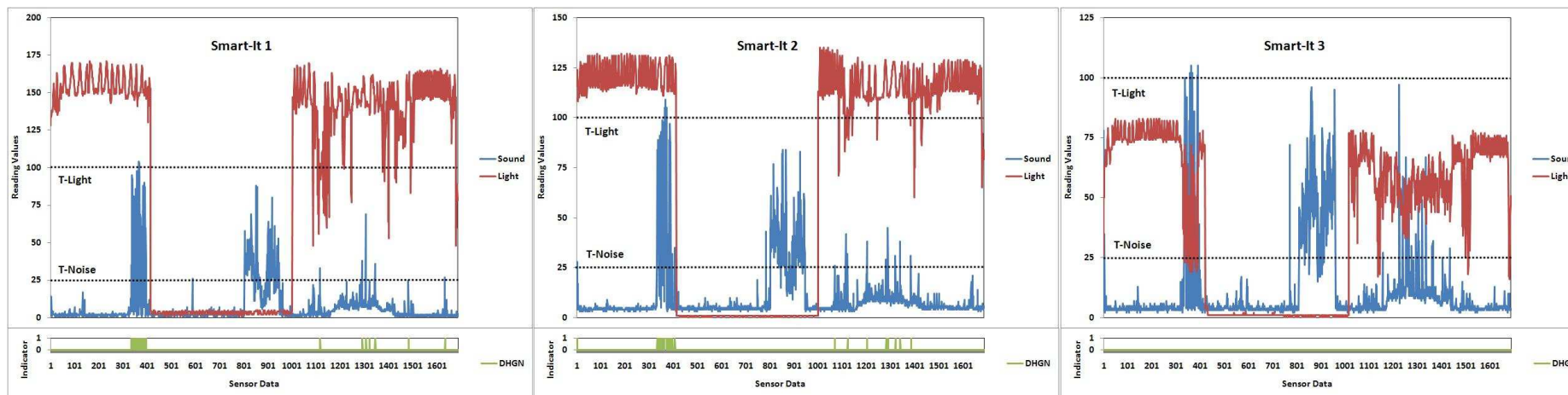
has been detected. At a given time t , the base station might receive a number of signals from the network.

Analysis and Results. Analysis of DHGN distributed event recognition scheme has been carried out using a simulation approach. The sensory data taken from the research by Catterall et al. (2003) have been used to investigate the performance of our proposed classifier. The data consists of five Smart-It wireless sensor nodes readings that detect various environmental conditions such as light, sound intensity, temperature, and pressure. WA test was performed to detect the occurrences of light and sound simultaneously. The simulation involves assigning a DHGN subnet to each Smart-It sensor data. Recognition tests were performed over 1690 datasets. For comparison, similar tests using support vector machine (SVM) and self-organizing map (SOM) have also been conducted. This analysis extends the research by (Muhamad Amin and Khan, 2009) through a comparison between this approach and SOM. This research has used SVMLight (Joachims, 1999) implementation with both linear-type and 2-degree polynomial kernel and SOM Toolbox (Vesanto, Himberg, Alhoniemi and Parhankangas, 2000) with default configuration.

DHGN retrieves sensory readings in the form of binary representation using the discussed threshold-signature technique. Table 5.4 shows the adopted threshold classes.

Figure 5.16 shows the results of the recognition test conducted on sensor datasets for Smart-It 1, 2, 3, 4, and 5. In general, datasets with high level of both noise and light represent event and vice versa. The DHGN detects these event occurrences only in Smart-It 1, Smart-It 2, and Smart-It 5. Smart-It 3 and 4 produces no output since the datasets do not meet the conditions as shown in Table 5.4.

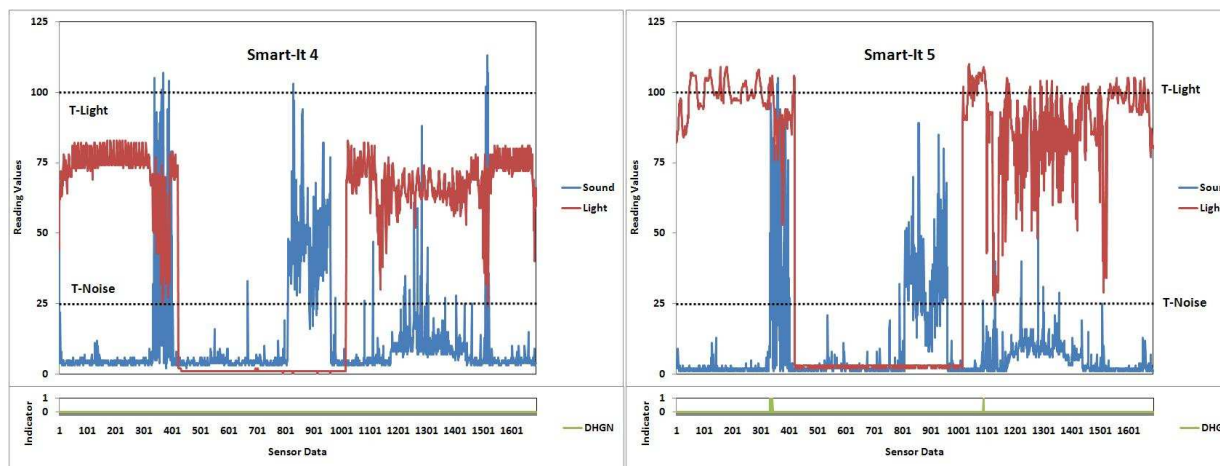
A recognition accuracy analysis was conducted to find the optimum classification scheme between DHGN, SVM and SOM classifiers for event detection. Table 5.5 shows a comparative analysis on different accuracy parameters used for sensory data from Smart-It



(a) Smart-It 1.

(b) Smart-It 2.

(c) Smart-It 3.



(d) Smart-It 4.

(e) Smart-It 5.

Figure 5.16: DHGN event detection results for test using 1690 sensor datasets (x-axis). Note that Smart-It 3 and Smart-It 4 produce non-events since noise and light exposure readings are well below the threshold values (*T-Light* and *T-Noise*).

1, Smart-It 2, and Smart-It 5. The accuracy parameters used include precision, recall, accuracy and error values, as described in Section 4.4.3. In this experiment, SVM and SOM classifiers use a set of six readings in their training set, while DHGN only uses datasets with two entries. For both SVM classifiers, six support vectors have been initialised and implemented for classification purposes. The value of the best result (selected feature) for each parameter is underlined.

SmartIt	Classifier	Parameters			
		Precision	Recall	Accuracy	Error
1	DHGN	<u>1.0000</u>	<u>1.0000</u>	<u>1.0000</u>	<u>0.0000</u>
	Linear SVM	0.9388	0.7419	0.9888	0.0112
	Poly-2 SVM	<u>1.0000</u>	0.7538	0.9905	0.0095
	SOM	<u>1.0000</u>	0.8367	0.9953	0.0047
2	DHGN	<u>1.0000</u>	<u>1.0000</u>	<u>1.0000</u>	<u>0.0000</u>
	Linear SVM	0.9074	0.6364	0.9805	0.0195
	Poly-2 SVM	<u>1.0000</u>	0.6279	0.9811	0.0189
	SOM	<u>1.0000</u>	0.5741	0.9864	0.0136
5	DHGN	<u>1.0000</u>	<u>1.0000</u>	<u>1.0000</u>	<u>0.0000</u>
	Linear SVM	0.1923	0.6250	0.9858	0.0142
	Poly-2 SVM	0.5000	0.3077	0.9923	0.0077
	SOM	0.8000	0.5000	0.9970	0.0030

Table 5.5: Comparative analysis on recognition accuracy parameters between DHGN and other classifiers for event recognition using sensory data obtained from three wireless sensors (Smart-It 1, Smart-It 2, and Smart-It 5).

From the results obtained, DHGN has been shown to exert high recognition accuracy with zero error value and perfect recall. Both SVM and SOM classifiers have also demonstrated high precision and accuracy. However, their recall value is comparatively low against DHGN. This is due to the low true positive values obtained during the classification process. Furthermore, both algorithms have also recording small error values.

In general, DHGN's overall recognition accuracy shows better results than the other two algorithms. In regards to the precision value, SOM and polynomial SVM however exhibit similar performances (Smart-It 1, and Smart-It 2) except for Smart-It 5 in which SOM outperforms polynomial SVM. The polynomial SVM however produces slightly better results than the linear approach (for Smart-It 1, Smart-It 2, and Smart-It 5). This shows that SVM approach depends heavily on the types of kernel being implemented (either linear or polynomial) and the nature of data used. This data dependency problem limits the flexibility of SVM for event detection within WSN. Although SOM produces

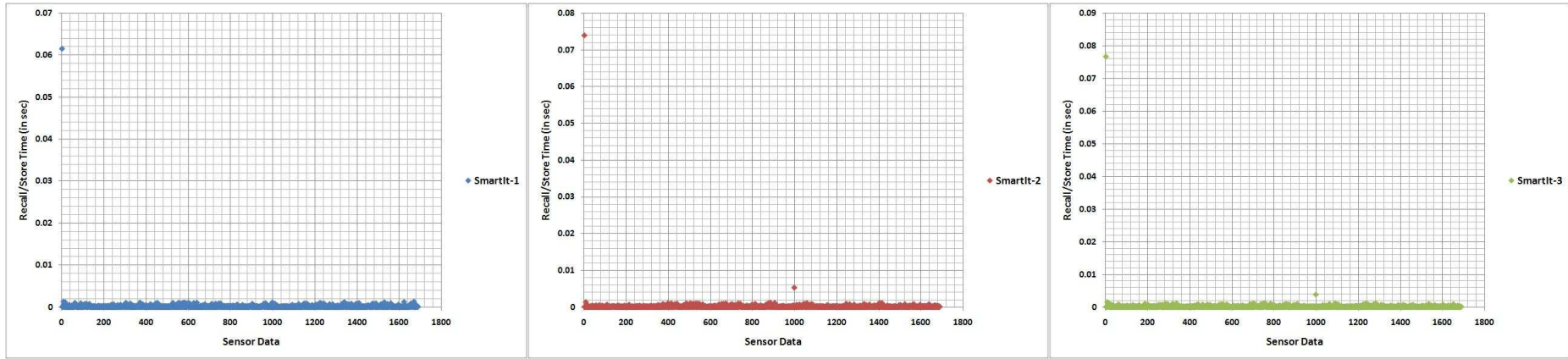
better overall results than SVM, this iterative algorithm is known to be resource intensive and hence impractical to be deployed in the resource-scarce wireless sensor networks. DHGN on the other hand, is a lightweight, single-cycle learning algorithm that is able to provide high detection accuracy when used together with our simple threshold-signature technique.

This research also included a performance analysis to measure the recognition time incurred for each sensor data collected by each sensor node. The simulation was conducted using a fully-distributed configuration with message-passing implementation using MPICH-2 package under GNU C program, and was executed on VPAC HPC machines. Figure 5.17 shows the results of the performance analysis of DHGN distributed event detection on five different wireless sensor nodes (Smart-It 1 to Smart-It 5).

The scheme only takes on average, less than 0.002 seconds for a sensor data to be recalled or memorised. This fast recognition time has been achieved through a simple recognition procedure using a collaborative-comparison learning (CCL) mechanism as described in Section 3.1.5. Furthermore, the average recall time for each sensor data remains consistent for entire sensor data collection. This indicates significant consistency of our approach in regards to the recognition time per pattern, for a recognition application involving large amount of data. These results have also exhibit a potential for real-time development of pattern recognition scheme within a resource-constrained network such as WSN.

5.2.5 Summary

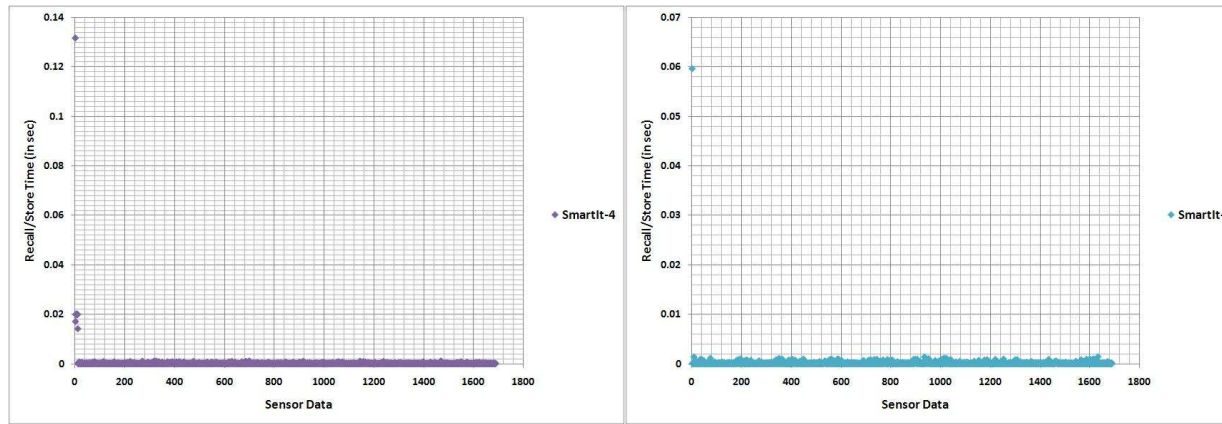
In this section, the capability of DHGN DPR scheme to provide a scalable and robust approach for pattern recognition has been presented. DHGN harnesses the distributed computing potentials by considering available resources within a network through its adaptive network granularity. This adaptiveness has been acquired using different system configurations in DHGN implementation. Through both clustered and fully-distributed DHGN configurations, pattern recognition was able to be conducted, regardless of the network limitations in terms of node's processing and storage capabilities. Moreover, the results of this adaptivity study have shown that DHGN performs fast recognition and able to recall or store up to 20000 patterns within on average less than a second. This effect further



(a) Smart-It 1.

(b) Smart-It 2.

(c) Smart-It 3.



(d) Smart-It 4.

(e) Smart-It 5.

Figure 5.17: Recognition time for each sensor data in 1690 sensor datasets (x-axis) for all Smart-It nodes using DHGN distributed pattern recognition scheme.

proves that DHGN single-cycle learning approach is capable of providing fast and accurate recognition.

It is important to note that DHGN configurations that have been described do not have to be exclusively applied in both coarse- and fine-grained networks. In some contexts, clustered DHGN may be more appropriately applicable in fine-grained networks and vice-versa. For instance, in the area of multi-sensory WSN networks. Further works need to be carried out in the area of adaptive network granularity, in order to enable DHGN to be able to be applied in different kinds of application domains and systems configurations.

The next section will further analyse the performance and robustness of DHGN DPR scheme, with a focus on fault tolerance mechanism for distributed system deployment.

5.3 DHGN Fault Tolerance Mechanism

DHGN distributed pattern recognition scheme adopts resource-awareness in its implementation. For resource-constrained environment such as wireless sensor networks (WSNs), the use of computationally expensive processes may lead to resource exhaustion, in terms of power and storage. Hence, it is imperative that pattern recognition applications such as DHGN are able to conserve the node's energy.

Fault tolerance is an important aspect in distributed systems. It focuses on the flexibility of a system to handle an event of failure. Fault tolerance refers to the capability of a system to perform adequately in the event of failure in some of its components (Schalkoff, 1997).

Fault tolerance is considered an important aspect in the design and implementation of DHGN as a distributed pattern recognition algorithm. Initial design and implementation of DHGN assumes that the capability of computational network to provide sufficient processing capacity, such as in the grid environment. Nevertheless, with regards to different kinds of networks and applications, additional consideration should also be put in place for DHGN implementation in resource-constrained and fragile environment such as mobile ad hoc network (MANET) or wireless sensor network (WSN).

Two levels of fault in DHGN execution for pattern recognition have been identified. These are subnet and node fault. Error resulting in the entire subnet malfunction is

known as subnet fault, while error due to loss of node function and capability is known as node-fault.

5.3.1 Subnet Fault

In the network environment, occurrences of errors and malfunctions are unavoidable, but can be prevented, diagnosed and treated. The use of network fault management paradigm helps in sustaining the availability of particular network. Some of the possible failures that may happen include communication failures, network overloading, and security attacks.

In the case of DHGN implementation on distributed network environment, network fault may affect the performance and accuracy of the recogniser, due to its in-network processing feature. For instance, consider a DHGN network with n subnets, and each subnet is capable of handling subpatterns of length p . A failure on a single subnet will affect the result of the recognition, due to insufficient processing capacity of the entire network to function for pattern length of $(n - 1)p$. An increase in the number of faulty subnets may leads to significant deterioration of the DHGN performance.

The performance degradation of DHGN implementation implies that the recogniser has limited capacity of recognising the entire pattern for a specified period of time. In this case, an increase in the number of faulty subnets would linearly degrade performance of the system. Hence, there is a need for a mechanism to be applied in DHGN when executing under fragile or best-effort network environment.

Fault management scheme provides a capability to detect, isolate, and resolve the failures. This research applies a holistic approach in managing fault that occurs at subnet level.

Given a scenario where one or more subnets failed to commit to the recognition process, the coordinator node (or SI module in DHGN architecture) will be alerted when the faulty subnet(s) is/are unable to deliver responses to recognition request. In the case where the faults occurred during execution, SI module is able to detect these whenever top node in the respective faulty subnet failed to deliver recall/stored index of subpattern that has been sent to it.

Once the SI module has been made aware of possible fault, it may have to initiate a response by relegating recognition tasks to available subnets. This approach may require rearrangement of subnets and /or processing nodes, as to cater for recognition process of

remaining patterns. Figure 5.18 shows task re delegation method that could be applied in the case of fault occurrences at subnet level in DHGN execution.

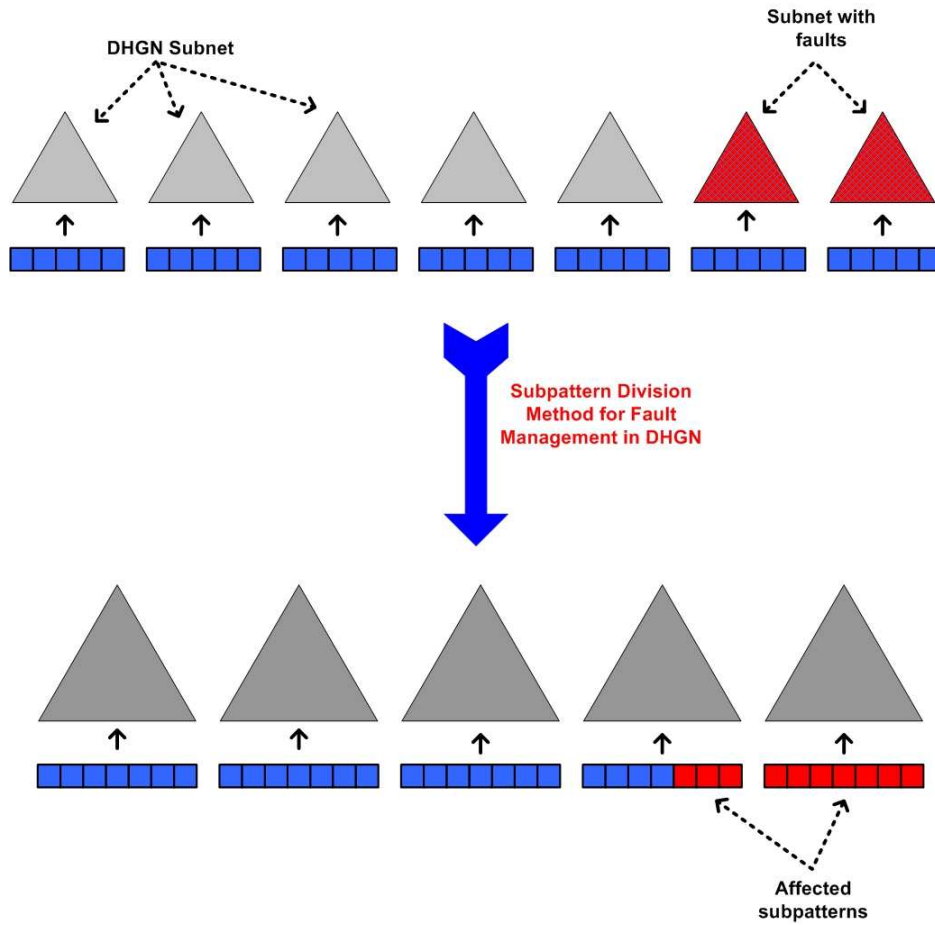


Figure 5.18: Subpattern-division method for task re delegation in DHGN fault management scheme.

In this figure, the two red-coloured subnets have malfunctioned due to faults. As a result, the network is incapable of continuing the recognition process, due to insufficient number of subnets available. To overcome this shortage of subnets, DHGN network must be restructured in the sense that the size of each subnet may be increased, in order to handle subpatterns that have been directed to the faulty subnets.

The subpattern-division method is used with basic assumptions that the non-affected subnets are capable of providing additional resources to be utilised for the remaining recognition tasks and each subnet is initially of equal sizes. In this approach, the subpattern-to-subnet position will be rearranged to accommodate the length of subpattern(s) that has/have been ignored by the malfunctioned subnet(s). Consider a DHGN scheme for binary patterns with subpattern length x and n number of subnets. Given y faulty subnets,

the new subnet arrangement will accommodate z subpattern length using the following equation:

$$z = x + \frac{yx}{n - y} \quad (5.2)$$

Figure 5.19 shows the subpattern length and subnet size increments, following an increase in the number of faulty subnets for DHGN with 50 subnets and binary patterns with 7-bits length.

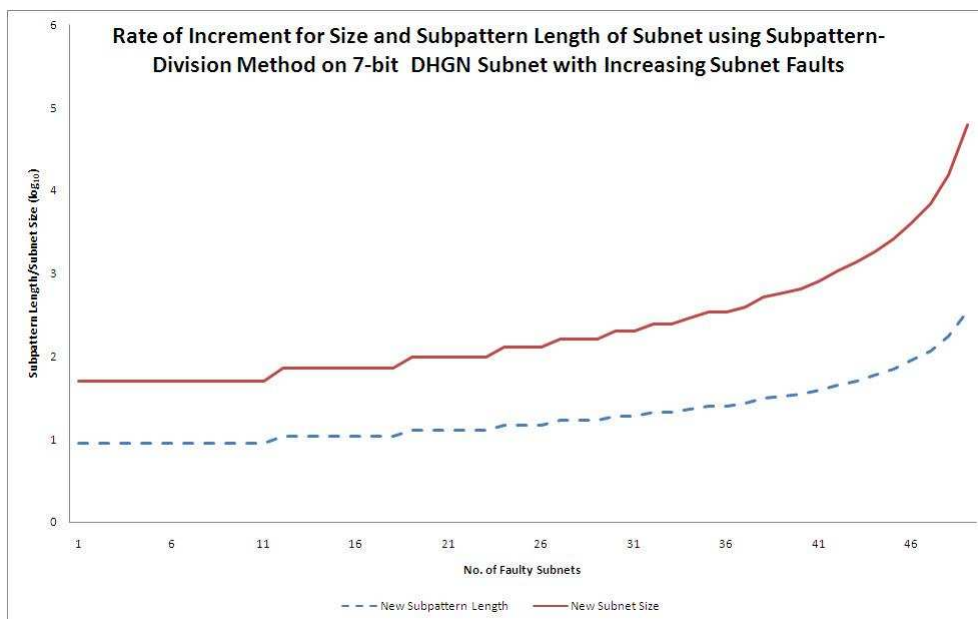


Figure 5.19: Size and subpattern length of subnet using subpattern-division method on DHGN network with increasing subnet faults.

Note that for large number of faulty subnets, this method does not seem to be applicable, since the subnet size overheads tend to be extremely high.

5.3.2 Node Fault

In most best-effort networking environment, failures among processing nodes are common. These are mainly due to communication failures, security attacks, or simply processing node malfunctions. In critical systems, failure of a single node may affect the performance of the entire network. Moreover, if the system is built upon a high-cohesion infrastructure (tightly-coupled design), it will also have a significant influence on the outputs of the system.

Neural network systems for pattern recognition take different forms of fault tolerance. These include tolerance against inexact computations and network mapping. Two different mechanisms for fault tolerance have been introduced by Panduranga, Rao and Deshpande (2007). These include training algorithm modification and explicit augmenting redundancy. In addition, Jiang, Liu and Wang (2006) proposed the use of pseudo-attractor and region of attraction concepts for estimating fault tolerance within a feed-forward neural network system.

In DHGN, we consider both forms of fault tolerance as a single task to be solved. This is due to the fact that in its implementation, inexact network mapping may lead to error in the recognition procedure, since DHGN applies (*value, position*) matching approach. A single node failure may result in the entire subnet to produce incorrect recognition output.

Failures at processing node level do not necessarily fail the entire subnet from carrying out the recognition process. Hence, there should be a mechanism for detecting, isolating, and resolving the faulty nodes.

In order to be able to conduct a recognition process, existing DHGN design requires that the number of processing nodes are sufficient enough to handle the length and dimension of the specified subpattern. In the case of faulty nodes occurrences, there is a need for DHGN to work on a constrained condition where the number of processing nodes n_{GN} , is less than the required capacity for a given subpattern with length l and dimension r , as shown in the following equation:

$$n_{GN} < r \left(\frac{l+1}{2} \right)^2 \quad (5.3)$$

In this context, there is a need to consider a fault tolerance mechanism for handling node failures in DHGN scheme. The researcher proposes a compressed model of DHGN implementation using multi-value DHGN (MV-DHGN) as described in Section 3.5, that enables it to perform recognition process with less number of GNs while maintaining similar level of accuracy.

5.4 Conclusions

This chapter presented an important aspect in DHGN implementation, i.e. resources. DHGN as a distributed application requires constant awareness of network resources, in

order to effectively performed recognition procedures. Understanding of different levels of network granularity helps in determining a suitable configuration for DHGN distributed pattern recognition deployment. Consequently, this improves the robustness and scalability of the proposed approach.

The performance of DHGN recognition scheme has been demonstrated, in regards to its execution and recall times. Furthermore, the ability of the proposed scheme to deliver single-cycle learning mechanism has enable it to adapt to large-scale data sets and its ability to perform parallel pattern recognition using a message-passing model has also been presented.

Apart from granularity, fault tolerance is an important aspect to be considered. Network or node failure may lead to disruption of recognition process, hence imposing delays in producing outputs. Several approaches were proposed to deal with this aspect, including subnet redelegation and compressed DHGN model using multi-value DHGN (MV-DHGN).

The next chapter will examine a case study on DHGN implementation in adaptive granular networks, specifically in fine-grained networks for distributed event detection applications within WSN networks.

Chapter 6

Distributed PR Applications within Fine-Grained Networks

The ability to acquire resource-awareness characteristic has been discussed in the previous chapter, and is essential for the design of distributed applications, including pattern recognition. DHGN as a distributed pattern recognition scheme, has been developed with adaptive granularity consideration in its design. This enables it to be deployed in both coarse- and fine-grained networks.

Parallel pattern recognition scheme for applications within fine-grained systems such as WSN has been initially introduced by Khan and Mihailescu (2004). In this research, a distributed recognition framework for detecting stress patterns from a simple finite element model stored within WSN network has been proposed. Further research work has also been carried out within a domain of pattern recognition applications in WSN, including the works on GN-based pattern recognition/event detection (Baquer et al., 2005; Baig et al., 2006). Moreover, a distributed pattern recognition model for event detection has also been proposed by Nasution and Khan in their Hierarchical Graph Neuron (HGN) implementation (Nasution and Khan, 2008).

The motivation for this chapter lies in the aforementioned applications using GN-based algorithms. DHGN, with its features including high scalability and granular-adaptive, can be applied in real applications such as event detection. With the ability to distribute recognition process within the body of network, DHGN enables recognition scheme to be deployed in small and resource-constrained devices such as sensor nodes in WSN network.

This chapter provides a discussion on DHGN implementation on real applications. As an example, a case study on distributed pattern recognition application for a lightweight event detection scheme within a fine-grained network will be presented. To achieve this, a DHGN implementation with one-shot learning recognition capability for event detection is proposed, specifically within WSN network. DHGN, as described in Chapter 3, has the potential to recognise and classify multi-dimensional sensory input for identifying natural or man-made phenomena through clustered and hierarchical graph-based representation of input patterns for use within fully-decentralised networks. DHGN has the ability to divide-and-distribute recognition tasks throughout the network in a fine-grained manner for minimising the energy use. Thus, it is highly-suited for resource-constrained networks such as WSN. In addition, as being discussed in both Chapter 2 and 3, DHGN provides a comparable level of accuracy in comparison with HGN implementation. However, as described in Section 3.3, scalability of DHGN outperforms HGN, in regards to the number of processing GN nodes requirement. DHGN is also capable of integrating vast networks of sensors into intelligent macroscopes for observing our surroundings. These will bring unprecedented capabilities within our reach that transform the way we deal with phenomena occurring over large distances and inaccessible regions.

Apart from applications such as stress pattern detection, this research intends to consider beyond a simple event detection application and move towards a scheme involving critical real-time detection applications such as forest-fire detection. This kind of applications require fast response time and lightweight recognition model for in-network detection involving large-scale real-time data. With single-cycle learning and low-computational complexity characteristics as being described in Chapter 3, DHGN has a potential for such applications. Furthermore, as discussed in Chapter 5, DHGN has an ability to conduct recognition procedure with parallel processing and low-memory utilisation. This is mostly suitable for fine-grained parallel systems such as WSN networks.

The objectives of this chapter are as the followings:

- i. To present a case for distributed pattern recognition implementation on event detection within a fine-grained WSN network.
- ii. To conduct a review on WSN and other related event detection schemes.

- iii. To analyse the performance and accuracy aspects of DHGN for event detection application within a fine-grained network implementation.
- iv. To present and evaluate a study related to distributed forest-fire detection using DHGN recognition scheme within WSN network.
- v. To propose an integrated WSN-Grid infrastructure for complex spatio-temporal event detection using DHGN distributed pattern recognition scheme.

The outline for this chapter is as follows. Section 6.1 provides an overview of WSN technology and explains the current event detection schemes within WSN and some significant issues related to it. Section 6.2 describes the proposed dimensionality reduction technique on sensory data. Details on our proposed DHGN-WSN integration will be further described in section 6.3. Section 6.4 reports on the case study on forest fire detection using DHGN-WSN scheme. Finally, section 6.5 concludes the chapter.

6.1 WSN Event Detection

With regards to different levels of network granularity, existing breakthrough in communication technologies have not only enhanced the performance of existing coarse-grained networks capabilities such as cloud and grid computing. Research has also leads to the rapid growth of emerging fine-grained networks such as wireless sensor networks (WSNs). These networks emerged from the confluence of wireless communication, extensive computational schemes, and sophisticated sensor technology. WSNs in particular, are created from a collection of self-organised wireless and battery-powered devices with sensing capabilities. The future of this kind of networks is promising, as been mentioned by Stankovic (2008), “The potential of these systems is nothing short of revolutionary. This technology will affect all aspects of our lives, bringing about substantial improvements in a broad spectrum of modern technologies ranging from healthcare to military surveillance”. Apart from this promising future, the emergence of these self-organised networks comprising of large number of tiny processing devices have also lead towards an ability for parallel and distributed computing deployment within such fine-grained systems.

Unfortunately, the current scenario in WSN deployment is still far away from its tremendous potential. WSN has only been demonstrated for humble applications such

as meter reading in buildings and basic form of ecological monitoring. Achieving full potential of this technology requires an intelligent computational scheme which at present is still missing.

Common approach implemented within existing WSN applications usually involve a number of processing steps including sensory data capture and conveyance of these data to a central entity known as the base station for further refinement and analysis. Consequently, this approach would lead to a system bottleneck, if it is scaled up for widespread use. Furthermore, processing delay would intermittently occur due to latency between data capture/aggregation and processing time. These limitations make WSN less suitable for real-time monitoring applications. Therefore, we require a new approach for data processing within WSN that acquires the abilities to process their sensory data in situ and with decentralised manner and to generate highly condensed and sophisticated outputs internally. These abilities will alleviate the bottleneck problem within WSN through on-site computations, and improve its performance by reducing the processing delay experienced using existing approach.

Figure 6.1 shows a generic wireless sensor node architecture. Currently, there are a number of commercially available wireless sensor nodes of different types of applications. These include Berkeley Mica Mote (<http://www.xbow.com>) and UCLA iBadge (Park, Locher, Savvides, Srivastava, Chen, Muntz and Yuen, 2002). The specifications of the Berkeley Mica Mote sensor node that is used in a number of surveillance networks (Levis and Culler, 2002; Lewis, 2004) are also listed in Table 6.1.

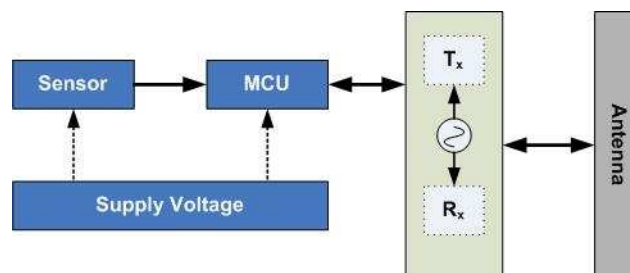


Figure 6.1: A generic wireless sensor node architecture.

On a macro level, WSN is built up from a network of wireless sensor nodes that are linked together through a common entity known as the base station (also commonly known as sink). Due to limited power and processing capabilities, communications between sensor

Dataset	Subpattern Size (bits)
CPU:	8-bit 4 MHz
Memory:	128KB Flash and 4KB RAM
Communication:	916 MHz 40 Kbps Radio
Power:	2 AA Batteries

Table 6.1: Berkeley Mica Mote sensor node specifications.

nodes and base station usually involve a series of data aggregation techniques to reduce the volume of traffic enroute to the base station.

6.1.1 WSN Deployment Issues for Event Detection

Issues with WSN deployment across wide area of applications encircled towards its resource-constrained characteristics, which include limited communication bandwidth, power, processing capability and memory capacity (Culler, Estrin and Srivastava, 2004). In addition, any algorithm that may entail computations, communications, or storage resources within a sensor node would lead to quick exhaustion of the limited battery power available per node. The limited energy and computational resources of sensors imply that the data processing and transmission must be kept to a minimum in order to conserve energy (Khan and Muhamad Amin, 2009).

In solving these issues, systems designers must be able to produce a well-managed design for WSN deployment in which it will provide long-term reliability to the network. An effective design should include principles such as data-centric mechanism, localised algorithms, and lightweight middleware. In this chapter, a new design for WSN deployment is proposed for event detection which incorporates these principles for highly-reliable sensor networks.

One of the primary purposes of the existence of WSN is to provide capabilities for monitoring, detecting, and reporting various significant occurrences of events in the sensory domain. An event can be defined as a behavioural change over time on a certain dynamic phenomenon (Guralnik and Srivastava, 1999). An example is the change in rainfall amount, ranging from light to heavy to extreme. The behavioural change mentioned here could be either a change involving single environmental parameter value or changes involving composite parameters. In explaining this, Li, Lin, Son, Stankovic and Wei (2004) proposed an event hierarchy terminology that differentiates between atomic events and

compound events. Atomic event can be determined based on an observation of a sensor, while compound event cannot be determined from a single observation. Rather, compound event is a collection of observations on different types of sensors. For instance, forest fire is a compound event in which observations could be made on four different parameters including temperature, relative humidity, wind speed, and rainfall.

Research in the area of event detection using WSN is commonly classified into two groups. These are performance-specific research and application-specific research. The performance-specific research concerns the efficiency of the event detection scheme. The main research goal in this area is to develop an event detection scheme with minimum energy consumption and extended lifetime of the WSN network. Alternatively, application-specific research focuses on the development of event detection mechanism that provides accurate and reliable detection for predefined applications such as intrusion detection or phenomenon detection. The common goal of this research area is to obtain efficient mechanism for event detection that deploys specific data processing algorithm that is able to provide accurate and reliable detection using WSN network. This section will further described these two common research areas.

Performance-specific Event Detection Schemes

Most of the recent research works on performance-specific event detection schemes are looking into efficient localisation and routing mechanism that could be deployed within a WSN network. Localisation and routing are the two important factors in determining the optimum coverage and performance of WSN network. Furthermore, these studies have also considered multiple event detection.

A collaborative event detection and tracking in wireless heterogeneous sensor networks has been proposed by Shih, Wang, Chen and Yang (2008). In this research, emphasis has been put into tracking procedure and localization of sensors attribute region for event detection. Event detection scheme known as COLLECT (Collaborative Event Detection and Tracking) has been introduced. A collaboration of different types of sensor nodes is used for event detection and tracking. Three main procedures involved: vicinity triangulation, event determination, and border sensor node selection. The scheme allows event detection and tracking to be conducted simultaneously. However, the scheme requires significant distinction of sensor nodes and their attributes according to its sensing capability.

Furthermore, it also requires extensive collaboration of sensor nodes to derive towards maximum accuracy in the event detection.

Banerjee, Xie and Agrawal (2008) introduces multiple-event detection scheme with fault tolerant within WSN. They propose the use of polynomial-based scheme that addresses the problems of Event Region Detection (PERD). There are two-components involved, including event recognition and event report with boundary detection. For event recognition, they adopt min-max classification scheme which classifies event according to the sensor reading values. These values would then be transformed into polynomial coefficients and passed through a data aggregation scheme. The proposed event detection scheme has enabled a 33% savings in the communication overhead experienced by the network.

Another important contribution in this event detection with performance-specific research is on the work conducted by Ai, Hou, Li and Beyah (2009) in Authentic Delay Bounded Event Detection System (ADBEDS) for WSN. ADBEDS implements iterative event detection scheme using event detection tree. This approach is responsible for simultaneous event detection and packet routing mechanisms. ADBEDS support singular and composite event monitoring. Important aspects within ADBEDS implementation include energy efficiency and authenticity within WSN deployment for event detection. ADBEDS implements user specified bounded delay for event detection. Energy efficiency is achieved through sleep-awake alternation between sensor nodes.

The development of energy-efficient scheme for event detection within WSN has also been carried out by Baqer (2008) using GN pattern recognition scheme with voting capabilities. This work provides a foundation for energy-efficient pattern recognition scheme to be deployed within WSN infrastructure for real-time applications such as structural health monitoring (SHM). This research intends to extend the capabilities of parallel pattern recognition scheme using voting-GN towards a more scalable scheme using DHGN distributed approach.

Application-specific Event Detection Schemes

Application-specific schemes for event detection refer to the area of research involving development of application middleware for WSN. This middleware provides enhanced capability and accuracy for event detection using sensor networks. Several machine learning

algorithms have been applied by a number of research studies, including Fuzzy-ART neural network, multi-layer perceptrons (MLPs), and Self-Organizing Maps (SOMs).

The use of Adaptive Resonance Theory (ART) neural network for event tracking was introduced by Kulakov and Davcev (2005b). Further classification scheme for event detection within WSN has also been introduced in Kulakov and Davcev (2005a). In these research, the use of artificial neural networks (ANNs) in the form of an ART network has been used as pattern classifier for event detection and classification. The scheme offers reduction in communication overhead with only cluster labels being sent to the sink, instead of the overall sensory data. However, the implementation of ART neural network incurs excessive iterative cycle to achieve optimum cluster matches.

The research by Kulakov and Davcev (2005b) on ART neural network for event tracking has also been further researched by Li and Parker (2008) in their study on intruder detection using a WSN with fuzzy-ART neural networks.

Self-organisation for event detection has also been a major focus in application specific research within WSN networks. Catterall et al. (2003) propose a concept of distributed event classification through the use of Kohonen self-organising map (SOM) approach (Kohonen, 2000). The occurrence of events, which are signified by changes in sensor parameter values, could be mapped into clusters representation. The proposed scheme however, imposes significant iterative learning procedure and the classification process is carried out on each input unit, rather than collective input units.

6.1.2 Summary

Existing schemes for event detection using WSN commonly involve centralised processing at the sink or base station. Efforts to minimise the tendency for this singular processing base have been shown by the research on both performance and application-specific research works. However, a complete decentralisation has yet to be achieved. There are several factors related to this issue. These include complex learning algorithms for event detection and tightly-coupled schemes being deployed for event detection. With a reference to Table 6.1, any algorithm that may entail computations, communications, or storage resources within a sensor node would lead to quick exhaustion of the limited battery power available per node. The limited energy and computational resources of sensors

imply that the data processing and transmission must be kept to a minimum in order to conserve energy.

It is evident through the research by Kulakov and Davcev (2005b) and Catterall et al. (2003) that they require extensive learning procedures to derive clusters of events. Consequently, the inputs from the sensors would need to be processed separately and thus incur additional communication overhead for inter-nodes communication. In addition, the proposed schemes do not take into account the variable data processing latency for each sensor nodes in which some inputs might require longer processing time than the others.

The research conducted by Shih et al. (2008) and Banerjee et al. (2008) offer significant contribution in the efficiency of communication schemes for event detection using WSN. However, the tendency for centralised processing is somewhat undeniable. Furthermore, approaches for distinguishing different roles of specific nodes within WSN are still within a scope of further discussion, due to the nature of WSN network which consists of uniformly-equivalent resource-constrained sensor nodes.

This chapter proposes a holistic solution for event detection using WSN. It incorporates a distributed pattern recognition scheme within WSN network and provides on-site and localised computation. This chapter details the implementation of DHGN single-cycle learning distributed pattern recognition algorithm. Within this scheme, a dimensionality reduction approach has been employed for minimising the need for complex computation, as well as the incurrence of communication overhead within the network. The proposed scheme is also capable of providing scalable detection, enabling allowance for the outgrowth of event classes. Furthermore, an integration with computational grid for further complex event analysis is viable throughout this scheme. Finally, the proposed lightweight event detection scheme also equipped with a detailed workflow of the event detection process. Details of the DHGN distributed pattern recognition scheme can be further referred to in Chapter 3.

6.2 Integrated DHGN-WSN Scheme

With distributed and lightweight features of DHGN, an event detection scheme for WSN network is able to be carried out at the sensor node level. It acts as a front-end middleware that could be deployed within each sensor nodes in the network, forming a network of event

detectors. Hence, our proposed scheme minimizes the processing load at the base station and provides near real-time detection capability. Preliminary work on DHGN integration for WSN has been conducted in (Amin and Khan, 2008). Two distinctive configurations for DHGN deployment within WSN have been proposed, as been further described in Chapter 5.

When integrating DHGN within WSN for event detection, mapping each DHGN subnet into each sensor node is considered, using clustered configuration as described in Chapter 5. The proposed scheme is composed of a collection of wireless sensor nodes and a sink. Deployment of WSN in two-dimensional plane with w sensors, represented by a set $W = (w_1, w_2, \dots, w_n)$, where w_i is the i th sensor is examined. The placement for each of these sensors is uniformly located in a grid-like area, $A = (x \times y)$, where x represents the x-axis coordinate of the grid area and y represents the y-axis coordinate of the grid area. Each sensor node will be assigned to a specific grid area as shown in Figure 6.2. The location of each sensor node is represented by the coordinates of its grid area (x_i, y_i) .

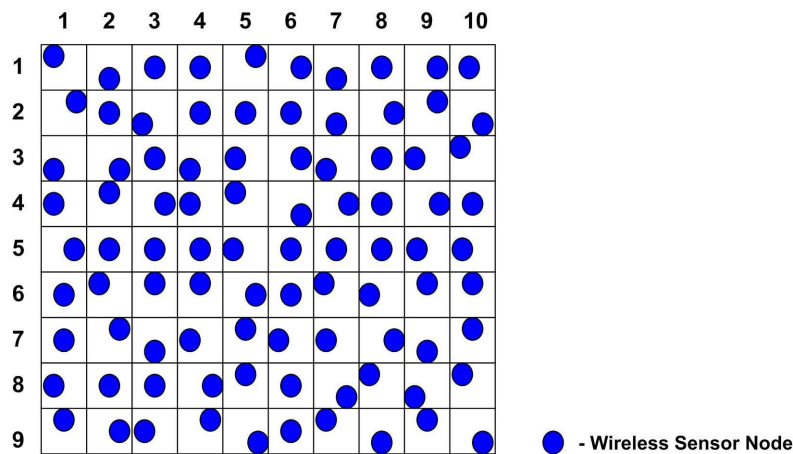


Figure 6.2: Sensor nodes placement within a Cartesian grid. Each node is allocated to a specific grid area.

For the communication model, a single-hop mechanism for data transmission from sensor node to the sink is proposed. The use of “autosend” approach is planned as proposed by Saha and Bajcsy (2003) to minimise error due to the loss of packets during data transmission. The proposed scheme does not involve massive transmission of sensor readings from sensor nodes to the sink, due to the ability of front-end processing approach. It has been concluded from previous research that single-hop mechanism is the most

suitable approach for DHGN deployment. On the other hand, communication between the sink and sensor nodes is done using broadcast method.

6.2.1 Dimensionality Reduction in Sensory Data

Event detection usually involves recognition of significant changes or abnormalities in sensory readings. In heterogeneous sensor networks, specifically, sensory readings could be of different types and values, e.g. temperature, light intensity, and wind speed. In DHGN implementation, these data need to be pre-processed and transformed into acceptable format, while maintaining the values of the readings.

In achieving a standardised format for pattern input from various sensory readings, the use of adaptive threshold binary signature scheme for dimensionality reduction and standardisation technique is proposed for multiple sensory data. This scheme has originally been developed by Nascimento and Chitkara (2002) in their studies on content-based image retrieval (CBIR). Further discussion on this binary signature scheme has been presented in this thesis and can be referred to in Section 3.2.2. Binary signature is a compact representation form that capable of representing different types of data with different values using binary format.

Given a set of n sensory readings $S = (s_1, s_2, \dots, s_n)$, each reading s_i would have its own set of k threshold values $P_{s_i} = (p_1, p_2, \dots, p_k)$, representing different levels of acceptance. These values could also be in the form of acceptable range for the input. The following procedures show how the adaptive threshold binary signature scheme is being conducted:

- i. Each sensor reading s_i , is discretised into j binary bins ($B^i = b_1^i b_2^i \dots b_j^i$) of equal or varying capacities. The number of bins used for each data is equivalent to the number of threshold values P_{s_i} . This bin is used to signify the presence of data which is equivalent to the threshold value or within a range of the specified p_i values using binary representation.
- ii. Each bin would correspond to each of the threshold values. Consider a simple data as shown in Table 6.2. If the temperature reading is between the range 20-25 degrees Celsius, the third bin would be activated. Thus, a signature for this reading is 01000.

- iii. The final format of the binary signature for all sensor readings would be a list of binary values that correspond to specific data, in the form of $S_{bin} = b_1^1 b_2^1 b_1^2 b_2^2 \dots b_j^n$, where b_j^k represent the binary bin for k th sensor reading and j th threshold value.

Temperature Threshold Range (°C)	Binary Signature
0 - 20	10000
21-40	01000
41-60	00100
61-80	00010
81-100	00001

Table 6.2: Example of a simple temperature readings with respective binary signature.

6.2.2 DHGN Event Classification

DHGN distributed event detection scheme involves a bottom-up classification technique, in which the classification of events is determined from the sensory readings obtained through WSN. As been discussed before, our approach implements adaptive threshold binary signature scheme for pattern pre-processing. These patterns would then be distributed to all available DHGN subnets for recognition and classification purposes.

The recognition process involves finding dissimilarities of the input patterns from the previously stored patterns. Any dissimilar patterns will create a response for further analysis, while similar patterns will be recalled. This research will conduct supervised single-cycle learning approach within DHGN that employs recognition based upon the stored patterns. The stored patterns in our proposed scheme include the set of ordinary events that could be translated into normal surrounding/environmental conditions. These patterns are derived from the results of the analysis conducted at the base station, based upon the continuous feedback from the sensor nodes. Figure 6.3 shows our proposed workflow for event detection.

The proposed event detection scheme incorporates two-level recognition: front-end recognition and back-end recognition. Front-end recognition involves the process of determining whether the sensor readings obtained by the sensor nodes are either could be classified as extraordinary event or simply a normal surrounding condition, through the use of DHGN pattern matching mechanism. Conversely, the spatial occurrence detection is conducted through the back-end recognition. In this approach, the use of signals sent

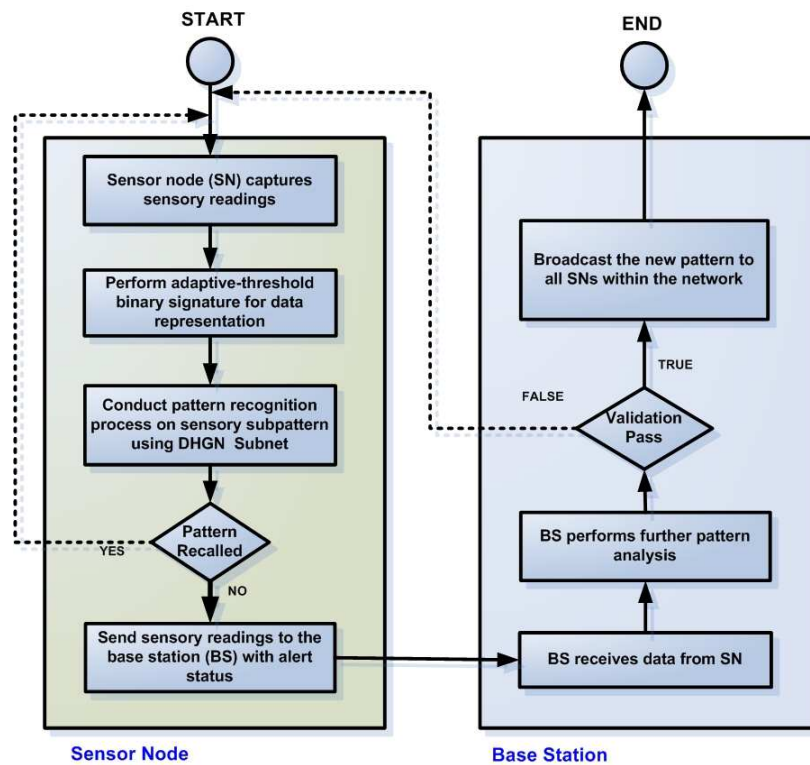


Figure 6.3: A process workflow for DHGN distributed event detection within WSN.

by sensor nodes is considered as pattern for detecting event occurrences at specific area or location. This chapter will explain more details of our front-end recognition scheme.

Pattern Matching at Sensor Level

In event detection application using DHGN algorithm, the occurrences of abnormal events are detected using a pattern matching approach. Sensory readings are considered as patterns, and any significant changes in the structure of normal patterns are classified as events or critical events that need to be reported back to the base station. The use of clustered DHGN configuration maps each sensor node with DHGN subnet that able to accept a number of different sensory readings as a single subpattern. The following algorithm describes our proposed pattern matching approach for event detection at sensor level.

In this algorithm, the output of the pattern matching process is in the form of signal, alerting SI Module in base station, of a new event that has been detected. Base station will respond by performing spatio-temporal analysis on the readings obtained. This analysis will be further described in Section 6.3.5.

Algorithm 7 Pattern Matching Function at Sensor Level

```

1: given  $n$  sensory readings for  $t$  time:  $S_t = (s_1, s_2, \dots, s_n)$ 
2: convert  $S_t$  to binary signature  $B_t$ . Therefore  $f(\text{binsig}) : S_t \mapsto B_t$ 
3:  $trigger = FALSE$ 
4:  $eventAlert.Sensor = FALSE$ 
5: repeat
6:   for  $i = 0$  to  $MAXREADINGS$  do
7:     {check for matched subpattern (sensory readings) within sensor data storage}
8:     if  $new.B_t == s[i].Sensor$  then
9:       { $new.B_t$ : new readings, matching process is conducted using DHGN algorithm}
10:      exit FOR
11:     else
12:        $s[MAXREADINGS + 1].Sensor = new.B_t$ 
13:        $trigger = TRUE$ 
14:        $eventAlert.Sensor = TRUE$ 
15:     end if
16:   end for
17: until  $trigger = TRUE$ 
18: send  $eventAlert.Sensor$  and  $s[MAXREADINGS + 1].Sensor$  to SI Module function
   at base station
19:  $MAXREADINGS = MAXREADINGS + 1$ 

```

6.2.3 Performance Metrics: Memory Utilisation

Memory utilisation estimation for DHGN algorithm involves the analysis of bias array capacity for all the GNs within the distributed architecture, as well as the storage capacity of the SI Module node. A detailed analysis of bias array capacity has been presented in Chapter 3. Based upon the analysis one could derive the fact that DHGN offers efficient memory utilisation due to its efficient storage/recall mechanism. Furthermore, it only uses memory to store newly-discovered patterns, rather than storing all pattern inputs. Figure 6.4 shows the comparison between the estimated memory capacities for DHGN processing cluster with increasing subpattern size against the maximum memory size for a typical physical sensor node (referring to Table 6.1).

The memory capacity requirements for DHGN as shown in the Figure 6.4, have been derived from the equations 3.12 - 3.19 in Chapter 3. As the size of subpattern increases, the requirement for memory space is considerably increases. It is noted that small subpattern sizes only consume less than 1% of the total memory space available. Therefore, DHGN implementation is best to be deployed with small subpattern size.

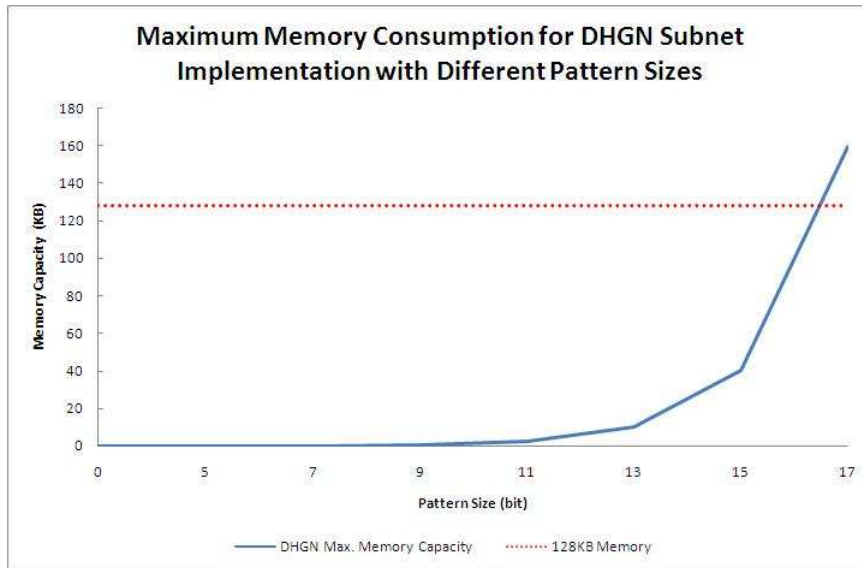


Figure 6.4: Maximum memory consumption for each DHGN subnet for different pattern sizes. DHGN uses minimum memory space with small pattern size.

6.3 Case Study: Forest Fire Detection using Integrated DHGN-WSN

In recent years, forest fire has become a phenomenon that affects both human and the environment. The damages incurred by this event cost millions of dollars in recovery. Current preventive measures seem to be limited, in terms of its capability and thus require active detection mechanism to provide early warnings for the occurrence of forest fire. In this chapter, a preliminary study on the adoption of DHGN distributed pattern recognition scheme for forest fire detection using WSN is presented.

6.3.1 Existing Approaches

There are a number of distinctive approaches that have been used in forest fire detection. These include the use of lookout towers using special devices such as Osborne fire finder (Fleming and Robertson, n.d.), video surveillance systems such as in the works of Arrue, Ollero and de Dios (2000).

There are also a few research studies on forest fire detection using WSN, including the works of Pripuzic, Belani and Vukovic (2010). The main interest is in the implementation of forest fire detection using Fire Weather Index (FWI) and Fine Fuel Moisture Code (FFMC) for standard detection measurement. FWI and FFMC have been introduced by

Canadian Forest Service (CFS) and (de Groot, Wardati and Wang, 2005). FWI is used to describe the spread and intensity of fires, while FFMC is used as a primary indicator for a potential forest fire. At this stage, our interest is mainly focuses on early detection for potential forest fire. Hence, this research is concentrating on the use of FFMC values for fire detection.

6.3.2 Dimensionality Reduction on FFMC Values

FFMC value is derived from an extensive calculation involving environmental parameter values including temperature, relative humidity, precipitation, and wind speed. The approach using DHGN recognition scheme is looking into reducing this burden experienced by the back-end processing within the sink, by providing a front-end detection scheme that enables only valid readings that will be sent for further processing.

Table 6.3 shows the FFMC value versus ignition potential level. This FFMC value provides an indication of relative ease of ignition and flammability of fine fuels due to exposure to extreme heat. In general, fires usually begin to ignite at FFMC values around 70, and the highest probable value to be reached is 96 (de Groot et al., 2005).

Ignition Potential	FFMC Value
Low	0 - 76
Moderate	77 - 84
High	85 - 88
Very High	89 - 91
Extreme	92+

Table 6.3: Ignition potential versus FFMC value.

The DHGN implementation performs dimensionality reduction on the FFMC values, by combining the levels for ignition potential into two stages: High Risk and Low Risk, as shown in Table 6.4. This approach could be used to determine the possibility of forest fire occurring, given certain values of sensory readings.

Ignition Potential	FFMC Value
Low Risk	0 - 84
High Risk	85+

Table 6.4: Modified FFMC classification for DHGN event detection scheme.

6.3.3 Methodology

The implementation of DHGN for forest fire detection involves a series of steps that reduces the expensive computation of FFMC values at the base station. The research proposes a distributed detection scheme that enables each sensor node to perform simple recognition process using DHGN to detect any abnormal readings obtained from its surroundings.

The first processing step in our proposed recognition scheme is the reduction of sensory data dimension using adaptive threshold binary signature approach. In this approach, we assume that each sensor node is composed of multiple sensors including temperature, relative humidity, precipitation, and wind speed. The readings would be converted into binary string representation or binary signature, as discussed in Chapter 3.

The second step is the actual recognition process, in which the binary signature is treated as subpattern and being introduced into specific DHGN processing cluster within each of the sensor nodes. It is assumed that DHGN processing cluster in this context has taken place as a block of memory space that could be used for simple DHGN recognition process. In addition, it is also assumed that each node is handling a subpattern (sensory readings) which collectively could become an overall pattern for the whole sensor nodes within the network. The recognition process is conducted by using reference subpatterns which consist of normal event pattern/readings.

Once the sensor node detected abnormal occurrence of subpattern (subpattern is not being recalled), it will send a signal to the base station for further analysis. This signal consists of all the sensory readings and event flag. The base station would then compute the FFMC value for the readings. Continuous signals being sent to the base station could be interpreted as an increment towards potential risk of fire. Therefore, early process of prevention could be executed at the specific location within the area of the sensor nodes.

A test was conducted on the accuracy of this scheme and a comparison with Kohonen's self-organizing map (SOM). Forest fire data was taken from Cortez and Morais (2007). The DHGN simulation was performed on computational grid environment for this dataset with 517 items. Three distinctive readings were taken from the dataset, which include temperature, relative humidity and wind speed. The precipitation (rainfall) values for this dataset have not been included in the analysis as it has shown minimal effect to the FFMC values. Table 6.5 shows the bits allocation for each of the readings. This bits

allocation eventually will be represented as a binary signature. The results of this test are presented in the following subsection.

Data	Bit Allocation
Temperature	2 bits
Relative Humidity	3 bits
Wind Speed	2 bits

Table 6.5: Sensory data with allocated binary signature bits.

6.3.4 Classification/Recognition Results

In this study, a supervised classification test was performed on DHGN event detection scheme, and the results compared with Kohonen's self-organizing map (SOM). The sole purpose of this comparison is only to demonstrate DHGN's ability to perform classification of events with high recall accuracy. In doing this, it was possible to benchmark the proposed scheme with commonly used classifiers such as Kohonen SOM. It is not the main intention of this research to replace SOM as an established classification tool. The SOM toolbox for Matlab that has been developed by Vesanto et al. (2000) was used. The results from this test have shown that this approach produces equivalent recall accuracy with both minimum training and training data. Furthermore, this scheme only requires binary input patterns.

Multiple sensory readings that are used to detect the occurrence of critical events, are considered as follows; given a set of x sensory readings (r_1, r_2, \dots, r_x) where $r_i \in R$ and $i = 1, 2, \dots, x$. A dimensionality-reduction technique known as threshold-signature was performed that converts each reading value to its respective binary signature. The threshold-signature technique utilises the threshold classes to represent a single data range into a binary format. Given the same sensory reading and H -threshold class, the equivalent binary signature that implies $b_i \rightarrow r_i$ is in the form of $b_i \in \{0, 1\}^H$. Therefore, for x -set sensory readings (r_1, r_2, \dots, r_x) will be converted into a set of binary signatures (b_1, b_2, \dots, b_x) . The following data in Table 6.6 shows samples of temperature threshold range with its equivalent binary signature.

If the output index from DHGN subnet matches the stored pattern for the critical event, then a signal is transmitted to the base station in the form of data packet (node id, timestamp, class id). The class id parameter is the identification for class of event that

Temperature Threshold Range (°C)	Binary Signature
0 - 20	10000
21-40	01000
41-60	00100

Table 6.6: Temperature threshold ranges with respective binary signatures.

has been detected. At a given time t , the base station might receive a number of signals from the network.

The training data used only signifies the normal event data (FFMC values lower than 84). For similar number of training data used, DHGN produces higher classification accuracy as compared to Kohonen SOM. Table 6.7 shows the training data that have been used in this classification test. The error value obtained using DHGN only incurs 0.1122, while SOM experienced high error value of up to 0.9439 for 3-class mode, as shown in Table 6.8.

Data	1	2	3
FFMC Value	≤ 84	≤ 84	≤ 84
Temperature (°C)	0-40 (10)	0-40 (10)	0-40 (10)
Relative Humidity	> 70 (001)	≤ 40 (100)	> 70 (001)
Wind Speed (km/h)	≤ 3 (10)	≤ 3 (10)	> 3 (01)
Binary Signature	1000110	1010010	1000101

Table 6.7: Training data set in the form of specific threshold ranges used in classification test. Binary digits in brackets represent signature for the respective data range.

Classifier	Error Value	1 - Error Value
DHGN	0.1122	0.8878
SOM (3 classes)	0.9439	0.0561
SOM (4 classes)	0.5532	0.4468
SOM (5 classes)	0.4487	0.5513
SOM (6 classes)	0.1103	0.8897

Table 6.8: Comparison on classification accuracy between DHGN and Kohonen SOM classifiers for forest fire detection. Different numbers of training data were used for each SOM implementation.

As shown in Table 6.8, the test on Kohonen SOM was extended to observe the effect of the training data increment on its classification accuracy, and abnormal event data (FFMC values higher than 84) was added. Note that an increase in the number of training data improves the accuracy of SOM classifier. Nevertheless, higher class value may leads to

higher complexity in obtaining a good classification output. Through this observation, DHGN's capability to perform classification and recognition of event data using small classes was demonstrated. Furthermore, the results produced are comparable with SOM implementation on higher number of classes.

The test also reveals that DHGN offers higher accuracy with minimum training data, in comparison with SOM approach. Furthermore, our distributed approach requires no training iteration, as it adopts a single-cycle learning mechanism. Comparatively, SOM requires high training iteration to achieve high classification accuracy. Figure 6.5 shows the number of iterations incurred for different number of training data being used in this test.

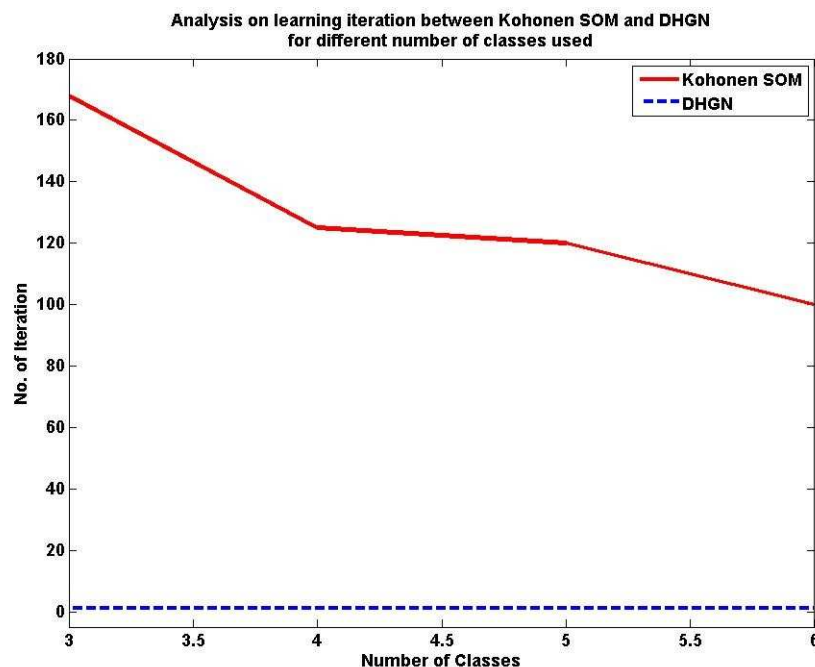


Figure 6.5: Analysis on learning iteration between Kohonen SOM and DHGN for different number of classes used in training.

The results of this test have also provided evidence to support the claim that the proposed scheme is capable of providing high classification accuracy with minimum effort. Thus, makes it deployable over resource-constrained networks such as WSN.

6.3.5 Spatio-Temporal Analysis of Event Data

The spatio-temporal analysis is a process of observing the frequency and distribution of events within the wireless sensor networks. It is conducted at the base station, since it has

the bird's eye view of the overall network. Figure 6.6 shows a scenario of spatio-temporal analysis using our proposed scheme.

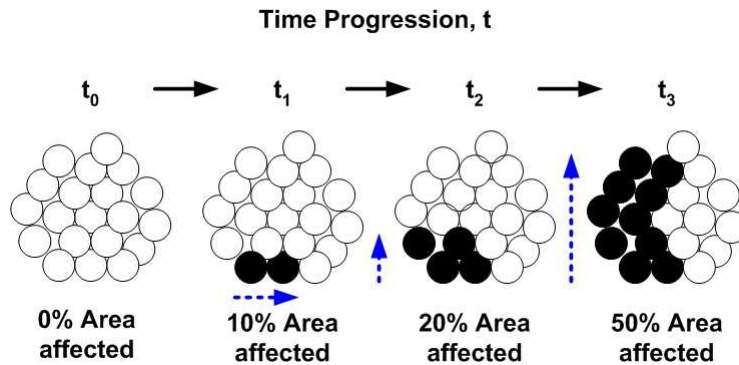


Figure 6.6: Analysis of event data triggered by the sensor nodes and received by the base station.

The dashed arrow shows the direction over time as the event occurs within the network. A brief process workflow for the proposed spatio-temporal recognition process is shown in Figure 6.7.

6.3.6 Summary

This section presents a case study on forest fire detection using DHGN distributed pattern recognition algorithm within WSN network. The proposed implementation involves minimum modification towards existing WSN infrastructure. Furthermore, based on the results of the classification test, DHGN has shown to perform well with minimum training data and within a single-cycle learning mechanism. This makes the proposed approach more viable for WSN deployment in forest fire detection.

There are several benefits and advantages in our DHGN implementation for event detection within WSN network. This new approach offers low memory consumption for event data storage using simple bias array representation. Furthermore, this scheme only stores subpatterns/patterns that relate to normal event, rather than keeping the records of all occurring events. The results also show that the approach is most effective for small subpattern size, since it uses only a small portion of the memory space in a typical physical sensor node in WSN network.

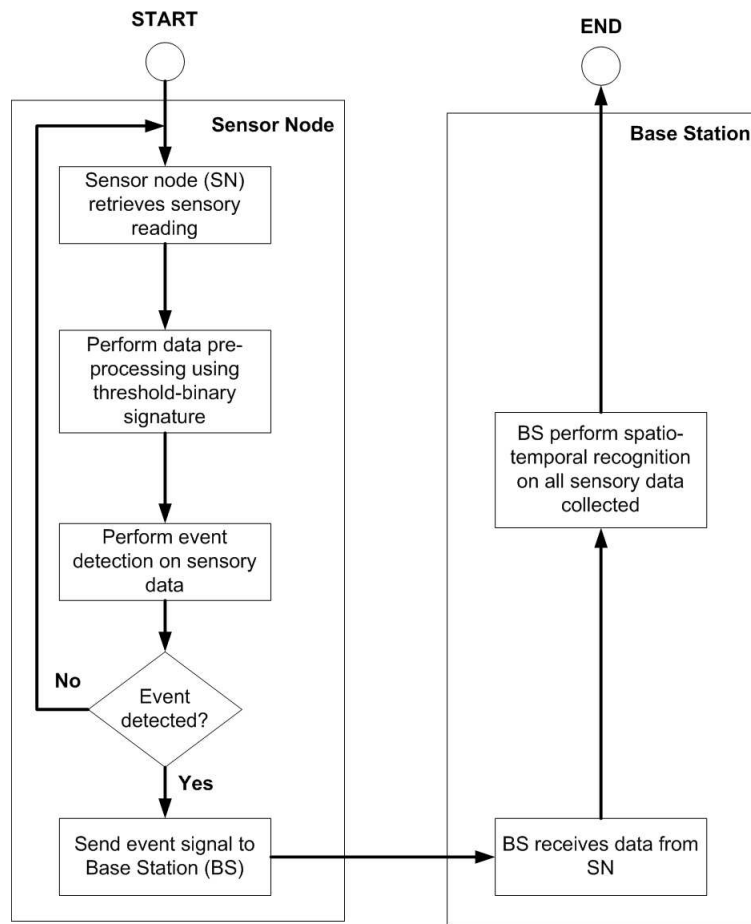


Figure 6.7: Process workflow for the proposed spatio-temporal event detection scheme using DHGN.

On a different note, DHGN is also susceptible to some limitations. Firstly, DHGN simple data representation would require significant advanced pre-processing at the front-end of the system. This might not be viable for strictly-resource constrained sensor nodes, where processing capability is very limited. In addition, DHGN single-hop communication for event detection scheme is not viable for large area monitoring, due to high possibility of communication error due to data packet loss during transmission. Our existing DHGN implementation has also been focusing on supervised classification. There is a need for unsupervised classification technique to be deployed for rapid event detection scheme.

These limitations of DHGN distributed event detection scheme would perhaps be the path for future research. Future research intentions of this researcher include looking into event tracking scheme using DHGN distributed detection mechanism, as well as providing

unsupervised classification capability for rapid and robust event detection scheme. Furthermore, future research may involve looking forward into implementation of this scheme in large-area monitoring using multi-hop communication strategy.

6.4 Conclusions

In this chapter, a case for distributed pattern recognition applications within a fine-grained system was presented. The research adopted a study on DHGN pattern recognition in event detection within WSN, as an example of such implementation within a fully-distributed and self-organised network. This case study has demonstrated the capabilities of DHGN to perform a distributed and lightweight detection mechanism for occurrences of events within a resource-constrained network such as WSN. There are several benefits and advantages in our DHGN implementation. The new approach offers low memory consumption for event data storage using simple bias array representation. Furthermore, this scheme only stores subpatterns/patterns that relate to normal event, rather than keeping the records of all occurring events. The research has also demonstrates that this new approach is most effective for small subpattern size, since it uses only a small portion of the memory space in a typical physical sensor node in WSN network.

In addition to this efficient memory usage, DHGN also eliminates the need for complex computations for event classification technique. With the adoption of single-cycle learning and adjacency comparison approaches, DHGN implements non-iterative and lightweight computational mechanism for event recognition and classification.

DHGN is a distributed pattern recognition algorithm. By having this distributed characteristic, DHGN would be readily-deployable over a distributed network. With such feature, DHGN has the ability to perform as a front-end detection scheme for event detection within WSN. Through divide-and-distribute approach, complex events could be perceived as a composition of events occurring at specific time and location. This new approach would be able to be used in event tracking in the future. However, the discussion on event tracking is not within the scope of this chapter. Nevertheless, our proposed scheme has been demonstrated to perform efficiently within an event detection scheme such as forest fire detection using WSN.

A case study on forest fire detection using DHGN-WSN infrastructure has also been presented. DHGN performances have been compared with Kohonen SOM and the results have shown that DHGN offers more efficient approach. Nevertheless, this comparison does not intend for a replacement of Kohonen SOM with DHGN, but more of comparative indication of DHGN's ability for event detection.

Despite all the benefits being discussed previously, DHGN is also susceptible to some limitations. Firstly, DHGN simple data representation would require significant pre-processing mechanism at the front-end of the system. This might not be viable for strictly-resource constrained sensor nodes, where processing capability is very limited. In addition, DHGN single-hop communication for event detection scheme is not viable for large area monitoring, due to high possibility of communication error due to data packet loss during transmission. The existing DHGN implementation has also been focusing on supervised classification. There is a need for unsupervised classification technique to be deployed for rapid event detection scheme.

These limitations of DHGN distributed event detection scheme would perhaps be the path for future research, which includes looking into event tracking scheme using DHGN distributed detection mechanism, as well as providing unsupervised classification capability for rapid and robust event detection scheme. Furthermore, looking into implementation of this scheme in large-area monitoring using multi-hop communication strategy is also a potential area for future research.

Chapter 7

Conclusions

7.1 Summary

The research conducted in this thesis has focused on the scalability problem in pattern recognition (PR). Scalability has been found to be highly significant in pattern recognition when dealing with the data deluge phenomenon. With the current outgrowth of data, particularly for their increasing complexity and size, the existing recognition schemes failed to provide optimum solutions for high-dimensional and large-scale patterns. Theory on distributed processing approach has been developed as a solution for the increasing issue of scalability in the existing CPU-centric pattern recognition schemes. A summary of the contents of this thesis is as follows.

- i. A review has been conducted on current implementations of scalable pattern recognition (Chapter 2). The review has indicated three different approaches towards solving the scalability problem within PR schemes, namely data, learning, and distributed approaches. As described in Sections 1.2 and 2.3, the data approach implements reduction or modification of data, while the learning approach tends to reduce the complexity of memorisation and recognition processes. Nevertheless, both the data and learning approaches do not fully overcome this scalability problem. This is mostly due to after-effects of these schemes, including loss of data integrity and low recognition accuracy. Moreover, the computational cost of using the learning approach is significantly expensive, making it less suitable for deployment in resource-constrained environments. The distributed approach has been shown to exhibit high scalability for PR implementation, that involves large and complex datasets. This

is mainly due to its ability to distribute data and processes within computational network(s), and to extend the processing capacity for patterns with large size and dimension. Nevertheless, the existing distributed pattern recognition schemes are unable to achieve optimum scalability, due to the tightly coupled and highly complex nature of recognition algorithms applied within those schemes.

- ii. This thesis has also showcased fundamental research on the distributed approach for pattern recognition. The position taken in this thesis is that a distributed pattern recognition scheme with single-cycle learning and bottom-up approach will remain scalable for any given size or dimensions of data, if sufficient computational resources are available. In this research, two important characteristics of a scalable distributed pattern recognition scheme have been identified. These are single-cycle learning mechanism and in-network processing capability. An effective and simple learning mechanism for memorising patterns is essential as part of the low-resource consumption approach for recognition procedure. This is accomplished within our proposed scheme through a learning mechanism that performs recognition within a single-cycle phase. From a different perspective, in-network processing capability provides a facility for expandable computational resources to be implemented. This makes a contrast with the existing recognition procedure with tightly-coupled schemes designed for single-CPU processing.
- iii. Based on the above position, a distributed scheme was proposed. This scheme incorporates both single-cycle learning mechanism and in-network processing capability for scalable pattern recognition involving large and complex data. These data are not only limited to numerical and textual data. Rather, the proposed scheme is highly-capable of handling complex patterns, such as high-dimensional images and multi-sensory readings. Complexity analysis of the proposed recognition algorithm was conducted using distributed approach. The study has indicated that the proposed scheme requires low computational complexity in its recognition procedure.
- iv. Apart from scalability considerations and the proposed distributed pattern recognition scheme, this research has also covered the works on recognition with multiple features. This research demonstrates that the proposed distributed pattern recognition scheme is capable of analysing complex patterns using multiple features. The

recall accuracy obtained from the simulation indicated that the scheme has been able to produce low error values for multi-feature recognition on complex patterns including facial images and handwritten character objects (see Chapter 6). Furthermore, the simulation results have indicated the ability of the proposed scheme to produce high recall accuracy with the minimum number of training data.

- v. In this research, a study on capability of the proposed scheme to perform pattern recognition on different levels of network granularity was established. This granularity ranged from coarse-grained networks such as the computational grid, to fine-grained networks including wireless sensor networks (WSNs). This capability was accomplished through the use of simple recognition procedure with low processing and storage capacity requirements.

From an overall perspective, this thesis has delivered a fundamental knowledge on distributed scheme as a solution for the scalability issue in pattern recognition applications that involve large-scale and complex data. Proofs of its high scalability and low complexity have also been presented. The adaptive granularity feature of the proposed scheme has also provided greater capacity for scalability considerations. Furthermore, the outcomes of the research have indicated a strong capability of the proposed distributed scheme to perform recognition on complex data with multiple features, while keeping its algorithmic complexity at minimum. Hence, simple pattern recognition procedure involving large and complex data is viable by the mean of distributed processing approach.

Even though the proposed distributed approach was presented as a solution for the scalability issue, it is best to note that the proposed scheme is not a replacement for existing pattern recognition schemes. Numerous initiatives have been made in developing highly-accurate pattern recognition schemes. This research does not only looks at accuracy, but also considers a distributed approach for the scalability issue in its implementation. As a result, the proposed scheme is scalable and highly suited for applications in parallel and distributed architectures.

7.2 Research Contributions

In this thesis, four key contributions have been made. These key contributions are recapitulated below, in the sequence that they appear in the thesis.

Initial discussions on scalability and distributed pattern recognition have been presented in Chapter 2. These comprise overall literature reviews conducted in this research, specifically on existing initiatives on distributed pattern recognition. Furthermore, background knowledge have also been added to the literature, on the aspects of pattern distribution techniques for recognition (See Section 2.4), together with current single-cycle learning approaches in distributed pattern recognition schemes (Sections 2.5-2.7).

The primary research contribution lies in the proposed distributed pattern recognition scheme, Distributed Hierarchical Graph Neuron (DHGN), (Chapter 3). There are three distinctive characteristics of the scheme that make it highly-suitable for scalable implementation for pattern recognition involving large-scale data. These include:

- i. *Bottom-up algorithmic architecture.* DHGN is a distributed pattern recognition scheme with in-network processing capability. Unlike any other distributed schemes that implement CPU-centric processing approach in distributed environment, DHGN has been developed purely with a network processing principle, making it highly-deployable within any computational networks (Section 3.1.2). The existing distributed pattern recognition schemes mainly rely on adaptive learning mechanisms such as incremental learning, to perform recognition in a distributed manner. This kind of learning has been initially developed with single-processing capacity, and requires extensive computational requirements. Hence, it makes this learning less applicable in a distributed environment. DHGN performs its recognition procedure using a body of network, by implementing distributed learning mechanism, in which each processing unit is capable of memorising a unit/segment of patterns.
- ii. *Divide-and-distribute technique for pattern analysis.* Pattern analysis in DHGN distributed scheme involves a process of dividing and distributing patterns into subpatterns for recognition process. DHGN works as a collection of subnets that perform recognition on subpatterns. This divide-and-distribute technique makes a substantial contribution to reduce the complexity of existing Hierarchical Graph Neuron (HGN) implementation, by reducing the number of processing nodes (labelled as Graph Neuron (GN)) up to 95% (See Section 2.7). This technique also improves the scalability of HGN, allowing more subnets to be added with an increase in the size of patterns. In addition, the divide-and-distribute technique in DHGN pattern

recognition scheme also introduces an effect known as error encapsulation, in which small distortion occurrence within a pattern is unlikely to affect recall accuracy of the overall pattern (Section 2.7.2). DHGN also performs a dual-layer recognition, in order to obtain overall information (at both pattern and subpattern levels) of the given pattern. This creates a holistic view for recognition and classification of patterns.

- iii. *Single-cycle learning mechanism.* DHGN implements a collaborative learning approach known as Collaborative-Comparison Learning (CCL) (see Section 3.1.5). This learning approach performs pattern memorisation and recall within a single cycle, without further iterations. This is different from existing learning approaches such as Hebbian-based learning that heavily rely on iterative weight or value adjustments to obtain optimum outputs. Furthermore, CCL is capable of performing memorisation with a low-impact on storage requirements. This was achieved by storing only unique bias entry composition within each GN node (Section 3.1.4). In relation to this single-cycle learning, DHGN is also capable of performing recognition procedure with minimum number of training patterns (Sections 3.4, 4.3, and 4.4.)
- iv. *Binary signature approach data representation.* DHGN has adopted a binary signature scheme for dimensionality reduction and data representation. This approach offers a high representation factor for complex patterns using binary data format.

In an overall perspective, DHGN is capable of performing scalable pattern recognition. The results obtained from a number of simulations have shown that its recall accuracy is considerably high for distorted pattern recognition. In examining the accuracy of DHGN recogniser, it was found that the proposed scheme is capable of producing perfect recall for up to 20% distortion on binary character images (Section 3.4). At this level of distortion, even human eye can barely associate the original pattern with distorted ones. Besides its high accuracy, DHGN imposes low computational complexity (Section 3.3). Comparative analysis of recognition accuracy and complexity has been carried out between DHGN and other recognisers/classifiers such as HGN, Hopfield Network, and Kohonen SOM. The results of this analysis have indicated DHGN's relatively low complexity in performing recognition process, in comparison with these highly-iterative approaches. In terms of Big-O complexity estimation, DHGN only impose linear complexity as low as $O(n)$

in its recognition process, where n represents a single executable instruction within the procedure. This was achieved using our collaborative-comparison learning approach. This study also extend its comparative analysis to include neural networks with fast learning mechanism such as Instantaneously-Trained Neural Networks (ITNN).

The second key contribution of this thesis lies in the distributed properties of CCL learning mechanism as a solution for an important pattern recognition problem within this scalable distributed framework. This problem relates to the recognition involving multiple features. In this thesis, a multi-feature pattern recognition implementation using DHGN distributed scheme has been considered (Chapter 4). This research has demonstrated DHGN capability to perform pattern recognition that involves multiple features analysis and evaluation. The proposed solution enables high-dimensional patterns such as images, to be represented with a number of different features in recognition process, without incurring high computational costs. Furthermore, the multi-feature scheme allows extensive amount of features to be analysed, without impending on its scalability. This is achieved using the DHGN distributed architecture that provides scalable resources in the form of computational networks. The results of DHGN multi-feature implementation on greyscale facial images and handwritten numeral objects have indicated high recall accuracy for a large number of patterns used. Therefore, this solution offers significantly important contribution in the area of complex pattern recognition. Furthermore, comparative analysis with other combined-classifier approaches for multi-feature pattern recognition has indicated DHGN's high recall accuracy for complex patterns with minimum number of training dataset used.

The final key contribution of this research lies in the resource-awareness characteristic of DHGN. This contribution has been extensively described in Chapter 5, with a case study in Chapter 6. DHGN, being a purely distributed pattern recognition scheme, has acquired a behaviour that relates to its adaptability against different levels of network granularity. This resource-awareness characteristic provides capability for DHGN to perform its recognition procedure with minimum storage and processing capacity. Apart from this characteristic, DHGN has also been proven to perform fast recognition procedure, in which for a single recall/store process of each subpattern, the execution time will only

lasts about less than 0.1 seconds (60-70 milliseconds). Therefore a real-time DHGN pattern recognition is a possibility, based upon the case on its implementation within a WSN distributed event detection scheme (See Chapter 6).

7.3 Future Research

This research has primarily aimed to create a scalable scheme for pattern recognition involving complex and large-scale data. For this reason, a distributed approach for scalable pattern recognition has been proposed, which incorporates in-network processing capability and single-cycle learning mechanism. DHGN as a distributed pattern recognition scheme has been able to provide a scalable recognition framework with high recall accuracy and low computational complexity.

It is possible to suggest two directions to further improve the DHGN scheme for scalable pattern recognition, namely focus on algorithm and application.

7.3.1 Algorithm-Specific Research

Two specific improvements on DHGN scheme that may be carried out as future works have been identified. These include:

- i. *Bias array design.* The existing bias array design heavily relies upon unique bias entries stored within the array. This enables efficient savings on storage space, as only unique entries, representing unique patterns will be memorised. However, as more unique patterns being stored, the array size may significantly be large and the physical storage of each GN could no longer be able to accommodate such outgrowth in size. Further research on bias array design may be carried out for an efficient design can be proposed as a solution to this possible limitation.
- ii. *Network structure.* An important aspect of DHGN architecture is its network structure, specifically in regards to network composition. Even though DHGN has been able to significantly reduce the required number of processing nodes, this number still remain high, for large pattern size and dimension. Further study has to be carried out in this respect, by looking at other forms of structural representation, apart from the existing pyramidal and hierarchical form.

7.3.2 Application-Specific Research

Four possible areas of application developments have been identified, to further improve DHGN implementation:

- i. *Object recognition.* In the evaluation of the DHGN distributed pattern recognition, this thesis did not include pattern recognition that involves objects such as human movements, as well as object tracking. In addition, a future work may also be carried out on the recognition mechanism on patterns/objects involving structural distortion, such as rotation, transformation and relocation.
- ii. *Spatio-temporal event detection.* Distributed event detection using DHGN within WSN has been proposed in Chapter 6 of this thesis. DHGN provides capability for event detection to be deployed within a WSN, by performing recognition procedure at the sensory level. To extent this capability for WSN real-time event detection, we propose a future work on DHGN-based event detection to include spatio-temporal analysis of event. This analysis involves a process of observing the frequency and distribution of event, such as forest fire within a given sensory network. This could be achieved by implementing dual-recognition procedure that entails initial event detection occurring at the sensor level using DHGN algorithm and event distribution detection that is performed at the base station using DHGN algorithm on computational grid.
- iii. *Cloud data management.* The efficiency of the existing cloud system in dealing with data intensive applications through parallel processing, essentially lies in how data is partitioned among nodes, and how collaboration among nodes are handled to accomplish a specific task (Wu and Wu, 2009). Dependency on cloud file systems such as Hadoop Distributed File Systems (HDFS) for data storage and retrieval can create single-points of failure for Map/Reduce infrastructure, especially at master nodes. In fact, if they go down, all running jobs are lost. As a result, and to address the aforementioned concerns in relation to data storage and retrieval in cloud, any data access scheme should aim to handle partitioning between processing nodes, as well as node collaborations in a robust manner. These two features are still lacking in the current data access mechanisms. Hence, new data management approaches need to be investigated for cloud computing environments. DHGN's

pattern matching capability and the small response time, that remains insensitive to the increases in the number of stored patterns, can make this approach remarkably suitable for clouds. Moreover, the DHGN does not require definition of rules or manual interventions by the operator for setting of thresholds to achieve the desired results, nor does it require heuristics entailing iterative operations for memorisation and recall of patterns.

- iv. *Content-based image retrieval (CBIR)*. Current approaches in large-scale image retrieval are highly-dependable upon keyword meta-tags search. However, there is considerably weak association between these keywords and their targeted images, due to semantic disconnection between words and visual content. Future works entailing the use of DHGN in content-based image retrieval applications is possible. DHGN has been shown to produce high recall accuracy in pattern recognition involving both heterogeneous (e.g. handwritten character objects) and homogeneous (e.g. facial images) images. Furthermore, DHGN exerts low-computational requirements with fast recognition time, making it suitable for large-scale and real-time deployment.

References

- Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities, *IEEE Data Eng. Bull.* **32**(1): 3–12.
- Ai, C., Hou, H., Li, Y. and Beyah, R. (2009). Authentic delay bounded event detection in heterogeneous wireless sensor networks, *Ad Hoc Netw.* **7**(3): 599–613.
- Akyama, M. T. and Kikuti, M. (2001). Recognition of character using a morphological associative memory, *SIBGRAPI '01: Proceedings of the 14th Brazilian Symposium on Computer Graphics and Image Processing*, IEEE Computer Society, Washington, DC, USA, p. 400.
- Al-Hertani, H. and Ilow, J. (2005). Pattern recognition based detection and localization in a network of randomly distributed sensor nodes, *ISDA '05: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 412–419.
- Albiol, A., Monzo, D., Martin, A., Sastre, J. and Albiol, A. (2008). Face recognition using hog-ebgm, *Pattern Recognition Letters.* **29**(10): 1537–1543.
- Amin, A. H. M. and Khan, A. I. (2008). Parallel pattern recognition using a single-cycle learning approach within wireless sensor networks, *PDCAT: Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2008, Dunedin, Otago, New Zealand, 1-4 December 2008*, pp. 305–308.
- Anderson, C. (2008). The end of theory: The data deluge makes the scientific method obsolete. accessed on 03/03/2010.
<http://www.wired.com/science/discoveries/magazine/16-07/>

- Arrue, Bego n. C., Ollero, A. and de Dios, J. R. M. (2000). An intelligent system for false alarm reduction in infrared forest-fire detection, *IEEE Intelligent Systems* **15**(3): 64–73.
- Auwatanamongkol, S. (2007). Inexact graph matching using a genetic algorithm for image recognition, *Pattern Recognition Letters* **28**(12): 1428 – 1437.
- Baig, Z. A., Baqer, M. and Khan, A. I. (2006). A pattern recognition scheme for distributed denial of service (ddos) attacks in wireless sensor networks, *ICPR (3)*, pp. 1050–1054.
- Banerjee, T., Xie, B. and Agrawal, D. P. (2008). Fault tolerant multiple event detection in a wireless sensor network, *J. Parallel Distrib. Comput.* **68**(9): 1222–1234.
- Baqer, M. (2008). *Energy Ecient Event Recognition for Wireless Sensor Networks*, PhD thesis, Faculty of Information Technology, Monash University.
- Baqer, M. and Khan, A. (2007). Energy-efficient pattern recognition approach for wireless sensor networks, *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pp. 509–514.
- Baqer, M., Khan, A. I. and Baig, Z. A. (2005). Implementing a graph neuron array for pattern recognition within unstructured wireless sensor networks, *EUC Workshops*, pp. 208–217.
- Battiti, R. and Colla, A. M. (1994). Democracy in neural nets: voting schemes for classification, *Neural Netw.* **7**(4): 691–707.
- Beck, M., Dongarra, J. and Plank, J. S. (2005). Netsolve/d: A massively parallel grid execution system for scalable data intensive collaboration, *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 10*, IEEE Computer Society, Washington, DC, USA, p. 223.1.
- Bengoetxea, E. (2003). *Inexact Graph Matching Using Estimation of Distribution Algorithms*, PhD thesis, Département Traitement du Signal et des Images, Ecole Nationale Supérieure des Télécommunications.
- Berglund, E. and Sitte, J. (2006). The parameterless self-organizing map algorithm, *Neural Networks, IEEE Transactions on* **17**(2): 305–316.

- Black, P. E. (2008). big-o notation. accessed on 06/04/2010.
<http://www.itl.nist.gov/div897/sqg/dads/HTML/bigOnotation.html>
- Blazejewski, A. and Coggins, R. (2004). Application of self-organizing maps to clustering of high-frequency financial data, *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 85–90.
- Bondi, A. B. (2000). Characteristics of scalability and their impact on performance, *WOSP '00: Proceedings of the 2nd international workshop on Software and performance*, ACM, New York, NY, USA, pp. 195–203.
- Bouvier, J. (2006). Notes on convolutional neural networks.
<http://cogprints.org/5869/>
- Bow, S.-T. (2002). *Pattern Recognition and Image Preprocessing*, Marcel Dekker, Inc., New York, NY, USA.
- Caetano, T. S., McAuley, J. J., Cheng, L., Le, Q. V. and Smola, A. J. (2009). Learning graph matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**: 1048–1058.
- Canny, J. (1986). A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(6): 679–698.
- Cao, J., Ahmadi, M. and Shridhar, M. (1995). Recognition of handwritten numerals with multiple feature and multistage classifier, *Pattern Recognition* **28**(2): 153 – 160.
- Casali, D., Costantini, G., Perfetti, R. and Ricci, E. (2006). Associative memory design using support vector machines, *Neural Networks, IEEE Transactions on* **17**(5): 1165–1174.
- Catterall, E., Van Laerhoven, K. and Strohbach, M. (2003). Self-organization in ad hoc sensor networks: an empirical study, *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, MIT Press, Cambridge, MA, USA, pp. 260–263.
- Cheng, J. and Wang, K. (2007). Active learning for image retrieval with co-svm, *Pattern Recognition* **40**(1): 330 – 334.

- Cheung, Y.-M. and Law, L. (2007). Rival-model penalized self-organizing map, *Neural Networks, IEEE Transactions on* **18**(1): 289–295.
- Chitkara, V., Nascimento, M. A. and Mastaller, C. (2001). Content-based image retrieval using binary signatures, *Technical Report 00-18*, University of Alberta.
- Choi, H.-C. and Oh, S.-Y. (2006). Efficient human-like memory management based on walsh-based associative memory for real-time pattern recognition, *IJCNN*, pp. 3657–3663.
- Chow, T. W. S. and Huang, D. (2008). Data reduction for pattern recognition and data analysis, *Computational Intelligence: A Compendium*, Springer, Berlin / Heidelberg, pp. 81–109.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks, *Mach. Learn.* **20**(3): 273–297.
- Cortez, P. and Morais, A. (2007). Data mining approach to predict forest fires using meteorological data, *New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence*, pp. 512–523.
- Cruz, B., Sossa, H. and Barrón, R. (2007). A new two-level associative memory for efficient pattern restoration, *Neural Process. Lett.* **25**(1): 1–16.
- Culler, D. E., Estrin, D. and Srivastava, M. B. (2004). Guest editors' introduction: Overview of sensor networks, *IEEE Computer* **37**(8): 41–49.
- de Groot, W., Wardati and Wang, Y. (2005). Calibrating the fine fuel moisture code for grass ignition potential in sumatra, indonesia, *International Journal of Wildland Fire* **14**: 161–168.
- Dong, J.-X., Krzyzak, A. and Suen, C. (2005). Fast svm training algorithm with decomposition on very large data sets, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**(4): 603–618.
- Duin, R. P. W. and Tax, D. M. J. (2000). Experiments with classifier combining rules, *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, Springer-Verlag, London, UK, pp. 16–29.

- Favata, J. T. and Srikantan, G. (2002). A multiple feature/resolution approach to hand-printed digit and character recognition, *International Journal of Imaging Systems and Technology* **7**: 304–311.
- Fei, B. and Liu, J. (2006). Binary tree of svm: a new fast multiclass training and classification algorithm, *Neural Networks, IEEE Transactions on* **17**(3): 696–704.
- Fei-Fei, L., Fergus, R. and Perona, P. (2006). One-shot learning of object categories, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **28**(4): 594–611.
- Fleming, J. and Robertson, R. G. (n.d.). Fire management tech tips, *Technical report*, San Dimas CA, USA.
- Foo, S. K., Saratchandran, P. and Sundararajan, N. (1995). Comparison of parallel and serial implementation of feedforward neural networks, *J. Microcomput. Appl.* **18**(1): 83–94.
- Fox, G. C., Aktas, M. S., Aydin, G., Donnellan, A., Gadgil, H., Granat, R., Pallickara, S., Parker, J., Pierce, M. E., Oh, S., Rundle, J., Sayar, A. and Scharber, M. (2005). Building sensor filter grids: Information architecture for the data deluge, *SKG '05: Proceedings of the First International Conference on Semantics, Knowledge and Grid*, IEEE Computer Society, Washington, DC, USA, p. 2.
- Frank, A. and Asuncion, A. (2010). UCI machine learning repository. accessed on 06/10/2010.
<http://archive.ics.uci.edu/ml>
- Garai, G. and Chaudhuri, B. (2007). A distributed hierarchical genetic algorithm for efficient optimization and pattern matching, *Pattern Recognition* **40**(1): 212 – 228.
- Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA.
- Ghosh, S. (2006). *Distributed Systems: An Algorithmic Approach*, CRC Press.
- Giorgetti, G., Gupta, S. K. S. and Manes, G. (2007). Wireless localization using self-organizing maps, *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, ACM, New York, NY, USA, pp. 293–302.

- Gropp, W., Thakur, R. and Lusk, E. (1999). *Using MPI-2: Advanced Features of the Message Passing Interface*, MIT Press, Cambridge, MA, USA.
- Guoqing, Y., Songcan, C. and Jun, L. (1992). Multilayer parallel distributed pattern recognition system model using sparse ram nets, *Computers and Digital Techniques, IEEE Proceedings on* **139**(2): 144–146.
- Guralnik, V. and Srivastava, J. (1999). Event detection from time series data, *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, New York, NY, USA, pp. 33–42.
- Hafez, R., Haroun, I. and Lambadaris, I. (2005). Building wireless sensor networks. accessed on 18/05/2010.
<http://www.mwrf.com/Article/ArticleID/11071/11071.html>
- Hassoun, M. and Watta, P. (1996). The hamming associative memory and its relation to the exponential capacity dam, *Neural Networks, 1996., IEEE International Conference on*, Vol. 1, pp. 583–587.
- Hattori, M., Fukui, A. and Ito, H. (2002). A fast method of constructing kernel patterns for morphological associative memory, *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, Vol. 2, pp. 1058–1063.
- Hebb, D. O. (1988). The organization of behavior, *Neurocomputing: foundations of research* pp. 43–54.
- Hecht-Nielsen, R. (1989). *Neurocomputing*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Hey, A. J. G. and Trefethen, A. E. (2003). The data deluge: An e-science perspective. accessed on 06/10/2010.
<http://eprints.ecs.soton.ac.uk/7648/>
- Hongtao, S., Feng, D. D. and Rong-chun, Z. (2002). Face recognition using multi-feature and radial basis function network, *VIP '02: Selected papers from the 2002 Pan-Sydney workshop on Visualisation*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 51–57.

- Hopfield, J. and Tank, D. (1985). Neural computation of decisions in optimization problems, *Biological Cybernetics* **52**: 141–152.
- Hsiao, S.-J., Sung, W.-T. and Fan, K.-C. (2002). Web-based distributed pattern recognition system, *Information Visualisation, International Conference on* p. 375.
- Huang, G.-B., Mao, K., Siew, C.-K. and Huang, D.-S. (2005). Fast modular network implementation for support vector machines, *Neural Networks, IEEE Transactions on* **16**(6): 1651–1663.
- Ikeda, N., Watta, P., Artiklar, M. and Hassoun, M. H. (2001). A two-level hamming network for high performance associative memory, *Neural Networks* **14**(9): 1189 – 1200.
- Jain, A. K., Duin, R. P. and Mao, J. (2000). Statistical pattern recognition: A review, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1): 4–37.
- Jiang, H., Liu, T. and Wang, M. (2006). Direct estimation of fault tolerance of feed forward neural networks in pattern recognition, *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pp. 864 –869.
- Joachims, T. (1999). Making large-scale support vector machine learning practical, pp. 169–184.
- Kak, S. (2002). A class of instantaneously trained neural networks, *Information Science-Applications: An International Journal* **148**(1-4): 97–102.
- Kalos, A. (2005). Automated heuristic growing of neural networks for nonlinear time series models, *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, Vol. 1, pp. 320–325.
- Kamath, C. and Musick, R. (2000). Scalable data mining through fine-grained parallelism: The present and the future, *Advances in Distributed and Parallel Knowledge Discovery* pp. 29–77.
- Kasabov, N. K. (1996). *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, MIT Press, Cambridge, MA, USA.

- Kbir, M. A., Maalmi, K., Benslimane, R. and Benkirane, H. (2000). Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules, *Pattern Recognition Letters* **21**(6-7): 503–509.
- Khan, A. I. (2002). A peer-to-peer associative memory network for intelligent information systems, *Enabling Organisations and Society Through Information Systems: The Proceedings of The Thirteenth Australasian Conference on Information Systems, Melbourne, Victoria, Australia*, pp. 317–326.
- Khan, A. I., Isreb, M. and Spindler, R. S. (2004). A parallel distributed application of the wireless sensor network, *HPCASIA '04: Proceedings of the High Performance Computing and Grid in Asia Pacific Region, Seventh International Conference*, IEEE Computer Society, Washington, DC, USA, pp. 81–88.
- Khan, A. I. and Mihailescu, P. (2004). Parallel pattern recognition computations within a wireless sensor network, *ICPR (1)*, pp. 777–780.
- Khan, A. I. and Muhamad Amin, A. (2007). One shot associative memory method for distorted pattern recognition, *AI 2007: Advances in Artificial Intelligence*, Springer, Berlin/Heidelberg, pp. 705–709. accessed on 06/10/2010.
<http://www.springerlink.com/content/g4315t3170t54h01>
- Khan, A. I. and Muhamad Amin, A. H. (2009). Integrating sensory data within a structural analysis grid, *Parallel, Distributed and Grid Computing for Engineering*, Saxe-Coburg Publications, Stirlingshire, UK.
- Khan, A. I., Muhamad Amin, A. H. and Raja Mahmood, R. (2010a). Lightweight event detection scheme using distributed hierarchical graph neuron in wireless sensor networks, *Wireless Sensor Networks*, In-Tech Publications. In-Press.
- Khan, A. I., Muhamad Amin, A. H. and Raja Mahmood, R. A. (2010b). An on-line scheme for threat detection within mobile ad hoc networks, *Research in Mobile Intelligence*, John Wiley & Sons, Inc.
- Kim, J. H., Yoon, S. H., Kim, Y. H., Park, E. H., Ntuen, C. A. and Sohn, K. (1992). Efficient matching algorithm by a hybrid Hopfield network for object recognition, *in*

- S. K. Rogers (ed.), *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Vol. 1709 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 908–916.
- Kimmel, R., Shaked, D., Elad, M. and Sobel, I. (2005). Space-dependent color gamut mapping: a variational approach, *IEEE Transactions on Image Processing* **14**(6): 796–803.
- Kohonen, T. (2000). *Self-Organizing Maps*, 3rd edn, Springer.
- Kokiopoulou, E. and Frossard, P. (2006). Distributed svm applied to image classification, *Multimedia and Expo, 2006 IEEE International Conference on*, pp. 1753–1756.
- Kosko, B. (1988). Bidirectional associative memories, *IEEE Trans. Syst. Man Cybern.* **18**(1): 49–60.
- Kosko, B. (1992). *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Kulakov, A. and Davcev, D. (2005a). Distributed data processing in wireless sensor networks based on artificial neural-networks algorithms, *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications*, IEEE Computer Society, Washington, DC, USA, pp. 353–358.
- Kulakov, A. and Davcev, D. (2005b). Tracking of unusual events in wireless sensor networks based on artificial neural-networks algorithms, *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, IEEE Computer Society, Washington, DC, USA, pp. 534–539.
- Kumar, V., Shekhar, S. and Amin, M. B. (1994). A scalable parallel formulation of the backpropagation algorithm for hypercubes and related architectures, *IEEE Trans. Parallel Distrib. Syst.* **5**(10): 1073–1090.
- Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience.
- Lawrence, S., Giles, C. and Tsoi, A. C. (1996). Convolutional neural networks for face recognition, pp. 217–222.

- LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series, in M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press.
- Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**(11): 2278–2324.
- Levis, P. and Culler, D. (2002). Maté: a tiny virtual machine for sensor networks, *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ACM, New York, NY, USA, pp. 85–95.
- Lewis, F. (2004). Wireless sensor networks, *Smart Environments: Technology, Protocols and Applications*, John Wiley & Sons, Inc.
- Lewis, R. A., Hall, C. J., Hufton, A. P., Evans, S., Menk, R. H., Arfelli, F., Rigon, L., Tromba, G., Dance, D. R., Ellis, I. O., Evans, A., Jacobs, E., Pinder, S. E. and Rogers, K. D. (2003). X-ray refraction effects: application to the imaging of biological tissues, *Br J Radiol* **76**(905): 301–308.
- Li, S., Lin, Y., Son, S. H., Stankovic, J. A. and Wei, Y. (2004). Event detection services using data service middleware in distributed sensor networks, *Telecommunication Systems* **26**(2-4): 351–368.
- Li, Y. and Parker, L. E. (2008). Detecting and monitoring time-related abnormal events using a wireless sensor network and mobile robot, *IROS*, pp. 3292–3298.
- Li, Y., Tang, Z., Xia, G. and Wang, R. (2005). A positively self-feedbacked hopfield neural network architecture for crossbar switching, *Circuits and Systems I: Regular Papers, IEEE Transactions on* **52**(1): 200–206.
- Lin, X., Soergel, D. and Marchionini, G. (1991). A self-organizing semantic map for information retrieval, *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, NY, USA, pp. 262–269.

- Lobo, V., Bandeira, N. and Moura-Pires, F. (1998). Ship recognition using distributed self organizing maps, *Proceedings of the 1998 International Conference on Engineering Applications of Neural Networks (EANN98)*, pp. 326–329.
- Löwe, M. (1999). On the storage capacity of the hopfield model with biased patterns, *IEEE Transactions on Information Theory* **45**(1): 314–318.
- Mavroforakis, M. and Theodoridis, S. (2006). A geometric approach to support vector machine (svm) classification, *Neural Networks, IEEE Transactions on* **17**(3): 671–682.
- Miller, E., Matsakis, N. and Viola, P. (2000). Learning from one example through shared densities on transforms, *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 1, pp. 464–471.
- Moore, G. E. (2000). Cramming more components onto integrated circuits, pp. 56–59.
- Morrill, J. P. (1998). Distributed recognition of patterns in time series data, *Commun. ACM* **41**(5): 45–51.
- Mu, X., Watta, P. and Hassoun, M. (2007). A weighted voting model of associative memory, *Neural Networks, IEEE Transactions on* **18**(3): 756–777.
- Muhamad Amin, A. H. and Khan, A. I. (2008a). Commodity-grid based distributed pattern recognition framework, *AusGrid '08: Proceedings of the sixth Australasian workshop on Grid computing and e-research*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 27–34.
- Muhamad Amin, A. H., Raja Mahmood, R. A. and Khan, A. I. (2008). Analysis of pattern recognition algorithms using associative memory approach: A comparative study between the hopfield network and distributed hierarchical graph neuron (dhgn), *CIT-WORKSHOPS '08: Proceedings of the 2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, IEEE Computer Society, Washington, DC, USA, pp. 153–158.
- Muhamad Amin, A. and Khan, A. I. (2008b). Single-cycle image recognition using an adaptive granularity associative memory network, *AI 2008: Advances in Artificial*

- Intelligence*, Springer, Berlin/Heidelberg, pp. 386–392. accessed on 06/10/2010.
<http://www.springerlink.com/content/u8550287378r8815>
- Muhamad Amin, A. and Khan, A. I. (2009). Collaborative-comparison learning for complex event detection using distributed hierarchical graph neuron (dhgn) approach in wireless sensor network, *AI 2009: Advances in Artificial Intelligence*, Springer, Berlin/Heidelberg, pp. 111–120. accessed on 06/10/2010.
<http://www.springerlink.com/content/074871630060g843>
- Nadal, J.-P. (1989). Study of a growth algorithm for a feedforward network, *Int. J. Neural Syst.* **1**(1): 55–59.
- Nagy, G. (2005). Interactive, mobile, distributed pattern recognition, *CIAP05*, pp. 37–49.
- Nascimento, M. A. and Chitkara, V. (2002). Color-based image retrieval using binary signatures, *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, ACM, New York, NY, USA, pp. 687–692.
- Nasution, B. B. (2007). *Trusted Transaction Secure Network: Agent-Based Distributed Security Control System for Traffic on the Internet*, PhD thesis, Faculty of Information Technology, Monash University.
- Nasution, B. B. and Khan, A. I. (2008). A hierarchical graph neuron scheme for real-time pattern recognition, *IEEE Transactions on Neural Networks* **19**(2): 212–229.
- Nasution, B. B., Khan, A. I. and Kendall, E. A. (2005). Incorporating graph neurons (gns) to the trusted transient simple network (ttsn) security control system architecture, *IASTED Conf. on Software Engineering*, pp. 13–19.
- Nebauer, C. (1998). Evaluation of convolutional neural networks for visual recognition, *Neural Networks, IEEE Transactions on* **9**(4): 685–696.
- Ng, W. W., Dorado, A., Yeung, D. S., Pedrycz, W. and Izquierdo, E. (2007). Image classification with the use of radial basis function neural networks and the minimization of the localized generalization error, *Pattern Recognition* **40**(1): 19–32.
- Nguyen, D. and Ho, T. (2006). A bottom-up method for simplifying support vector solutions, *Neural Networks, IEEE Transactions on* **17**(3): 792–796.

- Nilsson, N. (1996). Introduction to machine learning: An early draft of a proposed textbook. accessed on 06/10/2010.
<http://robotics.stanford.edu/people/nilsson/mlbook.html>
- Nowicki, D. and Dekhtyarenko, O. (2004). Kernel-based associative memory, *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, Vol. 1, pp. – 744.
- Pal, S. K. and Mitra, P. (2004). *Pattern Recognition Algorithms for Data Mining: Scalability, Knowledge Discovery, and Soft Granular Computing*, Chapman & Hall, Ltd., London, UK, UK.
- Panduranga, P. P., Rao, D. and Deshpande, A. G. (2007). Fault tolerance analysis of neural networks for pattern recognition, *Computational Intelligence and Multimedia Applications, International Conference on* 1: 222–226.
- Park, S., Locher, I., Savvides, A., Srivastava, M. B., Chen, A., Muntz, R. and Yuen, S. (2002). Design of a wearable sensor badge for smart kindergarten, *Wearable Computers, IEEE International Symposium* p. 231.
- Pripuzic, K., Belani, H. and Vukovic, M. (2010). Early forest fire detection with sensor networks: Sliding window skylines approach, *Knowledge-Based Intelligent Information and Engineering Systems*, Springer Berlin / Heidelberg.
- Raja Mahmood, R., Muhamad Amin, A. and Khan, A. (2008). A lightweight, fast and efficient distributed hierarchical graph neuron-based pattern classifier, *International Journal of Intelligent Engineering and Systems (IJIES)* 1(4): 9–17. accessed on 06/10/2010.
<http://www.inass.org/publications.asp?id=502>
- Ritter, G. and Sussner, P. (1996). An introduction to morphological neural networks, *Pattern Recognition, International Conference on* 4: 709–717.
- Ritter, G., Sussner, P. and Diaz-de Leon, J. (1998). Morphological associative memories, *Neural Networks, IEEE Transactions on* 9(2): 281–293.

- Román-Godínez, I., López-Yáñez, I. and Yáñez-Márquez, C. (2009). Classifying patterns in bioinformatics databases by using alpha-beta associative memories, *Biomedical Data and Applications* pp. 187–210.
- Rosenblatt, F. (1957). The perceptron — a perceiving and recognizing automaton, *Technical Report No.85-460-1* .
- Rueda, L. and Herrera, M. (2008). Linear dimensionality reduction by maximizing the chernoff distance in the transformed space, *Pattern Recognition* **41**(10): 3138–3152.
- Rumelhart, D. E. and Zipser, D. (1988). Feature discovery by competitive learning, *Connectionist models and their implications: readings from cognitive science*, Ablex Publishing Corp., Norwood, NJ, USA, pp. 205–242.
- Saha, S. and Bajcsy, P. (2003). System design issues in single-hop wireless sensor networks, *The IASTED International Conference on Communications, Internet and Information Technology 2003 (CIIT 2003), Scottsdale, AZ, USA, 17-19 November 2003*.
- Schalkoff, R. J. (1991). *Pattern recognition: statistical, structural and neural approaches*, John Wiley & Sons, Inc., New York, NY, USA.
- Schalkoff, R. J. (1997). *Artificial Neural Networks*, McGraw-Hill Companies.
- Schlimmer, J. C. and Granger, Jr., R. H. (1986). Incremental learning from noisy data, *Machine Learning* **1**(3): 317–354.
- Shih, K.-P., Wang, S.-S., Chen, H.-C. and Yang, P.-H. (2008). Collect: Collaborative event detection and tracking in wireless heterogeneous sensor networks, *Computer Communications* **31**(14): 3124 – 3136.
- Simard, P. Y., Steinkraus, D. and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis, *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, IEEE Computer Society, Washington, DC, USA, p. 958.
- Song, F., Liu, H., Zhang, D. and Yang, J. (2008). A highly scalable incremental facial feature extraction method, *Neurocomputing* **71**(10-12): 1883 – 1888. Neurocomputing for Vision Research; Advances in Blind Signal Processing.

- Srinivas, A. V. and Janakiram, D. (2005). A model for characterizing the scalability of distributed systems, *SIGOPS Oper. Syst. Rev.* **39**(3): 64–71.
- Stankovic, J. A. (2008). Wireless sensor networks, *IEEE Computer* **41**(10): 92–95.
- Sulehria, H. and Zhang, Y. (2008). Study on the capacity of hopfield neural networks, *Information Technology Journal* **7**(4): 684–688.
- Sussner, P. and Valle, M. (2006). Gray-scale morphological associative memories, *Neural Networks, IEEE Transactions on* **17**(3): 559–570.
- Talukder, A., Sheikh, T. and Chandramouli, L. (2004). Real-time intelligent pattern recognition, resource management and control under constrained resources for distributed sensor networks, *Neural Networks, IEEE Proceedings International Joint Conference on* **2**: 1321 – 1326.
- Theodoridis, S. and Koutroumbas, K. (2003). *Pattern Recognition*, Academic Press, San Diego, CA, USA.
- Tsang, I., Kwok, J. and Zurada, J. (2006). Generalized core vector machines, *Neural Networks, IEEE Transactions on* **17**(5): 1126–1140.
- Turkoglu, I. and Arslan, A. (2001). Optimisation of the performance of neural network based pattern recognition classifiers with distributed systems, *Parallel and Distributed Systems ICPADS 2001, Proceedings of the Eighth International Conference on* pp. 379–382.
- Vazhkudai, S. S., Ma, X., Freeh, V. W., Strickland, J. W., Tammineedi, N. and Scott, S. L. (2005). Freeloader: Scavenging desktop storage resources for scientific data, *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, Washington, DC, USA, p. 56.
- Vesanto, J., Himberg, J., Alhoniemi, E. and Parhankangas, J. (2000). Self-organizing map in matlab: the som toolbox, *In Proceedings of the Matlab DSP Conference*, pp. 35–40.
- Vivanco, R. A., Demko, A. and Pizzi, N. J. (2005). Scopira: A pattern recognition application framework for biomedical datasets, *ICMLA '05: Proceedings of the Fourth International Conference on Machine Learning and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 165–170.

- von Laszewski, G., Foster, I. T. and Gawor, J. (2000). Cog kits: a bridge between commodity distributed computing and high-performance grids, *Java Grande*, pp. 97–106.
- von Laszewski, G., Gawor, J., Peña, C. J. and Foster, I. T. (2002). Infogram: A grid service that supports both information queries and job execution, *HPDC*, pp. 333–342.
- von Laszewski, G., Hategan, M. and Kodeboyina, D. (2006). Work coordination for grid computing, *Grid Technologies*, WIT Press, Southampton, UK.
- Wang, H., Zheng, H. and Azuaje, F. (2007). Poisson-based self-organizing feature maps and hierarchical clustering for serial analysis of gene expression data, *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **4**(2): 163–175.
- Wang, X., Ma, J., Wang, S. and Bi, D. (2007). Dynamic energy management with improved particle filter prediction in wireless sensor networks, *ICIC (1)*, pp. 251–262.
- Wilson, R. C. (2009). Parallel hopfield networks, *Neural Comput.* **21**(3): 831–850.
- Wilson, R. C., Hancock, E. R. and Luo, B. (2005). Pattern vectors from algebraic graph theory, *IEEE Transactions on Pattern Analysis and Machine Intelligence.* **27**(7): 1112–1124.
- Wu, S. and Chow, T. (2007). Self-organizing and self-evolving neurons: A new neural network for optimization, Vol. 18, pp. 385–396.
- Wu, S. and Wu, K.-L. (2009). An indexing framework for efficient retrieval on the cloud, *IEEE Data Eng. Bull.* **32**(1): 75–82.
- Wythoff, B. J. (1993). Backpropagation neural networks: A tutorial, *Chemometrics and Intelligent Laboratory Systems* **18**: 115–155.
- Yang, A., Jafari, R., Kuryloski, P., Iyengar, S., Sastry, S. S. and Bajcsy, R. (2007). Distributed segmentation and classification of human actions using a wearable motion sensor network, *Technical report*, Electrical Engineering and Computer Sciences, University of California at Berkeley. accessed on 17/03/2010.
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-143.html>

- Yang, F. and Paindavoine, M. (2003). Implementation of an rbf neural network on embedded systems: real-time face tracking and identity verification, *Neural Networks, IEEE Transactions on* **14**(5): 1162–1175.
- Yu, D., Ma, L. and Lu, H. (2007). Lottery digit recognition based on multi-features, *Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE*, pp. 1–4.
- Zaremba, M., St-Laurent, L., Niemann, O. and Richardson, D. (2000). Integration of self-organizing maps with spatial indexing for efficient processing of multi-dimensional data, *GIS '00: Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, ACM, New York, NY, USA, pp. 77–82.
- Zhao, Z.-Q., Huang, D.-S. and Sun, B.-Y. (2004). Human face recognition based on multi-features using neural networks committee, *Pattern Recognition Letters* **25**(12): 1351 – 1358.
- Zhu, H., Zhang, L., Sun, H. and Xiao, R. (2008). Face detection using multi-feature, *Advances in Cognitive Neurodynamics ICCN 2007*, Springer Netherlands, pp. 921–925.

Appendix A

Extended Analysis and Results

A.1 Greyscale Image Recognition

This section presents extended results obtained from the analysis on DHGN distributed pattern recognition implementation for heterogeneous greyscale images using binary signature for colour representation. Some of the results have been reported in Chapter 4.

A.1.1 The Dataset

A recognition test using DHGN scheme on 40 greyscale images from 5 different classes has been implemented. Each class represent similar images with random pixel noise distortions and rotations. Figure A.1 shows all the images used in this test.

Each image used is of size 512×512 with 256-greyscale levels. Figure A.2 shows histograms of all the images used in this test. These histograms indicate different greyscale levels between each image class, proving their heterogeneity. Note that with increasing level of noise introduced, the structure of each histogram also changes accordingly.

In this recognition test, 6 colour quantisation levels are considered, in creating patterns that represent the greyscale images using binary signature scheme as described in Section 3.2.2.

A.1.2 Extended Results

In this subsection, a set of extended results of the recognition test that has been performed are presented. The analysis on recognition accuracy for greyscale image recognition using DHGN takes two important parameters namely recall and error values. The results

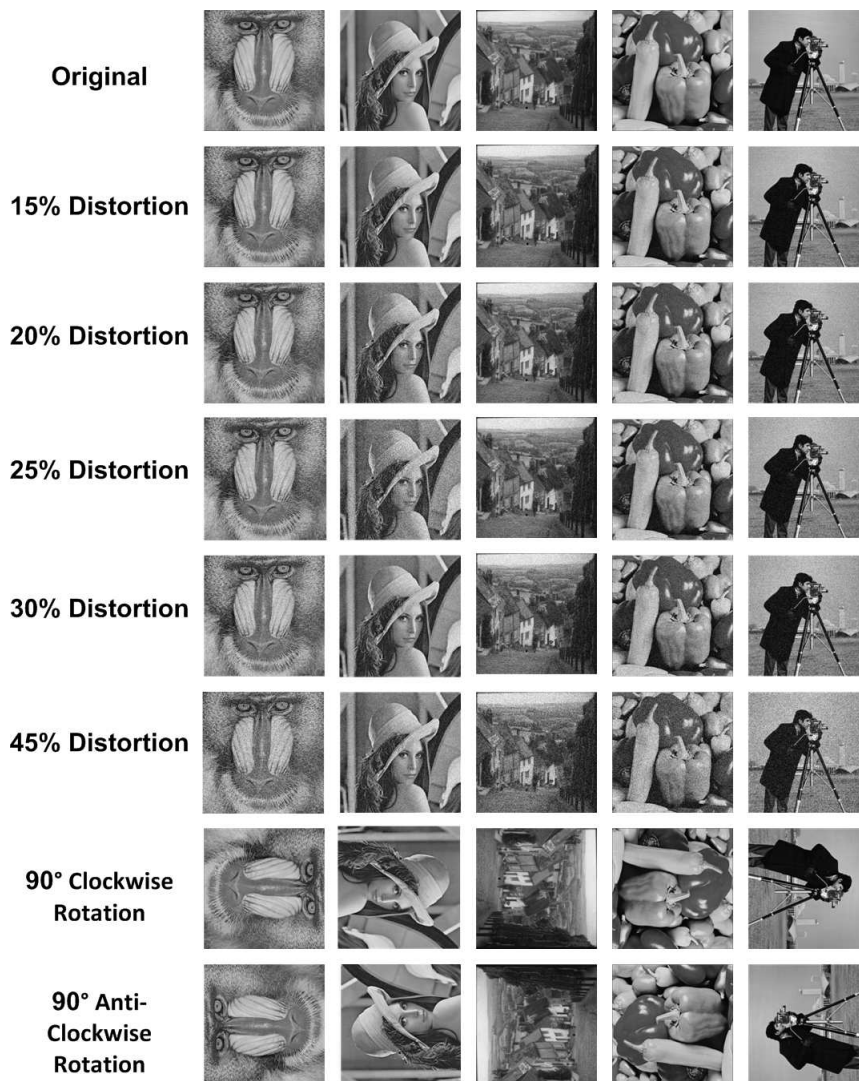


Figure A.1: 40 heterogeneous greyscale images used in DHGN image recognition test. These images belongs to 5 different classes, namely Babboon, Lena, Goldhill, Peppers, and Camera.

indicate that DHGN with binary signature scheme on different quantisation levels have produced high recall values. This is mainly due to DHGN's ability to recognise distinctive colour or greyscale level of different set of images. On the other hand, error values for DHGN scheme have been recorded considerably high. On average, the error value for DHGN recognition scheme on the greyscale images is 0.153. This has been mainly influenced by the use of global binary signature scheme that implements a single signature pattern to represent a particular image. This perhaps can be improved, using local binary signature scheme as described in Section 3.2.2.

Figure A.3 shows the error values obtained for each image class on different quantisation levels.

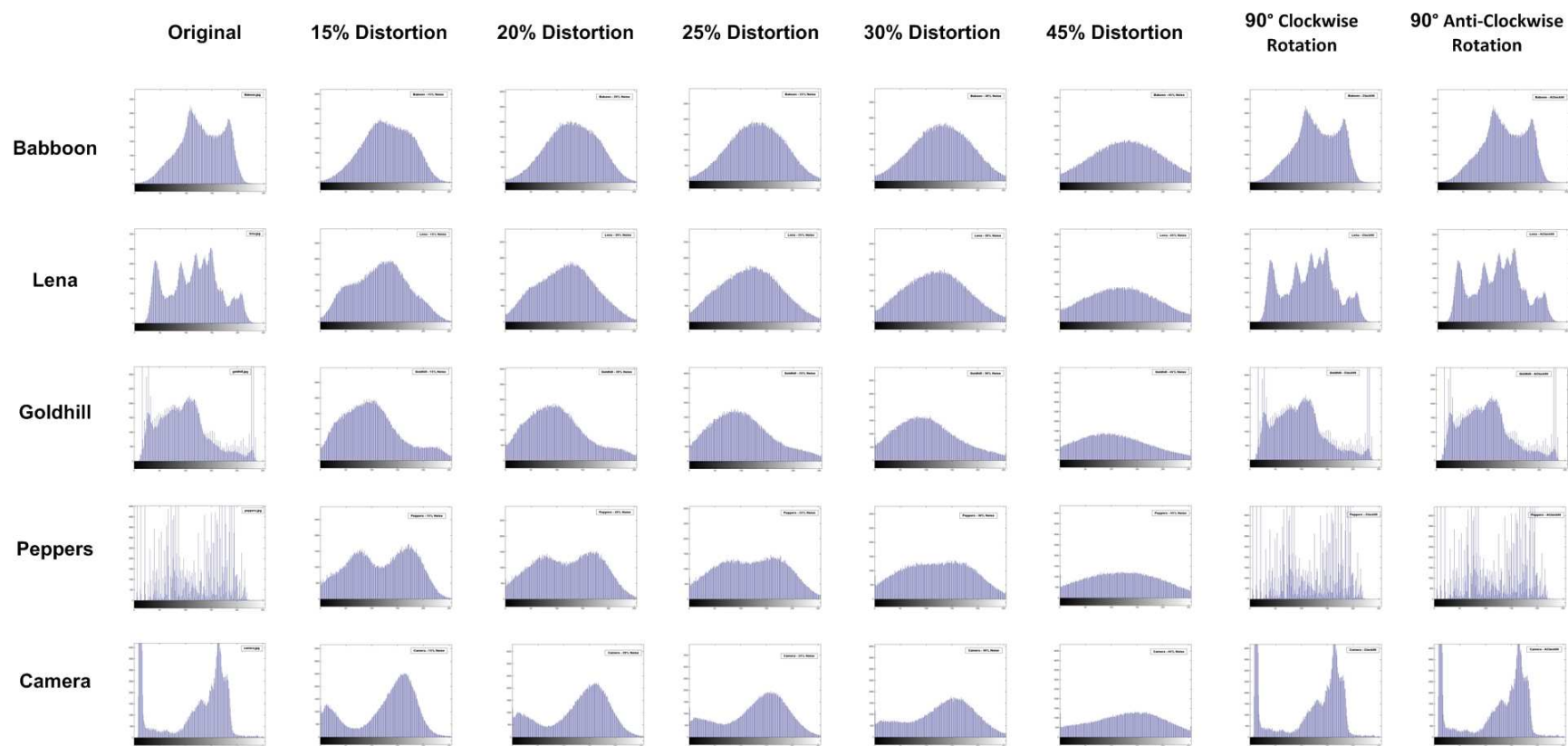


Figure A.2: Image histograms for 40 greyscale images used in DHGN image recognition test. Each image comprises 256-greyscale levels.

Note that in this test, DHGN performed recognition with the lowest error values for a quantisation value 5. A particular issue that needs to be solved in using this binary scheme is in finding the best representation for the image. Different quantisation level will affect the accuracy of the scheme, due to improper representation of patterns.

Figure A.4 shows the recall values derived from the image recognition test conducted.

From the recall analysis, we also discovered that DHGN has been able to produce high recall values. About 31% of the entire test patterns obtained perfect recall for different noise distortion levels and rotational directions in this recognition test.

A.2 Negative Image Detection using DHGN

Negative image contains the most contradictive pixel information to the original image. However, the structure of the image remains intact. Therefore, in pattern recognition context, we can deduce that negative image is a form of full spatial-distortion. This distortion is uniform, i.e. the pixel information changes uniformly across the entire image. Figure A.5 shows an example of binary character image with its negative representation.

This study has made observations of how DHGN networks are able to positively react to negative images. The results of the analysis have shown that smaller subnets, together with minimum voting scheme adopted by the SI module node, enables the network to give accurate responses to the negative images. The size of the input subpattern determines the capacity of spatial information each subnet holds. Thus, the capacity of spatial information of an image within a DHGN subnet is directly proportional to the size of input subpattern.

A.2.1 Subpattern Size Reduction

This study has discovered that a reduction in the size of input subpattern B_s will increase the recall accuracy of the DHGN network, in terms of its $V_{P_x}^{\max}$ or $V_{P_x}^{\min}$ values. DHGN enables recognition of negative images through the concept of spatial encapsulation. It involves the process of encapsulating the spatial information through a decomposition process. In this context, an image could be decomposed into subpatterns with various decomposition sizes. This work is focusing on uniform decomposition, i.e. decomposition of pattern into subpatterns of equal sizes. This process encapsulates spatial information of an image. Large encapsulation usually reduces the detailed characteristics of an image.

This is due to the fact that large encapsulation collectively holds more information as compared to small encapsulation.

Given a binary pattern $P_k = (011011010)$ and its distorted pattern $\tilde{P}_k = (011011011)$. If DHGN subnet with an input subpattern size $B_s^k = 9$ is used, thus no decomposition is made. The output of the recognition process using maximum voting will results in a new pattern being detected, since $V_{P_k}^{\max} = 0$. However, if we decompose the patterns into equal-sized subpatterns $B_s^k = 3$, thus we have three DHGN subnets working on the subpatterns:

$$\begin{aligned} P_k^1 &= (011) & \tilde{P}_k^1 &= (011) \\ P_k^2 &= (011) & \tilde{P}_k^2 &= (011) \\ P_k^3 &= (010) & \tilde{P}_k^3 &= (011) \end{aligned}$$

The output of recognition process using maximum voting will results in a recall of pattern P_k . This is due to the responses obtained from all DHGN subnets that produces $V_{P_k}^{\max} = P_k$ (two subnets give recall responses to the distorted input, while a subnet recorded a new subpattern).

This scenario explains the encapsulation effect in DHGN pattern recognition scheme. The distortion in this example is well-encapsulated within a subpattern and thus, improving the recall accuracy of the entire DHGN network. This encapsulation effect also plays similar role in minimum voting scheme as described previously.

Proof: A reduction in the size of input subpattern B_P of pattern P will possibly increase the number of votes (maximum or minimum) V_P , given that $0 \leq V_P \leq n$, where n represents the number of subnets used.

$$B_P = \frac{P}{n}, \quad \text{Given } \exists B_P \in \mathbb{N}: B_P \text{ is odd}$$

$$n = \frac{P}{B_P} \tag{A.1}$$

$$\text{Given } 0 \leq V_P \leq n, \quad \therefore 0 \leq V_P \leq B_P$$

As the value of B_P decreasing, therefore the possible values of V_P will increase.

A.2.2 Negative Image Recognition

Apart from the previous discovery, we have also able to show that given a negative image $\overline{I_{bin}}$ of original binary image I_{bin} , the amount of vote obtained from the recognition process using minimum voting will always be zero.

With an ability to provide a bird's eye view of the entire binary image, the DHGN network would be able to detect total changes in the pixel values within the image. Therefore, changes from original image to negative image will leads to massive changes in image's pixel information. Thus, none of the subnets are able to recall the original image. However, this could be taken as full recall, when the voting mechanism is switched between maximum and minimum. Figure A.6 shows a recognition result for a set of binary character images and the minimum voting value derived from all subnets.

Note that all subnets will only respond with zero minimum voting value, given a negative of image 'E'. This condition however will change with an increase in the size of input subpattern, as shown in Figure A.7.

The main justification of this effect is such that with an increase in the size of input subpattern, DHGN network is not able to detect the spatial structure of the image. Small encapsulation enables the localisation of spatial structure within the image and hence, able to be memorised by the network. Further description of this localisation effect can be referred to in Section 2.7.

A.2.3 Recognition Tests and Results

This research has conducted a series of recognition tests involving binary character images using min-max DHGN pattern recognition scheme. Four different sets of 100 x 100 pixels greyscale character images (26 characters from A to Z) were used as shown in Figure A.8.

The first set of images act as stored patterns while the rest of the images act as the test patterns. Three different recognition tests were conducted using different input subpattern sizes, namely 5-, 7-, and 9-bits, and subsequent simulation tools have been developed using C programming language with Message Passing Interface (MPI) capabilities for parallel processing using MPICH2 toolkit (Gropp et al., 1999). The following subsections describe in details of the results of these recognition tests.

9-bits Input Subpatterns

Table A.1 shows the results of the recognition test using 9-bits input subpatterns. In this test, 1111 subnets were used for 100 x 100 pixels greyscale homogeneous character images. Note that for Dataset I, the result of the maximum voting gives a perfect recall for the entire set of character images. Dataset II and III are negative images of the stored character images, as outcomes of the image translation process. The outputs from DHGN network have shown that minimum voting is able to provide perfect recall for almost entire images in those datasets. There are few exceptions including letters E, F, and L. The main reason for this is that with 9-bits input subpattern representation, the encapsulation effect is not able to encapsulate the spatial structure of the character images of those letters. Therefore, all the three letters will respond with minimum voting. The same situation also applied to letter O and Q.

		Patterns																									
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dataset I	max	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	min	W	W	W	W	W	W	W	Q	W	Q	W	W	W	W	W	W	W	W	W	W	W	Q	Q	G	Q	W
Dataset II	max	W	W	W	T	W	W	W	T	W	M	W	W	W	W	M	W	M	M	M	J	M	M	M	M	M	
	min	A	B	C	D	E,F,L	E,F,L	G	H	I	J	K	E,F,L	M	N	O,Q	P	O,Q	R	S	T	U	V	W	X	Y	Z
Dataset III	max	W	M	W	T	W	W	W	T	W	M	W	W	W	W	M	W	M	M	M	J	M	M	M	M	M	
	min	A	B	C	D	E,F,L	E,F,L	G	H	I	J	K	E,F,L	M	N	O,Q	P	O,Q	R	S	T	U	V	W	X	Y	Z

Table A.1: Results of image recognition test using DHGN recognition scheme with 9-bits input subpatterns.

7-bits Input Subpatterns

With a reduction in the size of input subpattern, the results of the recognition test has been improved significantly using both maximum and minimum voting schemes for Datasets I, II, and III, as shown in Table A.2.

		Patterns																									
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dataset I	max	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	min	W	W	W	W	W	W	W	Q	W	Q	G	W	W	W	W	W	W	W	W	W	W	Q	Q	G	Q	W
Dataset II	max	W	W	W	M	W	W	W	W	M	W	W	W	W	W	M	W	W	W	M	J	M	M	M	M	M	
	min	A	B	C	D	E,F,L	E,F,L	G	H	I	J	K	E,F,L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dataset III	max	W	W	W	M	W	W	W	T	W	M	W	W	W	W	M	W	W	W	M	J	M	M	M	M	M	
	min	A	B	C	D	E,F,L	E,F,L	G	H	I	J	K	E,F,L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Table A.2: Results of image recognition test using DHGN recognition scheme with 7-bits input subpatterns. There is an improvement on the accuracy of this DHGN scheme.

In this test, 1428 subnets were used for similar datasets as described earlier. Note that there is an improvement in recall outputs for Datasets II and III. The DHGN network is able to recognise and differentiate character images O and Q. The reason for this is

that with a reduction in the size of subpattern, DHGN network is able to encapsulate the spatial structure of the character images more precisely. Nevertheless, this network is still unable to recognise and differentiate character images E, F, and L.

5-bits Input Subpatterns

This recognition test involves 2000 subnets. The results obtained from this test show that DHGN network is able to fully recall all the images in Datasets I, II, and III, as shown in Table A.3.

		Patterns																									
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dataset I	max	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	min	W	W	W	W	W	W	W	Q	W	Q	G	W	W	W	W	W	W	W	W	W	W	Q	Q	G	Q	W
Dataset II	max	W	W	W	M	W	W	W	W	M	W	W	W	W	W	M	W	W	W	M	W	M	M	W	M	M	
	min	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dataset III	max	W	W	W	M	W	W	W	T	W	M	W	W	W	W	M	W	W	W	M	W	M	M	W	M	M	
	min	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Table A.3: Results of image recognition test using a recognition scheme with 5-bits input subpatterns. DHGN is able to recall all the characters using minimum voting scheme on both dataset II and III.

A.2.4 Discussions

The results of recognition tests conducted have shown an interesting phenomenon for both positive and negative image recognition. DHGN produced perfect recalls for all the characters in Dataset I using maximum voting scheme, regardless of its configuration, i.e. size of the input subpattern used. This outcome reveals the capability of DHGN max-voting mechanism to recall all the test character images that contain similar colour proportions as the stored images. Furthermore, with voting technique being applied, the approximation process to find the nearest matched images is possible. Figure A.9 shows character-to-maximum vote ratios for selected character images in Dataset I that has been collected using DHGN-9 (9-bit input subpattern) scheme.

On negative image recognition perspective, DHGN has the ability to recognise character images with negative colour. Nevertheless, its ability is significantly improved with an increase in the amount of subnets involved during computation, i.e. smaller input subpattern being used. The character-to-maximum vote ratios for negative characters A, E, F, L, O, and Q in Dataset III that has been used in all DHGN configurations are shown in Figure A.10.

These tests reveal that DHGN is able to provide accurate recall for both positive and negative images, given a condition that the size of input subpattern is able to withhold the spatial-structure information of the images. However, it is best noted that there is also a significant increase in the number of DHGN subnets being used.

The results obtained from these series of recognition tests have revealed two important factors in the performance of DHGN network for image recognition. These are the number of DHGN subnets being used and the recall accuracy of DHGN network. To get high recall accuracy for binary character images, more DHGN subnets are required as a direct result in the reduction of the size of input subpatterns. Figure A.11 shows an approximate comparison between recall accuracy for the binary character images against the number of DHGN being used in the tests.

Note that the slopes of both graphs are similar. Thus, it could be deduced that the reduction in recall accuracy is directly proportional to a decrease in the number of subnets available for the network. However, in real implementation, a single subnet may be used for multiple sets of input subpatterns, without having to allocate a unique subnet for each input subpattern. This is an idea for recyclable DHGN subnets. Further research work will be carried out to ensure the outgrowth of these subnets does not significantly affects the recall accuracy of the overall network.

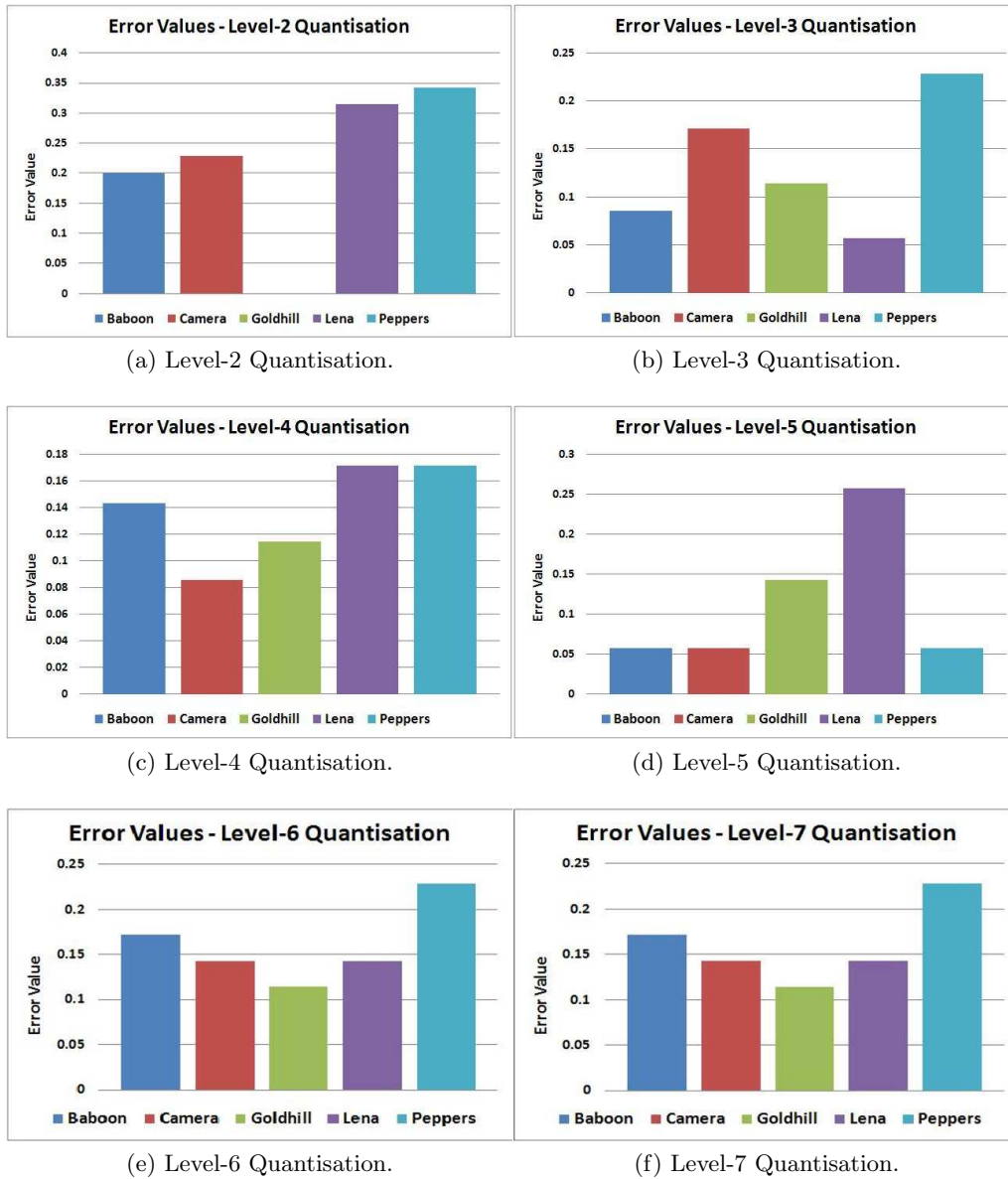


Figure A.3: Error values obtained for each quantisation level from recognition test on 5 classes of greyscale images with noise distortion and rotation.

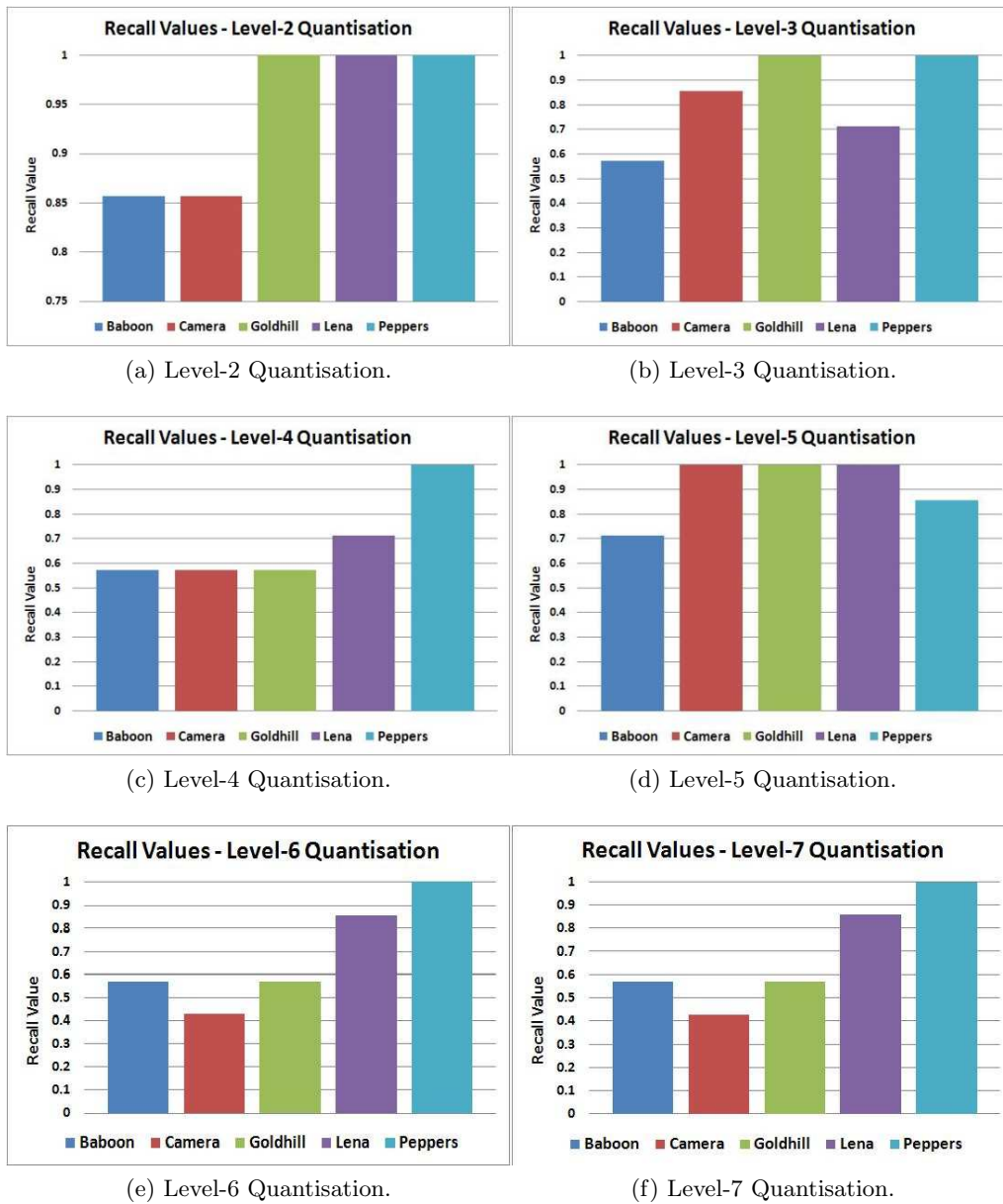


Figure A.4: Recall values obtained for each quantisation level from recognition test on 5 classes of greyscale images with noise distortion and rotation.

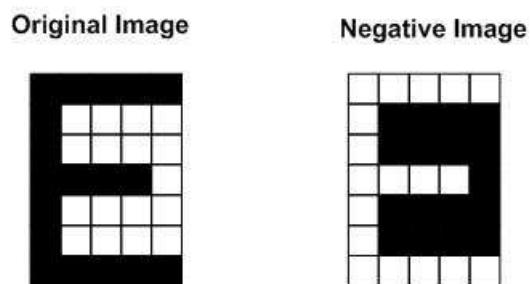


Figure A.5: Binary character image 'E' with its negative.

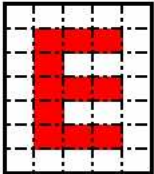
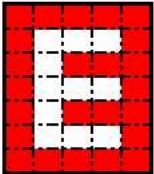
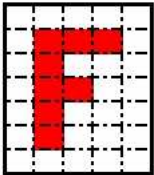

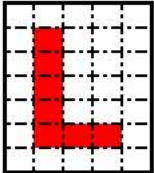
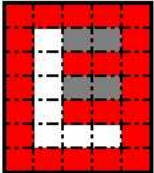
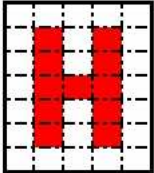
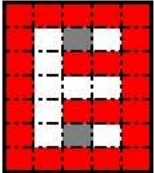
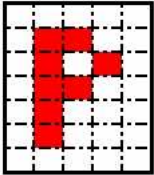
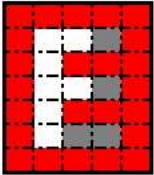
Stored Pattern	Test Pattern	Similarities
		0
		3
		4
		2
		4

Figure A.6: Recognition results using DHGN image recognition scheme. The similarities represent the minimum voting obtained from all subnets. The grey areas within the test patterns show the pixel value of test pattern that is similar to stored pattern.

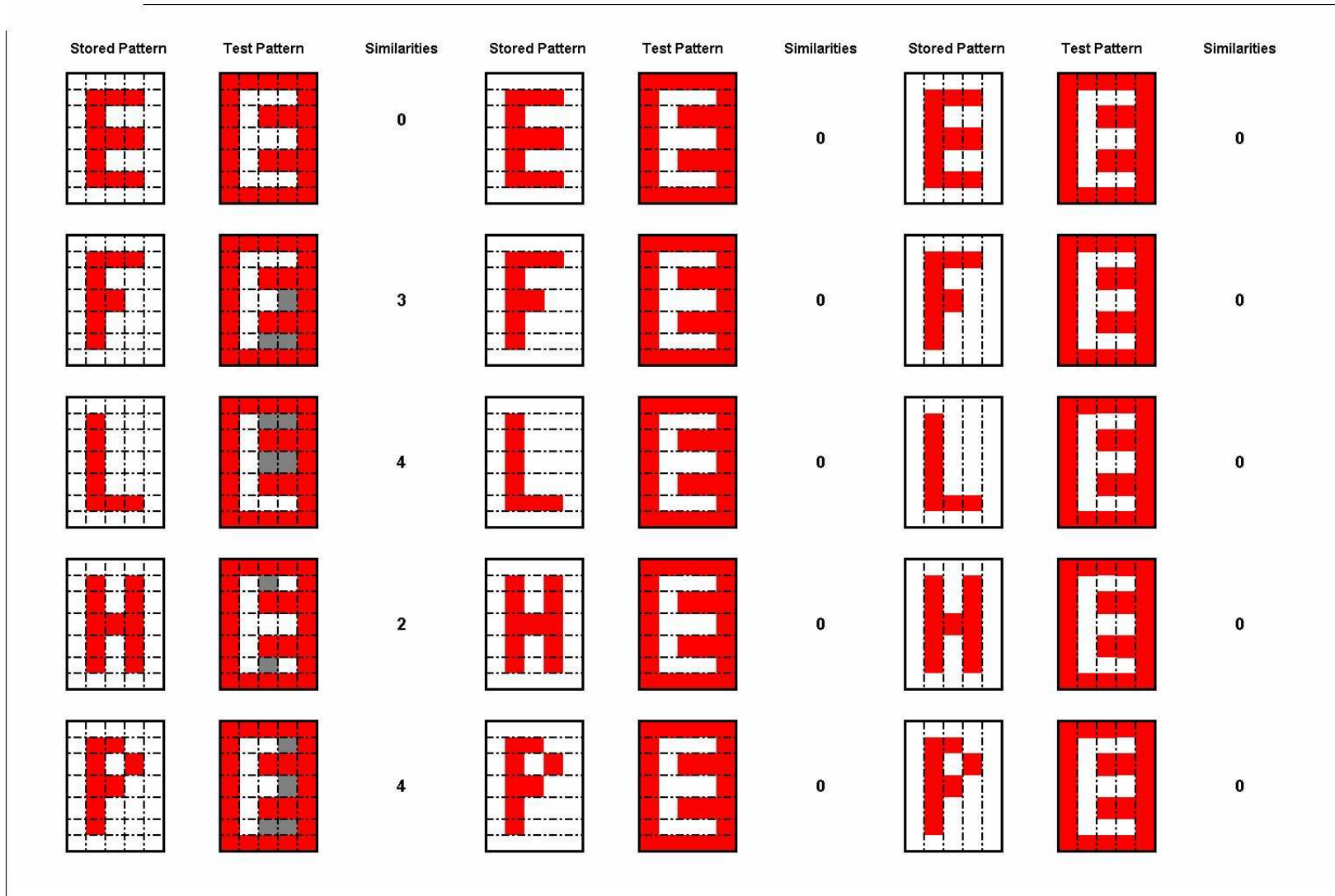


Figure A.7: Extended recognition results showing the effect of input subpattern size to the minimum voting scheme.

Stored Patterns	Dataset I	Dataset II	Dataset III
A N	A N	A N	A N
B O	B O	B O	B O
C P	C P	C P	C P
D Q	D Q	D Q	D Q
E R	E R	E R	E R
F S	F S	F S	F S
G T	G T	G T	G T
H U	H U	H U	H U
I V	I V	I V	I V
J W	J W	J W	J W
K X	K X	K X	K X
L Y	L Y	L Y	L Y
M Z	M Z	M Z	M Z

Figure A.8: Datasets used in DHGN image recognition tests.

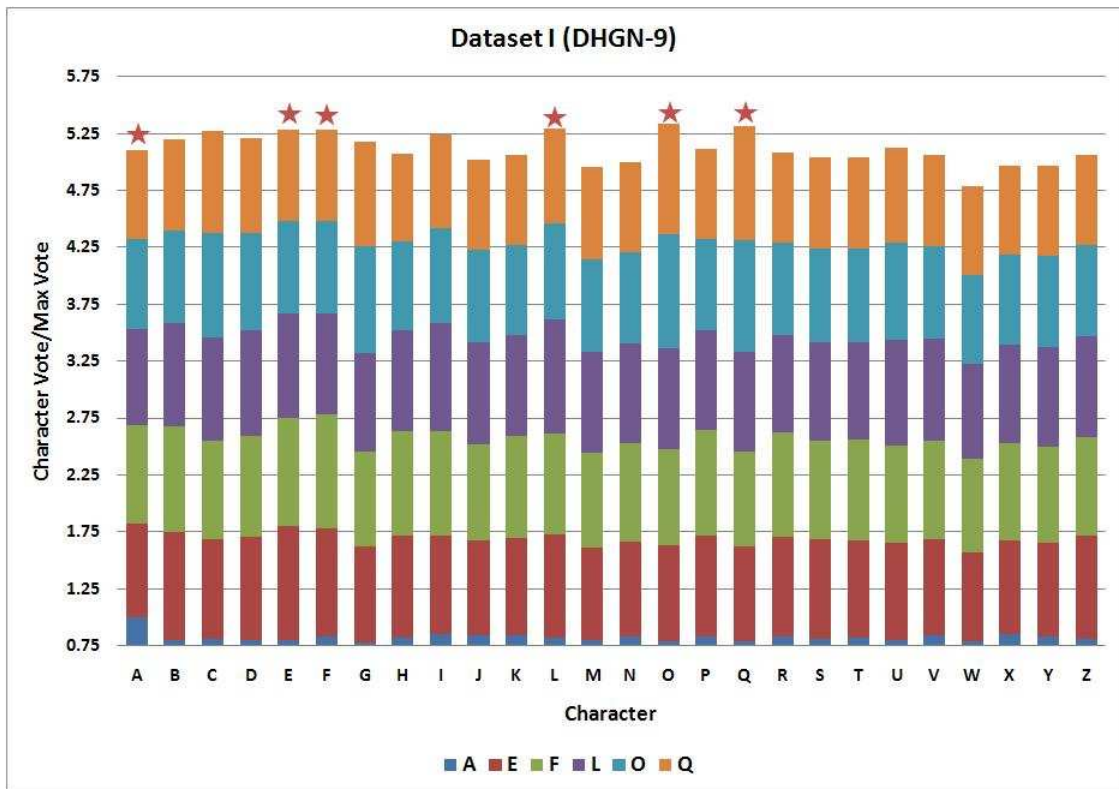
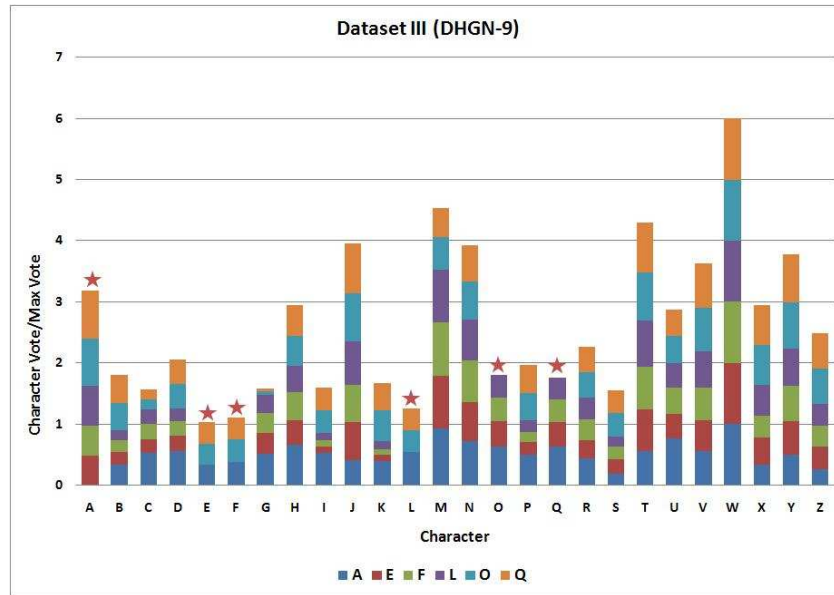
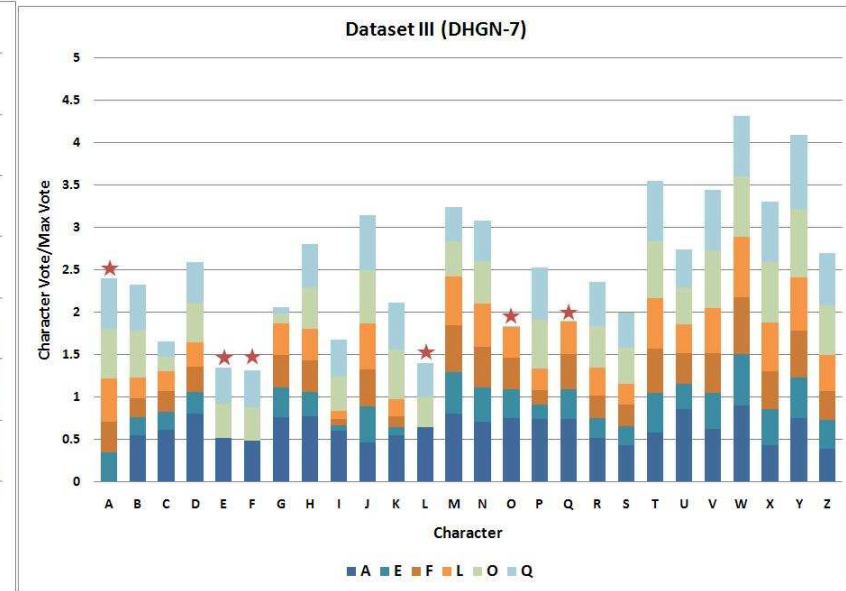


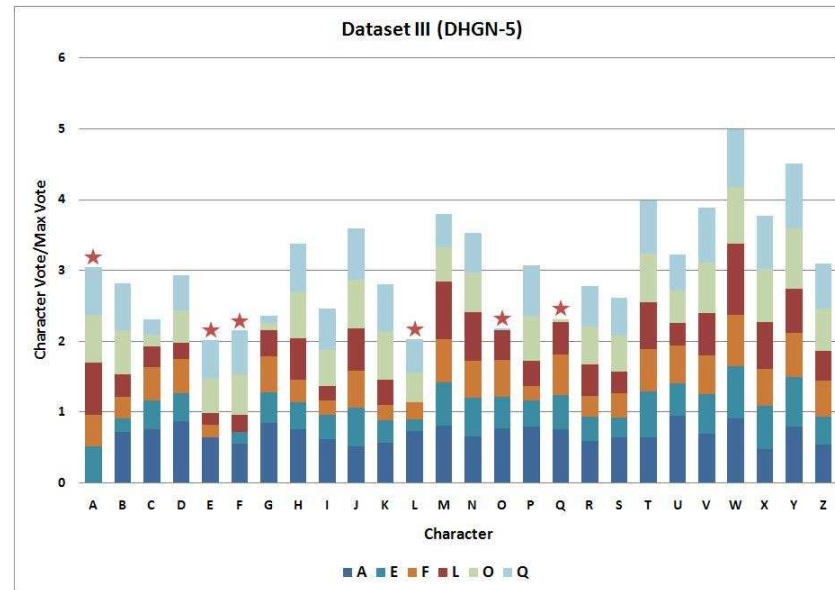
Figure A.9: Character-to-maximum vote ratio for characters A, E, F, L, O, and Q in Dataset I retrieved using DHGN-9 recognition scheme. Note that the star represent column for respective test character.



(a) DHGN-9.



(b) DHGN-7.



(c) DHGN-5.

Figure A.10: Character-to-maximum vote ratio for characters A, E, F, L, O, and Q in Dataset III retrieved using different DHGN input subpattern size. Note that the star represents column for respective test character.

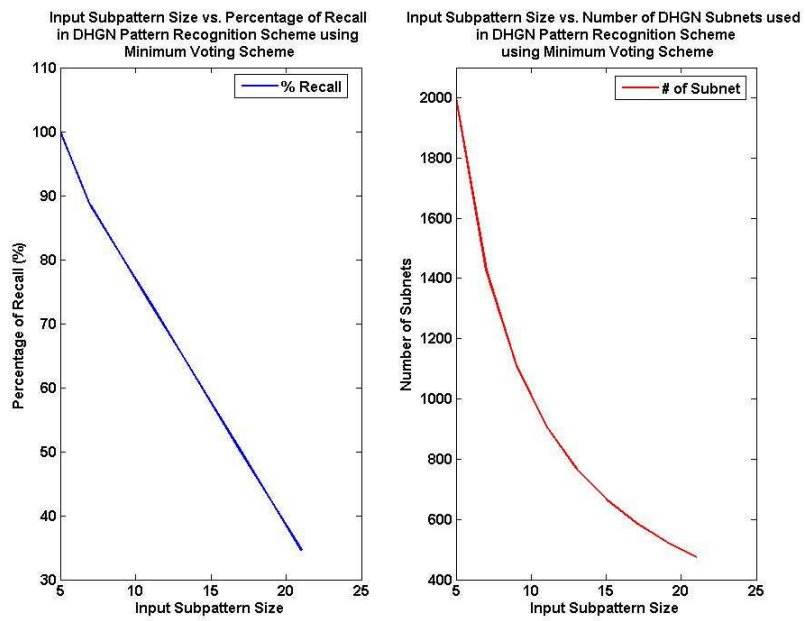


Figure A.11: Comparison between percentage of recall and number of subnets against the input subpattern size in DHGN pattern recognition with minimum voting scheme.

Appendix B

Algorithms and Pseudocodes

B.1 Single-Value (Original) DHGN

The original DHGN implementation as described in Chapter 4 comprises 4 important functions. These are SI Module, voting, adjacency comparison, and bias calculation. Figure B.1 shows a context diagram for the functions within the DHGN architecture. The pseudocode for each function is listed in this Appendix.

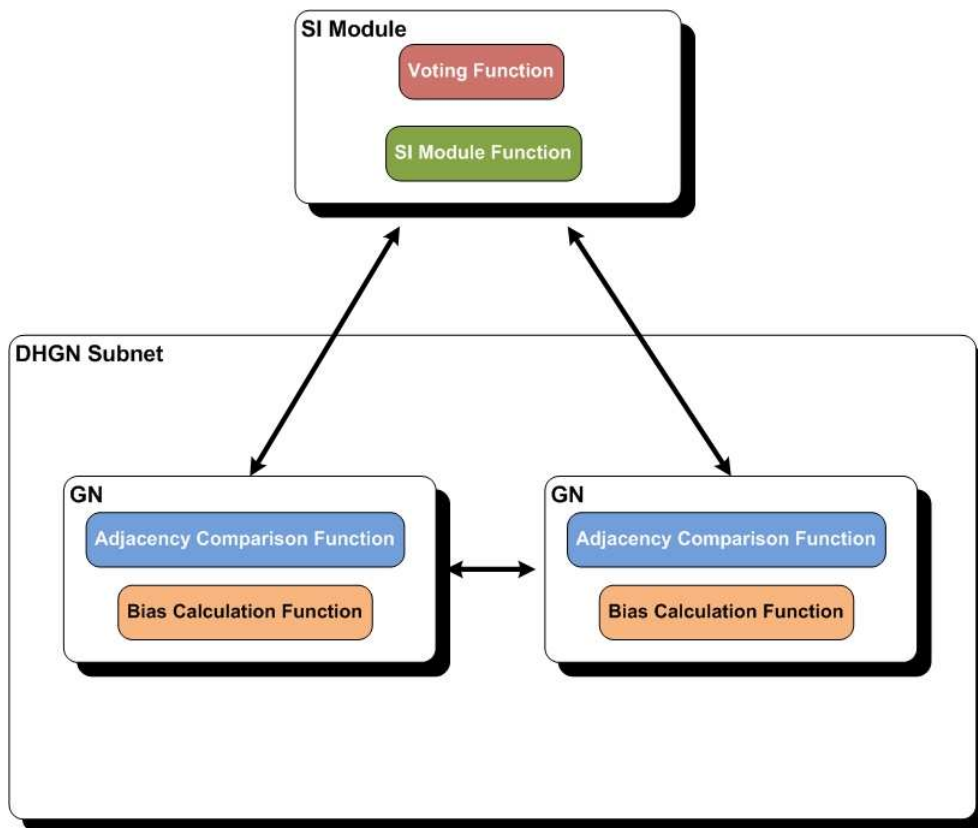


Figure B.1: Context diagram showing important functions within DHGN implementation.

The SI Module function deals with how patterns are communicated from the SI node to all other GN nodes within all the subnets. Three distinguished commands have been used, namely *init*, *store*, and *abort*. These represent initialisation, recall/store, and abort processes respectively. This function communicates directly with each subnet, via the base-layer GNs.

Algorithm 8 SI Module Function

Require: $GN_{top} = 2 \times \left(\frac{size+1}{2}\right)^2$ {Determine the top GN id from subpattern size}

- 1: **repeat**
- 2: read *input* from *file*
- 3: broadcast *input* to all *GN*
- 4: **if** *command.input* equals *command.init* **then**
- 5: get next *input* from *file*
- 6: **else if** *command.input* equals *command.store* **then**
- 7: **for** $i = 1$ to GN_{top} **do**
- 8: get *msg* from $GN(i)$ listen for MPI messages from each GN
- 9: **if** $i == GN_{top}$ **then**
- 10: $idx.input = msg.GN_{top}$
- 11: **return** *idx.input*
- 12: **end if**
- 13: **end for**
- 14: invoke VOTING function
- 15: get next *input* from *file*
- 16: **else if** *command.input* equals *command.abort* **then**
- 17: abort
- 18: **end if**
- 19: **until** end of *file*

The Voting function implements voting procedure for results of the recognition performed at subpattern level. Each subnet will communicate each index retrieved (*idx*) to SI Module node. SI Module node will then perform this function.

Adjacency comparison function involves a process of communicating entries between adjacent GNs within each DHGN subnet. Once a particular GN is activated, it will perform this function to obtain subpattern entries from adjacent nodes. This function produces bias entries for Bias calculation function. Note that in this pseudocode, we implemented binary pattern recognition, in which each layer within DHGN subnet consists of two levels of GN. Level 0 corresponds to value 0 while Level 1 reflects value 1 in binary patterns.

Bias calculation function performs bias matching process within the bias array structure of each GN node. Each bias entry received will be matched with stored entries. If entry has been found, then the index will be recalled. Otherwise, new index will be generated.

Algorithm 9 Voting Function

```

1: for  $i = 0$  to  $max.pattern$  do
2:   for  $j = 0$  to  $max.subnet$  do
3:     get  $idx[i][j]$ 
4:   end for
5: end for
6: for  $i = max.storage$  to  $max.pattern$  do
7:   for  $j = 0$  to  $max.subnet$  do
8:     for  $k = 0$  to  $max.storage$  do
9:       if  $idx[i][j] == idx[k][j]$  then
10:         $idx[i].count[k] ++$ 
11:       end if
12:     end for
13:   end for
14: end for
15: for  $i = max.storage$  to  $max.pattern$  do
16:   for  $j = 0$  to  $max.storage$  do
17:     find max  $idx[i].count[k]$ 
18:     return max  $idx[i].count[k]$ 
19:   end for
20: end for

```

Algorithm 10 Adjacency Comparison Function (Base Layer GNs)

Require: $active.GN = true$ and $size = subpatternlength$

```

1: check  $GN$  position within subnet
2: if  $edge.GN == left$  then
3:   send  $idx.bias.GN(this)$  to  $GN + 1$  and  $GN + (size + 1)$ 
4:   receive  $idx.bias.GN$  from either  $GN + 1$  or  $GN + (size + 1)$ 
5: else if  $edge.GN == right$  then
6:   send  $idx.bias.GN(this)$  to  $GN - 1$  and  $GN + (size - 1)$ 
7:   receive  $idx.bias.GN$  from either  $GN - 1$  or  $GN + (size - 1)$ 
8: else if  $edge.GN == middle$  then
9:   send  $idx.bias.GN(this)$  to  $GN + 1$  and  $GN + (size + 1)$ 
10:  send  $idx.bias.GN(this)$  to  $GN - 1$  and  $GN + (size - 1)$ 
11:  receive  $idx.bias.GN$  from either  $GN + 1$  or  $GN + (size + 1)$ 
12:  receive  $idx.bias.GN$  from either  $GN - 1$  or  $GN + (size - 1)$ 
13: end if

```

Algorithm 11 Bias Calculation Function

```

1: receive  $entry.pattern$  from ADJACENCY COMPARISON function
2: for all  $test.pattern$  in  $pattern$  do
3:   for  $i = 0$  to  $MAXBIAS$  do
4:     if  $entry.pattern == element.bias[i].GN$  then
5:        $idx.pattern.GN = i$ 
6:       exit FOR
7:     else
8:        $idx.pattern.GN = MAXBIAS + 1$ 
9:     end if
10:  end for
11:  return  $idx.pattern.GN$ 
12: end for

```

B.2 Multi-Value DHGN

In terms of the modular functions, the multi-value DHGN implementation is very similar to the original DHGN implementation. However, the adjacency comparison function only requires communication of entries between GNs within a single level per layer as shown in the following pseudocode:

Algorithm 12 Adjacency Comparison Function (Base Layer GNs) for Multi-Value DHGN

Require: $active.GN = true$ and $size = subpatternlength$

```

1: check  $GN$  position within subnet
2: if  $edge.GN == left$  then
3:   send  $idx.bias.GN(this)$  to  $GN + 1$ 
4:   receive  $idx.bias.GN$  from  $GN + 1$ 
5: else if  $edge.GN == right$  then
6:   send  $idx.bias.GN(this)$  to  $GN - 1$ 
7:   receive  $idx.bias.GN$  from  $GN - 1$ 
8: else if  $edge.GN == middle$  then
9:   send  $idx.bias.GN(this)$  to  $GN + 1$ 
10:  send  $idx.bias.GN(this)$  to  $GN - 1$ 
11:  receive  $idx.bias.GN$  from  $GN + 1$ 
12:  receive  $idx.bias.GN$  from  $GN - 1$ 
13: end if

```

The rests of the modular functions are similar to the single-value DHGN functions described in previous section.