

ADDENDUM

P 11 Section (1.3): Add at the end of point (i):

“A real time application is the application that requires converging to a decision within a predetermined time limits”

P 14 section (1.4) para 2, first sentence: delete “a novel pattern detection scheme for WSNs” and read “a novel pattern detection scheme for WSNs called Cellular Graph Neuron (CGN)”

P 15 line 1: delete “The scheme reduces” and read “The scheme is called Cellular weighted Graph Neuron (CwGN) and it reduces”

P 34 section (2.3.3), 10th line: delete “the states are rectangle or circle” and read “the states can be selected as rectangle and circle”

P 35 para 1, 9th line, delete “ i is the class number,”

P 45 2nd line: delete “The neurons in the comparison layer are fed” and read “Each neuron in the comparison layer is fed”

P 57 para 1, 9th line: delete “increases exponentially” and read “increases in a quadratic manner”

P 61 para 1, 1st line: delete “propos” and read “proposed”

P 63, 8th line: delete “The memory size of a sensor is intuitively small” and read “The available memory in each node is small”

P 65 para 1, 11th line: delete “ability to address randomness problems” and read “ability to deal with problems that involve random occurrence of events and patterns”

P 66 para 1, 3rd line: delete “S” and read “ S ”

P 66 para 1, 7th line: delete “ v ” and read “ v ”

P 66 para 1, 8th line: delete “grows exponentially” and read “grows in a quadratic manner”

P 70 para 1, 6th line: delete “if we are to the problem of interest” and read “if we are to solve the problem of interest”

P 78 section (3.2) para 2, 4th line: delete “attempting to achieve” and read “attempting to achieve are”

P 78 section (3.2) para 2, 4th line: delete “low scheme complexity” and read “low time complexity”

P 80 Equation (3.1): delete “ $\varepsilon \in V$ ” and read “ $\varepsilon_i \in V$ ”

P 80 Definition (3.2): Add to the end of the definition “In other words, an index i is a unique number that describes P_i ”

P 81 Section (3.2.2), 1st line: delete “The CGN network consists of a set of GN networks where each GN network reports to another one with reaching the S&I.” and

read “The CGN network is composite of multiple GN networks. Each GN network performs a set of computations and report its outcomes to another pre-assigned GN network. This operation continues until delivering all computations to the S&I.”

P 81 Definition (3.2), 3rd line: delete “ x ” and read “ x_i ”

P 81 Definition (3.2), 4th line: delete “ $a, v \in V$ ” and read “ $a_i, v \in V$ ”

P 81 Definition (3.2), 6th line: delete “ $\varepsilon \in V$ ” and read “ $\varepsilon_i \in V$ ”

P 85 para 1, 7thline: delete “ $i=1$ ” and read “ $i=1$ ”

P 86 Definition (3.6), 7th line: delete “ j ” and read “ l ”

P 88 section (3.2.4), 3rd line: delete “patter” and read “pattern”

P 98 Proposition (3.1): delete “ $\varepsilon \in V$ ” and read “ $\varepsilon_i \in V$ ”

P 106 para 2, 2nd line: delete “increases exponentially” and read “increases in a quadratic manner”

P 114 Figure (3.15), Y axis: delete “Accurcy” and read “Accuracy”

P 115 para 1, 6th line: add before “Additionally,”:

“The analysis of the scheme shows that it is capable of performing recognition tasks within a predictable single learning cycle that suits real time applications.”

P 125 Figure 4.1, legend: delete “CWGN” and read “CwGN”

P 127 Definition (4.2), 4th line: delete “as follows.” and read “. If $\Delta\omega_{iC} = |\omega_i - \omega_C|$, and $j = \min\{\Delta\omega_{iC}, 1 \leq i \leq n\}$. Then, $R_p=j$. This can be also represented using the following expression.”

P130 para 1, 2nd line: delete “odd numbers” and read “odd number”

P 131 Definition (4.4): Comment: The (\rightarrow) notation means communicate (or send).

P 131 Figure 4.3, legend, 2nd line: delete “(L_1, L_2, \dots, L_m)” and read “(L_1, L_2, \dots, L_{2m})”

P 150: delete Equation (4.16) and read

$$“T_{total} = T_{send} + T_{sense} + 2.T_{send} + 2.T_{compare} + T_{add} + T_{dev} + T_{send} \cdot (10^{\frac{\log S}{2}} - 1) + T_{add} \cdot 10^{\frac{\log S}{2}} + T_{SI}”$$

P 150: delete Equation (4.17) and read

$$“T_{total} = \left(2 + 10^{\frac{\log S}{2}}\right) \cdot T_{send} + (1 + T_{add}) \cdot 10^{\frac{\log S}{2}} + T_{sense} + 2.T_{compare} + T_{dev} + T_{SI}”$$

P 150: delete Equation (4.18) and read “ $T_{total} = 3.T1 + 5T2 + (T1 + T2) \cdot 10^{\frac{\log S}{2}}$ ”

P 150: delete Equation (4.19) and read “ $T_{total} = 3.T1 + 4T2 + (T1 + T2) \cdot 10^{\frac{\log S}{2}} + (2.T2)\log_2(Mp)$ ”

P 154: delete Equation (4.22) and read “ $N_{zones} = S \cdot \left(\frac{\tau - 3.T1 - 4.T2 - (2.T2)\log_2(Mp)}{T1 + T2}\right)^{-2}$ ”

P 164 para 2, 2nd line: delete “firstly” and read “first”

P 202 para 1, 15th line: add before “However,”:

“The results show the ability of the CwGN scheme of recognizing transformed patterns such as translated and dilated hill and valley patterns with higher detection accuracy levels compared to iconic methods.”

P 209, 7th line: add before “This means”:

“Also, this shows that the scheme is capable of mapping sensory information into patterns and use the mapped patterns to solve complex decision making problems such as the wall following robot problem.”

P 212, 7th line: Comment: The threshold 200 was empirically selected based on a series of experiments to ensure that high peaks of the contour images are marked in the binary patterns.

P 214 para 2, 8thline: delete “mS” and read “ms”

P 232 para 2, 8th line: delete “have been chose” and read “have been chosen”

P 242 Definition (6.1), 2nd line: delete “of the system” and read “of the GA system”

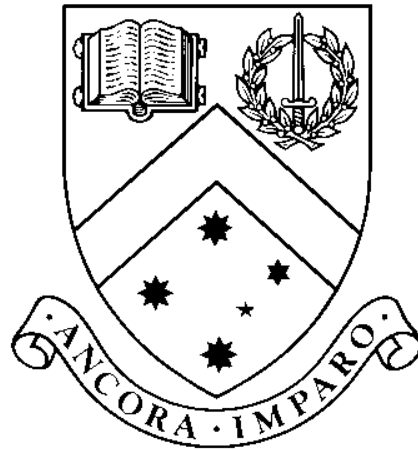
P 242 Definition (6.1), 5th line: delete “ $d(M_i) = \min\{(m_j, OP): m_j \in M_i\}$ ” and read “ $d(M_i) = \min\{d(m_j, OP): m_j \in M_i\}$ ”

Distributed Pattern Recognition Schemes for Wireless Sensor Networks

by

Waleed Mohammed J. Alfehaid

BSc of Computer Engineering (King Saud University)
MSc of Information Technology (Monash University)



Thesis

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy (0190)

**Faculty of Information Technology
Monash University**

November, 2013

©Copyright

Waleed Mohammed J. Alfahaid

2013

Under the Copyright Act 1968, this thesis must be used only under the normal conditions of scholarly fair dealing. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.

Abstract

Smart systems are increasingly in use in daily life applications, replacing old-fashioned processes and procedures as a result of technological evolution. However, these systems can be limited in their resources capacity. Wireless Sensor Networks (WSNs) are considered to be one form of such resource-constrained smart systems. One of the main goals of WSNs is to sense physical activities so as to detect events in an area of interest. Adaptive and machine learning techniques have been proposed and implemented to work in conjunction with WSNs to serve a number of applications, such as physical activities detection, network security threats detection, artificial intelligence applications and decision making support. Pattern recognition is one of the most useful machine learning techniques that can perform event detection for WSNs. However, the nature of WSNs poses extreme challenges for the implementation of these learning techniques so that they can serve the goals of different types of applications. Such networks have limited resources available for performing learning operations. Additionally, WSNs are of a dynamic nature in terms of network deployment and the appearance of activities in the field of interest. Global events can also span very large regions requiring vast quantities of data exchange and processing in order to detect such events. These challenges become critical when detection time limits are required by applications such as mission critical and online applications.

The aim of this research project is to propose pattern recognition schemes that are capable of addressing the limitations associated with resource-constrained networks such as WSNs. The research first investigates the existing learning techniques for WSNs and their limitations. Then the research proposes novel collaborative in-network global pattern recognition based event detection schemes that are light-weight, scalable and suit resource-constrained networks such as WSNs well. The proposed schemes address the limitations and challenges for WSNs to provide reasonable detection capabilities for mission critical, online, and decision making applications. The proposed schemes adopt the distributed and parallel recognition mechanisms of Graph Neuron (GN) in order to minimise recognition computations and communications and thus will lead to maintaining low levels of limited resources consumption. The distributed network structure of the proposed schemes will result in loosely coupled connectivity between a network's nodes and avoid iterative learning. Hence, the proposed schemes will perform recognition operations in a single learning cycle of predictable duration, which will provide online learning capabilities that can support mission critical applications. In addition to minimal resources and time requirements, the distributed structure of the schemes will sustain large-scale networks in performing pattern recognition operations.

To deal with a WSN's dynamic nature and limited prior knowledge of events, a pattern transformation invariant scheme is proposed in this research. The proposed scheme implements a weighting mechanism that searches the

edges and boundaries of patterns and replaces traditional local information storing. This mechanism allows the scheme to identify dynamic and continuous changes in patterns. Consequently, the scheme will be capable of performing recognition operations in dynamic environments and will also provide a high level of detection accuracy using a minimal amount of available information about patterns. Required protocols for performing scheme operations are also presented and discussed.

Theoretical and experimental analysis and evaluation of the presented schemes is conducted in the research. The evaluation includes time complexity, recognition accuracy, communicational and computational overhead, energy consumption and lifetime analysis. The scheme's performance is also compared with existing recognition schemes. This shows that the scheme is capable of minimising computational and communicational overheads in resource-constrained networks, enabling those networks to perform efficient recognition activities for patterns that involve transformations within a single learning cycle while maintaining a high level of scalability and accuracy. The results show that the scheme's time complexity is proportional to the square root of the pattern size which allows the network to scale up to adopt large patterns. It is shown that a network that implements mica 2 motes and requires 3.0625 milliseconds to send a single message can perform recognition operations within a single learning cycle duration ranging between 126.4 and 323.1 milliseconds for 10,000- and 40,000-node network settings respectively. The results also show that energy requirements can be decreased up to 89.66

per cent by using the proposed schemes in comparison to other recognition techniques. In terms of efficiency, theoretical and experimental analyses show that the proposed schemes are capable of dealing with transformed patterns with a high level of accuracy. The analyses show that the scheme is able to detect translated patterns, rotated or even flipped patterns with a rotational angle of up to 23 degrees, and dilated patterns with a dilation level of up to 26 per cent. The results show that the proposed schemes have features that will be best suited for implementing pattern recognition applications on resource-constrained networks such as WSNs.

The research also discusses the use of the proposed pattern-recognition-based schemes in different machine learning and artificial intelligence (AI) applications. This aims to explore new research opportunities that can lead to enhancing existing schemes' performance by involving the proposed schemes in different technological disciplines and models. Two disciplines are presented as examples in this context: optimisation and classification. In the first example, a new model that involves the proposed schemes in the process of optimisation techniques, in this case genetic algorithms (GA), is presented. The proposed model enhances the performance of traditional GA in terms of speed and accuracy. The second example proposes a classification model using the pattern-recognition-based proposed schemes. The proposed model shows a high level of classification capabilities compared with other well-known existing schemes.

Declaration

In accordance with Monash University Doctorate Regulation 17 / Doctor of Philosophy and Master of Philosophy (MPhil) regulations the following declarations are made:

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Waleed M. Aalfahaid
November 14, 2013

This thesis is dedicated to my beloved parents, my wife Hind, and my children, who have inspired and supported me in my pursuit of higher education. This thesis is also dedicated to my brother, Abdullah, and my father-in-law, who passed away as I worked on my degree, before I had a chance to say goodbye to them.

Acknowledgements

It would not have been possible for me to accomplish this research project without the support and help I received from people involved in this project and those who were around me during the pursuit of my degree. Hence, I would like to take this opportunity to thank and acknowledge them.

First and foremost, I thank my GOD Allah the Almighty for all blessings and help I have received in my life and during my pursuit of my degree.

I am grateful and thankful for my supervisors Dr. Asad Khan and Prof. Bala Srinivasan, for their help, guidance and support in developing my thesis. This research would not have been possible without their contribution and support. They trained and guided me in building up my knowledge and in developing my skills, and finally in writing this thesis. They also helped me during the hard and sad days that my family and I experienced during my study.

I would like to thank Dr. Noriaki Sato for his help and support in improving the structure of my thesis and training me on how to tackle different types of academic writing issues. I also thank Ms. Anna Thwaites and Ms. Alison Caddick for proofreading my thesis. I also thank my friends Dr. Aron Mani and Mr. Yahia Alnashri for reviewing some parts of my mathematical proofs. Special thanks for my colleagues Mr. Mohammed Alnaeem, Ms. Amiza Amir, Dr. Anang Amin and Mr. Amir Basirat for their help and support.

Great thanks are also given to my father Mohammed and my mother Hessa for encouraging me to pursue my degree and for their love, patience and everything they have given me ever since I was born until this moment. Special thanks also for my adored wife Hind Altwaijry for surrounding me with love and care and lighting up my life, making it possible for me to achieve my goals in this life. She has always been patient, supportive and helpful. I also thank my children, Khaled, Bader, and Abdullah, for being with me all the time, making my life enjoyable and giving me great happiness in my life. Many thanks are due also to my brothers, sisters, nephews and nieces for their love, support and faithful wishes.

I also thank my friends Fahid Alqahtani, Hamad Alsuhaim, Saleh Alrubaish, Mohammed Alhabeeb, Abdullah Almuhaideb, Suliman Alsaloom, Adel Binjahlan, Salem Alkhalaf, Omar Alluhydan, Saleh Alsuhaim, Khaled Alobyki, Abdullah Binjahlan, and the rest of my friends for supporting me and being my family during my PhD studies.

Thank you all for making my dreams come true.

Waleed M. Alfahaid

Monash University

November 2013

Outcomes/Publications

The outcomes of this thesis research have been published as follows:

1. Waleed M. Alfahaid and Asad I. Khan, “Cellular microscopic pattern recogniser – A distributed computational approach for macroscopic event detection in WSN,” in *Computational Science (ICCS)*, Proceedings of the *11th International Conference on Computational Science*, vol. 4, pp. 66-75, 2011.
2. Waleed M. Alfahaid and Asad I. Khan, “A combined pattern recognition scheme with genetic algorithms for robot guidance using wireless sensor networks,” in *Proceedings of 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pp. 759-764, 2012.

Table of Contents

Abstract	i
Declaration	v
Acknowledgements	vii
Outcomes/Publications	ix
Table of Contents	x
List of Tables	xv
List of Figures	xvi
1 Introduction	1
1.1 Challenges and services of WSNs.....	5
1.2 Motivation and Objectives	7
1.3 Contributions	11
1.4 Thesis Outline.....	14
2 Pattern Recognition in WSNss	17
2.1 Introduction	17
2.2 Wireless Sensor Networks (WSNs)	20
2.2.1 Common WSN network topologies	21
2.2.2 WSN applications.....	23
2.2.3 WSN network architecture.....	25
2.3 Pattern Recognition in WSN	28
2.3.1 Threshold-based techniques.....	30

2.3.2	K-nearest neighbour	31
2.3.3	Statistical approaches	34
2.3.4	Neural networks	39
2.3.5	Support vector machines (SVM).....	49
2.3.6	Graph neuron (GN)	51
2.3.7	Structural and conditional methods.....	57
2.4	Requirements of Pattern Recognition in WSNs	61
2.5	Comparing Existing Schemes.....	65
2.6	Possible Solution	70
2.7	Summary	73
3 Cellular Graph Neuron (CGN) for Pattern Recognition in WSNs..		
	75
3.1	Introduction	75
3.2	Overview of CGN	78
3.2.1	CGN structure	79
3.2.2	CGN network	81
3.2.3	Memory and bias array.....	87
3.2.4	Network operations	88
3.2.5	S&I operations	90
3.3	Obtaining pattern in CGN	94
3.4	Complexity of CGN schemes.....	98
3.4.1	CGN network size	98
3.4.2	CGN communications	101

3.4.3	CGN network time	103
3.5	Evaluating CGN Performance	105
3.5.1	CGN and Hopfield	105
3.5.2	First test	108
3.5.3	Second test	110
3.6	Summary	114
4 Cellular Weighted Graph Neuron (CwGN) for Transformation		
	Invariant Recognition in WSN.....	117
4.1	Introduction	117
4.2	Transformation Invariant Recognition Techniques.....	120
4.3	Overview of Cellular Weighted Pattern Recogniser (CWGN)	123
4.3.1	Stimulator and interpreter (S&I)	125
4.3.2	CwGN network	128
4.3.3	Pattern edge search.....	135
4.4	CWGN Communication Scheme	140
4.4.1	CWGN communication requirements.....	141
4.4.2	CWGN communication protocol	145
4.5	Complexity of CwGN Algorithm.....	147
4.6	Zoning Approach for Online Recognition.....	151
4.7	CwGN Message Sequence Models	155
4.7.1	Frame-slotted asynchronous CwGN model	158
4.7.2	Frame-slotted synchronous CwGN model	163

4.7.3	Multi-channel CwGN models	167
4.8	Effects of Pattern Transformation on CwGN Recognition ...	172
4.8.1	Pattern translation.....	173
4.8.2	Pattern dilation	176
4.8.3	Spatial rotation	179
4.9	Summary	181
5	Experimental Evaluation of CwGN Schemes	184
5.1	Introduction	184
5.2	Accuracy analysis of CwGN	186
5.2.1	CwGN accuracy using uniform patterns (shapes dataset).....	187
5.2.2	CwGN accuracy using non-uniform patterns (contours dataset)	195
5.3	Comparing CwGN Accuracy with other Schemes.....	201
5.3.1	Hill-Valley problem	201
5.3.2	Wall following robot problem.....	204
5.4	CwGN Communicational Overhead Analysis	211
5.4.1	Activation process analysis	211
5.4.2	Energy and time analysis	214
5.5	Summary	226
6	Using CwGN Scheme in Enhancing Optimisation and Pattern Matching Applications Performance	230
6.1	Introduction	230

6.2 Hybrid CwGN-GA Scheme for Autonomous Robot Navigation using WSN.....	235
6.2.1 Approach	238
6.2.2 Performance enhancement	241
6.2.3 Autonomous robot navigation using GA	243
6.2.4 Simulation	244
6.3 Human Activity Recognition Using WSNs.....	248
6.3.1 Opportunity dataset	252
6.3.2 CwGN for activity recognition.....	253
6.4 Summary	259
7 Conclusions and Future Work	262
7.1 Summary of the Research.....	262
7.2 Research Contributions and Outcomes	265
7.3 Future Work	270
References	274

List of Tables

Table 2.1: Comparison of existing pattern recognition schemes for WSNs.	71
Table 3.1: Command messages from BS to network nodes.....	97
Table 3.2: Description of the terms used for complexity estimation. ...	99
Table 4.1: Description of terms used in CwGN complexity analysis.	148
Table 4.2: Summary of existing MAC protocol types for WSNs.	157
Table 4.3: Summary of communication time overhead (T_{comm}) limit estimates for each CwGN message sequence model.	172
Table 5.1: Recognition accuracy results of different schemes for the hill and valley dataset	203
Table 5.2: Recognition accuracy results of different schemes for the wall following robot dataset.....	206
Table 5.3: Recognition accuracy results of robot navigation simulation for different schemes.....	209
Table 5.4: Average activated nodes.	213
Table 5.5: Summary of assumptions used in the simulation.....	216
Table 6.1: Recognition accuracy results of opportunity challenge dataset for different schemes.....	256

List of Figures

Figure 1.1: The main components of a WSN sensor	4
Figure 2.1: The three WSN network topologies	22
Figure 2.2: Classification of existing pattern recognition schemes for WSNs.	29
Figure 2.3: KNN classification example.....	33
Figure 2.4: A simple Bayesian belief network.....	36
Figure 2.5: The structure of a feed-forward NN	40
Figure 2.6: The Hopfield network structure.....	41
Figure 2.7: RNN structure.....	44
Figure 2.8: ART network architecture	46
Figure 2.9: SVM classes separation.....	50
Figure 2.10: A simple four node GN array	52
Figure 2.11: Simple HGN structure	54
Figure 2.12: DHGN structure.....	56
Figure 2.13: A simple decision tree example.....	61
Figure 3.1: The two main components of the CGN scheme.	80
Figure 3.2: Active CGN nodes.....	82
Figure 3.3: CGN track of m neuron positions.	83
Figure 3.4: CGN network to adopt a 9 elements binary pattern	86
Figure 3.5: A CGN NP node's memory structure.....	88
Figure 3.6: CGN node learning operations steps block diagram.	93

Figure 3.7: S&I learning operation steps block diagram.	94
Figure 3.8: Block diagram of S&I voting steps.	95
Figure 3.9: Pattern divided into sub-patterns by S&I	96
Figure 3.10: Number of communications in Hopfield and CGN networks based on pattern size.....	107
Figure 3.11: Time derived from Big-O analysis for Hopfield and CGN	108
Figure 3.12: Original bitmap image patterns for A, I, J, S, X and Z with sample of recalled distorted images.	112
Figure 3.13: Accuracy recall percentage for the CGN.....	113
Figure 3.14: Memorisation and sample recognition patterns representing written numbers from 0 to 5.	113
Figure 3.15: Average accuracy levels obtained by CGN, Naïve Bayes and back propagation networks.....	114
Figure 4.1: The CWGN communication model.....	125
Figure 4.2: S&I operations for memorising and recalling patterns.....	129
Figure 4.3: Track exchange communications	131
Figure 4.4: CwGN network.....	135
Figure 4.5: CwGN network node operations for weight calculation. .	137
Figure 4.6: Track linking CwGN network communication scheme. ..	141
Figure 4.7: Report messages from a node to its alternative assigned inner nodes.	144

Figure 4.8: The estimated recall time of the CwGN network with respect to increasing pattern size.....	152
Figure 4.9: The estimated recall time of the CwGN network as a function of the number of stored patterns	152
Figure 4.10: Parallel CwGN network zones	154
Figure 4.11: A general MAC frame structure [35].	157
Figure 4.12: CwGN FS-Async message sequence model.....	159
Figure 4.13: CwGN FS-Async message sequence model scenarios...	159
Figure 4.14: Message sequence for FS-Async report communications	161
Figure 4.15: : CwGN Message sequence for FS-Sync exchange communicational model.....	165
Figure 4.16: CwGN message sequence for FS-Sync exchange communicational model	166
Figure 4.17: Message sequence for FS-Sync report communicational model.....	167
Figure 4.18: CwGN MC-Async message sequence for exchange communications when all adjacent nodes to a sending node are active.	169
Figure 4.19: CwGN Message sequence for MC-Async exchange communicational model.....	169

Figure 4.20: CwGN MC-Sync message sequence for exchange communications when all adjacent nodes to a sending node are active.	170
Figure 4.21: CwGN message sequence for MC-Sync exchange communicational model	170
Figure 4.22: Possible types of pattern transformations.	180
Figure 5.1: Shapes used as the training dataset for the first test series .	189
Figure 5.2: Sample of altered patterns used as the testing dataset for the shapes test series.	189
Figure 5.3: Sample of altered patterns used as the testing dataset for complex translation and combination of translation and rotation transformations.	189
Figure 5.4: CwGN network accuracy in detecting spatially rotated patterns for the shapes dataset.	190
Figure 5.5: CwGN network accuracy in detecting spatially rotated patterns for the shapes dataset.	191
Figure 5.6: CwGN network accuracy in detecting translated patterns for the shapes dataset	193
Figure 5.7: CwGN network accuracy in detecting dilated patterns for the shapes dataset	194
Figure 5.8: CwGN network accuracy in detecting dilated patterns for the shapes dataset	194

Figure 5.9: Process of producing contours training dataset.	196
Figure 5.10: The rest of the non-uniform shape training patterns.....	196
Figure 5.11: CwGN network accuracy in detecting spatially rotated patterns for the contours dataset.....	197
Figure 5.12: CwGN network accuracy in detecting spatially rotated patterns for the contours dataset.....	198
Figure 5.13: CwGN network accuracy in detecting dilated patterns for the contours dataset.....	198
Figure 5.14: CwGN network accuracy in detecting dilated patterns for the contours dataset.....	199
Figure 5.15: CwGN network accuracy in detecting translated patterns for the contours dataset	200
Figure 5.16: Three samples of hill pattern.	203
Figure 5.17: Three samples of valley pattern.....	203
Figure 5.18: The ROC space and plots of the hill and valley classes for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes.....	204
Figure 5.19: The ROC space and plots of the classes used in the wall following robot problem for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes	207
Figure 5.20: Room setting for the wall following robot navigation problem	208
Figure 5.21: Robot route obtained by CwGN	209
Figure 5.22: Example of CwGN nodes activation	213

Figure 5.23. A simple parallel KNN network.	217
Figure 5.24. Average number of communications for CwGN Async, Sync, and parallel KNN for the shapes and contours datasets.	217
Figure 5.25: Average energy consumption for each network for each dataset in mJ.	219
Figure 5.26: Average energy consumption for each network for both shapes and contours datasets in mJ.	220
Figure 5.27: Average lifetime for CwGN and parallel KNN networks	220
Figure 5.28: Available energy for each node in CwGN networks dealing with the shapes dataset.	222
Figure 5.29: Available energy for each node in in CwGN networks dealing with the contours dataset.	223
Figure 5.30: Average learning cycle time in milliseconds (ms) for a CwGN network.	224
Figure 6.1: The memorisation phase of hybrid CwGN-GA scheme.	240
Figure 6.2: The optimisation phase of hybrid CwGN-GA scheme.	241
Figure 6.3: A training map from the contours dataset and a sample of altered maps used in the testing dataset.	245
Figure 6.4: A comparison of the performance between the proposed combined CwGN-GA and autonomous GA.	2477
Figure 6.5: Connectivity between neighbouring nodes in a 3-D CwGN track.	254

Figure 6.6: The ROC space and plots of the activity classes for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes.	256
--	-----

Chapter 1

Introduction

The advance of modern technology in recent decades has had a huge impact on the way organisations, businesses, and individuals interact with various technological environments and the means of using their services. Expectations are growing as new technological capabilities are improving. Smart systems are more in use in our daily life, replacing old-fashioned processes and procedures as a result of technological evolution. Embedded systems are one kind of such smart systems that interact with human lives and provide a variety of services that can be used in such areas as industrial, military, medical, and agricultural applications. Embedded systems are usually small components with limited computational resources that are designed to perform specific tasks. Such systems exist in numerous applications in our daily life and can scale in their complexity from performing a single task in electrical appliances to performing complex control systems decisions [1]. As a consequence of complexity variations, not every embedded system can be classified as a smart one. Krishnamurthy [2] says,

A smart system exhibits the three important properties: (i) Interactive, collective, coordinated and parallel operation (ii) Self-organization through emergent properties (iii) Adaptive and Flexible operation.

According to the Krishnamurthy [2], this means that a system should be capable of interacting with the environment, capable of being dynamic in relation to the emergence of new unpredictable properties, and flexible to environmental changes by using adaptive learning techniques if it is to be considered a smart system. In simpler terms, Chandrasekaran [3] says,

The public does not expect a smart system to do everything that people do. It does expect a smart product to be flexible, adaptive, and robust.

The existence of such systems allows the linking of real world activities to computerised capabilities, paving the way for more interactive methods to automate processes and make accurate decisions.

With the advent of ad hoc networks, embedded devices can form a wireless network to communicate with each other in order to perform more complex tasks [4]. Wireless Sensor Networks (WSNs) are a specific type of such networks. A WSN consists of a number of smart sensor nodes that sense physical activities such as motion, heat, speed, and many other environmental parameters, and provide solutions for multiple applications such as climate sensing, factory monitoring, traffic monitoring, and pollution measuring [5]. A sensor node is small in size. It generally varies from the size of a grain to the size of a coin [6]. A sensor node consists of four main components: a power unit, a sensing unit, a communication unit, and a processing unit. Because of the limited size of a sensor, the power source that a sensor is usually equipped

with is limited and is often achieved by using batteries. This will affect the lifetime of the sensor node. To overcome this issue, it is also possible that sensor nodes be equipped with alternative power resources, such as tiny solar panels, that can help in increasing a node's life time. Sensing units should be capable of doing two tasks: observing the surrounding environment and transferring the readings to digital data. For example, they might detect the surrounding temperature and convert the reading into a digital form. The communication unit of a sensor allows it to communicate with other nodes in the network. It is considered to be one of the physical layer elements that sends and receives radio-based signals to and from other nodes. It can also be a laser- or infrared-based unit. A processing unit is essentially responsible for doing the computations needed by the sensor and is usually combined with a memory unit. Other components can also be added to a sensor node, based on application requirements. Examples of such components may be a GPS unit for localisation purposes or a mobiliser for node location change estimation. The main components of a WSN sensor are shown in Figure 1.1 [5, 7].

A WSN can scale to thousands of densely deployed nodes in a vast area in order to perform its tasks [8]. A WSN node is susceptible to failures for many reasons, which can lead to communications and information losses between peer nodes. One reason for such failures could be the harsh environmental conditions in which WSNs are usually deployed, such as those of underwater WSNs. Another cause of failure could be dead batteries as sensor nodes are limited in their energy resources. An adversary attack could

also have harmful effects on a node's functionality. In fact, environmental factors, limited energy resources, adversary attacks, module failures, fabrication problems, and being out of communication range are the most frequent causes of WSN sensor failure [9]. Common dense WSN network deployment is useful in compensating failed nodes to maintain network functionality. On the other hand, some applications require topological or functional changes of a WSN [10]. Consequently, WSNs are usually designed to be robust in order to self-adapt to node failures, topological changes, and functional modifications [11].

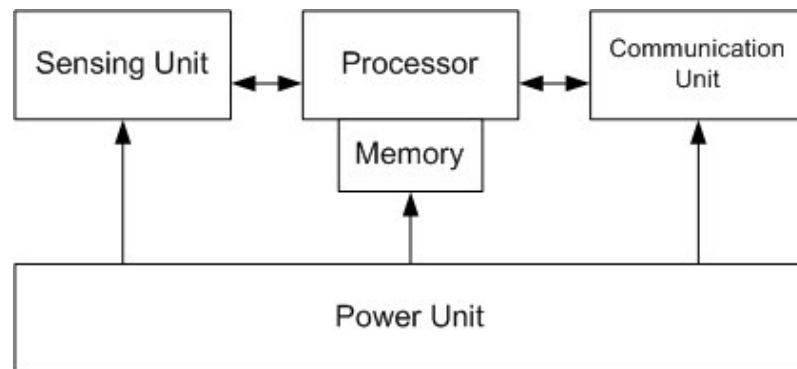


Figure 1.1: The main components of a WSN sensor [5].

Adaptive learning techniques have been implemented and proposed in conjunction with WSNs to offer solutions for many problems and scenarios. Adaptive learning allows WSNs to learn patterns occurring in areas under surveillance and recognise these patterns when they occur again. This process is known as Pattern Recognition (PR) and is useful for decision making around such issues as fire detection, gas leak detection, environmental phenomena

warnings, and so forth. Considering that sensor nodes are limited in their energy, memory and computational resources, traditional applications, techniques, and protocols that are in use in networks in general are usually not applicable in WSNs. Hence, adaptive learning and PR techniques such as Neural Networks (NN), template matching, fuzzy logic, and nearest neighbour are usually tailored to WSNs' limitations. The success of WSNs in adopting such techniques puts WSNs among smart systems that sense the environment, adapt to changes, and learn from experience in order to provide proper means for automated and accurate decision making processes [12].

1.1 Challenges and services of WSNs

Wireless sensor networks are described as wireless ad hoc networks as they share multiple features and characteristics. However, there are several differences between the two platforms, making WSNs a unique type of ad hoc network. The differences are carried by the WSNs' node size, deployment environment, and flexibility requirements. These factors make the design and implementation of WSNs challenging. They can be described in the following way [5, 7, 8, 10, 12, 13]:

- i. **Limited energy:** A WSN node is small in size, which makes it limited in its energy, memory, and computational resources. The lifetime of a sensor node is limited and depends on the tasks the sensor is required to perform. If a sensor is required to maintain continuous computations or communications, its lifetime will be shortened.

- ii. **Limited memory:** This directly affects the ability of sensor nodes when dealing with computations, queries, and communications, as these three tasks require a part of the memory to store the result, keep track of communications, and prepare information for query response.
- iii. **Limited communication:** The communication range of a WSN sensor is short. Sensors are usually equipped with small communication devices as they are small in size, causing restrictions such as limited power of transmission.
- iv. **Node failure:** Sensor nodes in WSNs are susceptible to failure and damage for reasons such as physical environmental effects and limited energy. In some applications, sensors are deployed in hostile environments exposing them to physical dangers such as fire, animals, storms, and floods. If a sensor survives such effects, it will sooner or later die due to energy consumption.
- v. **Network size:** In WSNs, the number of sensors can scale to thousands as some applications need to cover a very large area in order to efficiently sense the occurring physical features. Additionally, sensors are usually densely deployed in WSNs to overcome node failure problems.
- vi. **Dynamics of the network:** WSNs commonly require changes to their network size and topology. These changes stem from the need to add nodes to the network, nodes failures, and communications failures. In addition, the environment a WSN is monitoring is expected to be

dynamic, requiring a set of network changes over time. Other applications require mobile sensor networks, which are of a dynamic nature in themselves.

- vii. **Identification issues:** In general, sensor nodes in WSNs do not have unique identifications (ID) that distinguish one node from another. This is due to the large-scale sensor network size and the dense deployment of nodes. This means that sensors use broadcast messages to communicate with neighbouring nodes rather than using point to point communication.

Applications and network management protocols for wireless sensor networks require a set of services in order to be functional. These services include data aggregation, security, sensor deployment, localisation, coverage, optimisation, and routing. These services should take WSNs' limitations and challenges into consideration in order to provide proper quality of service (QoS) levels for WSN applications and management protocols [13].

1.2 Motivation and Objectives

Wireless sensor networks are used for sensing real environments and are useful for countless applications. A sensor node is responsible for detecting and sensing events in a limited surrounding area. However, global events can span over very large regions, requiring large numbers of sensor readings to be detected. WSNs can scale to thousands of communicating embedded sensors

deployed over a large geographical area. These sensor readings can be used in a collaborative manner to detect global events. Considering WSNs' limitations, designing and implementing collaborative global event detection applications and schemes is challenging.

Sensory data from thousands of sensors are expected to be processed and analysed in order to detect global events. Processing such information can be achieved by using one of two common processing paradigms: centralised processing or in-network processing. In centralised processing, data will be forwarded to one (or a set) network entity for processing. Many issues are related to using this paradigm in global event detection using WSNs. Due to limited WSN node capabilities, processing data in one entity will exhaust its computational and communication resources. Data collection and aggregation towards selected nodes for processing in a WSN creates communication bottlenecks and can result in a single point of failure problem, threatening the availability of a service or application [14]. Hence, distributing processing information among nodes to create an in-network processing paradigm is favourable as data aggregation and processing tasks are distributed over the entire network, avoiding the creation of communicational and processing bottlenecks [15].

Global events occurring in large regions can arise randomly and in various forms according to environmental factors or the use of mobile entities [10]. Such random phenomena can result in there being a limited number of event patterns available to be used in global event detection in the area under

surveillance. This randomness of event occurrence can be tackled by using adaptive learning techniques that are capable of searching for similarities between a stored event and a currently encountered one. These techniques store event patterns, allowing WSNs to learn from experience and develop information about patterns. The changing feature of events could come in different forms, such as an event pattern dilation, rotation, translation, or as a combination of these variances. Consequently, efficient detection mechanisms that are capable of discovering events even if they occur in different forms are required for global event detection. However, these techniques should address WSNs' limitations and challenges in order to provide acceptable levels of QoS for beneficial applications.

WSNs support a variety of mission critical and decision-making applications such as battlefield monitoring, robot guidance, and structural health monitoring. Such applications have unique requirements if they are to be beneficial. These requirements are driven by the fact that critical applications should result in decision making at a certain point in time. In general, a WSN node communicates directly or indirectly with one or more nodes in the network, called *base station* and *sink* nodes, or to an external monitoring server to deliver sensed and processed information. Fast reporting and delivering of information to base station nodes, sink nodes, and monitoring servers is one of the key requirements in mission critical applications. This would include efficient communication methodology and a global decision-making mechanism. Reliable and accurate detection is another requirement for accurate

decision-making applications in order to avoid wrong actions or delayed responses. In addition, these applications require the network to be fault tolerant. This means that a WSN should be capable of dealing with noisy patterns and faulty nodes [16].

Detection accuracy versus managing energy resources is the main challenge for providing proper event and pattern detection capabilities for WSN applications [17]. Existing event detection and pattern recognition schemes for WSNs use neural networks, support vector machines (SVM), fuzzy inference systems (FIS), and other detection techniques. These techniques are generally tailored to provide detection capabilities for specific applications or problem scenarios. However, these techniques may fulfil some of the event detection and decision-making application requirements (i.e. detection accuracy) while failing in respect of other requirements (i.e. light-weight detection) by adding high communicational or computational overhead to a WSN. Hence, pattern recognition schemes which lead to event detection in WSNs that fulfil global event detection for decision-making applications requirements by balancing detection accuracy and WSN resource-constrained are required.

The aim of this thesis is to develop and implement collaborative in-network global event detection schemes that are light-weight, scalable, and best suit resource-constrained networks such as WSNs. The proposed schemes address the limitations and challenges for WSNs to provide reasonable QoS for applications. Additionally, these schemes address the randomness of event

occurrences by using adaptive learning techniques and the occurrence changes related to event patterns such as rotation, translation, and dilation. The schemes simplify computations for energy conservation and speed up recognition by leveraging the parallel distributed processing capabilities of WSNs.

Another aim of the thesis is to design decision-making support models using the proposed event detection techniques. The proposed schemes will be used in conjunction with decision-making approaches such as Genetic Algorithms (GA) to speed up the process of optimisation and obtain more accurate and automated decisions by learning from experience. The detection and decision-making techniques proposed in this thesis can be applied to different WSN platforms and any other resource-constrained network environments for numerous applications and scenarios.

1.3 Contributions

The main contributions of the thesis are:

- i. **Developing new global pattern recognition based event detection schemes for resource-constrained networks such as WSNs:** The developed schemes address the limited resources of WSNs by performing communicational and computational tasks in a distributed manner. The distributed techniques will be more suited for real time and real life WSN applications, especially those that require event detection over a large area in terms of communications and computations complexity.

ii. **Developing a pattern transformation invariant scheme for WSNs:**

In large regions that are under surveillance, events arise randomly and changeably. An event could occur in a certain part of the region with a set of characteristics and it take a long time for a similar event to occur in a different part of the region with variations to the previous characteristics. Thus, the proposed detection scheme will address randomness and changing phenomena by adopting techniques that make it possible to store events and recognise transformation in patterns such as translation, dilation, rotation, or a combination of these factors.

iii. **Developing communication protocols for pattern recognition in resource-constrained networks:**

Communication protocols are needed for WSNs to be functional in terms of detection techniques. Such protocols will be presented in this thesis. They will describe the tasks and communications required by network nodes to learn and recognise patterns, from sensing the data to concluding the result.

iv. **Integrating recognition schemes with decision-making methods to enhance decision-making process performance:**

There are several types of applications that use WSNs for decision-making purposes, such as real time applications. In this thesis a pattern recognition based decision-making model is proposed. The aim of the model is to speed up the decision-making process by detecting events and suggest suitable and reliable solutions. By developing such a model, it is

possible to automate the entire decision-making process for large monitored regions, starting with sensing and ending with taking an action.

v. **Designing pattern recognition based classification model for WSNs:**

Complex classification problems are common challenging tasks for resource-constrained networks such as WSNs. In this thesis a pattern recognition based classification model is proposed. Such a model demonstrates the ability of the proposed schemes to perform classification tasks with minimal resource requirements using pattern recognition capabilities while maintaining high accuracy compared to other classification schemes. This model shows the advantage of using pattern recognition capabilities in solving complex classification problems.

vi. **Analysing and evaluating proposed schemes:** Analysis and simulations for the proposed pattern recognition and decision-making schemes will be conducted in this thesis. This will include time complexity analysis, analysis of pattern translations effects and storing patterns, determining accuracy levels of patterns detection, and decision-making process performance analysis. Additionally, this thesis will provide a comparison between proposed schemes and other existing pattern recognition schemes in term of accuracy, communication overhead, computational overhead, and storage requirements. The proposed integrated decision-making model will also

be compared with other models to evaluate the speed enhancement gained by applying the proposed model.

1.4 Thesis outline

There are seven chapters in this thesis. In chapter two, a background on pattern recognition and event detection in wireless sensor networks will be presented. There will be a detailed analysis of existing pattern recognition schemes for WSNs, including threshold-based, template matching, nearest neighbour, statistical, syntactical, fuzzy logic, and neural networks techniques. Additionally, issues related to implementing such schemes for global event detection will be discussed. Finally, a set of metrics to evaluate the suitability of existing schemes for detecting changing events using WSNs will be presented and a comparison between these schemes provided.

In Chapter three, a novel pattern detection scheme for WSNs will be presented. The scheme will provide template matching and noisy pattern recognition capabilities in a light-weight and distributed manner that suits WSNs. This scheme is fault-tolerant and speeds up recognition by leveraging the parallel distributed processing capabilities of WSNs. Extended analysis of the presented scheme will be provided, including time complexity and simulation tests. Finally, the proposed scheme will be tested and compared with existing schemes in terms of accuracy and time requirements.

In Chapter four, a novel efficient event detection and pattern recognition scheme that addresses the problem of changing events such as

rotation, dilation, and translation will be presented. The scheme reduces the number of nodes, required computations, memory resources, and number of communications needed for performing event detection in WSNs. This is achieved by adopting distributed and parallel design, along with efficient activation processes. The network models and protocols of the scheme will be presented in this chapter. Additionally, this chapter will present a parallel model that allows the scheme to function under online operations' constraints. Extended theoretical analysis of the presented scheme will be provided, covering time complexity and predictions of pattern transformation recognition accuracy.

In Chapter five, tests and results that evaluate the presented scheme's performance levels will be provided. The chapter starts by evaluating recognition accuracy levels that can be obtained by the proposed scheme using different types of transformed datasets. Then, the scheme's network performance will be evaluated in terms of time and energy. Finally, the scheme will be compared with other existing recognition techniques in terms of recognition accuracy, using different types of standard datasets.

In Chapter six, the capabilities of the proposed schemes will be demonstrated using two models serving two different application domains. The first model will demonstrate the ability of the proposed schemes to enhance the performance of sensory-dependent decision-making systems. A pattern recognition based decision-making scheme will be presented as an application of the proposed detection schemes. The proposed decision-making scheme

combines detection schemes with genetic algorithms (GA) in order to speed up finding the optimal solution process. The proposed scheme will be analysed and evaluated against normal random schemes to measure speed-up enhancements. The second model will discuss the classification problem of human activity recognition as an example of a mission critical application. The aim is to demonstrate the ability of the proposed pattern based recognition schemes to learn and recognise complex patterns using a minimal amounts of information and resources to perform classification tasks. Using a standard dataset, the proposed model will be compared with existing schemes in terms of performance and resources requirements.

Chapter seven concludes the thesis by summarising the contributions presented in this thesis. Additionally, issues related to proposed schemes and future work will be presented in this chapter.

Chapter 2

Pattern Recognition in WSNs

2.1 Introduction

The main task of a wireless sensor network (WSN) is to sense a physical or network environment and detect events occurring in the field of interest or in the monitored network [18]. According to Chandy [19], an event in general may be seen as “changes in the real state”. More specifically, Johansson [20] define an event as “Changes that take place in one or more elements within a large group of these elements”. In sensor networks, Boonma and Suzuki [21] and Ortmann et al. [22] describe an event as observable changes in sensors’ readings. More precisely, Zoumboulakis and Roussos [23] indicate that an event is a collection of sensor readings that describe a specific or an abnormal activity. In general, we define an event in WSNs as a change in the state that describes a specific state of predefined phenomena in the field of interest. Examples of event detection in WSNs include the detection of fires in forests, a border intruder, network congestion or a network security attack.

Detecting such events can be performed by processing and analysing sensory information obtained by sensor nodes. Pattern recognition is one

commonly used approach for event detection in WSNs and has become the focus when dealing with event detection problems in the literature, especially when detecting complex events [24]. Watanabe [25] defines a pattern as the “opposite of chaos”. Catania et al. [26] define a pattern as “a compact and rich in semantics representation of raw data”. In this research, a pattern is defined as a set of raw sensory data that describes the main characteristics of an event. In other words, a pattern can be seen as the signature of an event.

Pattern recognition is highly affected by the limited resources offered by WSNs, including limited energy and limited computational, communicational, and memory resources. In addition to limited resources, WSNs carry other challenges for event detection. These challenges are related to the nature of the environments that WSNs are usually deployed in. For instance, WSNs are usually deployed in hostile environments, making sensors susceptible to physical damage and intentional tampering. Additionally, sources of electricity are not usually available for running sensor nodes, requiring these sensors to be operated using limited batteries. WSNs are also usually required to communicate in an ad hoc manner using low frequency radio signals due to the absence of wires in deployed environments. Moreover, WSNs are usually deployed as large numbers of sensors in order to monitor an area of interest, which therefore requires the use of low cost sensors. This will limit the types of instruments and resources that these sensors can be supplied with. [27]. As a consequence, pattern recognition in WSNs is a matter of trade-

off between detection accuracy, the use of limited available resources and dealing with existing challenges [28].

Other challenges for pattern recognition in WSNs are application related. WSNs support a variety of real time applications, including battlefield monitoring, environmental monitoring, emergency relief, microsurgery, and even more for different disciplines such as health, environment, education, surveillance, and others. Such applications have unique requirements in order to be beneficial. These requirements are driven by the fact that real time applications should result in decision making at a certain point of time. Fast reporting to the WSN sink or to a monitoring server is one of these requirements. This would include efficient communication methodology and a global decision making mechanism. Reliable and accurate detection is another requirement for critical mission applications in order to reduce false alarms. In addition, these applications require the network to be fault tolerant. This means that a WSN should be capable of dealing with noisy patterns and faulty nodes [16].

Luo et al. [17] state that detection accuracy versus managing energy resources is the main challenge for providing proper event and pattern detection capabilities for WSN applications. Existing event detection and pattern recognition schemes use neural networks, support Vector Machines (SVM), Fuzzy Inference Systems (FIS), and other detection techniques. These techniques are generally tailored to provide detection capabilities for specific applications or problem scenarios. The techniques may fulfil some of the

requirements (e.g. detection accuracy) while failing at others (e.g. light weight detection). Hence, what are required are pattern recognition schemes which lead to event detection in WSNs that fulfil real time application requirements by balancing detection accuracy and WSN resource constraints.

In this chapter, existing pattern recognition schemes for WSNs are presented and analysed. In section 2.2, the chapter starts by briefly introducing WSNs, including network topologies, applications, and network architecture. Section 2.3 presents pattern recognition in WSNs. This includes the existing recognition schemes that have been used in the literature to solve the problem of pattern recognition in WSNs. Section 2.4 analyses the requirements for solving such recognition problems, especially when using WSNs for real time and decision making recognition problems. Section 2.5 compares existing schemes based on the requirements and discusses the issues related to these schemes. Section 2.6 discusses possible methods that can be used to overcome the existing schemes' limitations to provide real time recognition for large scale WSNs. Section 2.7 concludes the chapter.

2.2 Wireless Sensor Networks (WSNs)

Wireless sensor networks (WSNs) are a specific type of ad hoc network. A WSN consists of a number of smart sensor nodes that sense physical activities such as motion, heat, speed, and many other environmental parameters. WSNs provide solutions for multiple applications such as climate sensing, factory monitoring, traffic monitoring, and pollution measuring. A

WSN can scale to thousands of densely deployed nodes in order to perform its tasks. Dense WSN networks are useful because sensor nodes are generally susceptible to failures. A sensor node is small, its size generally varying from the size of a grain to the size of a hand. Sensor nodes are limited in their energy, memory, and computational resources, thereby limiting WSNs. Consequently, traditional applications and protocols that are in use for networks in general are usually not applicable in WSNs [6, 7, 29-31].

As discussed in Chapter 1, a sensor node consists of four main components: a power unit, a sensing unit, a communication unit, and a processing unit. Some sensors may also include other components to enhance performance or to perform specific tasks. Solar panels and GPS devices can be examples of such additional components. Due to the limited size of a sensor, the capabilities and resources of its components are limited [5, 7]. Sensors are usually wirelessly connected and densely deployed to construct large scale WSNs in order to sense and monitor physical environments. The connectivity between WSN sensors can be done according to different network topologies as discussed in the following sub-section.

2.2.1 Common WSN network topologies

Three types of network topologies are commonly in use in WSNs: star, peer-to-peer, and two-tier. In the star topology, a central device is in control of the network's communications. This device can be a node, an access point or any communication unit that is capable of linking nodes. In general, a central

device has larger memory capacity, higher processing capabilities and more energy resources than other nodes in the network. The main issue that restricts the use of such a topology is the presence of single point failures. The peer-to-peer network topology allows each node in the network to directly communicate with its neighbours within its communication range. If a node needs to communicate with more distant nodes, routing protocols may be used allowing some nodes in the network to act as routers. The use of such a topology allows the network to be more fault-tolerant and flexible. However, managing a WSN that has a peer-to-peer network topology can be challenging. The two-tier network topology is a combination of star and peer-to-peer topologies. The network in this topology is divided into groups. Each group is connected using a star topology. The central devices communicate using a peer-to-peer topology. The three topologies are depicted in Figure 2.1 [32, 33].

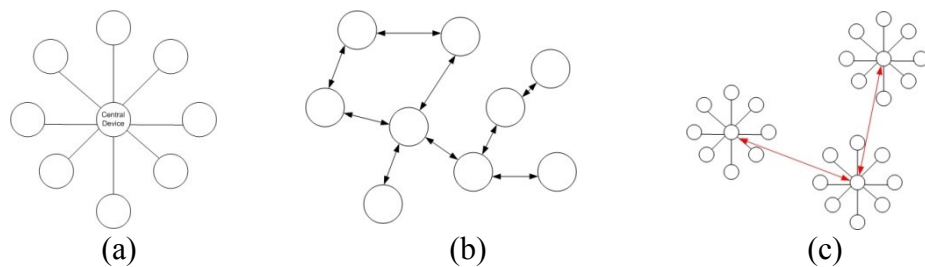


Figure 2.1: The three WSN network topologies: (a) star, (b) peer-to-peer, (c) two-tier.

2.2.2 WSN applications

Iyer et al. [34] classify the different applications of WSNs as data gathering, object tracking, and event detection. In data gathering, each sensor node sends its readings to a sink or a base station either periodically or in accordance with the sink request. Hence, the main goal for data gathering is solely to obtain information about the field of interest without in-network decision making. Intuitively, this means that the sensor nodes of a WSN do not collaborate to perform computations and/or memorisations on gathered data in this type of application. Object tracking focuses on monitoring the movement and the state of one or more objects that enter the field of interest and can use data gathering applications to achieve this goal. Event detection can be considered a higher level abstraction of the data that represents a unique occurrence or a feature in the WSN dataset. As such it is a challenging task that requires dealing with computational complexity within the resource constraints of the WSN while often also requiring real time solutions. Furthermore, the WSN datasets in close proximity to events are often being continuously generated, meaning a dataset must be analysed just-in-time before the next dataset is collected.

Chen and Varshney [16] classify WSN applications based on their requirements as follows:

- i. **Event-driven (event detection):** Event-driven applications in WSNs are most likely to be real time applications where the network is analysing sensory data to detect a specific or a set of events. The events

in these applications are usually infrequent, which means that sensors can remain in sleep mode for most of the time. However, these events are expected to be mission critical and require quick recognition and reporting. In this application category, the detection requirements are fast reporting, distributed recognition, reliable, and accurate detection, noisy patterns detection, and location information providing.

- ii. **Query-driven (sink-initiated):** In this type of application, data is gathered based on sink commands. Applications of this sort are usually interactive and mission critical. The sink can send query commands to obtain sensory data in order to take an action. The requirements of these applications are fast reporting, fast distribution of sensory data, reliable reporting, and noisy pattern detection, They can also require location information reporting.
- iii. **Continuous and periodic reporting:** This is where WSN nodes are continuously reporting information to the sink according to a specific timeframe. These applications can be either real time or asynchronous. For real time applications, the emphasis is on fast information reporting while noisy patterns and lost information are tolerated (to a certain extent).
- iv. **Hybrid models:** This is where an application may combine two or more of the applications presented above. An example of these applications is tracking-based applications where the network is interested in detecting intruders in a specific location. The requirements

of these types of applications depend on how many application types they are using. That includes all the above requirements.

2.2.3 WSN network architecture

WSNs are a network of sensor nodes. A set of protocols is necessary to perform networking functions such as processing information and communicating with other sensors. In networking, such protocols are segmented into layers that differentiate between the roles for each protocol. The standard networking segmentation is the Open System Interconnection (OSI) network reference model, which was developed to standardise the protocols of networking by the ISO organisation [35]. Hence the term ISO reference model is also used to denote the standard model. In this model, the network protocols are divided into seven layers: Application, Presentation, Session, Transport, Network, Datalink and Physical. Each layer is assigned specific roles in the networking process.

WSNs carry unique features and limitations. Consequently, other network architecture models are available for WSNs in the literature. A common model for WSNs contains five layers: Application, Transport, Network, Data-link and Physical [13]. In this sub-section, the tasks for each layer are briefly presented and discussed.

The Application layer provides high level protocols and applications that are commonly available on a WSN base station rather than the rest of the network's sensors [36]. The absence of this layer from sensors is due to the

high level of processing it requires compared to the limited computational capabilities offered by sensors. A WSN base station is expected to include much higher computational capabilities that enable the hosting of such high level processing requirements and achieve a comprehensive outcome for the whole network.

Transport layer protocols provide reliable communication services between two ends in the network. In this layer, the protocols ensure the highest possible level of Quality of Service (QoS). This can be achieved by offering services such as message segmentation, flow control, congestion control, and message retransmission for lost packets [37]. The techniques for implementing and designing transport protocols in WSNs affect the QoS and throughput of a WSN network and should vary from one application to another as different applications have different QoS tolerance levels.

The Network layer deals with routing packets within the network. This includes developing mechanisms for building routing tables to allow sensors to direct their messages and redirect incoming messages from other sensors to the proper hop. Unlike traditional networks, WSNs do not provide IP addresses and hence do not provide IP routing capabilities. The design of network layer protocols in WSNs should take into consideration network scaling, routing fairness, and security issues, along with the existence of resource constraints. In addition, network protocols in such network environments should address the problem of sensors' limited lifetime by involving fault tolerance procedures [13].

The Datalink layer or Medium Access Control (MAC) protocols are used as an underlying layer of the network layer protocols. MAC protocols control and manage the access of the shared wireless medium between WSN nodes [38]. In WSNs, the communication environment is noisy and nodes' resources are limited. Consequently, MAC protocols designed for WSNs pay attention to power consumption and attempt to reduce collisions between communicational nodes to avoid retransmission of packets [7]. Nodes in WSNs usually alternate between active mode and low power consumption sleep mode to conserve energy resources. Hence, WSN MAC protocols should consider ways of communicating with nodes that are in sleep mode in order to avoid occupying communicational channels and increase the throughput of the network [39]. Traditional MAC protocols for WSNs allow each node to have only one communication at a time. However, recent WSN MAC protocol research trends have moved towards multichannel MAC capabilities that allow a WSN node to have multiple communications at a time to support multi-task operations [39].

The lowest layer in the model is the WSN Physical layer. This layer includes all physical components of sensors such as chips, transceivers, and processors [36]. Small sensor design results in limited resources for WSNs as limited memory, computational, and communicational resources are available per sensor. Consequently, these constraints have to be addressed in designing WSN protocols in any layer in the model. Sensors mostly work in a

collaborative manner in order to tackle these limitations and create larger interactive resources to deal with detection problems.

2.3 Pattern recognition in WSNs

Duda et al. [40] and Wittenburg, et al. [41] divide the process of pattern recognition into three main stages:

- i. **Sampling:** Sensing data and describing the sensed object.
- ii. **Feature extraction:** Obtaining the main features of sensed objects. The main goal of feature extraction is to minimise the amount of data needed to describe a pattern, which then reduces the number of computations and resources needed to recognise it.
- iii. **Classification:** Using extracted features to classify objects into categories.

This section will focus on the classification aspects of the pattern recognition process in WSNs, which can be seen from different perspectives. For example, Predd et al. [42] discuss the classification of distributed learning and recognition in WSNs from the network structure perspective. Predd et al. [42] classify learning approaches as supervised, unsupervised, kernel, and distributed. Their focus is to classify distributed learning in terms of fusion centric and ad hoc network topologies. Nakamura et al. [43] present different classification perspectives for data fusion, one such application being pattern recognition. Nakamura et al. [43] classify pattern recognition into the following areas: Bayesian, Dempster-Shapfer, Fuzzy Logic, Neural Networks,

and Semantic. Any pattern recognition approach is also classified as either supervised or unsupervised learning. In supervised learning, a teacher entity provides a set of patterns to train the pattern recognition system before it encounters incoming data. In unsupervised learning the system starts classifying incoming patterns immediately. In this section, the different pattern recognition methods and schemes that have been covered in the literature on WSNs will be discussed. These schemes can be classified as threshold-based, K-nearest neighbour, statistical, neural networks, support vector machines, graph neuron, and conditional schemes. Figure 2.2 summarises the existing pattern recognition schemes for WSNs.

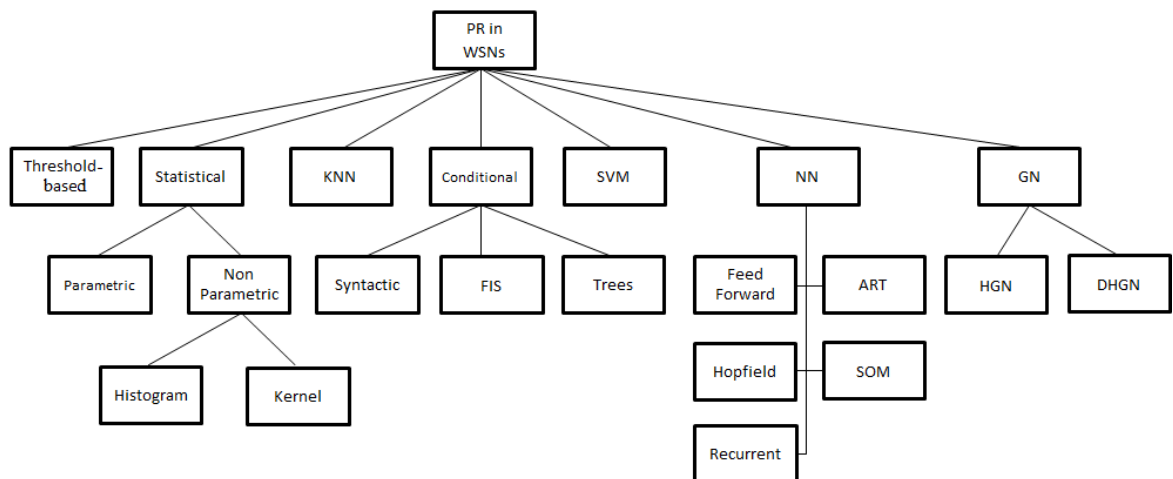


Figure 2.2: Classification of existing pattern recognition schemes for WSNs.

2.3.1 Threshold-based techniques

Threshold-based event detection and pattern recognition technique is one of the simplest and most widely used techniques in WSNs. Each sensor in this technique is assigned a threshold value or in some cases more than one threshold value. When a sensor's reading hits the assigned threshold value, it declares the detection of the event of interest. As an example of using threshold-based techniques in WSNs, Kim et al. [44] present a fence surveillance model that can detect intruders based on thresholds obtained from the average signal measurements of each sensor. If a node's reading exceeds its threshold, it sends a DETECT signal to the base station. Another example can be seen in the work of Jabbar et al. [45]. In this example, the authors proposed a threshold-based load balancing technique for routing problems in WSNs.

Threshold-based event detection and pattern recognition techniques are considered to be light-weight and simple techniques. Additionally, such techniques do not require complex network communication relationships between sensor nodes. However, three main issues are related to implementing threshold-based techniques for pattern recognition in WSNs. First, threshold-based pattern recognition cannot describe and address complex detection problems and thus it will cause a high level of false alarms when used in such problems. Second, such techniques are limited in terms of dealing with noisy patterns [46]. That WSN patterns are commonly noisy has been discussed in the literature. Hence, such limitation degrades the suitability of using such

techniques for pattern recognition in WSNs. Third, determining threshold values can be challenging in some applications [47].

Threshold-based techniques provide event detection and pattern recognition with light-weight capabilities in terms of computations and communications. However, such techniques provide problem specific solutions and they are limited in providing high accuracy detection levels in complex problems and noisy environments.

2.3.2 K-nearest neighbour

K-nearest neighbour (KNN) is one of the simplest non-parametric classification techniques [48]. Non-parametric classification assumes that the density distributions of pattern samples are unknown [40]. KNN computes distance or similarity measures between two data instances and makes a decision based on the result of the comparison. The distance between two samples is calculated according to a predefined function. One of the most popular distance functions in KNN is the Euclidian distance. This function can be represented as follows [49].

$$d(x, y) = \sqrt{\sum_{i=1}^n (a_i(x) - a_i(y))^2} \quad (2.1)$$

where $d(x, y)$ is the distance between instances x and y , n is the number of attributes, and a_i is the i 'th attribute of the instance.

The decision in a KNN algorithm is based on the number of nearest neighbours K . After calculating the distances to each neighbour, a KNN scheme will vote among K neighbouring instances to classify an incoming pattern according to labelled classes. Such voting can be described as follows [48].

$$C(x) = \text{majority}[C(N_1), \dots, C(N_k)] \quad (2.2)$$

where $C(x)$ is the class label of instance x , $C(N_i)$ is the class label of the i 'th nearest neighbour to instance x , k is the number of nearest neighbours assigned to KNN, and *majority* means the highest number of instances that have the same class label. A simple classification example using KNN is shown in Figure 2.3. In this example, seven instances are classified into two classes, C1 (Blue) and C2 (Red). The task is to classify instance x as one of these classes using KNN. The KNN uses Euclidian distance as the distance function. In Figure 2.3, (a) shows the use of $k=1$, (b) $k=3$, and (c) $k=5$. The classification process results in classifying x as C1 when using $k=1$, as the nearest neighbour is of class C1, C2 when $k=3$ as two instances out of three (majority) are of class C2, and x is C1 when $k=5$ as three instances out of five are of class C1.

KNN is commonly used as an outlier pattern recognition algorithm in WSNs [50]. This means that the normal activities of a network are modelled as pattern instances and stored in the network. A data instance is considered to be an outlier if its measure is far from the neighbouring instances that represent normal activities [50]. The main assumption that such techniques are based on is that an outlier occurrence takes place far from its neighbours [51]. The work

of Zhang et al. [52] is an example of the nearest neighbour outlier detection technique for WSNs. The authors propose a tree aggregation structure where each node sends some information to its parent. Then, the sink decides the global n outliers for the network and sends back this information to the network for verification. The process continues until all the network nodes agree on the outliers.

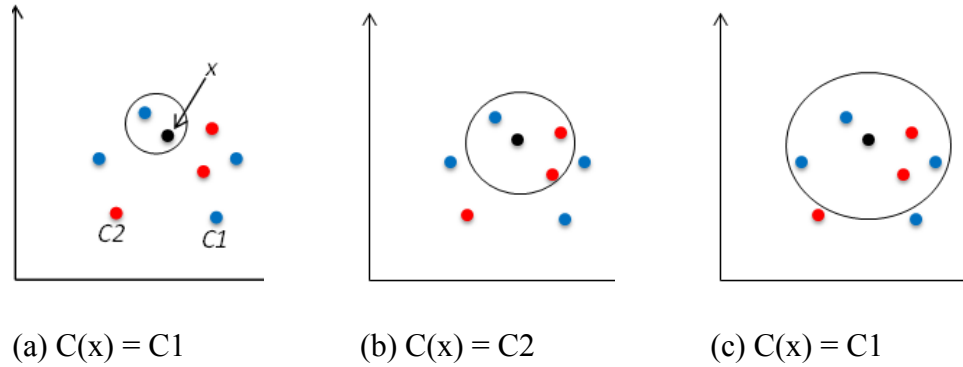


Figure 2.3: KNN classification example. (a) $k=1$, (b) $k=3$, and (c) $k=5$.

Several issues are related to using KNN as a classifier in general and as a pattern recognition technique in WSNs. First, KNN depends on the use of a distance function. In some classification problems the standard Euclidian distance function does not lead to accurate classification [49]. Hence, more complicated functions might be required. Second, KNN accuracy and complexity is dependent on the choice of the number of neighbours (k). Such dependency requires k to be tuned in such a way that it balances complexity and accuracy. Moreover, such dependency could lead to tuning k according to probability distribution of data, making the process data-dependent [48].

Third, the decision process in KNN is based on majority function. In the case of tie voting, complex algorithms must be used to break the tie decision. Such complex algorithms could lead to higher KNN complexity [48]. Fourth, KNN requires large memory resources to memorise distances between data instances, especially when used in WSNs [53]. Fifth, KNN requires high computational resources to compute distances, which makes this technique lack scalability when implemented in WSNs [50].

2.3.3 Statistical approaches

Statistical pattern recognition schemes are dependent on the probability of the occurrence of a pattern and the *Bayesian decision rule*. The theory of these approaches is based on two major assumptions: recognition decision is achieved in terms of probability, and the probabilities of occurrence values are known [40]. In these approaches, the patterns are classified in terms of *state of nature*, where each pattern may be assigned to one state. The priority probability describes the likelihood of a pattern being of a certain class (natural state). For example, if we are trying to recognise whether a shape is a rectangle or a circle, the states are rectangle and circle. The priority probability of an input pattern is determined in accordance with historical data that links the input pattern's features (such as input location) to one of the states (i.e. $P(\text{rectangle})=0.26$ if the pattern is located in the bottom of an image and $P(\text{circle})=0.74$). The decision is made based on the highest prior probability, which can be expressed as follows.

$$\text{if } P(\text{rectangle}) > P(\text{circle}), \quad \text{then the Result is rectangle} \quad (2.3)$$

In order to avoid making the same decision whenever the same situation is encountered, such statistical models use *class-conditional-probability* to reduce the classification error rate (i.e. the shape could be a rectangle even if located in the bottom of an image) and can be described as an extra feature that supports the decision making process. For example, the colour of the shape could be used as a variable that discriminates between a circle and a rectangle. In order to make a decision about an incoming pattern, statistical approaches use the Bayesian decision rule. Let x be the statistical variable, i is the class number, and C_i is class number i . The Bayesian decision rule can be described as follows [40].

$$P(C_i|x) = \frac{p(x|C_i)p(C_i)}{p(x)} \quad (2.4)$$

where $P(C_i|x)$ is the classification of the incoming pattern given x (posterior), $p(x|C_i)$ is the conditional probability density of class C_i given x , $p(C_i)$ is the prior probability of class C_i , and $p(x)$ is the evidence probability of x for j number of entered (stored) classes that can be calculated according to the following Equation [40].

$$p(x) = \sum_{i=1}^j p(x|C_i)p(C_i) \quad (2.5)$$

In practical classification problems, statistical approaches use more than one variable (i.e. feature). If the statistical relationships and dependencies between variables are known, *Bayesian belief networks* are used for solving

classification problems. Figure 2.4 shows a simple example of a Bayesian belief network. In this example, four variables A, B, C, and D and their dependencies are available. The final decision is made based on the dependencies statistics.

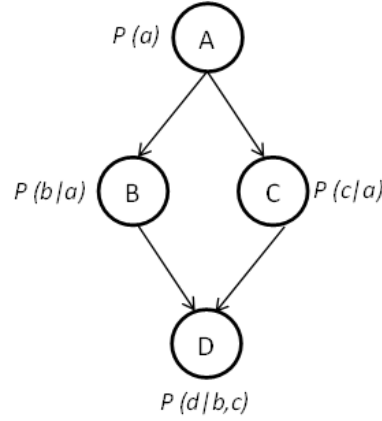


Figure 2.4: A simple Bayesian belief network.

On the other hand, *Naive Bayes* statistical classification is used when the statistical relationships and conditional dependencies between variables are unknown. In this classification method, the assumption that variables (e.g. a, b, and c) are conditionally independent is taken into account and can be represented as follows [40].

$$P(a, b|c) = P(a|c)P(b|c) \quad (2.6)$$

Non-parametric statistical classification methods assume that statistical density distribution is not available. Hence, such techniques obtain probability densities from a set of training samples. This assumption is based on the fact that in most classification problems, probability density of classes is unknown [40, 50]. Such techniques use distance thresholds based on probability

observations to decide the incoming pattern's class. According to Zhang et. al. [50], such techniques are commonly used in WSNs as outlier detection methods and can be classified as histogram and kernel approaches. Histogram approaches count the probability of the occurrence of data classes and instances and compare incoming patterns with the calculated probabilities. Kernel approaches create probability distribution functions and use thresholds to determine an instance class.

Mittal et al. [54] presents Bayesian belief network approaches for weather status detection. Their technique obtains weather attributes such as humidity and temperature values from WSNs and then applies a two-step method for classification. The first step constructs the relationship between obtained attributes and the second step performs the recognition based on the constructed relationships. Elnahraway and Nath [55] present a Naïve Bayesian distributed method to detect faulty sensors. Their proposed technique provides an outlier detection method using spatio-temporal classification where each node evaluates its readings probability according to one of many predefined classes. Wu et al. [56] presents a Naïve Bayesian based technique for medical application. In their work they used WSNs to monitor patients and detect abnormal gait patterns. Sun and Edward [57] present a non-parametric distributed statistical approach to detect specific events (e.g. loud cheering) in sports stadiums. Each sensor in a WSN deployed in a stadium decides the occurrence of an event locally, based on noise levels, and then sends the result

to a cluster head. The cluster head then detects the event based on the optimal median amongst the collected information from all participating sensors.

Using statistical pattern recognition approaches in WSNs carries a number of challenges. In most classification problems, such as the ones in WSNs, the prior knowledge of probability distribution is rare [40, 50, 58]. Consequently, implementing most parametric statistical approaches becomes unfeasible due to the lack of such knowledge. Non-parametric statistical approaches are more feasible since such approaches do not require prior information about probability distribution. However, the accuracy of these techniques is highly dependent on the number of available training samples as they construct probability distributions based on available samples [40, 58]. In WSNs, numbers of samples of patterns and events are limited due to the randomness feature of information obtaining in WSNs [10, 50]. That is, the occurrence of an event may be captured on rare occasions. Moreover, obtaining enough information about an existing pattern to construct probability distributions is limited due to WSNs' communicational and computational limitations [42]. These limitations make the use of non-parametric approaches in WSNs challenging. In addition, some non-parametric approaches such as histogram techniques require high communicational overheads to obtain histogram information [50]. Such requirements contradict the limited communicational capabilities of WSNs. Other non-parametric approaches such as kernel techniques require defining thresholds in order to estimate probability densities. However, determining such thresholds may be challenging [50].

2.3.4 Neural networks

Neural networks (NNs), also referred to as Artificial Neural Networks (ANNs), are computational methods that offer parallelism in pattern learning and recognition [43]. Associative Memory (AM) is one neural network approach that is capable of storing and retrieving patterns in a distributed manner. AM has been discussed in the literature as being able to provide pattern recognition solutions based on the capability of recalling stored templates [59]. Additionally, AM networks are capable of dealing with noisy patterns and considered to be a robust solution [60]. Simply, AM depends on using small memory chunks available in computational units to achieve distributed memory management. In this sub-section, various types of NNs are discussed.

2.3.4.1 Feed-forward NNs

The Feed-forward is a supervised neural network that consists of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to each neuron in the above layer by variable weight values [61]. Figure 2.5 shows the structure of feed-forward networks [61, 62]. Such networks are commonly used in pattern recognition applications [40, 63]. The computations in feed-forward NNs take place in the hidden layers. Each hidden layer calculates the inner product of inputs with weights, which is called the network's activation function. The connections between neurons are usually called synapses and the values of these synapses are called synapse weights,

which are calculated using a non-linear activation function. In this approach, the activation of a neuron depends on a predefined weight and a bias unit assigned to neurons [40].

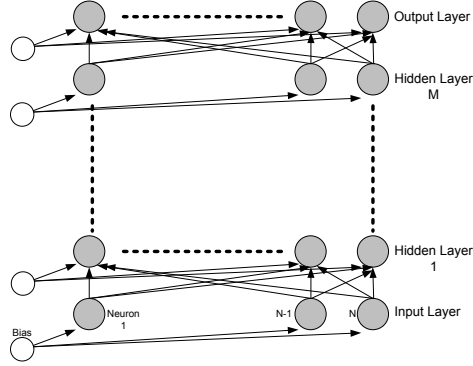


Figure 2.5: The structure of a feed-forward NN that has N inputs and M hidden layers [61].

Awad et al. [64] use a feed-forward based recognition scheme for localisation and location estimation in WSNs. The proposed approach uses the feed-forward network to analyse received signal strength indicators (RSSI) to estimate the distance between two nodes. Rajkamal and Ranjan [65] use feed-forward networks to classify exchanged packets between sensors based on the nature of incoming data in order to control the traffic flow in a WSN. Radial basis function networks (RBFNs) are one type of the feed-forward network. RBFN consists of three layers: input, hidden, and output [66]. Ishizuka and M. Aida [67] use RBFN to achieve efficient low-power sensor placement. Tran and T. Nguyen [68] use RBFN as a kernel function for a support vector machine (SVM) technique in the localisation of WSNs' nodes.

2.3.4.2 Hopfield networks

The structure of Hopfield NNs [69] is based on a single-layer network where each neuron is connected to all other neurons. Figure 2.6 shows the structure of the Hopfield network. Each connection in the network is measured as a weight that is assigned during the pattern learning phase. Both connections that go from one neuron to another must have the same weight. The weight can be calculated according to the following function.

$$W_{ij} = \begin{cases} \sum_{r=1}^M P_i^r P_j^r, & i \neq j \\ 0, & i = j \end{cases} \quad (2.7)$$

where W_{ij} is the connection weight, P_i^r and P_j^r are the pattern number r for neurons i and j respectively, and M is the total number of patterns [59].

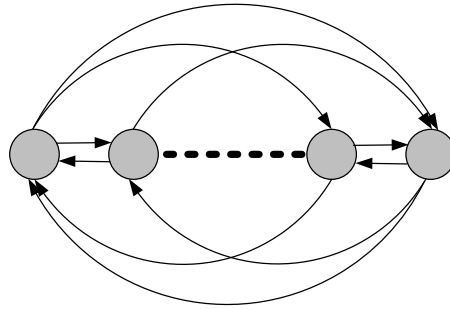


Figure 2.6: The Hopfield network structure [70].

The above function describes the Hopfield network in a discrete representation. G. Massini [70] argues that the Hopfield model is limited in terms of the number of patterns that can be stored and detected. A continuous model of a Hopfield network is possible to achieve by taking the differentiation

of the above discrete equation. In this model, a global convergence of the network is not guaranteed [70].

Hopfield neural networks are one of the simplest and most common types of NNs that have been used for pattern recognition problems in WSNs. For example, Chen et al. [71] used Hopfield networks for target tracking applications. The authors propose the use of a data fusion algorithm based on Hopfield networks to construct the relationships between a WSN's sensors' readings and existing target tracks. A target's track can be detected based on the obtained relationships. In another example, Tisza et al. [72] propose a multicast routing protocol for WSNs using Hopfield NNs. The proposed algorithm is based on the assumption that the routing information obtained by the network is incomplete. The proposed routing algorithm obtains the incomplete link's metrics from the WSN and uses Hopfield networks to create the best routing tree that fulfils certain quality of service (QoS) criteria (e.g. routing delay). Levendovszky et al. [73] propose a Hopfield NN-based datalink layer algorithm for WSNs. The proposed algorithm attempts to schedule data forwarding in WSNs based on specific QoS metrics. The obtained QoS metrics are fed into Hopfield networks in order to find data packets' optimum forward scheduling times.

2.3.4.3 Recurrent neural networks (RNN)

Recurrent Neural Network (RNN) is a multi-layered structured NN, also called a feedback neural network. The term feedback means that RNN

output is fed back into the input in order to reduce the error percentage and enhance the recognition accuracy [40]. Such links (i.e. from output to input) are not available in standard NNs [74]. Figure 2.7 depicts the structure of an RNN that has input layer, one hidden layer, and one output layer [40].

Connor et al. [75] classify RNNs under two categories: standard and relaxation. Standard RNNs work as standard NNs with feedback links. Relaxation RNNs perform learning and recognition continuously until feedback inputs reach a fixed predefined class. This guarantees a predictable convergence time. However, in some applications, such as time series prediction, it is impossible to achieve this target.

Barron et al. [76] and Moustapha and Selmic [74] use RNN-based techniques for fault detection in WSNs. They use an RNN to model a sensor node and its related communications with other nodes in the network. Their aim is to use previous output samples from communicating sensors in addition to the current and previous output samples of the modelled sensor as an input to the RNN model in order to detect failures in a dynamic environment. Another example is the work of Raju et al. [77]. The authors present a faulty data detection system for WSNs using RNN. The proposed system obtains the output of a sensor's neighbours to be fed as input into an RNN model in order to detect faulty information.

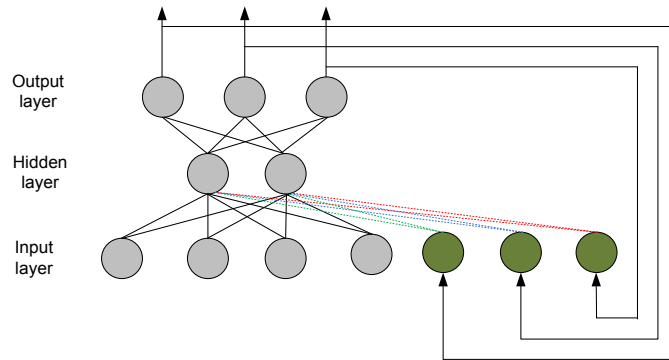


Figure 2.7: RNN structure [40].

2.3.4.4 Adaptive resonance theory (ART)

Adaptive resonance theory (ART) is a multi-layered unsupervised NN approach that overcomes the limited learning scalability of NNs. This limitation is called the stability-plasticity dilemma [78]. ART network architecture consists of three main layers: input, comparison, and recognition as shown in Figure 2.8. The input layer receives the pattern and stores it. The connectivity between input and comparison layers is one-to-one, meaning that each neuron in the input layer is connected to a corresponding neuron in the comparison layer using non-modifiable weights. On the other hand, each neuron in the comparison layer is connected to all neurons in the recognition layer using modifiable weights. A feedback connection is also available in the ART structure where each neuron in the recognition layer is linked to all neurons in the comparison layer. In addition, the architecture uses gain modules (G1 and G2) and the orienting subsystem (R). These are the signals

that control activating and deactivating neurons in the comparison and recognition layers [79]. The neurons in the comparison layer are fed with three inputs: the input pattern, the feedback pattern from recognition layer, and the gain value $G1$. Neurons in the recognition layer will receive two inputs, from comparison layer and $G2$. Recognition is based on calculating the weights and determining the winning neuron in the recognition layer. The highest weight neuron will be activated and compared to the stored patterns to find a match. If no matched pattern is found, the neuron will be de-activated and another neuron will be activated and compared. This process continues until the network finds a match. Otherwise, the incoming pattern will be stored [78].

Kulakov and Davcev [80] use ART networks as classifiers to detect unusual WSN nodes' behaviour in order to identify intruders. Yuan and Parker [81] present an ART-based WSN detection system to detect intruders in an unknown environment. Kumar et al. [82, 83] implement ART networks in WSNs to classify patterns in order to achieve clustering aggregation in unknown environments. From the above description and examples, we can see that ART networks offer scalability in terms of the number of stored patterns. In addition, these networks are useful for classifying patterns that have no prior information, such as statistics [82, 83]. However, there is no guaranteed convergence time, which degrades the suitability of such a scheme for use in WSNs.

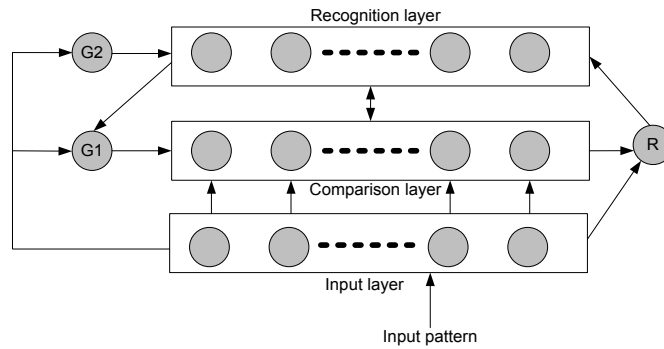


Figure 2.8: ART network architecture [78].

2.3.4.5 Self-organising maps

Self-Organising Maps (SOM) [84, 85], also called Kohonen maps, are unsupervised learning mechanisms and considered to be a type of artificial NN. The neurons in the network are arranged in a regular manner and can be the shape of one- or many-dimensional spaces. Each neuron in the network is assigned a random weight in the initialisation. The training process of SOM goes through two main steps:

- Competition, where training samples are presented to the network and compared to neurons' weights, with the neuron of the maximum value considered to be the winning neuron. The comparison in this phase is controlled by a discrimination function such as Euclidean distance or inner product.
- Adaptation, the winning neuron's weight, is updated in accordance with the learning rate parameter and the neighbourhood function.

The learning process goes in iterative cycles where the learning rate and the number of neighbours are reduced at the end of each iteration [86]. After the completion of the learning process, it is possible to present patterns to the network to perform classification operations. The weights of each presented pattern are compared with each neuron's weight and the neuron that has the closest weight is classified as the input vector.

Examples of using SOM in WSNs can be seen through the work of Giorgetti et al. [87] who proposes a localisation mechanism that determines nodes' coordinates in WSNs based on SOM. Another example is the work of Postolache et al. [88] who uses sensor networks and a SOM mechanism to confirm a sensor's failure and detect pollution events. Despite the classification properties offered by SOM, it requires centralised processing to compare weights and to come up with the output class. Thus, tailoring SOM for use in WSNs may be resource exhaustive.

2.3.4.6 Issues related to implementing NNs in WSNs

Neural Networks (NN) provide parallel pattern recognition capabilities for multiple problems. However, there are a number of issues that degrade the suitability of such techniques for pattern recognition in WSNs. One of the most prominent issues is the tightly coupled connectivity between neurons. In a single layered NN such as a Hopfield network, each neuron is connected to every other neuron in the network. In a multi-layered NN such as feed- forward networks, each neuron in a layer is connected to each neuron in an upper layer.

Such tight connectivity between neurons will require a high number of network communications between WSN nodes, which means high power consumption. In addition, such connectivity limits a WSN that implement a NN technique from scale up in terms of network size.

Pattern recognition using NN techniques involves an iterative process. This means that a network performs actions such as weight calculations in repetitive steps until reaching an optimum status. The number of these steps is usually unpredictable and in some cases is not guaranteed to lead to an optimal solution. Consequently, the convergence time of an NN technique is high. Hence, the suitability of such techniques in real time WSN applications is limited. Moreover, such iterative processes involve a large number of computations which will result in resource consumption when implemented in resource-constrained networks such as WSNs.

In some NNs such as Hopfield networks and some types of the feed-forward networks, predetermined synaptic weights and relationships between nodes are required. In addition, NNs in general require a large number of training samples in order to correctly classify incoming patterns. These requirements may be challenging in some applications, especially for environments where patterns are expected to occur randomly.

Generally, NNs offer distributed and parallel pattern recognition capabilities. However, the performance of such schemes is affected by the large number of communications, iterative processing, and high computational resources involved, as well as the non-guaranteed convergence time and the

predetermined weights and large number of pattern training samples that are required. These factors and requirements make the implementation of such schemes in resource-constrained networks such as WSNs either unfeasible or challenging.

2.3.5 Support vector machines

Cortes and Vapnik [89] present the principle of support vector machines (SVM) as a learning and classification mechanism. An SVM network maps input vectors to high dimensional feature space in a non-linear manner. Then, a linear decision surface is created in this high space by creating one or more hyperplanes to perform class separation. The optimal hyperplane between two classes can be obtained by maximising the margin between the classes' points.

Figure 2.9 shows an example of a two-dimensional separation problem for two different classes. The points located on a class margin line are called support vectors and the distance between two classes' margin lines is called the optimal margin. To perform classification, an input pattern is compared with support vectors to determine which class this pattern should belong to. In practice, SVMs use a classification function which can be calculated based on the kernel representation. Hence, the choice of the kernel will have impacts on the classification process of the network [90].

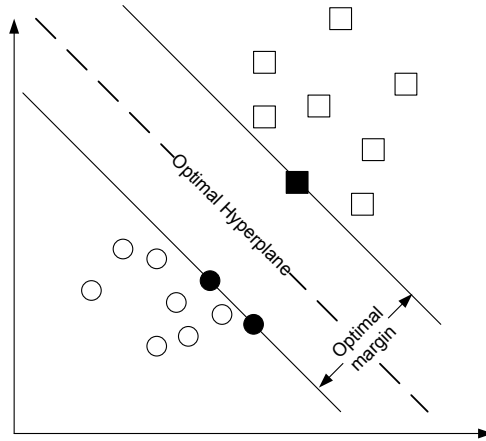


Figure 2.9: SVM classes separation for two classes in a two-dimensional space. Black vectors (samples) represent the support vector for each class [89].

Tran and Nguyen [68] use SVMs with an RBFN kernel for error tolerance localisation in WSNs. The authors propose the use of connectivity information, such as number of hops, as metrics to classify WSNs and estimate sensors locations. Xue et al. [91] propose target classification mechanisms based on SVMs that overcome samples' false rates in WSNs. The authors propose the use of energy consumption metrics to construct an SVM-based classifier for WSNs in two paradigms, namely, centralised and distributed. The centralised paradigm represents the traditional SVM classifier. In contrast, the distributed method attempts to use samples (i.e. nodes) close to hyperplanes in order to reduce classification overhead costs. The main goal of the distributed classifier is to allow a set of sensors to communicate with a set of cluster heads to construct an SVM classifier. Abu Sajana et al. [92] also use SVMs to detect physical intrusion attacks on WSNs containing PIR sensors. The goal is to reduce false alarms caused by detecting windblown vegetation. They propose

the use of HAAR transformation and frequency binning along with SVMs to solve the addressed classification problem.

The main challenge of using SVMs for classification problems in WSNs comes from the fact that a technique that implements an SVM classifier requires centralised processing capabilities in order to create hyperplanes and classify incoming patterns based on computed information. Another challenge is related to the number of training samples required. In order to create separating hyperplanes between classes and correctly classify instances, SVMs require a large number of training datasets. Such requirements may be challenging in applications that expect patterns to occur randomly. Another challenging issue when using SVMs is their dependency on kernel functions. The use of kernel functions will tie SVM techniques to the issues related to the kernel itself. For example, an SVM technique that implements an NN method as its kernel function will suffer from tightly coupled connectivity between nodes and the iterative processing associated with NNs. Hence, the choice of the type of kernel function plays a very important role in determining the suitability of an SVM technique for use in WSNs.

2.3.6 Graph Neuron (GN)

Graph Neuron (GN) [93, 94] is a scheme that creates AM in a fully parallel-distributed manner over fine grained WSN. GN nodes only communicate adjacently and in a loosely coupled fashion. Hence, GN offers light-weight one-shot learning capabilities in a decentralised manner. These

Despite the light-weight pattern recognition capabilities offered by GN, the recognition accuracy of GN is affected by the limited perspective of each neuron as each node only knows about its immediate neighbours. This leads to the *crosstalk* problem. For example, if a GN network with a pattern size of five memorised patterns *abcdf* and *fbcd*e, the network would falsely recall the pattern *abcde* that was never presented to the network. Using GN in WSNs is also affected by the constraint that each node is required to communicate with a single entity (i.e. base station) in order to perform pattern recognition operations. Such requirements increase the number of communications and overhead over the network.

2.3.6.1 Hierarchal GN (HGN)

Hierarchal Graph Neuron (HGN) [95] fixes the crosstalk problem by using a pyramidal framework for obtaining a higher perspective on the incoming pattern. HGN creates a set of layers above the neurons that receive the incoming pattern. The goals are to provide higher oversight over an incoming pattern and to minimise direct communications between nodes and the base station. The structure of HGN is built using layers of GN arrays in a pyramidal logical shape that allows a single node (the top node in the pyramid) to classify the incoming pattern and communicate with the base station. Figure 2.11 shows an example of simple binary HGN that handles a five elements pattern size.

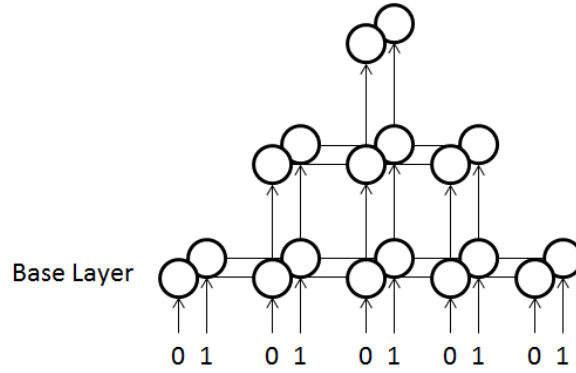


Figure 2.11: Simple HGN structure for a 5 elements binary pattern.

The incoming pattern is firstly processed by the base layer of HGN (i.e. base GN). And then each neuron sends its calculations to its corresponding higher level neuron. This process continues until the top node of the structure. This allows neurons in higher levels to build up a higher knowledge about the incoming pattern. The top node decides the pattern's index based on a given command from the base station (memorise or recall). However, if the top node fails to classify the pattern, the base node communicates with lower level nodes to vote for an answer. It is noticeable that the neurons in layers higher than the base layer monitor and manage nodes. That is, these nodes do not receive pattern elements. Instead, these nodes receive index numbers calculated by the base layer (and lower level) nodes.

HGN solves the problem of crosstalk associated with GN schemes. However, the size of HGN can scale substantially with the increase in pattern size due to the use of managing neurons in its structure. If the pattern size is S and the number of possible values of a pattern element is v , then the size of the

HGN network ($HGNsize$) can be calculated according to the following equation [95].

$$HGNsize = v \left(\frac{S + 1}{2} \right)^2 \quad (2.8)$$

HGN also attempts to reduce direct communications between each node in the network and the base station. However, with the presence of noisy patterns, an HGN network's top node will fail to classify the incoming pattern and the base station will communicate with nodes in lower layers to vote for an answer. Consequently, an HGN scheme's communications are affected by noisy patterns.

2.3.6.2 Distributed HGN (DHGN)

Distributed HGN (DHGN) [96] attempts to solve the large scale of HGN and reduce the number of direct communications required for voting. DHGN splits an incoming pattern into sub-patterns so they can be processed by multiple HGN networks. Figure 2.12 shows an example of a DHGN structure for a pattern size of 20 that has been split into 4 sub-patterns and sent to 4 HGNs where each HGN processes 5 elements. Each HGN network processes the assigned sub-pattern and presents its final result through its top node. The base station conducts a voting process between the top nodes of the GN networks in order to make a decision on an incoming pattern.

DHGN reduces the number of nodes required for constructing the network by limiting the number of managing neurons. However, the use of

managing neurons leads to an increase in DHGN size with increase of pattern size. For example, in a uniform distribution of pattern size S , let Sp be the sub-pattern size and n HGN networks. Accordingly, a DHGN network size (DHGNsize) can be calculated according to the following formula.

$$DHGNsize = n.v \left(\frac{Sp + 1}{2} \right)^2 \quad (2.9)$$

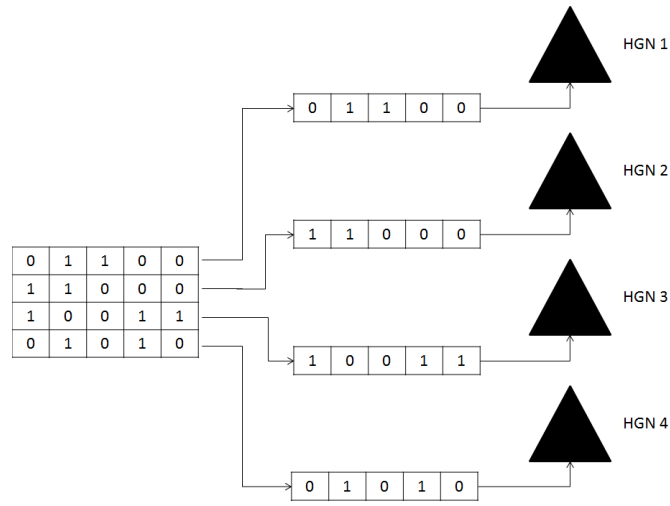


Figure 2.12: DHGN structure for a 20 bits pattern size that has been divided into 4 sub-patterns.

DHGN adopts an HGN scheme. It has been shown that when a pattern is distorted (i.e. a noisy pattern) the top node of an HGN will not make a decision about the incoming pattern. Instead, the base station conducts a voting process that involves nodes in lower layers in order to inspect the result. In contrast, DHGN avoids such processes in order to limit direct communications with the base station and also to speed up the detection process. The major problem that a DHGN network may encounter is when a distributed noise is

present. In such a case, a DHGN network may fail to reach a conclusion about the incoming pattern. For example, if each sub-pattern in the example given in Figure 2.12 has been changed by at least one bit, each HGN network would fail to reach a conclusion regarding the incoming pattern. In other words, all top nodes will give the result 0 (i.e. fail to recognise the pattern). Since the base station only conducts the voting process amongst the top nodes of all HGN networks and does not involve lower layers, the network is unable to determine the incoming pattern.

GN involves a limited number of communications and computations in performing learning operations. Such a feature makes GN a very good candidate for pattern recognition applications in resource-constrained WSNs. However, the accuracy of GN is limited due to the limited information available for each node. HGN and DHGN provide higher accuracy levels by involving a hierarchical network structure. Communications in both schemes are maintained at low numbers by adopting parallel and distributed mechanisms. However, the scalability of HGN and DHGN schemes is not best suited for large scale WSNs as the number of required nodes increases exponentially with the increase of the problem (pattern) size.

2.3.7 Structural and conditional methods

There are a set of structural and conditional classification methods in the literature that have been used for classification problems in WSNs. These methods attempt to create a relationship between pattern elements and are

commonly used when non-metric data is available [40, 58]. These techniques can be categorised as syntactical, fuzzy logic, and decision tree methods.

2.3.7.1 Syntactical classification

The syntactic model describes the relationship between sub-patterns and patterns by creating structural rules. It adopts language theory where letters form words and words form sentences based on grammatical rules. In this model, primitive elements and sub-pattern relationships are analysed to provide pattern recognition. The main challenge in the syntactic approach is to describe the relationships (rules) between sub-patterns so as to provide the capability of pattern recognition and identify primitives that describe patterns [97]. Such analysis is performed by using different schemes such as NNs, tree grammars, transformations, and more [98].

Latha et al. [99] use the syntactical method in semantic tracking for wildlife preservation using WSNs. The syntactical method is used as a processing stage that checks a node's detection with other nodes in the same cluster. Syntactic pattern recognition offers complex pattern recognition if there is no suitable statistical method available. However, grammars and recognisers (recognition) are complex, especially with noise [100]. Another issue with this technique is the large amount of training data required for training and creating relationships between sub-patterns [58].

2.3.7.2 Fuzzy logic

A system that implements fuzzy logic is usually called a fuzzy inference system (FIS). An FIS is capable of making conclusions by mapping inputs to outputs with the aid of membership functions, fuzzy sets, and rule base [101]. FIS is built up based on three main components: rule base, membership functions, and reasoning mechanisms [102]. The process of an FIS starts with classifying inputs to fuzzy sets in accordance with membership functions. For example, a temperature reading could be classified as high, medium or low. Then, the rule base is used to make conclusions based on the classified inputs. Rule base consists of a set of IF-THEN rules that take two or more variables to come up with a conclusion. An example of a fuzzy base rule can be described as follows.

$$\text{If } x \text{ is } A \text{ and } y \text{ is } B, \text{ then } z = f(x, y) \quad (2.10)$$

where x and y are the classified inputs and z is the output of the FIS. Deriving such rules for a FIS may requires prior knowledge about the relationships between variables [101]. Implementing these rules on variables is called fuzzy reasoning or an approximate reasoning mechanism of a FIS.

Marin-Perianu et al. [101] present a WSN activity recognition system based on FIS to assist workers in car assembling and training. Zarei et al. [103] propose a FIS congestion control scheme for WSNs to identify malicious node activities. Xiufang et al. [104] use FIS to measure the distance between WSN nodes in order to achieve better localisation. These are some examples of using

FIS for inference and recognition in WSNs. The main challenge for FIS is to derive the IF-THEN rules and fuzzy sets. According to Nakamura et al. [43], FIS is usually used to control neural networks' learning rates rather than being used for recognition. This will lead to the same issues with pattern recognition in WSNs that are present in NNs. In addition, in most WSN applications, concluding rules may be challenging.

2.3.7.3 Decision trees

Decision tree approaches are constructed from a set of nodes that are logically arranged in a tree-like shape. Each node makes a decision about the incoming pattern feature and, based on that decision, the process questions the next feature in lower level nodes. Figure 2.13 shows a simple decision tree example. In this example, five animal classes, eagle, sparrow, monkey, lion, and sheep are to be recognised by using four features: presence of wings, size, number of legs, and being a predator. The tree in the example shown in Figure 2.13 has four levels, including the root of the tree. The number of levels determines the depth of a decision tree. One of the most common decision tree structures used for classification is the binary decision tree. In such a structure each node makes one out of two decisions and inspects a single feature at a node to reduce complexity and time of recognition[58].

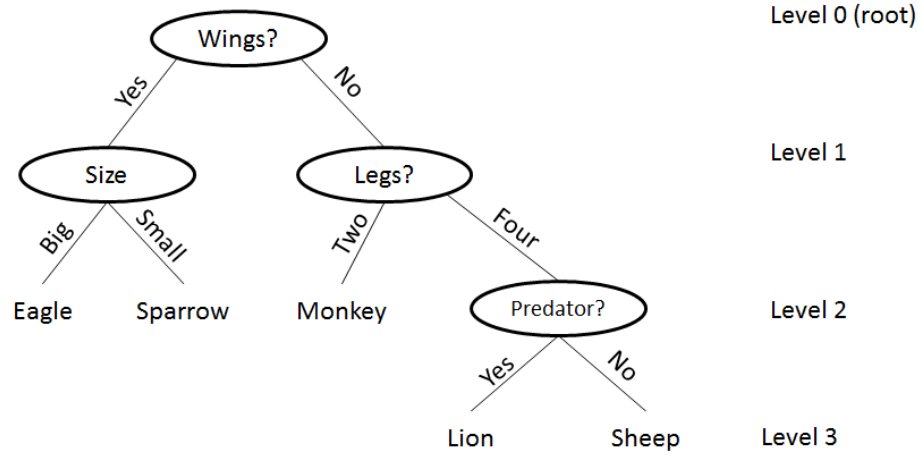


Figure 2.13: A simple decision tree example for animal classification.

As an example, Bahrepour et. al [105] propos a WSN event detection mechanism based on a decision tree technique. The proposed scheme distributes features into several trees where each tree makes one decision. Finally, a voting process takes place between the results obtained by the trees to determine the detected event. Decision trees are expected to involve limited computations and communications. However, decision tree techniques are affected by noisy patterns, which increase the scheme’s complexity, especially for large scale trees [106]. Hence, decision trees are more useful as decision making processes on top of another pattern recognition process.

2.4 Requirements of Pattern Recognition in WSNs

Pattern recognition in WSNs is affected by the limited physical design of sensor nodes, the nature of WSNs and the type of patterns a WSN is

attempting to deal with. In this sub-section, the requirements of performing pattern recognition in large scale WSNs for real time applications will be discussed.

Sensor nodes are generally designed to be small in size. Such design restricts the resources that can be included in each sensor. As discussed earlier, a sensor consists of four main components: a processing unit, a communication unit, a memory unit, and an energy source. The limited size of a sensor results in the limited size of these components. Consequently, each task assigned to each component can only use a restricted amount of resources. The energy source is one of the components that most affects the performance of a sensor and the design of a WSN. Generally, a sensor uses a battery that has a short lifetime. Moreover, in most applications, batteries are not likely to be replaced, which means that the lifetime of the battery determines the lifetime of the sensor. Since the energy source of a sensor is limited, energy consumption caused by another sensor's components must be reduced.

A sensor's communication is considered to be the most energy consuming task, and can drain the sensor's energy resources [18]. Hence, a pattern recognition scheme in WSNs should involve a limited number of communications per sensor in order to increase the lifetime of sensors. Computational capabilities of a sensor node in a WSN are constrained due to the small sensor size (small processor) and limited energy available. Consequently, involving large amounts of data processing in a sensor is an exhaustive task that will shorten the lifetime of the sensor. This is not only

because data processing requires energy, but also because the sensor will be kept in active mode for long periods of time. In addition, the higher the processing assigned to a sensor, the more time the sensor needs to obtain a result. If the amount of processing is large, the time needed to get a result out of this processing may be unexpected. Hence, a pattern recognition algorithm should involve a controlled amount of data processing for each sensor aligned with the sensor's computational resources in order to avoid energy consumption and to ensure a timely result. The memory size of a sensor is intuitively small. As a result, each sensor should hold the minimum amount of data it needs to process and detect patterns in a WSN pattern recognition scheme.

Other requirements for pattern detection result from the nature of the WSN network design. As WSNs are deployed in large numbers, network scaling is an important property for designing a recognition scheme for WSNs. Size scaling requires managing the way sensors are going to communicate with each other. The number of communications involved in a WSN scheme design is crucial as it will determine the number of communications each sensor is going to handle. This will have ramifications for the sensor's lifetime as well as the time needed to obtain a final result from these communications. In real time applications, convergence time is highly important. In recognition processes, sensors either send data to a fusion centre or to other sensors in the network in order to conclude pattern detection. Consequently, the convergence time of the network is highly dependent on the process of delivering

information from one point to another. Generally speaking, scaling a WSN should maintain a restricted method of communication to conserve energy resources and speed up the recognition process to support real time recognition applications in WSNs.

A pattern recognition scheme should have some invariant features. In WSNs, the need for such features increase because WSNs are dynamic and the nature of monitored fields of interest is changing. In other words, a stored pattern in a WSN pattern recognition scheme could appear in different form, such as location change or size dilation, in the field of interest. Or the topology of the WSN network or sensor locations may change, meaning the information stored within the network will have different distribution and relations. Another problem associated with the nature of WSNs is the restricted number of training samples available as events generally occur in some form of randomness [10, 50]. Hence, designing a pattern recognition scheme should address the restricted amount of training data available as well as the changing environment in WSN networks and fields.

Noisy patterns are another problem associated with large scale WSNs. Noisy patterns are a result of the monitoring environment and the limited lifetime of sensors. As a result of noisy patterns, damage to sensors, dead sensors, and lost packets could cause the loss of some parts of the detected pattern. Tolerance management is required to reduce the effect of lost parts of incoming patterns. It is worth noting that different applications may need different tolerance levels. However, in WSNs in general, a recognition scheme

should be able to detect events and patterns even if some parts of the detected pattern are lost.

In general, the main requirements of a pattern recognition scheme in large scale WSNs suitable for real time detection can be summarised as follows:

- restricted communications,
- restricted computations,
- limited memory requirements,
- ability to scale in terms of network size,
- Predicted convergence time,
- means to addresses invariance properties for dynamic networks and changing patterns,
- ability to address randomness problems, meaning that the scheme should maintain high accuracy with a restricted number of available training samples,
- ability to detect complex patterns, and
- ability to detect noisy patterns.

2.5 Comparing Existing Schemes

This sub-section compares the different presented pattern recognition schemes in section 2.3 for WSNs based on the requirements listed in the previous sub-section. The main goal of this research is to present recognition

schemes that can be implemented on large scale WSNs for real time applications and to support decision making processes in intelligent systems.

Most existing schemes are able to implement a number of nodes equal to the pattern size S . For example, Hopfield networks allow input/output neurons of size S . In contrast, HGNs and DHGNs require a larger number of nodes to adopt the same patterns, as can be seen from Equations 2.8 and 2.9. The higher number of nodes in HGNs and DHGNs is the result of requiring higher level neuron positions and the need to have one node for each possible value v in each position. It can be concluded from Equations 2.8 and 2.9 that the number of nodes (or sensors) grows exponentially with the increase of pattern size and the number of possible values for each pattern element.

On the other hand, the number of communications involved in neural networks such as Hopfield networks is high due to tightly coupled connectivity and iterative processing. The number of communications required for a Hopfield network is $(S \times (S-1))$ as each node is connected to every other node. Intuitively, the number of communications grows exponentially with the increase of pattern size as this number is related to the square of the pattern size. Since neural networks require iterations to reach an optimal state, communications between neurons are repeated several times, resulting in high communicational demand that would be exhaustive if implemented on sensors. Some statistical approaches such as histogram methods also involve a large number of communications in order to collect specific information from the network.

From a computational perspective, most of the existing schemes either provide distributed processing or require centralised processing. SVM, for example, requires centralised processing in order to create the required hyperplanes and classify patterns. Statistical approaches require global information to be available on a centralised component to compute distributions and perform recognition. There have been attempts to distribute statistical models amongst sensor nodes in WSNs and compute these distributions locally before sending the information to a fusion centre or a base station. The work of Luo et. al [17] is an example of this technique. However, the accuracy of such techniques would depend on the physical communication medium's noise tolerance and the thresholds computed to perform computations locally in sensors. Neural networks offer parallel and distributed functionality in terms of computations. However, the iterative process of neural networks requires a high amount of data processing. KNN techniques can be seen as simple, distributed approaches for pattern recognition. However, the computational complexity of a KNN scheme depends on the number of neighbours k . The higher the value of k , the more complex the scheme becomes. Hence, tuning the value of k plays a crucial role in determining a KNN scheme's computational simplicity.

Decision making support and real time applications require fast pattern detection. In this area, GN approaches such as HGNs and DHGNs offer one-cycle recognition that suits such applications. On the other hand, neural networks, SVM, and decision trees recognition schemes may require more time

to converge compared to other existing schemes. A neural network's convergence time to an optimum state depends on the number of iterations involved. A single iteration involves communications and computations to be performed by neurons. These activities can be time costly as the number of iterations grows. In SVM schemes, recognition time depends on the selection of the kernel. If the kernel chosen is one of the time costly techniques, such as neural networks, the detection time will intuitively increase. The choice of kernel in this case will be a trade-off between time and other factors such as accuracy. For decision trees, recognition time depends on the depth of a tree. The depth of a tree is the number of levels needed to perform recognition and depends on the number of attributes the tree is inspecting. The more attributes to inspect the greater the depth of the tree and hence the more time it takes to conclude a decision about a pattern. In addition to the depth of a tree, the method used to inspect each attribute affects the time cycle of detection.

The number of available training samples is commonly restricted in WSNs. Most existing detection schemes require a large amount of data to correctly recognise and classify patterns. Statistical approaches use training samples to construct distribution probabilities. The more samples the scheme is trained with, the higher the accuracy it achieves. Similarly, SVM requires large amounts of data to create separation hyperplanes. A limited amount of data could result in inaccurately setting hyperplanes and create large gaps between classes. Neural networks share the same requirement in order to accurately create weighting matrices. The limited number of training samples in this case

affects the invariant property of a recognition scheme. Statistical, SVM and neural network approaches are the best candidates for offering this invariant feature compared to other existing schemes. However, this feature is entangled with presenting a large number of training samples to a network implementing such schemes.

Memory requirements per node (or sensor) are limited in most existing schemes. However, in KNN, each node keeps information about distances to each of its neighbouring nodes. The amount of memory needed for each node in this case will depend on the value of k and the number of classes. In HGN, memory requirements increase in higher nodes in the hierarchy. In the base layer (i.e. input layer) each node is expected to hold up to (2^V) in its memory (v is the possible number of a pattern's element values) as each sensor communicates with its two direct neighbours. Each node in the top position of the hierarchy of the HGN is expected to hold up to $\frac{\text{Number of training samples}}{v}$.

The aim of event detection in WSNs is to reduce the amount of false alarms. Simple schemes, such as threshold-based scheme, seem to be perfect for simple problems. On the other hand, these schemes fail to deal with complex patterns, leading to false alarms. The nature of WSNs and the fields they monitor introduce recognition schemes to more complex problems. Sensors could run out of energy or lose information because of the noise in the physical transmitting medium. Consequently, schemes should be capable of overcoming such challenges and offer recognition capabilities despite the lost information. Several schemes encounter degradation of accuracy caused by

such problems. For example, a DHGN scheme might inaccurately classify patterns when distributed noise is present. If each cluster of a DHGN network is presented with noise, cluster heads will not be able to conclude sub-pattern detection and the final voting of the process could lead to inaccurate detection. Another example is decision tree schemes. These techniques may fail to correctly classify noisy patterns in large scale networks.

It can be seen that different schemes have different limitations in regards to the requirements of pattern recognition in large scale WSNs. Table 2.1 shows a comparison between existing pattern recognition schemes in WSNs. It can be seen from the table that none of the existing schemes can fulfil all the requirements set. Consequently, new schemes need to be proposed if we are to the problem of interest.

2.6 Possible Solution

Performing pattern recognition in WSNs requires tackling two problems: correctly classifying patterns and restricting use of constrained resources. Solving the problem of pattern recognition in WSNs is seen as a trade-off between accuracy and resources exhaustion [28, 107]. Existing solutions do not address the resource-constrained nature of WSNs and assume reliable message delivery in the network [18]. This causes such schemes to be resource exhaustive and to require heavy tailoring to suit WSN applications [108]. In fact, Tanengbaum et al. [27] concluded that existing techniques can only be implemented on limited scale WSNs.

Table 2.1: Comparison of existing pattern recognition schemes for WSNs.

Scheme	Comm- unications	Comp- utations	Memory	Network size	Time	Transform- ation invariant	Random patterns	Complex patterns	Noisy patterns
Threshold based	Low	Low	Low	Small	Low	No	No	No	No
KNN	Low	High/ Depends on k	High	Small	Low	No	Yes	Yes	Yes
Statistical	Low	High / Centralised	Moderate	Small	Low	Yes	No	Yes	Yes
Neural networks	High	High	Low	Small	High	Yes	No	Yes	Yes
SVM	Low	Centralised	Low	Small	Low	Yes	No	Yes	Yes
GN	Low	Low	Low	Small	Low	No	Yes	Yes	Yes
HGN	Low	Low	Moderate	Large	Low	No	Yes	Yes	Yes
DHGN	Low	Low	Moderate	Large	Low	No	Yes	Yes	No
Decision trees	Low	Dependant	Low	Small	Low	Yes	No	Yes	No
Target	Low	Low	Low	Small	Low	Yes	Yes	Yes	Yes

Using distributed approaches to solve the pattern recognition problem in WSNs appears in recent research. According to Giridhar and Kumar [109], sending information from each sensor to the base station or a fusion centre in a WSN is inefficient. Consequently, according to the authors, the whole network should perform as a distributed cooperative computational component. Wittenburg [108] suggests that the processing involved in application levels should be pushed and distributed in the network level in order to achieve conservative resource consumption schemes for WSNs. Chamberland and Veeravalli [18] suggest that the use of distributed pattern recognition is the most efficient method for WSNs. They mention that data should be computed by sensors locally, and only part of the resulting information should be sent, in order to conserve the WSNs' limited resources. However, the authors highlight that the choice of which information should be sent to the base station of a WSN is crucial to such implementation.

The main hypothesis of this research is that a fully distributed scheme, which works purely with localised node adjacency-based computation, is the best candidate for solving the problem of event detection in WSNs. Adjacency-based computations will promote WSNs' ability to deal with complex, invariant, and noisy patterns. Additionally, it is expected that using a loosely coupled connectivity scheme will scale up efficiently in terms of time and resources management if used in resource-constrained networks such as WSNs. By offering a fast, accurate, and scalable scheme that suits WSNs, it is

possible to use such schemes in decision making to solve far more complex and real time application problems.

2.7 Summary

This chapter has presented an overview of WSNs and the pattern recognition challenges that are associated with such networks. WSNs pose numerous challenges for complex applications such as pattern recognition. WSNs pose even more challenges if an application is a real time one and needs to be implemented on a large scale network. These limitations stem from the constraint resources that a WSN can offer, including computational, communicational and memory resources. Such limitations make solving pattern recognition problems in WSNs a trade-off between performance and resource consumption.

Existing techniques that provide solutions for the pattern recognition problem in WSNs are threshold-based, KNN, statistical, neural networks, SVM, GN, HGN, DHGN, and Conditional techniques. Each of these present several issues when implemented on WSNs. Examples of these issues include the requirements of centralised processing, iterative processing and large numbers of communications. Hence, each scheme would need to be heavily modified to be adopted by WSNs. Existing schemes can be implemented on limited scale size WSNs. Hence, new contributions should be made towards enhancing the scalability and performance of pattern recognition.

Distributed techniques conserve resources in a way which perfectly suits the nature of WSNs. Using such techniques allows the spread of computations across the network. Distributed processing can be achieved by allowing each node to locally process data and send the final result to another entity in the network. However, there must be a method for choosing which other nodes to communicate with, what data should be processed and what information should be sent.

In WSNs, minimising communications is one of the best resource utilisation methods. The reason for this is that sending a message from one node to another is the most energy-consuming task that a sensor can perform. Hence, this chapter proposes the use of the adjacency communication method to reduce the number and range of communications. Processing data gained from adjacent nodes allows the network to communicate and process data in a loosely coupled fashion. In addition to conserving resources, this would allow the network to limit recognition time. Such features will make the proposed methods good candidates for resource-constrained networks such as WSNs, allowing them to solve real time and complex problems. Additionally, these methods will be good candidates for hybrid use with other techniques, such as decision making algorithms, to support high level network processing and decision making processes.

Chapter 3

Cellular Graph Neuron (CGN) for Pattern Recognition in WSNs

3.1 Introduction

Wireless Sensor Networks (WSN) make it possible to sense physical parameters in a field of interest. These sensory data can be analysed in order to detect the presence of a physical activity or events in that field and take an action in accordance with the detected activity. Analysing sensory data is a computational and processing task that falls under two paradigms, centralised and in-network [7]. In centralised processing, data obtained by sensor nodes are aggregated to one machine that has the computational ability to analyse sensory information. However, this paradigm is considered inefficient in large scale resource-constrained WSNs, especially for applications that require limited latency time for data analysis in order to support decision making processes [109]. In contrast, in-network processing allows network nodes to perform computations and analysis on obtained data locally and in a distributed manner. This causes the network to act as a cooperative computational entity and this capability allows a WSN to reduce the computational and communicational complexity of processing nodes in the network.

The problem of event detection in WSNs can be solved by using in-network pattern recognition techniques. A pattern may be defined as a set of raw sensory data that describes the main characteristics of an event [26]. In-network pattern recognition techniques for WSNs include threshold-based, template matching, nearest neighbour, fuzzy logic, and neural networks, as discussed in Chapter 2. Existing pattern recognition schemes for WSNs are usually tailored to provide detection capabilities for specific applications or problem scenarios. These techniques may fulfil some event detection requirements while failing to address WSN resource limitation issues.

As discussed in Chapter 2, Graph Neuron (GN) is a scheme that creates associative memory (AM) in a fully parallel-distributed manner over fine-grained WSNs and offers light-weight one-shot learning capabilities. These characteristics make GN a good candidate for real-time pattern recognition applications in WSNs. However, the recognition accuracy of GN is affected by the limited perspective of each neuron, as each node only knows about its immediate neighbours. Consequently, the hypothesis of this chapter is that developing a GN-based scheme that addresses the accuracy limitations of GN would be the best option for solving the problem of pattern recognition in resource-constrained networks such as WSNs.

In this chapter, a pattern recognition scheme that is capable of detecting events and noisy patterns while addressing the resource constraints of WSNs is proposed. The scheme will adopt an in-network processing paradigm by including GN in its structure. Additionally, the scheme will solve the crosstalk

problem that affects GN accuracy by adopting network structures that allow certain nodes in the network to maintain more information about incoming patterns rather than being restricted to only adjacent nodes. The proposed network structure is designed to have the same network size as the GN network, which maintains the high scalability and the one-shot learning feature of GN.

The chapter starts by presenting the CGN scheme structure for pattern recognition in section 3.2. This covers the constraints related to deploying such a scheme and why these constraints are used. Additionally, this section analyses the relationship between CGN structural constraints and the network's size. The section also discusses the effect of incoming pattern size on the CGN network size. Additionally, this chapter describes the memory structure, computations and communications of a CGN. In section 3.3, the scheme's method of receiving incoming patterns is presented. This involves determining which nodes should participate in the learning process. Section 3.4 analyses the complexity of CGN in terms of memory size, number of communications, and learning cycle time. The aim of such analyses is to validate the suitability of CGN for use in WSN environments. This is followed by tests on the CGN scheme in section 3.5. These tests show the ability of CGN to act as a pattern recognition scheme despite the presence of noisy patterns. Additionally, this section presents a comparison between CGN and other existing pattern recognition schemes. The section compares CGN with Hopfield networks in terms of communications, computations, and time. Additionally, the section

presents a test of handwritten digits as an example for comparison of accuracy between CGN, Naïve Bayes, and back propagation neural networks, demonstrating CGN's superiority. Section 3.6 summarises the chapter.

3.2 Overview of CGN

This section describes the proposed CGN scheme. Part of this section has been published in [110]. As a first step to achieving efficient pattern recognition capabilities, the scheme provides the capability of template matching and noisy patterns recognition in resource-constrained environments such as WSNs. The main goal of the scheme is to minimise recognition time and increase network scalability. The scheme allows a WSN to collaborate and act as an associative memory in order to store and recognise patterns. This is achieved by creating a network of GN arrays and allows these arrays to communicate in order to conclude one result. This will allow the network to process and compute information in a distributed in-network paradigm.

The aim of the CGN network structure is to allow nodes to exchange information about an incoming pattern for storing and recalling operations using an in-network processing paradigm. Two goals that the structure is attempting to achieve, low scheme complexity and high pattern recognition accuracy. To achieve the first goal, the CGN network structure adopts a GN scheme as being well known for its low computational, communicational, and time complexity. This is due to the dependency on adjacency communications and computations in its structure for pattern recognition operations. Hence, the

CGN network structure consists of multiple GN arrays where each array is assigned to process a sub-pattern of an incoming pattern. To achieve the second goal, high pattern recognition accuracy, the CGN network structure provides links between GN arrays to give the whole network a broad overview of an incoming pattern. Since each GN array manages one sub-pattern, the links allow some GN arrays to overview the sub-patterns of other GN arrays. The aim is to have one array that has top overview over the whole incoming pattern. This array will make the final decision about an incoming pattern and report the result to the base station.

3.2.1 CGN structure

The CGN scheme involves two main entities, the stimulator and interpreter (S&I), which is an external computational node, and the CGN network, as shown in Figure 3.1. The two components communicate with each other in order to conclude one decision in a predictable learning duration. The S&I sends commands to the network and the network replies with an index number. The index number (I) is a unique integer number that describes the computation outcomes of the network. This index number can be used to represent a class or a pattern. A command that is sent by the S&I tells network nodes whether to memorise (store) a pattern or recall (search for) it. Also, the command will determine the method of obtaining the pattern (i.e. sense environment).

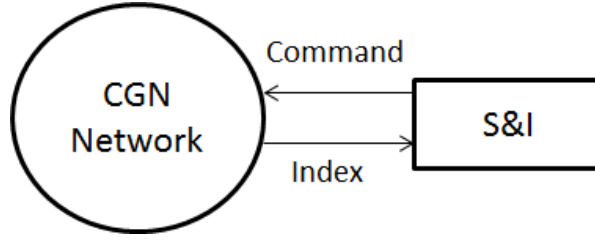


Figure 3.1: The two main components of the CGN scheme.

A CGN scheme's network structure and S&I depend on the problem's pattern size, the possible values of each pattern element, and the index number. Hence, we must first define these terms. In this research a pattern is defined as follows.

Definition 3.1: (Pattern) Given a set of possible values $V = \{x_1, x_2, \dots, x_v, \quad x_i, v \in \mathbb{N}\}$, a pattern is a set of elements that represent sensory information that can be sensed by a network's nodes or sent from the S&I to each node in the network and can be described as follows.

$$P = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S, \quad \varepsilon \in V, S \in \mathbb{N}\} \quad (3.1)$$

where ε_i is the i^{th} element of the pattern and S is the number of elements and is called the pattern size.

Definition 3.2: (Index number) Given a set of patterns $\{P_1, P_2, \dots, P_n\}, n \in \mathbb{N}$, an index number (I_i) is a unique number that describes P_i in the form $\{1, 2, \dots, n\}, n \in \mathbb{N}$. Hence, $I_i = i$.

3.2.2 CGN network

The CGN network consists of a set of GN networks where each GN network reports to another one with reaching the S&I. A GN network in the CGN network structure is called a *track* and each track consists of a set of *neuron positions* that communicate with each other using *exchange communications*, as described below.

3.2.2.1 Neuron position (NP)

Definition 3.2: (Neuron position) Given a pattern P that has a set of possible values $V = \{x_1, x_2, \dots, x_v, \quad x, v \in \mathbb{N}\}$, a neuron position (NP) is a set of v network nodes where each node is assigned to manage one x such that $NP = \{a_1, a_2, \dots, a_v, \quad a, v \in \mathbb{N}\}$. Where a_i is the i^{th} node in the NP .

Each NP is responsible for sensing or receiving one element of an incoming pattern such that $P_{\varepsilon, NP}: \varepsilon_i \rightarrow NP_i, \quad \varepsilon \in V, i \in \mathbb{N}$ where $P_{\varepsilon, NP}$ is the assignment of an incoming pattern's elements to NPs . An NP represents a column in GN. Hence, activation of NP nodes follows similar activation of GN nodes, as discussed in section 2.2.6. Based on the received element, one node in each NP is activated. If the element value is x_i , then the node number i in the NP is activated.

Definition 3.3: (Activate node) Given $NP = \{a_1, a_2, \dots, a_v, \quad a, v \in \mathbb{N}\}$ and a pattern element $= x_i$, an active node (AN) is the node that is assigned to manage the value x_i in the NP such that $AN = a_i$. Active nodes in the network

are the nodes that continue learning process operations. Figure 3.2 shows an example of activating nodes for a binary pattern in 3 NPs.

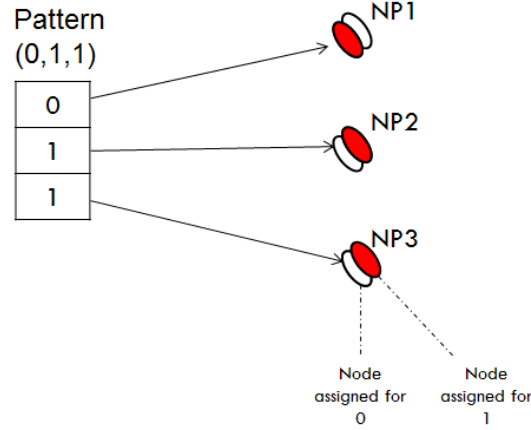


Figure 3.2: Active CGN nodes in response to the pattern (0,1,1). Red (shaded) nodes are the active ones.

3.2.2.2 Network track and communications

Definition 3.4: (Network track) A CGN network track (Trk) is a GN network that consists of a set of NPs where each NP communicates with its direct neighbour NP in the track. A CGN track can be described as follows.

$$Trk = \{NP_1, NP_2, \dots, NP_m, \quad m \in \mathbb{N}\} \quad (3.2)$$

Communications between NPs in the same track are called *exchange communications* and can be defined as follows.

Definition 3.5: (Exchange communications) Given a CGN network track (Trk) that consists of m NPs , exchange communications of a NP are two direct connections between the activated node in that NP and activated nodes in its direct neighbour (adjacent) *previous* (p) and *next* (n) NPs in the form $AN_i \rightarrow$

$AN_{i-1} : v$ and $AN_i \rightarrow AN_{i+1} : v$ respectively, where AN_i is the communicating (activate) node in NP_i and v is the value assigned to the AN_i . Figure 3.3 depicts a CGN track that consists of m NPs and v possible values. It is assumed that the first NP is directly adjacent to the last NP.

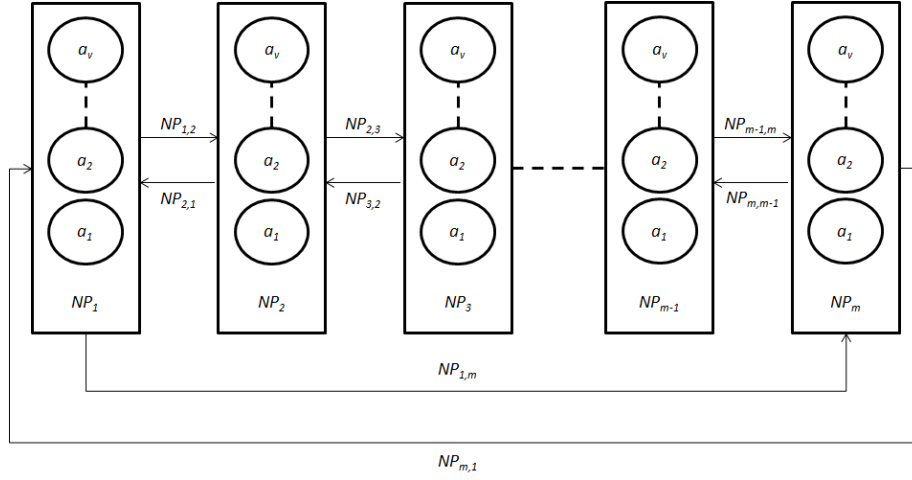


Figure 3.3: CGN track of m neuron positions.

The CGN network is designed in a cellular structure containing multiple tracks. The multiple tracks in the network structure aim to enable parallel processing and information exchange of incoming data. This is achieved by allowing each track to perform a set of recognition operations on a sub-pattern in parallel with other tracks. Such a structure also aims to enable the network to deal with multi-dimensional data types as each track will be assigned to process one dimension. The aim of the cellular structure is to deliver computations of network nodes to one track, called the *core track*, which contains only one NP, called the *core position*. To achieve this structure,

each track is formed using an odd number of *NPs*. Using odd numbers in determining track size ensures the formation of the cellular structure and simplifies reporting information between tracks. The deployment of the network begins by implementing the *core position* in the *core track* followed by the next odd numbered set of *NPs* in the next track and so on until all nodes have been deployed in the network. This results in tracks that hold odd numbers of *NPs* in the form $(1, 3, 5, \dots, 2n+1)$. It is important to highlight that node deployment in this section is a logical deployment method. In other words, deployment can be implemented by assigning each node its track and *NP* numbers. These numbers will be used to define the tasks that each node will perform, as will be described in the memory and network operations subsections later in this chapter. Algorithm 3.1 depicts the network deployment process.

Algorithm 3.1: CGN Network deployment

```

1. NetworkSize = PatternSize
2. TrackNumber = 1
3. TrackSize = 1
4. DeployedNP = 0
5. While (NetworkSize>0)
6.   Deploy a NP in Track(TrackNumber)
7.   NetworkSize--
8.   DeployedNP++
9.   if (DeployedNP ≥ TrackSize)
10.    TrackNumber++
11.    TrackSize = TrackSize + 2
12.  End if
13. End While

```

The aim is to provide the CGN with a cellular network structure that allows nodes to transmit their results to a core region, which is then responsible for delivering the final result to the S&I. The size of a track is the number of *NPs* it holds and can be calculated as follows.

$$S_i = 2i - 1, \quad i \in \mathbb{N}, i \geq 1 \quad (3.3)$$

where S_i is the size of the i^{th} track in the network. Here it is assumed that the first track is the core track and has the value $i=1$. In order to exchange information between tracks, each activated node delivers its computation outcomes (i.e. unique index number) to another activated node in a higher level track called *inner track*. Inner track of track i can be formally represented as follows.

$$Trk_{inner} = Trk_{i-1}, \quad i \in \mathbb{N}, i \geq 2 \quad (3.4)$$

This equation starts from the value $i=2$ because track 1 has no inner tracks. Instead it delivers its reports to the S&I directly. Conversely, an *outer track* can be described as the lower track level of track i and can be represented as follows.

$$Trk_{outer} = Trk_{i+1}, \quad i \in \mathbb{N}, 1 \leq i \leq N_{trk} - 1 \quad (3.5)$$

where N_{trk} is the number of the network's tracks. This means that the last track in the network has no further outer tracks. Inner and outer tracks are name conventions that will be used in this research to describe steps of the report communications process. The communications between the CGN network's tracks are called *report communications* and can be described as follows.

Definition 3.6: (Report communications) Given a CGN network that consists of a set of tracks, a report communication of an active node AN in an NP is the message (connection) between this node and the activate node in its direct assigned *inner* NP that contains the resulting index number (RI) and its value in the form $AN_{i,l} \rightarrow AN_{i-1,l} : \{RI, v\}, \forall l < S_{i-1}$ or $NP_{i,l} \rightarrow NP_{i-1,l-2} : \{RI, v\}, \forall l \geq S_{i-1}$.

where i is the active node's NP order in the track, j is the track number, RI is the computed index number, v is the value of the activated node in $NP_{i,l}$, and S_{i-1} is the size of the track number ($i-1$). Since each track is lower than its *outer track* by 2 NPs , two NPs track i will have no matching nodes in track $i-1$ and the report goes for the $NP_{i-1,l-2}$. Figure 3.4 depicts a 9 NPs CGN network, showing both exchange and report communications.

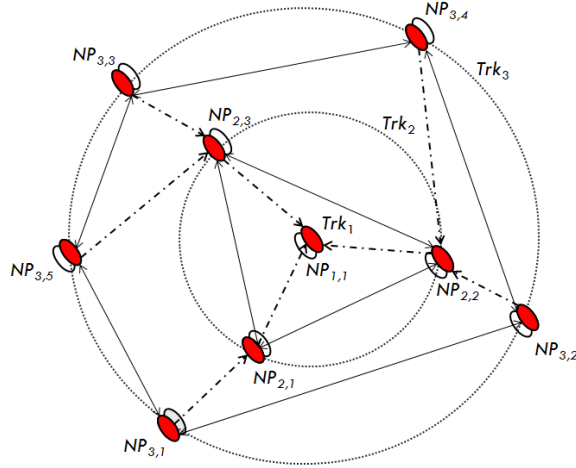


Figure 3.4: CGN network to adopt a 9 elements binary pattern. Red (shaded) nodes represent activated nodes in response to input pattern, solid arrows represent exchange communications, dotted arrows represent report communications and dotted circles show the CGN tracks.

3.2.3 Memory and bias array

GN involves initialising memory chunks in each node in order to hold the node's information and store the element combinations of a pattern encountered by the node and its adjacent nodes (i.e. previous and next). Such a memory chunk is called the *bias array* of a node and can be described as follows.

Definition 3.7: (Bias array) Bias array is a part of a CGN node's memory that stores information of memorised patterns by generating a unique *index number* (I) for each new combination of the adjacent activated nodes' values and reports. The index number is associated with the combination in the form $I \sim \{v_p, v_n, r_o\}$ in the memory, where v_p is the previous *NP* activated node's value, v_n is the next *NP* activated node's value, and r_o is the received report from an *NP* in the outer track. From Definition 3.6, the received report is the combination of the activated reporting node value and its computed index number in the form $r_o = \{RI_o, v_o\}$ where RI_o is the outer reporting node's resulting index value and v_o is the outer node's assigned value. Consequently, $I \sim \{v_p, v_n, RI_o, v_o\}$. It is assumed that report elements in the bias array elements are set to '0' in cases where no report is to be received. Figure 3.5 shows the representation of a CGN node memory structure.

Value (The assigned x value of the node in its NP)		Position	
		Track # (The track number of the NP)	NP # (The order of the NP in its track)
Bias Array			
Index	Bias $\{v_p, v_n, RI_o, v_o\}$		
I_1	$\{1,0,0,0\}$		
I_2	$\{1,1,0,0\}$		
\cdot	\cdot		
\cdot	\cdot		
I_{Npat}	$\{1,1,2,0\}$		

Figure 3.5: A CGN NP node's memory structure example that includes the network position of the node (track and NP numbers), its associated activation value and its bias array. N_{pat} means number of stored patterns in the network.

3.2.4 Network operations

The CGN network nodes perform two main operations, namely, memorisation and recall. In memorisation, network nodes store information about incoming pattern. In recall, network nodes search the stored information to find associated index numbers that describe the incoming pattern. Figure 3.6 shows a block diagram of the steps each node performs in order to memorise or recall a pattern. Here we use resulted index (RI) as the index number that an active node takes as the result of its computations and equals to the generated I in memorisation or the found I in recall. The steps can be explained as follows.

- i. **Receive pattern:** Each node receives the pattern element based on the command message received from the S&I. Each NP 's nodes receive the same pattern element value (x).

- ii. **Activation:** Based on the received pattern element value in each NP , the associated node to the received value is activated based on Definition 3.2 and the rest of the nodes are deactivated. Only activated nodes in the network continue the process.
- iii. **Exchange information:** Each activated node exchanges its value with the previous and next activated nodes using exchange communications.
- iv. **Receive reports:** Each node receives the reporting messages from outer track activated nodes. This step is excluded for nodes that are not assigned as an inner node.
- v. **Bias search:** after receiving all information (exchanged and reported) from neighbouring nodes, a node searches its bias array to find a match. If a match is found then the resulting index (RI) is assigned the associated index number (I) of that combination. Otherwise, the RI is assigned a new unique index number in memorisation or the value (0) in recall.
- vi. **Store information:** If the operation is to memorise the pattern and a new index number is assigned to the RI , the nodes associate the combination of neighbouring information with the RI and store it in the bias array.
- vii. **Report RI :** Each node reports the computed RI to its assigned inner node. Two nodes in each track are excluded from this step.

If the activated node is in the core track, it reports the *RI* directly to the S&I.

3.2.5 S&I operations

The S&I initiates the CGN process by sending commands to the CGN network and receiving the *RI* from the core node. After the S&I receives the CGN network's information, it begins the process of memorising or recalling a pattern. In memorisation, the S&I stores the concluded index number in its memory. This results in a set of patterns stored in a vector that can be described as follows.

Definition 3.7. (Pattern vector) Given a set of patterns $\{P_1, P_2, \dots, P_n\}$, the S&I memorises these patterns by obtaining each pattern's unique index number from the *core NP* in the CGN network, assigning the unique index number (I_i) to each pattern and storing the associations as a pattern vector in the S&I in the following form.

$$\vec{P} = \{I_1, I_2, \dots, I_i, \quad I_i \in \mathbb{N}\} \quad (3.6)$$

For example, the index number can be used to represent a class in classification problems. Storing index numbers in the S&I makes it possible to respond to query requests that may require information about stored patterns in the network. In recall, the declaration that a pattern has been detected depends on the CGN network's outcome. If a valid index number is returned by the network, the S&I declares that index number as the recalled pattern. Otherwise,

the S&I starts a voting process in order to obtain a valid outcome from the network. The following steps describe the S&I operations as shown in Figure 3.7.

- i. **Send command:** The S&I initiates the CGN learning process by sending a command to the CGN network's nodes. The command includes the operation type (memorise or recall) and the pattern obtaining method (e.g. obtain sensory information).
- ii. **Receive *RI*:** The S&I receives the *RI* from the active node in the core track *NP*.
- iii. **Store pattern:** If the operation is to memorise the pattern, the S&I stores the *RI* as the ID of the pattern and associates it with its description in its memory.
- iv. **Declare pattern:** If the operation is to recall a pattern and a valid *RI* is received (i.e. $RI \neq 0$), the S&I searches its pattern vector and declares the *RI* and its associated pattern description as the recalled pattern. Alternatively, the S&I initiates a voting process by sending queries to other nodes in the network to conclude a valid *RI*.

The voting process is initiated when the core node fails to deliver a valid index number to the S&I. The following steps describe the voting process, as shown in Figure 3.8.

- i. **Determine outer track:** Since the current contacting track failed to deliver a valid RI , S&I determines the contacting track as the outer track according to Equation 3.5. For example, if the core track $Trk=1$ fails to deliver a valid RI , the contacting track becomes $Trk=2$.
- ii. **Send query command:** The S&I sends query commands to all active nodes in the current contacting track (outer track) requesting their resulting indices (RI s). Each node replies by sending its RI .
- iii. **Vote RI :** After receiving all RI s from active nodes in the current track, the S&I finds the RI that has been received from the majority of nodes in the form $RI = majority(RI_1, RI_2, \dots, RI_m), m \in \mathbb{N}$. For example, if the track size is 3 and two active nodes reply with $RI=4$ and one node with $RI=1$, S&I determines $RI=4$. If one or more active nodes reply by invalid RI (i.e. $RI=0$) or in case of a tie, the S&I repeats the voting process with the outer track. This continues until contacting a track that results in a valid RI or until the vote reaches the outermost track.
- iv. **Declare pattern:** The S&I searches its pattern vector and declares the RI and its associated pattern description as the recalled pattern.

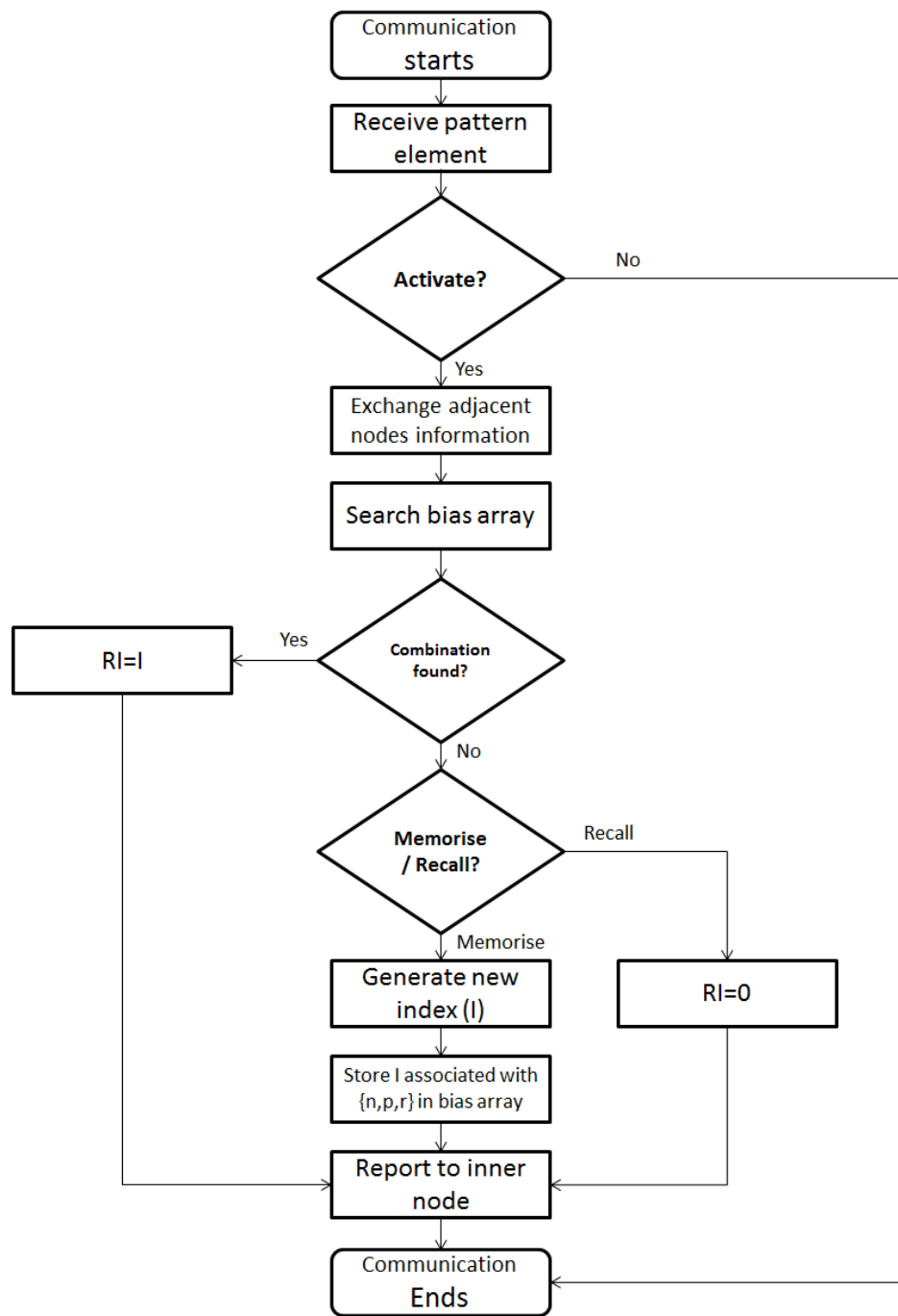


Figure 3.6: CGN node learning operations steps block diagram.

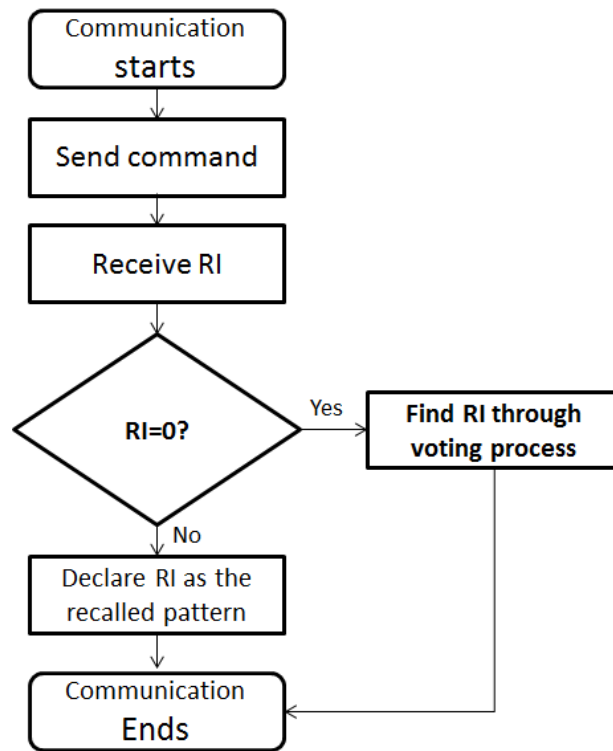


Figure 3.7: S&I learning operation steps block diagram.

3.3 Obtaining Pattern in CGN

To perform event detection and pattern recognition operations, CGN network pattern obtaining operations are discussed in this section. The CGN scheme has two types of operations, namely, memorisation and recall. CGN adopts the supervised pattern recognition manner. This means that a CGN network will be presented with a set of patterns to store and will then recall other patterns in accordance with the stored ones. These patterns can be imposed by the S&I or obtained by sensor readings. Performance of these operations is initiated by the S&I sending command messages to GN arrays in

each track. The command message from the S&I takes the form “ C, P ”, which means command (“ C ”= ‘command’) and pattern (“ P ”= ‘pattern’). S&I divides an incoming pattern into sub-patterns where each sub-pattern is managed by one track (GN array).

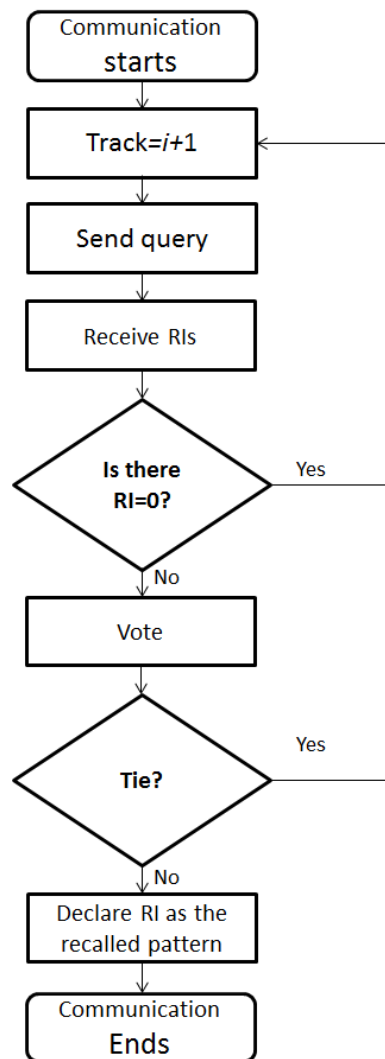


Figure 3.8: Block diagram of S&I voting steps.

The memorisation operation should always be initiated by the S&I. The S&I sends a command message to each node in the network of two parts “*M*, *P*”, which means memorise (“*M*”=‘memorise’) pattern element (“*P*”=pattern element). The pattern element part in the message can be provided by the S&I. Alternatively, the S&I will set the second part to “*S*”, meaning that the node should take its sensory information as the incoming pattern element (“*S*”=sense). Figure 3.9 shows an example of sending the binary pattern (1, 0, 1, 1, 0, 0, 1, 0, 1) to be stored in a 9 neuron positions size CGN network. The S&I will break the pattern into three sub-patterns (1), (0, 1, 1), and (0, 0, 1, 0, 1) and send these sub-patterns to tracks 1, 2, and 3 respectively. The S&I will send the commands “*M*, 1” or “*M*, 0” to each neuron position in the track based on the value assigned to the position. Alternatively, if the existing pattern in the sensed environment (for example, temperature readings) is to be stored, the S&I will send the command “*M*, *S*” to all network nodes.

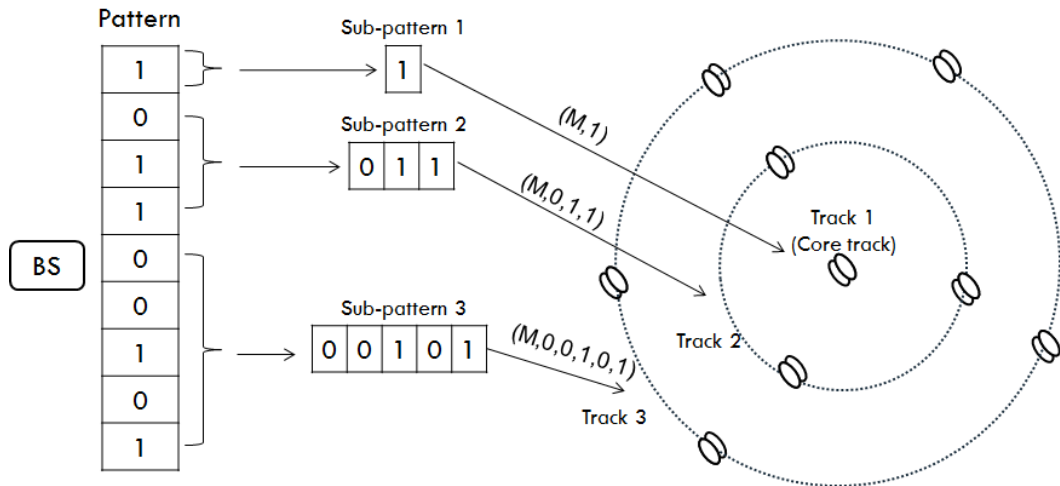


Figure 3.9: Pattern divided into sub-patterns by S&I. S&I in the base station (BS) divides a 9 size pattern into 3 sub-patterns and sends each sub-pattern to track for memorisation.

The recall operation, on the other hand, can be initiated automatically in a periodic manner or by the S&I. In a periodic recall operation, sensor nodes are given a time cycle where every sensor should take its readings as the incoming pattern. This suits automated and monitoring applications that require continuous recognition over the field of interest. S&I initiated recall operations are obtained by sending a command message of the form “ R, P ” to all network nodes, meaning recall (“R”= recall) pattern element. Similar to memorised commands, the pattern element part of the message can be provided by the S&I or requested to be sensed by sensor nodes. S&I initiated recall commands suit applications that require recognition at a certain point of time such as query-driven applications. Table 3.1 summarises the possible command messages that can be sent from the S&I to network nodes.

Table 3.1: Command messages from BS to network nodes.

Command	Description
(M,S)	Memorise (store) the sensory information
(M,X)	Memorise the value “X”, where X is a value of a given pattern element by the S&I
(R,S)	Recall the sensory information
(R,X)	Recall the value “X”, where X is a value of a given pattern element by the S&I

3.4 Complexity of the CGN Scheme

Analysis of the CGN scheme is conducted in the following section. The aim of a CGN scheme is to provide learning capabilities in resource-constrained WSNs while maintaining the scalability and speed of a GN scheme. Hence, the CGN network size, number of communications, and time analysis is presented. Table 3.2 summarises the terms used in CGN complexity estimation.

3.4.1 CGN network size

The CGN network size can be described in terms of the problem's pattern size as follows.

Proposition 3.1: Given a pattern $P = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S, \quad \varepsilon \in V, S \in \mathbb{N}\}$ and number of possible values v , the number of required NPs to construct a CGN network that can adopt P is $N_{NP} = S$ and the number of nodes required is $N_{nodes} = v \cdot S$.

Proof: According to the deployment process of the CGN network in algorithm 3.1 and the definition of a CGN track (Definition 3.4), each pattern element ε_i is represented using one NP in the network. Consequently, the number of NPs is equal to the number of a problem's pattern size S . According to Definition

3.4, each NP contains v nodes where each node is responsible for managing one value. Hence, the total number of nodes is $v.S$.

Table 3.2: Description of the terms used for complexity estimation.

Symbol	Name	Description
S	Network size (also pattern size)	Network size in terms of NP is equal to the pattern size
N_{trk}	Number of tracks	The number of tracks in the network
S_i	Track i size	Number of nodes in track number i
T_{exch}	Exchange time	The time required by nodes to conduct exchange communications
T_{report}	Report time	The time required by the network to perform report communications
T_{send}	Send time	The time required to send a message from one node to another
T_{sense}	Sense time	The time required by a node to obtain sensory information
T_{total}	Total network time	Time required by the CGN network to perform learning operations
N_{exch}	Number of exchange communications	Total number of exchange communications required by CGN network to perform learning operations
N_{comm}	Number of communications	Total number of communications required by CGN to perform learning operations
T_{opt}	Pattern obtaining time	Total time required for network nodes to obtain (sense or receive) an incoming pattern
$T_{activate}$	Node activation time	Time required by a single node to activate based on obtained pattern element
T_{bias}	Bias array search time	Time required by a node to search its bias array to find a matching index number

It can be seen from Proposition 3.1 that the CGN network maintains the same size as a GN network. The main point to consider in CGN network size is the multiplication by v . However, according to Definition 3.3, only one node in each NP is activated and participates in the learning process. Consequently, the number of activated nodes in the network during the learning process is equal to the number of NPs which is equal to S .

CGN network tracks are major components that can be used to estimate the complexity of a CGN scheme. The number of network tracks can be estimated as follows.

Proposition 3.2: Given a pattern $P = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S, \quad \varepsilon_i \in V, S \in \mathbb{N}\}$, the number of required tracks (N_{trk}) to construct a CGN network that adopts P is calculated as follows.

$$N_{trk} = \sqrt{S} \quad (3.7)$$

Proof: From Equation 3.3, a track size in terms of NPs can be estimated as $S_i = 2i - 1$, where i is the track number. Consequently, the total number of NPs (S) can be calculated as follows.

$$\begin{aligned} S &= \sum_{i=1}^{N_{trk}} 2i - 1 \\ S &= \sum_{i=1}^{N_{trk}} 2(i - \frac{1}{2}) \\ S &= 2. (\sum_{i=1}^{N_{trk}} i - \frac{1}{2}) \end{aligned}$$

$$S = 2. \left(\sum_{i=1}^{N_{trk}} i - \sum_{i=1}^{N_{trk}} \frac{1}{2} \right)$$

$$S = 2. \left(\sum_{i=1}^{N_{trk}} i - \frac{1}{2} \cdot \sum_{i=1}^{N_{trk}} 1 \right)$$

$$S = 2. \left(\frac{N_{trk}(N_{trk} + 1)}{2} - \frac{N_{trk}}{2} \right).$$

$$S = 2. \frac{N_{trk}^2 + N_{trk} - N_{trk}}{2}$$

$$S = N_{trk}^2$$

$$N_{trk} = \sqrt{S}$$

3.4.2 CGN communications

As described in Chapter 2, communication operations are one of the most important factors for energy consumption in WSNs. Consequently, the number of communications involved in performing pattern recognition using CGN can be used as the second aspect of scalability determination. The two CGN operation types (memorise or recall) need to be considered in estimating the number of communications in a CGN network. Both memorisation and recall operations involve exchange communications where each node sends its information to its adjacent nodes in the same track. That excludes the active node in the core position as it has no adjacent nodes in the structure with which to exchange information. Since the network size is equal to the pattern size (S), the number of exchange communications can be calculated as follows.

$$N_{exch} = 2S - 2 \quad (3.8)$$

Each active node in the network is required to give one report to its assigned inner node. Hence, the number of report communications can be estimated as $N_{report} = S$. This includes the report from the active core node to the S&I. The S&I sends a command (C, P) to each node in the network to initiate the memorisation and recall operations. Hence, the number of command communications will be equal to the total number of nodes $N_{cmd} = v.S$. The total number of communications involved in learning operations in a CGN network can be computed as the sum of the command, exchange and report communications and can be calculated according to the following equation.

$$N_{comm} = (3 + v)S - 2 \quad (3.9)$$

However, in the case of the core node failing to give a valid index number (i.e. replies by index '0') in a recall operation, the S&I contacts its outer tracks and starts the voting algorithm. The worst scenario in this case is if the S&I reaches the outermost track to obtain index voting information. In this scenario the S&I will contact all active nodes in the network (S), excluding the core node as it has already given its information using its report message. Hence, the S&I sends a query message and each active node replies with a message that contains the index number. This will require $2(S - 1)$ communications. Consequently, the total number of communications in such a case can be estimated according to the following equation.

$$N_{comm} = (3 + v)S - 2 + 2(S - 1) \quad (3.10)$$

$$N_{comm} = (5 + v)S - 4 \quad (3.11)$$

3.4.3 CGN network time

The computational and communication time overheads of a pattern of size S can be estimated by the duration of each CGN step. The first step is the pattern obtaining step. This step involves broadcasting the command message by the S&I, obtaining a pattern and node activation. In this step it is assumed that the command requests nodes to obtain a pattern through sensing the environment. It is also assumed that the broadcast command message is received in parallel by all nodes in the network. The estimated time required for this step is as follows.

$$T_{opt} = T_{send} + T_{sense} + T_{activate} \quad (3.12)$$

The following step exchanges sensory information by nodes. Taking parallelism into account, the time estimate can be described as follows.

$$T_{exch} = 2.T_{send} \quad (3.13)$$

This step is followed by report communications. This step involves searching the bias array and computing the index number. Taking parallelism into account, all active nodes in each track will perform bias search simultaneously. Excluding active nodes in the outermost track, each active node waits for reports from its outer track. Consequently, the reporting time can be estimated as follows.

$$T_{report} = \sqrt{S} \cdot (T_{send} + T_{bias}) - T_{send} \quad (3.14)$$

This includes the reporting message from the core node to the S&I. Consequently, the total duration of a learning cycle for a CGN network can be estimated as follows.

$$T_{total} = T_{opt} + T_{exch} + T_{report} \quad (3.15)$$

$$T_{total} = T_{send} + T_{sense} + T_{activate} + 2T_{send} + \sqrt{S} \cdot (T_{send} + T_{bias}) - T_{send} \quad (3.16)$$

$$T_{total} = \sqrt{S} \cdot (T_{send} + T_{bias}) + T_{sense} + T_{activate} + 2T_{send} \quad (3.17)$$

This time estimate works for both memorisation and recall operations in the network. However, for the worst case scenario in recall where the S&I contacts the outermost track, the voting process time should be added. In this case, the S&I will broadcast to each track and receive a response. The broadcasted message will be received in parallel by active nodes and the reply messages will also be sent simultaneously. Consequently, the total recall time in this case can be estimated as follows.

$$T_{total} = \sqrt{S} \cdot (T_{send} + T_{bias}) + T_{sense} + T_{activate} + 2T_{send} + 2T_{send} (\sqrt{S} - 1) \quad (3.18)$$

$$T_{total} = \sqrt{S}(3T_{send} + T_{bias}) + T_{sense} + T_{activate} \quad (3.19)$$

This time estimate shows that the learning cycle time is proportional to the square root of the problem size S . It can also be seen that the learning time cycle can be predicted a priori. From the analysis of a CGN scheme it can be concluded that the CGN network is capable of maintaining the scalability of

the GN network structure by involving S nodes in the learning process. This is due to the use of an activation mechanism that ensures that only one node in each NP is activated, according to Definition 3.2 and the analysis of Proposition 3.1. Additionally, the analysis shows that the scheme is capable of performing learning operations in predictable time while restricting the learning cycle to be proportional to the square root of S . Such features make the scheme a good candidate for implementation for large-scale real time problems.

3.5 Evaluating CGN Performance

This section presents a comparison and the tests conducted on the CGN scheme. CGN is compared with a Hopfield network in terms of numbers of communications and computation time. Two simulation tests were conducted on CGN. The first test aimed to test the tolerance levels of CGN with regard to noisy patterns. The test used distorted bitmap images of letters as patterns. The second test was conducted on handwritten digits to compare CGN with existing pattern recognition schemes such as Naïve Bayesian and back propagation networks. In these tests, it is assumed that the CGN network is deployed in a grid where each pixel is managed by one NP .

3.5.1 CGN and Hopfield

The main goal for a CGN scheme is to provide light-weight and fast pattern recognition capabilities for WSNs. Two metrics are considered to be

the main factors that affect the suitability of a scheme in providing such capabilities: number of communications and learning time. In WSNs, communications are the highest source of energy consumption and time latency. Consequently, the number of communications determines the level of resource consumption required where implemented in WSNs.

Hopfield networks require each node in the network to communicate with each other node in order to compute weights and conclude results. Since each node in the network has no connection with itself, each node requires a number of communications equal to $(S-1)$. Thus, the total number of communications in a Hopfield network can be calculated as $S(S-1)$ or S^2-S . This can be described as a quadratic relationship between the pattern size and the number of communications. In contrast, referring to Equations 3.9 and 3.11, the total number of communications of a CGN network can be described as a linear function in relation to the pattern size. Figure 3.10 shows the increase in the number of communications based on the network size for CGN and Hopfield networks.

It can be seen from the size relations in Figure 3.10 that the number of communications in Hopfield networks increases exponentially compared to a CGN. This indicates the amount of resource consumption reduction that a CGN network can offer compared to a Hopfield network. The time estimation can be described using Big-O notation. In this regard, pattern recognition operations can be used in determining the complexity of the two schemes. A Hopfield scheme goes through three processes in order to perform pattern recognition

operations: weight accumulation, weight determination and network propagation. In comparison, CGN involves a single learning cycle. Analysing a Hopfield network, the process of weight accumulation can be denoted as $O(S)$, the weight determination process can be denoted as $O(S^2)$ and network propagation as $O(S^3)$ using Big-O notation. In contrast, CGN pattern recognition time is proportional to the square root of S and can be denoted as $O(\sqrt{S})$. Figure 3.11 shows recognition time derived from Big-O notation analysis for both Hopfield networks and CGN, based on the assumption that each computation operation time is 1 microsecond.

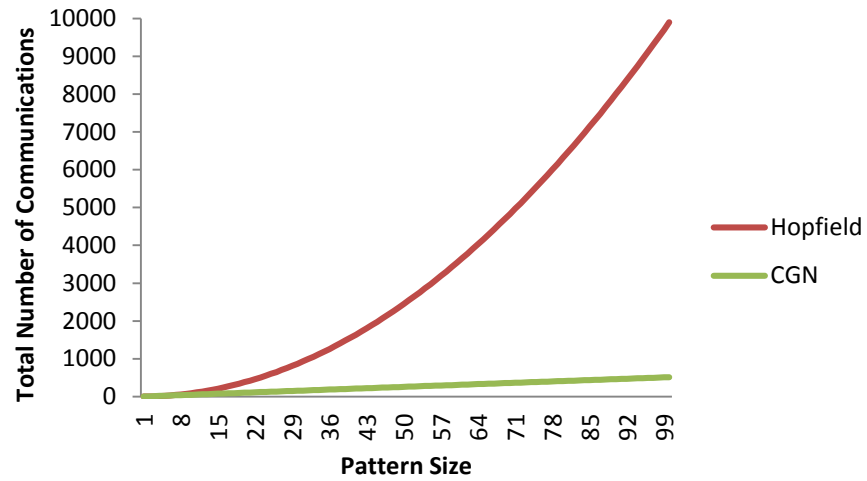


Figure 3.10: Number of communications in Hopfield and CGN networks based on pattern size.

It can be seen from Figure 3.11 and the Big-O notations that the pattern recognition computational time complexity of the CGN scheme is low

compared to the Hopfield scheme. That means a CGN network concludes its computations in much less time than a Hopfield network.

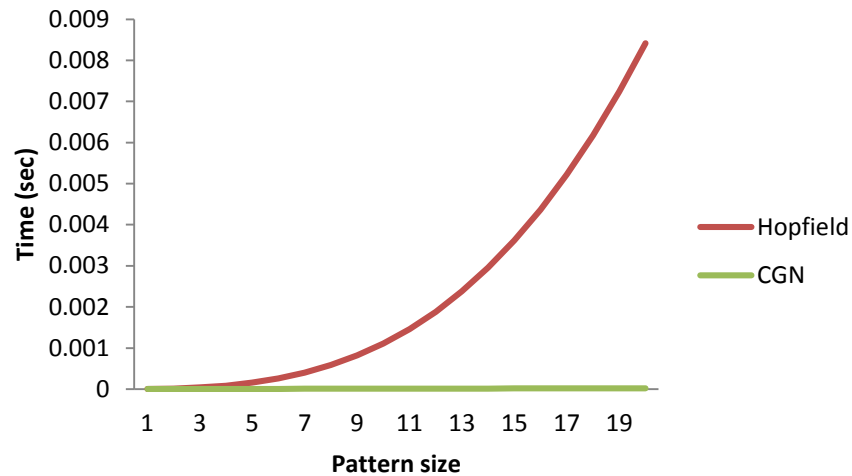


Figure 3.11: Time derived from Big-O analysis for Hopfield and CGN.

3.5.2 First test

The first test conducted on the CGN scheme aimed to determine the level of accuracy of a CGN network and the level of distortion it could tolerate. For this purpose, bitmap images were selected. Each pixel of the image holds either 0 or 1 value. By changing a pixel's value, the pattern will be changed. By testing each changed pattern against stored original patterns using a CGN network, it is possible to determine the tolerance and accuracy levels of the network. This can be achieved by storing a set of patterns and then generating altered patterns by changing a set of bits of each original one. It is assumed in

this test that each pixel is read by one *NP* that has two nodes, one is assigned to be activated with 0 value and the other with the value of 1.

In this test, letters “A”, “I”, “J”, “S”, “X”, and “Z” were represented as bitmap images of size 7x5 (35 pixels) to be stored in alphabetical order in a CGN network. These letters were chosen as they carry a level of distinction when represented in binary bitmap images. By applying Equation 3.2 and 3.7, 6 tracks and 36 neuron positions with two nodes for each were needed to construct a CGN network that could store the bitmap images. Since the patterns are binary bitmap images, each neuron position contained two nodes associated with 0 and 1 values. The images were presented to the CGN network for memorisation only once for each letter. Each image was then randomly distorted to varying degrees ranging from 1bit (2.78%) to 15bits (41.67%). Distortion is calculated as the number of changed bits (from 0 to 1 or vice versa) to the total number of bits (35 in this test). The distorted images were presented to the CGN network for recall. Figure 3.12 shows the original bitmap images and samples of distorted images and the recall results for the distorted samples.

It can be noted from Figure 3.12 that letters distorted by levels 25% and above are not visibly recognisable. It also can be noted that the CGN is able to detect patterns of characters “A”, “I”, “J”, and “S” with a high level of distortion — 13bit distortion level (36.11%). The accuracy of the CGN network’s recall is shown in Figure 3.13. The recall accuracy is calculated as the number of correctly recognised letters to the total distorted (altered) letters

of that letter presented to the network. For example, 100% recall accuracy for a bitmap pattern (A) means that all altered versions of letter A were recognised as letter A.

From Figure 3.13 it can be noted that the order of storing patterns has an effect on accuracy. It can be seen here that the letters stored first have higher accuracy results, especially after presenting a 16.7% level of distortion where four patterns scored recall accuracy below 80%. The reason is that when the CGN network cannot come up with a valid index number it goes through a voting process. In this case, the CGN network contacted the outermost track. Since the patterns are small binary images, the outermost track holds limited information about the pattern, which creates a high level of similarity between recalled and stored patterns. This test confirmed that the CGN scheme is capable of detecting distorted patterns. The results also show the ability of the CGN network to detect patterns of characters “A”, “I”, “J”, and “S” with a high level of distortion — a 13bit distortion level (36.11%) — which means that the network is tolerant to a high level of distortion.

3.5.3 Second test

This test aimed to present the ability of a CGN scheme to deal with complex and real life problems. It also aimed to compare the CGN’s accuracy with existing pattern recognition schemes. For this purpose, we chose a handwritten character recognition problem as such problems require memorisation of a high amount of training information. This test shows that a

CGN scheme is capable of performing a recognition operation using a minimal amount of training information (i.e. one sample for each class) while still maintaining a high level of accuracy compared with other schemes. In accomplishing this aim, a CGN network is capable of addressing the WSN randomness problem discussed in Chapter 2. The test used the dataset provided by [111] in [112]. The dataset contains 1593 handwritten patterns for numbers from 0 to 9. Each number was represented as a 16×16 binary pattern. Each pattern was produced by scanning and pre-processing numbers handwritten by 80 different people. The dataset has 10 classes where each class represents one number.

To construct a CGN network that is capable of adopting such patterns, 256 neuron positions distributed in 16 tracks were generated. Each neuron position contained two nodes (0 and 1). The selected classes for comparison were classes representing numbers from 0 to 5 as these numbers were distinguished in the handwritten representation. One pattern was selected randomly from each class for memorisation. This resulted in 6 memorised patterns to represent numbers from 0 to 5. The rest of the patterns in each class were used for recognition. This resulted in 955 recognition patterns. Figure 3.14 shows the 6 memorisation patterns and a sample of 6 recognition patterns.

The CGN is compared with Naïve Bayes and back propagation neural networks. Using a Weka tool [113, 114], a Naïve Bayes network was generated to perform storing and recognition operations. Figure 3.15 shows the average recognition accuracy of CGN compared to the Naïve Bayes and back

propagation NN schemes. The accuracy percentage is calculated as the number of correctly recognised patterns to the total number of recognition patterns. To compare with back propagation NN (BP), three BP implementations were used, generating a BP with a single hidden layer, generating a BP with two hidden layers, and generating a BP with three hidden layers. The number of learning iterations of each structure was set to ranges between 1 (single cycle) and 500 iterations. The best result obtained was with the implementation of BP with three hidden layers and 200 iterations. The results shown in Figure 3.15 for the BP NN are based on that structure.





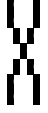
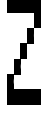

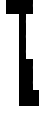


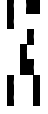
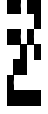





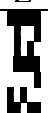
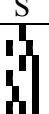
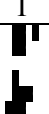




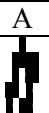



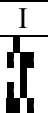

Original memorised patterns						
Distorted (5 bits) (13.89%)						
Result	A	I	J	S	X	Z
Distorted (9 bits) (25%)						
Result	S	I	J	S	X	Z
Distorted (13 bits) (36.11%)						
Result	A	I	J	S	I	S
Distorted (15 bits) (41.67%)						
Result	A	S	I	I	I	J

Figure 3.12: Original bitmap image patterns for A, I, J, S, X and Z with sample of recalled distorted images ranging from 2.7% to 41.67%. Black highlight indicates to one value. Zero values are not shown for clarity.

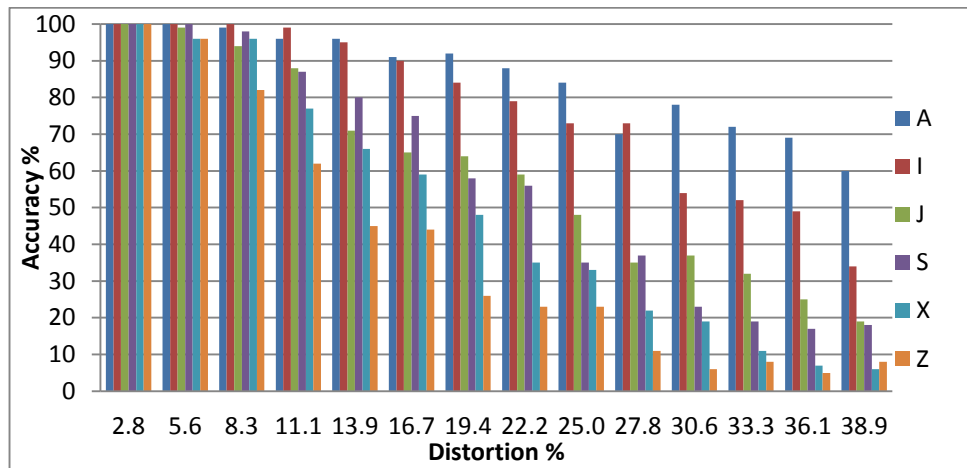


Figure 3.13: Accuracy recall percentage for the CGN using 100 randomly distorted patterns per memorised pattern.

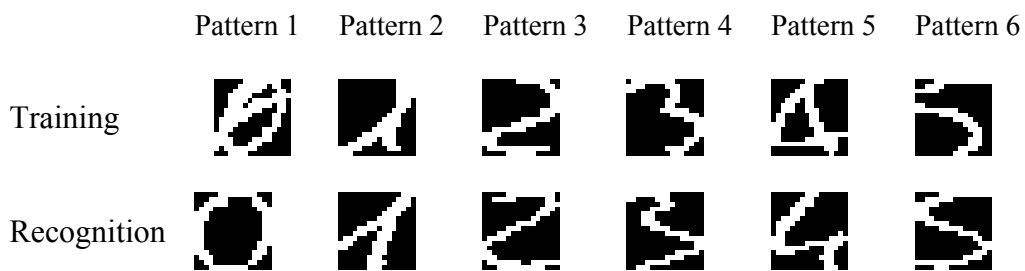


Figure 3.14: Memorisation and sample recognition patterns representing written numbers from 0 to 5.

Figure 3.15 shows that CGN can provide pattern recognition capabilities with better accuracy results than Naïve Bayes and BP NN schemes. These results were achieved by only using one pattern for each pattern for memorisation. In addition, the CGN scheme is capable of performing pattern recognition operations in a single cycle with the number of neuron positions equal to the pattern size. It is important to note that only one node in each neuron position participates in the learning operations. This means that the

CGN network requires a number of nodes that are equal to the pattern size to perform recognition operations. Compared to the back propagation setting for this example, CGN reduced the number of participating nodes, communications and iterations needed to perform pattern recognition while maintaining higher accuracy levels as back propagation networks involve more neurons to build the multi-layer structure, requiring tightly coupled connectivity between layers, and requiring 200 iterations to perform recognition operations. This test demonstrates the capability of a CGN network to perform complex and real life recognition problems by using a minimal amount of training information. This addresses the problem of randomness associated with WSNs.

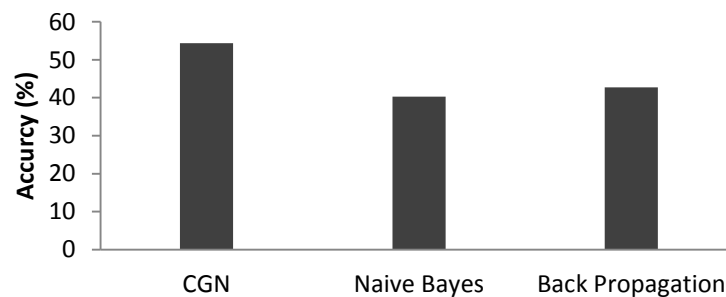


Figure 3.15: Average accuracy levels obtained by CGN, Naïve Bayes and back propagation networks.

3.6 Summary

This chapter has presented CGN schemes as a pattern recognition approach for WSNs. It has been shown that the structure of such schemes is

fully distributed and based on the relationships between adjacent nodes. Such capability limits the number of communications and computations as well as reducing the consumption of resources, which makes CGN a light-weight approach that is suitable for implementation in resource-constrained networks such as WSNs. Additionally, it has been shown that the CGN network structure is capable of maintaining a high level of scalability by involving each node in the network in receiving one element of an incoming pattern. This allows a CGN scheme to provide a scalable solution for large scale WSNs. Moreover, analysis of the scheme shows that the scheme is capable of performing learning operations in a single learning cycle that can be predicted. The analysis also shows that the learning cycle of a CGN grows in proportion to the square root of the problem size which makes the scheme favourable compared to other schemes that can involve iterative learning cycles with exponential growth in terms of time. Such features make the scheme a good candidate for use in real time applications where time estimates are required.

The two experimental tests show that CGN provides high recognition accuracy levels for noisy patterns compared to iconic pattern recognition schemes while involving a minimal amount of training information. From the analysis and tests conducted on the scheme, it can be concluded that CGN is a good candidate for implementation in large scale and limited resource WSNs for real time applications. Additionally, the use of adjacency information relationships in pattern recognition presented in this chapter will lead to

solving the problem of detecting transformed patterns, as will be discussed in the next chapter.

Chapter 4

Cellular Weighted Graph Neuron (CwGN) for Transformation Invariant Recognition in WSNs

4.1 Introduction

In the previous chapter, the CGN was presented as a light-weight and distributed pattern recognition scheme that involves a limited number of communications and computations. Such features suit resource-constrained systems and networks such as WSNs. It has been shown experimentally that CGN is capable of dealing with noisy patterns and some complex problems such as handwritten recognition. However, a CGN scheme is location sensitive due to its dependency on local nodes' information storage. In some real life applications, patterns and network nodes are subject to spatial and topological changes [115]. This means that a pattern can appear with a level of variations or transformations such as location change. For example, a malicious intruder pattern of a WSN can change its location in the network [116]. Another example can be seen in environmental surveillance systems where events such as forest fires and hurricanes may appear in different locations and at different magnitudes [117]. Such dynamics in pattern appearance can impact the

accuracy of a recognition system. Dealing with such dynamics could require a high amount of information that require patterns and their possible dynamics to be stored in a recognition system in order to efficiently recognise the presence of these patterns. However, such an approach might not be feasible, for two main reasons. First, it would require a high amount of resources to store and search patterns. Second, in some applications the amount of available information about patterns is limited, as discussed in Chapter 2. Consequently, a more efficient approach is required.

In this chapter, a novel approach called Cellular weighted Graph Neuron (CwGN) is proposed to deal with the transformations and dynamics of pattern recognition problems. As with CGN, the scheme adopts the GN approach to maintain minimal communicational and computational requirements to thus provide a light-weight pattern recognition scheme that suits resource-constrained systems and networks such as WSNs. Instead of storing pattern information locally on nodes, CwGN implements a weighting mechanism that searches the edges and boundaries of patterns. The main hypothesis in this chapter is that by describing patterns using their main edges and boundaries, it is possible to achieve an efficient recognition scheme that can detect transformations that may occur in these patterns. The scheme maintains limited communications and computations by involving local information exchange and reporting mechanisms that distribute resource consumption loads amongst the network's nodes. Additionally, this chapter will discuss the online recognition capability that can be achieved by using

CwGN. In achieving efficient recognition and fast reporting and by limiting resource consumption, a CwGN scheme shown to be the best option for real life applications that deal with complex problems in resource-constrained networks such as WSNs. This chapter also presents descriptions of the required protocols needed to make the proposed scheme applicable for implementation in network environments.

This chapter starts by presenting existing transformation invariant pattern recognition techniques, going on to discuss the limitations of these techniques. Section 4.2 presents an overview of existing techniques proposed for pattern transformation recognition. Section 4.3 presents the CwGN scheme, including memorisation and recall operations, network structure and the weighting technique. In view of the need for the scheme to be functional in networking environments, section 4.4 presents the communicational structure and requirements of the CwGN scheme and its network communication protocol. Section 4.5 analyses the complexity and the performance of CwGN with respect to learning cycle duration. Section 4.6 presents a proposed zoning model to support the online recognition capabilities of the scheme. In section 4.7, required message sequence models for the proposed scheme are presented. Section 4.8 discusses the effects of different types of pattern transformations on the recognition capacity of the proposed scheme. Section 4.9 summarises the chapter.

4.2 Transformation Invariant Recognition

Techniques

This section reviews the most relevant research in the area of pattern transformation-invariant recognition. Convolutional neural networks developed by Le Cun et al. [118] use multi-layered (deep learning) neural networks to encode high-dimensional patterns into a one-dimensional vector. That scheme allows for feature selection by adopting local receptive fields, local weight sharing, and downsampling architectures [119, 120]. The local receptive fields allow for the detection of the visual features of a pattern, the downsampling architecture reduces the dimensionality of the pattern, and the concept of weight sharing allows for a level of shift and scale invariant detection [120]. Convolutional neural networks have mostly been used for visual pattern recognition applications such as facial and handwritten recognition applications. Shock graphs [121], in contrast, attempt to recognise visual objects by determining the object's boundaries. This method builds connected curves and points that describe certain patterns in a tree-like graph. Sebastian et al. [122] show that using shock patterns allows for 2-D object transformation-invariant recognition for up to a certain level of transformation. They examined a set of pattern transformations such as articulation and deformation, illumination variations, and variations in the scale of objects. Alternatively, Map Seeking Circuits (MSC) [123] use the mathematical properties of pattern superpositions to perform template matching, which seeks a set of

transformations of an input visual pattern for a set of stored templates. The aim of the MSC approach is to reduce the growth in the computational complexity of template matching by parallelising computations in the hardware. MSCs provide solutions to visual applications such as stereo vision, shape recognition, and other transformation-recognition problems that can be solved using iterative processing and the decomposition of pattern transformations [124].

Despite the transformation detection capabilities discussed above, these schemes involve computationally intensive (e.g. iterative) operations that are poorly suited to real time sensory applications due to prolonged learning cycles, the need for large training datasets that are often not available, and reliance on specialised hardware. Based on a lattice algebra approach, morphological associative memories (MAM) [120, 125] are able to detect pattern transformation in a single step convergence. MAM methods use a morphological neural network structure that replaces multiplication and addition operations by addition and convergence maximisation. MAMs have been shown to be scalable in terms of pattern storage, and capable of detecting noisy patterns. However, the length of the MAM learning cycle cannot be fully estimated a priori, as it depends on the size and number of stored patterns [95]. Iftekharuddin [126] presents an online transformation scheme for automatic target recognition. His scheme addresses the problem of recognising rotation, translation, and scaling pattern translations in images by adopting adaptive circuit design, neural networks and reinforcement learning. Despite the level of

transformation recognition this scheme provides, it depends on conventional neural network structures such as feed-forward NNs. In this network structure, tightly coupled connectivity is required between neuron nodes in each layer. Sensory information systems such as WSNs have limited communicational capabilities, making the tightly coupled connectivity structure challenging to implement [28].

In general, the existing techniques can provide transformation invariant detection capabilities for pattern recognition problems. However, these techniques have significant limitations, especially if implemented in resource-constrained networks. Iterative operations and tightly coupled connectivity structures are the main problems faced in these schemes as such requirements involve huge resource consumption, especially when implemented on large scale networks. Some of these schemes require special centralised hardware settings to be functional. Such a requirement is not often feasible in resource-constrained systems and networks. Additionally, the reviewed schemes require a large database to store information about patterns in order to provide transformation invariant recognition capabilities. In environments and applications such as WSN applications, the amount of information available about incoming patterns is often limited. Another major issue related to these schemes is the uncertainty in the learning cycle convergence duration. This is due to the iterative operations and the dependency of learning cycle duration on the amount of stored information. Such constraints affect the suitability of these schemes for implementation in real time and mission critical applications

as no guarantees for time limits are available. Consequently, this chapter will present a light-weight pattern recognition scheme that aims to address these limitations and provide transformation invariant recognition capabilities. The proposed scheme minimises computations and communications by adopting local communicational and computational mechanisms in a single learning cycle. This will lead to minimised resource consumption and increase the scheme's scalability by avoiding iterative operations and limiting communications to adjacency nodes. This also makes it possible to estimate the learning cycle duration. In addition, the proposed scheme attempts to use minimal information to perform transformation recognition. With these features, the proposed scheme will be a light-weight, transformation invariant, and scalable scheme that is suitable for real time and mission critical applications for resource-constrained systems and networks such as WSNs.

4.3 Overview of Cellular Weighted Graph

Neuron (CwGN)

This section describes the structure of CwGN and the outcomes that can be expected from such architecture. Part of this section has been published in [127]. The main goal of developing the CwGN scheme is to provide efficient pattern recognition for WSNs while minimising resource consumption and network size. Olshausen and Field [128] state that coding techniques can reduce the use of resources and minimise the complexity of incoming patterns

so that they can be easily processed. The scheme presented in this section is based on such considerations. These authors [128] define coding technique as reproducing an incoming pattern and using a small number of active nodes for processing at any given time. To achieve the intended level of detection and minimise resource consumption, CwGN implements local adjacency computations for coding purposes in a fully distributed and parallel manner that is capable of detecting pattern transformations such as translation, dilation and rotation with minimal computational and communication requirements. CwGN uses weights rather than storing full information about an incoming pattern. Using weights allows CwGN to be location insensitive as weights are calculated locally and then accumulated throughout the whole network. Each node's weight describes its relationship to its neighbouring nodes. This translates an incoming pattern into a set of weights that are easier to process and can be used to distinguish each pattern from the rest.

Data instances are translated into weights that can be used to determine the boundaries of an incoming pattern. Each node's weight is a result of the relationship between its value and its neighbours' value in terms of change rate and edge type, which is calculated locally and only once. The change rate can be defined as the amount of average variance between a node's value and its adjacent nodes' values. Edge type is the order of the node's value compared to its neighbouring nodes that can be used to describe the pattern's boundaries. Peaks and troughs of a plotted pattern can be examples of two different edge types. The hypothesis behind such an approach is that patterns can be

efficiently recognised based on their boundary information. Since these boundaries are detected by local nodes' computations, the number of communications and amount of resources required is minimised.

Figure 4.1 illustrates the CwGN model. Similar to CGN, the model involves two main entities: the CwGN network and the stimulator and interpreter (S&I). The S&I is responsible for sending commands to the CwGN network. It receives weight and concludes the final decision about an incoming pattern. A command message can be to either memorise or recall a pattern. It also includes information about the pattern or commands the network to obtain sensory information. CwGN network nodes process the S&I command and reply with a weight (or set of weights). The S&I uses this weight (or summation of weights) to memorise or recall the sensed or sent pattern.

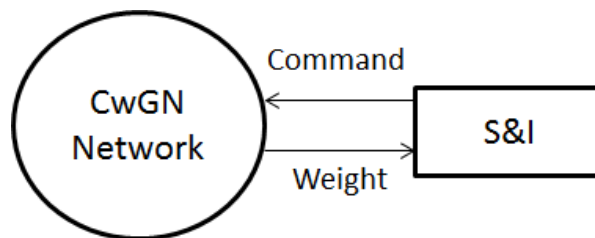


Figure 4.1: The CWGN communication model.

4.3.1 Stimulator and interpreter (S&I)

This component is responsible for sending commands to the CwGN network, receiving the weight and making the final decision about an incoming pattern. CwGN uses the same pattern obtaining method as CGN, as described

in section 3.2. Hence, the four commands (M,S), (M,X), (R,S), and (R,X) described in table 3.1 are used to train a CwGN network and recognise patterns. CwGN network nodes process the S&I command and respond with a weight (or set of weights). Weight computations will be discussed later in this chapter in the edge search section. The S&I uses this weight (or summation of weights) to memorise or recall the sensed or received patterns. After the S&I receives the weight from the CwGN network, it begins the process of memorising or recalling a pattern. In memorisation, the S&I assigns a unique index number to the pattern, associates this number with the resulting weight, and stores the index number and the associated weight in its memory. This results in a set of patterns stored in a vector that can be described as follows.

Definition 4.1: (Pattern vector) Given a set of patterns $\{P_1, P_2, \dots, P_n\}$, the S&I memorises these patterns by obtaining each pattern's weights (ω_i) from a CwGN network, assigning a unique index number (I_i) to each pattern and storing the associations of patterns and weights as a pattern vector in the S&I in the following form.

$$\vec{P} = \{(I_1, \omega_1), (I_2, \omega_2), \dots, (I_n, \omega_n) \mid I_i \in \mathbb{N}, \omega_i \in \mathbb{R}\} \quad (4.1)$$

In recall, the S&I searches the pattern vector to find a match. The declaration that a pattern has been detected depends on the difference between the CwGN network's weight and the weights stored in the pattern vector weights as follows.

Definition 4.2: (Recalled pattern) Given a weight calculated by the CwGN (ω_c) and a pattern vector, the recalled pattern (Rp) will be the index number with the smallest difference between the calculated weight value and the set of associated weights in the pattern vector as follows.

$$Rp = I[\min(\Delta\omega_{1C}, \Delta\omega_{1C}, \dots, \Delta\omega_{iC})] \quad (4.2)$$

where $\Delta\omega_{iC}$ is the difference between the i^{th} stored pattern weight and the weight calculated by the network for an incoming pattern (current pattern).

Figure 4.2 illustrates the S&I operations. These operations can be summarised as follows.

- i. **Send command:** The S&I sends the command which contains the operation (memorise or recall) and the pattern element obtaining method (direct receive or sense) to all network nodes.
- ii. **Receive weight:** The S&I receives the network's accumulated weight from the core node in the network.
- iii. **Memorise or recall:** If the incoming pattern is to be memorised, the S&I creates a unique index number, associates this index number with the network's weight, and stores this association in its pattern vector. In the case of recall, the S&I searches for the closest weight in its pattern vector to the network's delivered weight and declares its associated pattern as Rp .

4.3.2 CwGN network

CwGN network structure is similar to CGN network structure, as presented in Chapter 3. However, in CwGN, each neuron position contains only one node. The aim of the network structure is to have low pattern recognition scheme complexity, provide parallel processing, provide efficient recognition, and have a predetermined learning and recognition cycle duration. A low complexity scheme is achieved with a fully distributed structure that allows sensor nodes in the network to communicate only with adjacent nodes. CwGN's structure allows each node to communicate with two adjacent nodes for weight calculation and with one adjacent node for outcome reporting. Efficient recognition is provided by describing the patterns' boundaries in terms of weights in order to provide a transformation invariant recognition feature to the scheme. Using weights aims to allow the detection of any certain pattern's boundaries by any node in the network. In other words, the detection of any desired pattern's boundary is not associated with static nodes. Instead, any node in the network is expected to be able to derive the same weight for such boundary information. By using specific steps for weight reporting, CwGN will have a single learning and recognition time cycle that can be predicted and estimated. Once the final weight is delivered to S&I, CwGN does not need further information from sensor nodes in order to declare the detection of a specific pattern. Such a feature reduces the need for communications between S&I and participant nodes in the network.

The network structure depends on the size of the problem pattern. The deployment of the network uses the deployment algorithm described in Algorithm 3.1. However, it is important to note that CwGN deploys one node in each step to form network tracks, rather than deploying a set of NP nodes as in CGN. The deployment process begins by implementing the core node in the *core track* followed by the next odd numbered set of nodes in the next track and so on until all nodes have been deployed in the network. This results in tracks that hold odd numbers of nodes in the form $(1, 3, 5, \dots, 2n+1)$. The aim is to provide the CwGN with a cellular network structure that allows nodes to transmit their results to a core region that is responsible for delivering the final result to the S&I. These data are computed once to obtain weights that describe the input pattern.

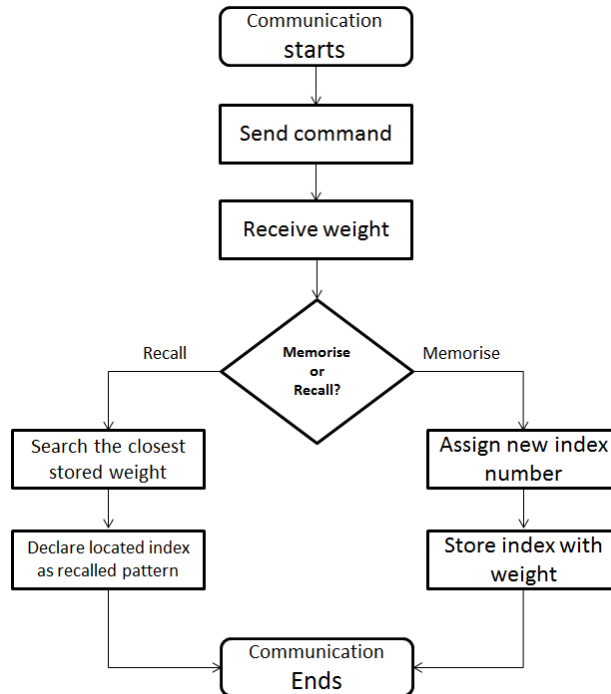


Figure 4.2: S&I operations for memorising and recalling patterns.

The network structure is composed of multiple tracks. A track is a GN array that consists of a set of odd numbers of nodes of the form $(S_i = 2i - 1)$, where S_i is the size of the track number i (i.e. the number of nodes in track i). As discussed in Chapter 3, involving multiple tracks in the network structure aims to enable parallel processing and information exchange of incoming data by dividing the pattern into a set of sub-patterns. This will also allow the network to deal with different pattern types that require multi-dimensional processing. On the other hand, using an odd number of nodes in each track makes the cellular network structure possible and restricts the number of communications between tracks. The cellular network structure allows accumulating nodes' computation outcomes to one track, called the *core track*, which is responsible for delivering these outcomes to the S&I.

Definition 4.3: (Network track) A CwGN network track (Trk) is a GN network that consists of a set of nodes where each node communicates with its direct neighbour nodes in the same track. A CwGN track can be described as follows.

$$Trk_i = \{ND_{i,1}, ND_{i,2}, \dots, ND_{i,m}, \quad m \in \mathbb{N}\} \quad (4.3)$$

where $ND_{i,l}$ is the l^{th} node in Trk_i . Each ND is responsible for sensing or receiving one element of an incoming pattern such that $P_{\varepsilon,ND}: \varepsilon_i \rightarrow ND_{i,l}$, $\varepsilon \in V, i \in \mathbb{N}$, where $P_{\varepsilon,ND}$ is the assignment of an incoming pattern's elements to ND s using Definition 3.1.

The communications between track nodes are called *exchange communications* and can be defined as follows.

Definition 4.4: (Exchange communications) Given a CwGN network track (Trk) that consists of m nodes, exchange communications of a node are two direct connections between a node and its direct neighbours (adjacent) *previous* (p) and *next* (n) nodes in the form $ND_i \rightarrow ND_{i-1} : v$, $ND_i \rightarrow ND_{i+1} : v$ respectively.

where $CND_{i,p}$ and $CND_{i,n}$ are the exchange communications between the i^{th} node and its previous and next nodes respectively, and v is the value assigned to the ND_i . The number of links in each track is equal to twice the number of its nodes ($2S_i$). Here, the last node in a track is assumed to communicate with the first node and vice versa. Figure 4.3 depicts a CwGN track that consists of m nodes. It is assumed that the first node is directly adjacent to the last node.

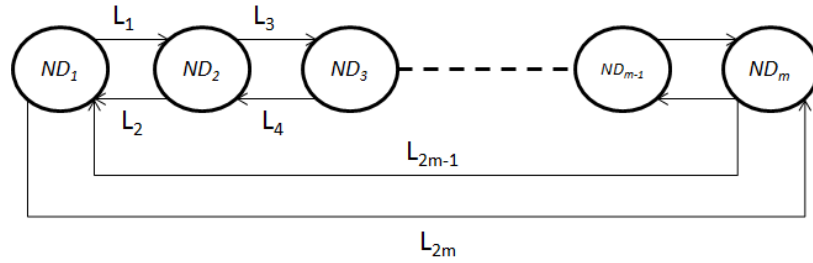


Figure 4.3: Track exchange communications. The arrows show the links between nodes (L_1, L_2, \dots, L_m).

Each node in the CwGN network receives its assigned command and pattern element, exchanges information with adjacent nodes, calculates its weight, and sends its calculated outcomes to another node in the network, named its *inner node*, which resides in the *inner track*, as described in Equation

3.4. The four commands (M,S), (M,X), (R,S), and (R,X) described in Table 3.1 are used to train a CwGN network and recognise patterns. However, it is important to highlight that CwGN does not use the node activation process included in CGN as each neuron position holds only one node. Instead, based on received data from S&I, each node is activated or de-activated according to activation criteria (e.g. high temperature). Another round of the activation process is performed by each node after performing exchange communications. Based on the received data, a node can decide whether to be activated or not. If the node is obtaining a pattern's boundary, it gets activated. Only activated nodes participate in pattern detection steps. This is to maintain limited use of node resources and to reduce the detection time by limiting the number of communicating nodes.

The activation process goes through two stages, namely, *value activation* and *edge activation*. A node's value activation can be achieved by examining its received value. If a node's value complies with certain user defined conditions, it gets activated. One of these conditions is reaching a certain threshold. This can be formally described as follows.

Definition 4.5. (Node value activation) Given a CwGN node $N_{i,j}$ that is assigned to a value $v \in P$, where $P = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$, $\varepsilon_i \in V$ is the incoming pattern, $N_{i,j}$ is activated if $v \geq \varphi$.

where i is the node's position in its track, j is the node's track number, ε_n is the n 'th element of P , V is the set of possible values, and φ is the node activation threshold.

A value activation of a node triggers the start of an exchange communication process for that node with its adjacent neighbours in the same track, as described in Definition 4.4. After receiving information from previous and next nodes, the activated node calculates its edge level according to Equation 4.4. Based on the resulting edge type, it either de-activates or goes for the second level of activation called *node edge activation*, which can be described as follows.

Definition 4.6. (Node edge activation) Given a CwGN value activated node $N_{i,j}$ and its adjacent nodes $N_{i+1,j}$ and $N_{i-1,j}$, the node continues to be active if its edge type $ED_{i,j} \in ED_A$.

where i is the node's position in its track, j is the node's track number, $ED_{i,j}$ is the node's edge type, and ED_A is the set of activation edges values such that $ED_A = \{ED_1, ED_2, \dots, ED_m\}$. For example, if $ED_A = \{DE, LE\}$, the node will get activated only if its edge type is double edge or left edge, as described in Equation 4.4. Otherwise, the node gets de-activated. Edge activated nodes are the only nodes that participate in the reporting communication described in Definition 4.5.

The inner node combines the weights it receives with its own weight and transmits the accumulated weight to its own inner node. This process

continues until a node (or set of nodes) is reached called the *core node*, which is responsible for delivering the accumulated weights to the S&I. Delivering weights to the inner nodes is called *report communication* and can be formally described as follows.

Definition 4.7: (Report communications) Given a CwGN network that consists of a set of tracks, a report communication of a node is the message (connection) between a node in that track and its direct assigned inner node that contains the resulting accumulative weight (ω) in the form $RND_{i,l}: ND_{i,l} \rightarrow ND_{i-1,l} : \omega_{i,l}, \forall l < S_{i-1}$ or $RND_{i,l}: ND_{i,l} \rightarrow ND_{i-1,l-2} : \omega_{i,l}, \forall l \geq S_{i-1}$. Where i is the track number, l is the node's number in track i , and S_{i-1} is the size of the track number $(i-1)$.

Figure 4.4 depicts a CwGN network of 9 nodes and 3 tracks. Figure 4.5 shows the steps that each node in the network performs in the learning process. These steps can be described as follows:

- i. **Receive command:** The node receives the broadcasted command from S&I which contains the operation (memorise or recall) and the pattern element obtaining method (direct receive or sense).
- ii. **Obtain pattern element:** Based on the command message, each node starts obtaining its assigned pattern element ε_i . Each node sets its value (v) according to the obtained pattern element.

- iii. **Exchange communications:** Each node performs exchange communications with its neighbouring nodes and calculates its weight.
- iv. **Report communications:** After compiling the weight, each node reports its weight to its assigned inner node. In this step, each node should wait for reports from outer tracks to accumulate its weight with the reported weights and then start the reporting itself. The core node in the network reports its accumulative weight to the S&I.

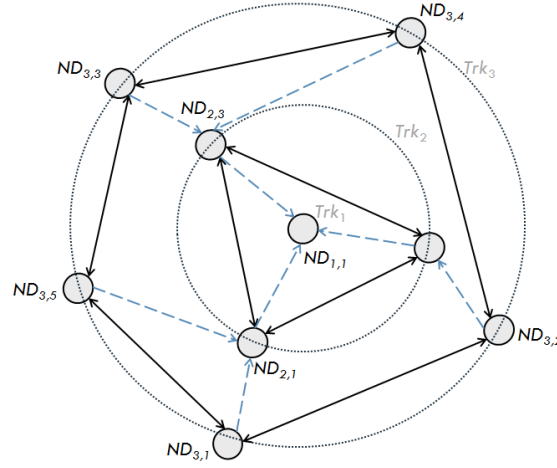


Figure 4.4: CwGN network that adopts a 9 elements pattern size. Solid arrows represent exchange communications, dotted arrows represent report communications.

4.3.3 Pattern edge search

The CwGN scheme represents patterns in terms of weights. The main goal of this approach is to enable pattern transformation detection (e.g. rotation, translation, and dilation). To achieve this goal, the CwGN scheme

searches for edges in the pattern's data domain to determine its boundaries. We assume that sensor nodes are deployed in a grid-like structure in the field of interest to obtain sensory information. To calculate a node's weight, its edge type is determined in accordance with its own value and the values received from its adjacent nodes using exchange communications. There are four edge types: not an edge (NE), when a node's value is less than or equal to the values of its neighbouring nodes; right edge (RE), when the node's value is only larger than the value of the right node (next value); left edge (LE), when the node's value is only larger than the value of its left node (previous value); and double edge (DE), when the value of the node is larger than the values of both adjacent nodes.

However, the types and numbers of edges can be changed according to the recognition problem. For example, the middle element in the pattern (1,1,0) is described as a right edge (RE), as the value of its right neighbouring element is lower than its value. Conversely, for the pattern (0,1,1), the middle element is considered as a left edge (LE). The edge type determination can be described as a function of the values of the node and its neighbouring nodes in the form of $ED = f(C_v, p_v, n_v)$, where ED is the edge type, C_v is the value of the current node, p_v is the value of the previous (predecessor) node in the same track and n_v is the value of the next node (successor) in the same track. This relationship can be described as the following piecewise function.

$$ED = \begin{cases} DE, & p_v < C_v > n_v \\ LE, & p_v < C_v \leq n_v \\ RE, & p_v \geq C_v > n_v \\ NE, & \text{Otherwise} \end{cases} \quad (4.4)$$

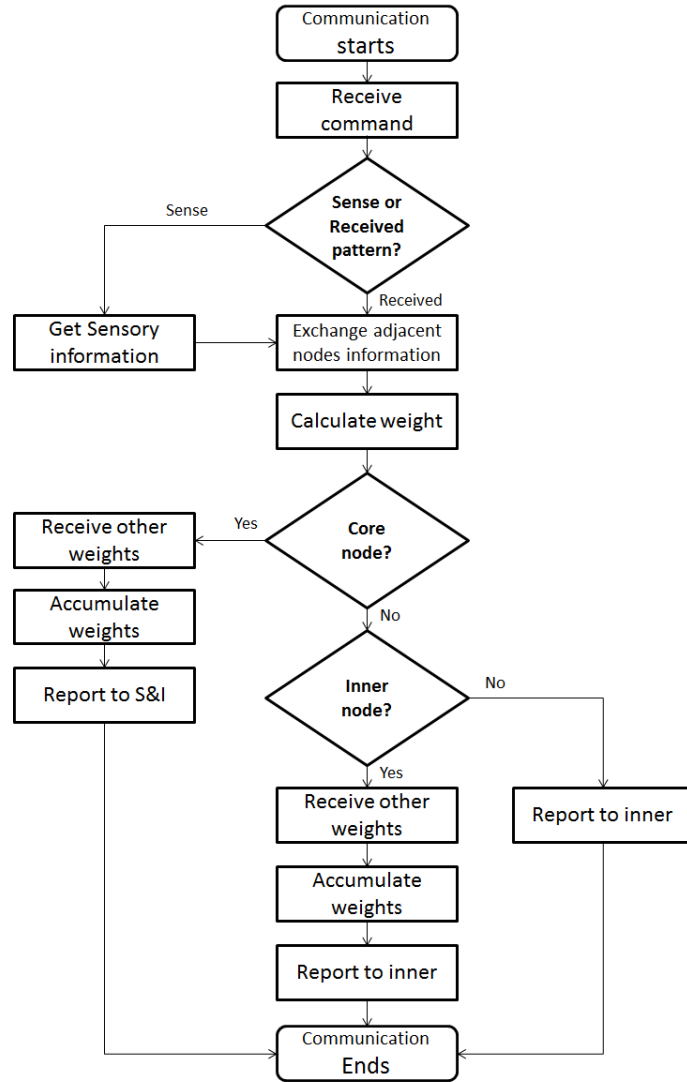


Figure 4.5: CwGN network node operations for weight calculation.

This convention for pattern description is suitable for binary pattern representation. However, in numerical representations we might encounter the

same edge type with a variance in levels. For example, the middle element in the pattern $P_1 = \{3,3,1\}$ is a right edge element, and the same applies for the pattern $P_2 = \{92,90,1\}$. However, these two patterns might be considered as two different types. Consequently, descriptions of edge levels in the CwGN scheme must be attained by factorising the percentage variance between a node's value and its neighbours' values in the same track. The variance ratio between a node (C) and its neighbouring node can be calculated as $VR_{cN} = \frac{(C_v - N_v)}{N_v}$. In this expression, N is a neighbouring node (previous or next), and C_v and N_v indicate the values of the current and neighbouring nodes. However, this function could lead to the problem of dividing by zero. Thus, in such cases an assumption should be made, such as considering the ratio to be equal to C_v . Accordingly, the variance ratio can be expressed as follows.

$$VR_{cN} = \begin{cases} \frac{C_v - N_v}{|N_v|}, & C_v > N_v \forall N_v \neq 0 \\ C_v, & C_v > N_v \forall N_v = 0 \\ 0, & \text{Otherwise} \end{cases} \quad (4.5)$$

where VR_{cN} is the value of the variance ratio of node C value to the neighbouring node N value. By implementing this equation, each node will calculate a ratio value that describes the difference between its value and its neighbour's value. Thus, we can differentiate between nodes, even between nodes with the same edge types. Note that this percentage will be equal to 0 if the relationship to the neighbouring node is not an edge (i.e. the current value is less than or equal to the neighbouring value).

A nodal weight can be calculated using the variance ratio and the edge type. However, other relational functions can also be used. Each edge type is assigned a fixed value. The choice of edge value can then be used to factorise weights and differentiate between edge types. A nodal weight is a function of its variance ratio to its neighbours, namely, the previous and next nodes, in the form of $\omega_c = f(VR_{cp}, VR_{cn})$, where ω_c is the node's c (current) weight, p is the previous node, and n is the next node. This function can be determined as the summation of VR_{cp} and VR_{cn} . To include the node's edge types in this function, we assume that each ED value in Equation (4.4) has been set to a constant number ED_v . Accordingly, the current node's weight is calculated as follows.

$$\omega_c = ED_v \cdot VR_{cn} + ED_v \cdot VR_{cp} \quad (4.6)$$

Considering the conditions included in 4.4 and 4.5, this equation will only have a value if the current node's value is higher than at least one of its neighbouring node's values. This is because the value is multiplied by zero when the current value is less than or equal to the value of the adjacent node.

To assign each pattern with a unique weight, each node reports its calculated weight to its assigned inner node in the network. Each inner node adds the reported weights and reports the result to its assigned inner node. This continues until each node in the core of the network has reported its accumulated weight value to the S&I. The S&I normalises the received accumulated weight by a predefined normalising factor Nf that can be the pattern (network) size (S). In other words, the i^{th} pattern's weight ($\rho\omega_i$) is the

sum of the entire network's node weights divided by a normalising factor (Nf) and can be calculated as follows.

$$\rho\omega_i = \frac{\sum_{m=1}^S \omega_m}{Nf} \quad (4.7)$$

4.4 CwGN Communication Scheme

In this section, the communications of a CwGN network are described. This includes adjacency communications, reporting communications, and the CwGN communication protocol. The CwGN scheme has two types of network communications scenarios: standard and track-linking communications. In the standard communication style, nodes exchange information with adjacent nodes in the same track. In this scenario, the incoming pattern is divided into several sub-patterns according to the number of tracks, and each sub-pattern's weight is calculated separately from those of other sub-patterns. This structure is useful when different data types and multi-dimensional patterns are to be processed using one network. The standard is described in Figure 4.4. Track-linking communication style aims to allow the calculation of weights between tracks or sub-patterns. This communication paradigm allows one node in each track to perform adjacent communications with nodes in its outer track. This will make it possible to avoid losing weights caused by dividing a pattern into sub-patterns and can be used for one dimensional patterns and single data types. Figure 4.6 shows the track-linking communication paradigm.

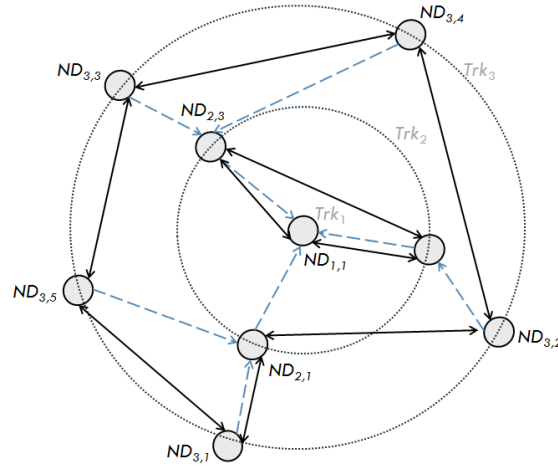


Figure 4.6: Track linking CwGN network communication scheme. The solid arrows indicate information exchange between adjacent nodes and dotted arrows indicate reporting communications to inner nodes.

4.4.1 CwGN communication requirements

To perform CwGN network node communications, it is assumed that a medium access control (MAC) protocol is present and available to support the network. MAC protocols control the communications of a network by setting the rules and steps for sending information amongst the network's nodes so as to share the available medium. In WSNs the efficiency of a MAC protocol will affect the sensors' lifetime by reducing transmission collisions, which reduces the number of retransmissions of packets [7]. In WSNs, nodes conserve energy resources by alternating between low power sleep mode and active mode. MAC protocol supports conserving energy resources for WSN nodes by determining timeslots for sleep and active modes. In addition, when using MAC protocols, each sensor can have a unique MAC address that differentiates it from other sensors, allowing direct communication between two nodes. For a

CwGN scheme, choosing a MAC protocol should take into account the special requirements and steps of the network's communications.

During the initialisation of a CwGN network, each node should be provided with a track number that it will work on. This allows the node to determine its communication process. For example, a node in the outermost track will send its information to adjacent nodes, receive adjacent nodes' information, and report its calculated weight to its assigned inner node. A node in a middle track performs the same steps except that it will wait for reports from outer track nodes before reporting the accumulated weights to its inner node. Determining a node's track can be performed statically or automatically during the initialisation phase of the CwGN network. Static track determination means that each node is provided with information about its track, adjacent nodes in the same track, and its assigned inner node to report its calculated or accumulated weight. This initialisation would be less complex in terms of computations and communications. However, the flexibility of adding new nodes to the network or adopting dynamic changes such as mobile nodes or clusters will be limited. Automatic track determination can be achieved by allowing each node to communicate with its neighbouring nodes and allowing the base station to determine its track, adjacent nodes, and inner nodes after the deployment of the network. This approach will provide more network flexibility to adapt to changes that may be required in the network design. However, this will lead to an increase in the number of communications in the

network. It is also important to take into account the distance between nodes and the communication ranges of sensors in determining the track size.

To ensure the functionality of the CwGN network, each node should be fed with sufficient information about how to react to failures or de-activated neighbouring nodes. In WSNs, it is common to have failure nodes that can no longer communicate due to running out of energy or physical damage. To overcome the effect of such phenomena on recognition, each node should take into account the steps for dealing with unavailable nodes. For adjacent nodes (i.e., predecessor and successor), a node should assume the value of a failure communicating node to be zero. This is to standardise the weight calculation with a zero level. To avoid losing weights in cases of inner node failures, each node should hold (or search for) information about alternative inner nodes. Each node should be supplied with a list of ordered alternative inner nodes in the inner track within its communication range. If a node cannot communicate with its inner node, it sends its weight to the first alternative inner node. If the alternative node is not responding, it sends it to the second alternative. This continues until the node reports its weight to one of its assigned inner nodes. If none of the alternative nodes respond, the node should report its weight to the base station. Figure 4.7 illustrates the alternative reporting process.

In Figure 4.7 (a), the reporting node, tries to send its weight information to alternative inner node 1 in the inner track after failing to communicate with its assigned inner node. In Figure 4.7 (b) the node sends its weight information to the base station after failing to communicate with all possible alternatives.

Since the number of communications of alternative nodes could increase due to failures occurring in assigned inner nodes, the number of alternatives should be determined in such a way as to avoid exhausting inner nodes that act as alternatives. Since each node is presumed to receive only one message at a time, the learning and recognition duration cycle will be affected by the number of communications increasing in alternative nodes, as will be discussed later in this chapter. Consequently, setting the total number of possible alternative inner nodes should take the effect on the duration of learning cycle into account. Finally, the base station should be able to predict the time needed to wait for reports from nodes that are unable to report to their assigned and alternative inner nodes before it declares the total weight of a pattern.

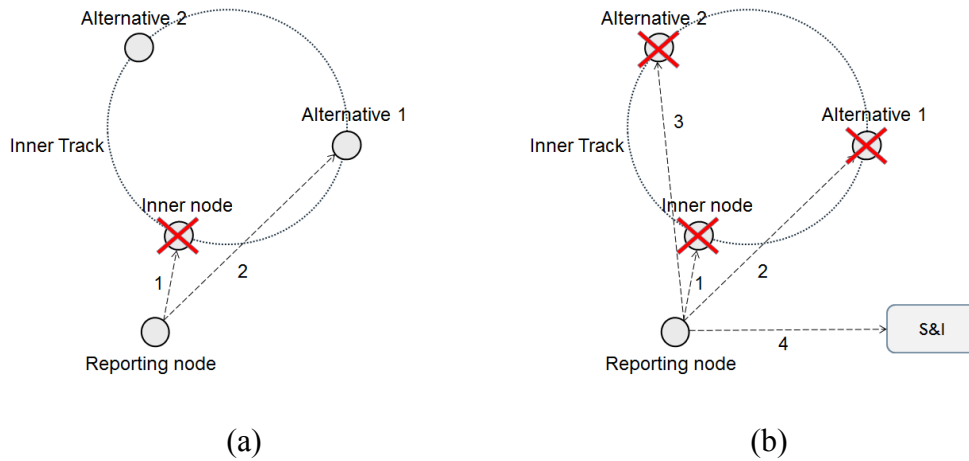


Figure 4.7: Report messages from a node to its alternative assigned inner nodes. (a) Reporting node attempts to send its report to the second alternative assigned inner node. (b) Reporting node reports directly to the BS after failing to report to all alternative inner nodes.

4.4.2 CwGN communication protocol

The CwGN communication protocol describes the main steps of CwGN network communications. By completing this protocol, a CwGN network will memorise or recall an incoming pattern. The protocol consists of four main steps as follows:

Step 1 $S\&I \rightarrow ND_1, ND_2, \dots, ND_m : (C_i, P_i) \{ (C_1, P_1), (C_2, P_2), \dots, (C_m, P_m) \}$

In the first step of the communication, the S&I (i.e. base station) sends the command C_i and the pattern elements P_i to each node in the network. As explained in section 3.2, a command can be either to memorise (M) or recall (R) and the pattern element can be a value to use for training, or (X) to initiate sensors to use the sensory information. Each node receives only one element of the pattern. For example, if the pattern is (1,5,7,4), the S&I will send the values 1, 5, 7, and 4 to the nodes 1, 2, 3, and 4 as pattern elements respectively. Obtaining sensory information can happen in two ways. In the first scenario, the S&I sends a memorise or recall command to nodes in order to start obtaining sensory information and continue the communication steps. In the second scenario, the nodes are programmed to obtain information periodically.

Step 2 $ND_i \rightarrow ND_p, ND_n: v_i$

Each node in the network ND_i starts the information exchange process with adjacent nodes. After receiving or obtaining the pattern element information, each node sends its value v_i to two nodes: previous ND_p , and next ND_n . These nodes are located in the same track on which the first node resides. The aim of this step is to allow each node to calculate its weight according to

Equation 4.6. After completing this step, each node will receive two values from its adjacent nodes representing the pattern elements by which a node should calculate its weight

Step 3 $ND_i \rightarrow N_{inner}$: ω_i

After obtaining pattern information from adjacent nodes and calculating its weight, each node reports its weight to its assigned inner node in the inner track. The weight will be a result of computations related to the node's value and adjacent nodes' values. This is applicable to nodes in the outermost track. For nodes in middle tracks, each node should wait for reports coming from outer track nodes. Once these reports are obtained, the node adds its calculated weight to the incoming weights according to Equation 4.6 and then starts the reporting step. The reporting communication step continues until reaching the core node. By the end of this step, the core node will obtain the accumulated weight of the present pattern.

Step 4 $ND_c \rightarrow S\&I$: ω_c

The core node (or set of nodes) ND_c will send the total pattern's weight ω_c to the base station in order to store or recall the pattern. The base station normalises the incoming weight by dividing Nf . The base station holds a database of trained patterns associated with their weights. If the pattern needs to be memorised, the base station assigns it a new index number and associates this index number with the total weight obtained by the network. If the pattern is to be recalled, the base station searches its database to find the closest value

to the total weight given by the CwGN network and declares it as the recalled pattern.

4.5 Complexity of CwGN Algorithm

This section estimates the complexity of the CwGN algorithm. Table 4.1 defines the terms used in complexity estimation. One of the goals of the scheme presented here is to provide real time recognition capabilities while maintaining a low level of resource use to suit WSNs. Hence, estimating the learning cycle duration is important to evaluate the scheme's feasibility in online operations. For such estimation, we assume that all network nodes are activated by a given pattern to estimate the maximum time required to learn or recall an incoming pattern. The computational and communication overheads of a pattern of size S can be estimated by the duration of each CwGN step. The first step is the pattern receiving step. This step involves broadcasting the command message by the S&I and sensing the pattern by the nodes. The estimated time required for this step is as follows.

$$T_{rec} = T_{send} + T_{sense} \quad (4.8)$$

The following step exchanges sensory information by nodes. Taking parallelism into account, the time estimate can be described as follows.

$$T_{exch} = 2.T_{send} \quad (4.9)$$

Table 4.1: Description of terms used in CwGN complexity analysis.

Symbol	Name	Description
N_{trk}	Number of tracks	The number of tracks in the network
S	Pattern (network) size	The problem size which equals the network size
S_i	Track i size	Number of nodes in track number i
Mp	Memorised patterns	Number of memorised patterns in the S&I
T_{add}	Addition time	Time for one node to complete an addition operation
T_{SI}	S&I time	The time required by the S&I to search or add a pattern to its database
$T_{compare}$	Compare time	The time required by a node to compare two weights
$T_{compute}$	Computing time	The time required by a node to perform weighting computations
T_{div}	Division time	The time required for a node to complete a division operation
T_{exch}	Exchange time	The time required by nodes to conduct exchange communications
T_{read}	Reading time	Time required by S&I to read a single pattern's stored weight
T_{rec}	Pattern receiving time	The time needed by a CwGN network to obtain an incoming pattern including the S&I command
T_{report}	Report time	The time required by the network to perform report communications
T_{send}	Send time	The time required to send a message from one node to another
T_{sense}	Sense time	The time required by a node to obtain sensory information
T_{total}	Total network time	Time required by the CwGN network to perform PR operations

This is followed by the computing of weights by nodes. Assuming that the relation function captures the variance level relations described in Equations 4.4 and 4.5 that delivers Equation 4.6, the computations involve

comparing received values with sensed information, addition, and division operations. The time required for this step can be estimated as follows.

$$T_{compute} = 2.T_{compare} + T_{add} + T_{dev} \quad (4.10)$$

The reporting step is the most time demanding step in the CwGN scheme as each reporting node waits to receive reports from other nodes in other tracks. Referring to the deployment process, the number of tracks in a CwGN network for a pattern size S is equal to the number of deployment iterations and, according to Equation 3.7, is equal to the square root of S . This can also be written as $N_{trk} = \sqrt{S} = 10^{\frac{\log S}{2}}$. Accordingly, reporting time can be estimated as the parallel sending time and weight accumulation time as follows.

$$T_{report} = T_{send} \cdot N_{trk} + T_{add} \cdot (N_{trk} - 1) \quad (4.11)$$

$$T_{report} = T_{send} \cdot (10^{\frac{\log S}{2}} - 1) + T_{add} \cdot 10^{\frac{\log S}{2}} \quad (4.12)$$

$$T_{report} = (T_{send} + T_{add}) \cdot 10^{\frac{\log S}{2}} - T_{send} \quad (4.13)$$

This excludes the outermost track from waiting for reports and includes the inner node's report to the S&I. Once the weights are accumulated and reported to the S&I, the S&I stores or recalls the incoming weight. The S&I time (T_{SI}) will be equal to T_{write} in cases of memorisation. Assuming that S&I performs a binary search to find the associated pattern, its recall time can be estimated as follows.

$$T_{SI} = (T_{read} + T_{compare}) \cdot \log_2(Mp) \quad (4.14)$$

The total CwGN network time can be estimated as follows.

$$T_{total} = T_{rec} + T_{exch} + T_{compute} + T_{report} + T_{SI} \quad (4.15)$$

$$T_{total} = T_{send} + T_{sense} + 2.T_{send} + T_{compare} + T_{add} + T_{dev} \\ + T_{send} \cdot (10^{\frac{\log S}{2}} - 1) + T_{add} \cdot 10^{\frac{\log S}{2}} + T_{SI} \quad (4.16)$$

$$T_{total} = \left(2 + 10^{\frac{\log S}{2}}\right) \cdot T_{send} + (T_{send} + T_{add}) \cdot 10^{\frac{\log S}{2}} + T_{sense} \\ + 2.T_{compare} + T_{dev} + T_{SI} \quad (4.17)$$

To simplify these calculations, we assume that communicational operations times ($T_{sense}, T_{send}, T_{report}$) are equal and denoted as ($T1$). Similarly, computational operations times ($T_{add}, T_{compare}, T_{dev}, T_{read}, T_{write}$) are assumed to be equal, and we denote them as ($T2$). Consequently, T_{SI} in memorisation will be equal to $T2$ because it equals T_{write} . Accordingly, substituting $T1$ and $T2$ into (4.17) can provide an estimate of the total network time required for memorisation.

$$T_{total} = 3.T1 + 4T2 + (2.T1 + T2) \cdot 10^{\frac{\log S}{2}} \quad (4.18)$$

In recall, T_{SI} will be equal to $(2.T2)\log_2(Mp)$, and the total network time required for recall can be estimated as follows.

$$T_{total} = 3.T1 + 3T2 + (2.T1 + T2) \cdot 10^{\frac{\log S}{2}} \\ + (2.T2)\log_2(Mp) \quad (4.19)$$

According to these equations, the CwGN network's response time is proportional to the square root of the network size in the form of $O(\sqrt{S})$. This minimises the effect of pattern size increase and provides the scheme with a high level of scalability. Figure 4.8 shows the estimated recall time as a

function of the pattern size (S); assuming 1 millisecond for $T1$, 1 microsecond for $T2$, and 10000 for Mp . Figure 4.9 shows the estimated recall time for a CwGN of $S=16$ according to the increase in the number of stored patterns Mp with the same set of time assumptions made in Figure 4.8. The figure shows that the response time increases in proportion to the square root of pattern size S . Figure 4.9 shows that the response time initially increases but then becomes insensitive to the increase in the number of stored patterns.

4.6 Zoning Approach for Online and Multi-Dimensional Recognition

It was shown in the previous section that the time complexity of CwGN is proportional to the square root of the pattern size. This means that time complexity has a low increase rate. However, for large scale online applications, such an increase could reach a level that makes recognition exceed the online criteria. Another problem is the heterogeneity of incoming data that may require multi-dimensional processing. An example of such a case would be having sensor nodes that detect one data type (e.g. temperature) and other nodes detecting a different data type (e.g. speed). It has been suggested in sub-section 4.3.2 that such problems can be handled using the multiple tracks network structure where each track deals with one data type. However, the constraints of the network structure that require a limited odd number of nodes in each track can restrict the usefulness of such a solution.

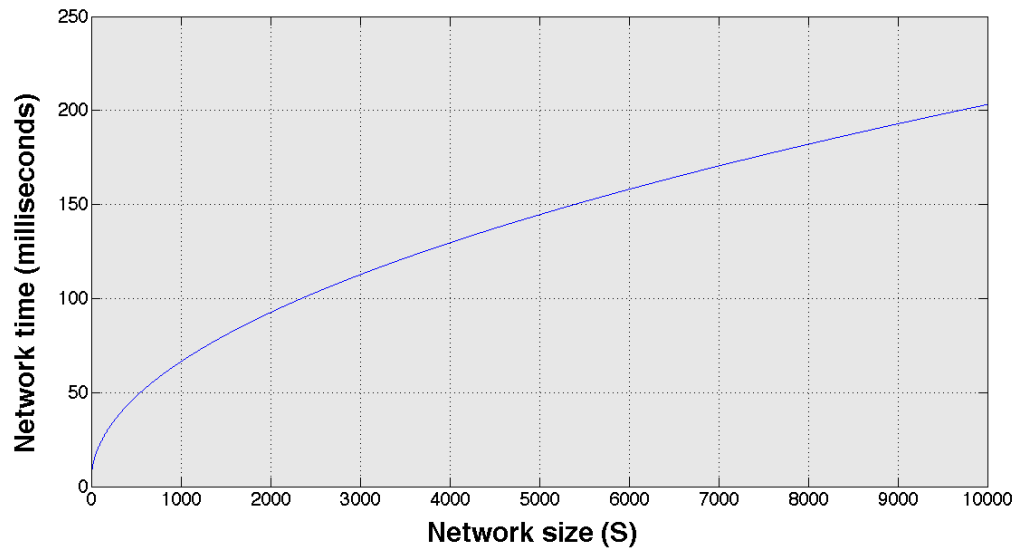


Figure 4.8: The estimated recall time of the CwGN network with respect to increasing pattern size.

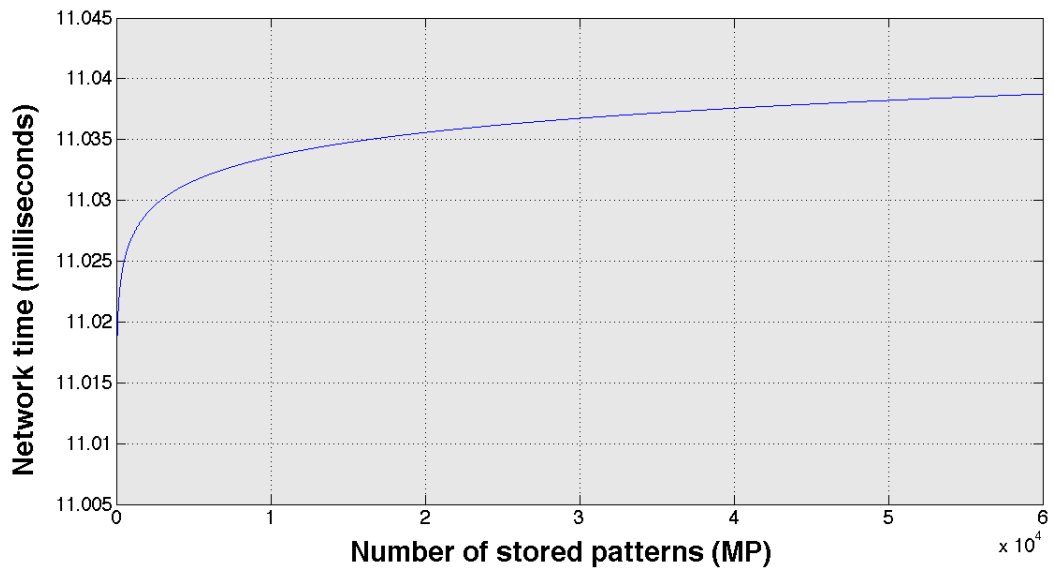


Figure 4.9: The estimated recall time of the CwGN network as a function of the number of stored patterns in the range from 0 to 60000 patterns.

To enable online learning and multi-dimensional data processing, the CwGN network is divided into smaller zones. Each zone is a CwGN network functions in parallel with other networks to fulfil the following constraint.

$$T_Z + T_{SI} \leq \tau \quad (4.20)$$

where T_Z is the network zone time and τ is the upper time limit for completing an online learning operation. In case of having heterogeneous data types, each zone is assigned to process only one type. Each zone feeds S&I with its concluded weight value, as shown in Figure 4.10. To determine the maximum number of zones (N_{zones}) that fulfil this constraint, it is assumed that T_Z is the maximum time for a network zone, that is, the time when all nodes in the network zone are activated. Additionally, it is assumed that the zones are uniformly distributed such that each zone has the same number of nodes. Consequently, T_Z will be equal to the total time of a CwGN network of size $\frac{S}{N_{zones}}$ and the network time can be estimated as follows.

$$T \frac{S}{N_{zones}} + T_{SI} = \tau, \quad N_{zones} \in \mathbb{N} \quad (4.21)$$

From Equations 4.18 and 4.19, the recall time will always be higher than the memorisation time when $Mp \geq 2$. This is due to the process by which the S&I searches for a matching pattern. Assuming that recall time is higher than memorisation time and substituting (S) with $(\frac{S}{N_{zones}})$ in Equation 4.19, the number of required network zones can be estimated using Equation 4.21 as follows.

$$N_{zones} = S. \left(\frac{\tau - 3.T1 - 3.T2 - (2.T2)\log_2(Mp)}{2.T1 + T2} \right)^{-2} \quad (4.22)$$

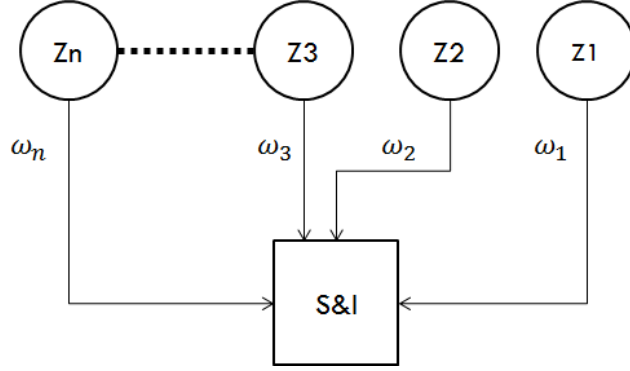


Figure 4.10: Parallel CwGN network zones

For example, if $\tau = 100$ milliseconds, $T1 = 1$ millisecond, $T2 = 1$ microsecond, and $Mp = 10000$ patterns, a CwGN network of size 2448 will be capable of performing learning operations within the time limit (i.e. 100 milliseconds). This means that the network needs to add one more zone when increasing the network size by more than 2448 nodes in order to ensure performance of recognition operations within 100 milliseconds. It is important to note that this assumes that all nodes in the network are activated, which is the worst case scenario. The complexity of the CwGN scheme can provide learning capabilities for large scale patterns within predictable duration limits. A CwGN network has a predictable logarithmic relationship between network size and response time. Additionally, the network response time for memorisation and recall operations is insensitive to increases in the number of

stored patterns. Therefore, the CwGN scheme offers a highly scalable network architecture enabling real time applications.

4.7 CwGN Message Sequence Models

Message sequence models for a CwGN network aim to govern the message exchange process between active nodes in the MAC layer, in accordance to the recognition scheme and protocol described in sections 4.3 and 4.4. In addition, the proposed models attempt to minimise communicational overheads by setting a timing sequence for network nodes to exchange and report messages thus restricting time delays caused by inactive nodes. This section will also use the abbreviations listed in table 4.1 when estimating communicational overheads.

Sub-section 2.2.3 discussed the MAC protocols' importance for WSN communications. Additionally, it has been highlighted that traditional protocols allow a single communicational channel for each node while new research presents multi-channel protocols to support multi-task operations. In this sub-section, the different types of existing MAC protocols will be presented, and MAC message exchange models for a CwGN scheme to function over these protocols will be proposed. The proposal of these models will help in estimating the time and resource requirements of a CwGN scheme as communicational overhead is the most time and resource consuming part of WSNs. The section will present different sequence models and possible scenarios for each model while analysing the time overhead for each scenario.

MAC protocols use the term *frame* to describe a message from a node in the network to another node in the data-link layer. A MAC frame is a sequence of bits that contains the necessary information to deliver a message from one node to another [35]. Figure 4.11 shows an example of a MAC frame. However, different protocols may have different frame structures and bit lengths for each field. The frame shown in Figure 4.11 contains six fields. The *Preamble* field contains a set of bits that are used to occupy the channel. A receiver will listen to a preamble bit sequence in order to start receiving an incoming message. In CwGN communication models there are two types of preamble, namely, *sending*, and *not sending*. The sending preamble indicates that a node is going to send a full frame to the receiver. The not sending preamble indicates that a node will have no frames to send. The not sending preamble will be used in reporting communications in order to allow a node to inform its inner node that it is not edge activated and it will not send weight information. The *Address* field is used to determine source and destination MAC addresses. The *Control* field is used for control purposes such as message sequences and acknowledgements. The *Checksum* field is used for error detection and correction purposes. The *Flag* field is used to indicate the end of transmission.

There are four types of WSN MAC protocols [39]: frame-slotted synchronous (FS-Sync), frame-slotted asynchronous (FS-Async), MC synchronous (MC-Sync), and MC asynchronous (MC-Async). Table 4.2 summarises these models and the abbreviations for each one that will be used

in the rest of this research. In this sub-section, a CwGN communicational model for each MAC protocol type will be presented. Additionally, different possible communicational scenarios will be discussed. This section will also use the abbreviations listed in Table 4.1 when estimating communication time complexity for each model.

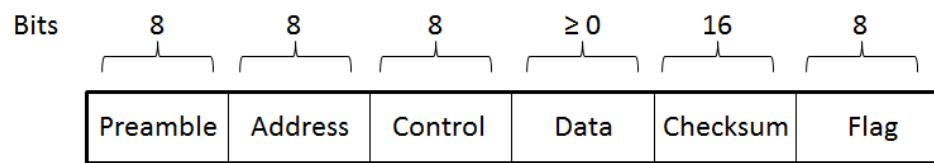


Figure 4.11: A general MAC frame structure [35].

Table 4.2: Summary of existing MAC protocol types for WSNs.

Protocol type	Abbreviation	Description
Frame-slotted asynchronous	(FS-Async)	Uses one MAC channel to send or receive a message in an asynchronous mode
Frame-slotted synchronous	(FS-Sync)	Uses one MAC channel to send or receive a message in a synchronous mode
Multi-channel asynchronous	(MC-Async)	Divides the MAC channel into several channels and sends or receives multiple messages at the same time in asynchronous mode
Multi-channel synchronous	(MC-Sync)	Divides the MAC channel into several channels and sends or receives multiple messages at the same time in asynchronous mode

4.7.1 Frame-slotted asynchronous CwGN model

In this model, each node is allowed to send or receive one frame at a time. The sending node starts by sending the preamble. A receiving node senses the channel. In exchange communications, a value active node (as described in Definition 4.5) will send a MAC frame that has a sending preamble with its value (v) in the data field to its next node in the track. The next node will be sensing the channel. Once a preamble is received it starts receiving the rest of the frame. After the sending node finishes sending the frame, it starts listening for sending preamble from the next node. If no preamble is received it starts sending to its previous node. Figure 4.12 and Figure 4.13 show the communication sequence scenarios for a value active node (n) and its neighbouring nodes. Figure 4.12 shows the normal message sequence when the next and previous nodes are both active. Figure 4.13 shows the scenarios when adjacent nodes to the sending one are inactive. Figure 4.13 (a) shows the sequence when one node (the next node: $n+1$) is active and the other node (previous node: $n-1$) is inactive. Figure 4.13 (b) shows the sequence when both neighbouring nodes to node n are inactive, W_t is the wait time that a node should wait before performing other computations or communications, P_t is the time required to send a full preamble field, and Δ_t is the error delay time that may occur in every communication between two nodes due to physical factors.

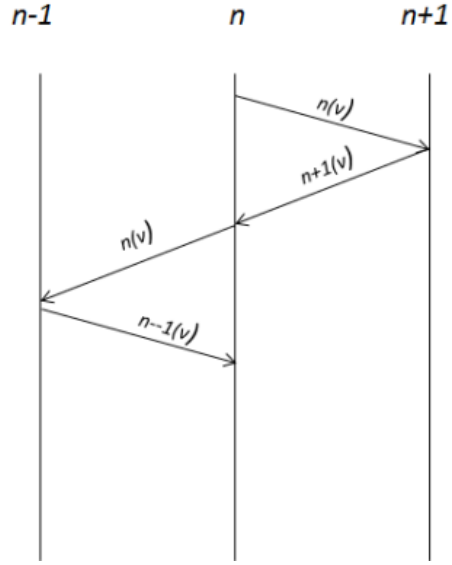


Figure 4.12: CwGN FS-Async message sequence model.

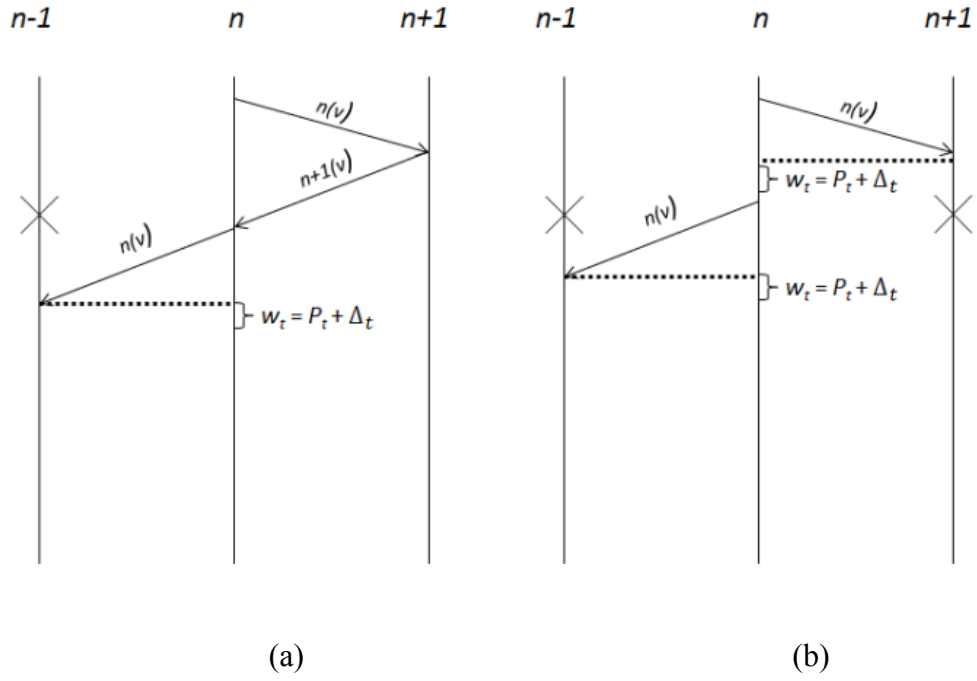


Figure 4.13: CwGN FS-Async message sequence model scenarios. (a) One adjacent node (previous) is inactive. (b) Both adjacent nodes are inactive.

In the normal scenario shown in Figure 4.12, the exchange time for the node (n) can be estimated as $4T_{send} + 4\Delta_t$ as it will send two frames and receive two messages, where T_{send} is the time required to send one frame, and Δ_t is the error delay time that may occur in every communication between two nodes due to physical factors. In the scenario shown in Figure 4.13 (a) the previous node ($n-1$) is inactive. In such a scenario the node n will wait an amount of time (W_t) that is equal to the time required to send a full preamble field (P_t) in addition to the Δ_t . Hence, the time estimation in such a scenario will be $3T_{send} + P_t + 4\Delta_t$. The same will be applicable if the next node is de-activated and the previous node is active. The last scenario described in Figure 4.13 (b) is when both neighbouring nodes to n are de-activated. In such a scenario the time estimation will be $2T_{send} + 2P_t + 4\Delta_t$. It can be concluded that the maximum time required for a node's exchange communications is.

$$T_{exch} = 4T_{send} + 4\Delta_t \quad (4.23)$$

P_t is shorter than T_{send} as the P_t is involved in sending only a sub-part of the whole frame. The minimum exchange time for a node can be estimated as follows.

$$T_{exch} = 2T_{send} + 2P_t + 4\Delta_t \quad (4.24)$$

Taking the parallelism design of a CwGN network into account, these limits can be used to estimate the entire network's maximum and minimum exchange time as all the network's nodes perform exchange communications in the same time.

To perform report communications, an edge activated node (as described in Definition 4.6) will first wait for reports from outer track nodes. Then, the activated node will send a MAC frame that has a *sending* preamble, with its accumulated weight (ω) in the data field to its assigned inner node in the inner track. The inner node will be sensing the channel. Once a preamble is received it starts receiving the rest of the frame. If the node is not an edge active node, the node sends a *not sending* preamble. This is to minimise the wait time for the inner node as preamble sending time is less than frame sending time. Figure 4.14 shows the two possible scenarios of a reporting process for a node ($n_{i,l}$), its outer track node ($n_{i,l-1}$), and its assigned inner node ($n_{i,l+1}$), where i is the node's position in its track and l is the node's track number. Figure 4.14 (a) shows the reporting sequence if all nodes are activated and Figure 4.14 (b) shows the sequence when ($n_{i,l-1}$) is inactive.

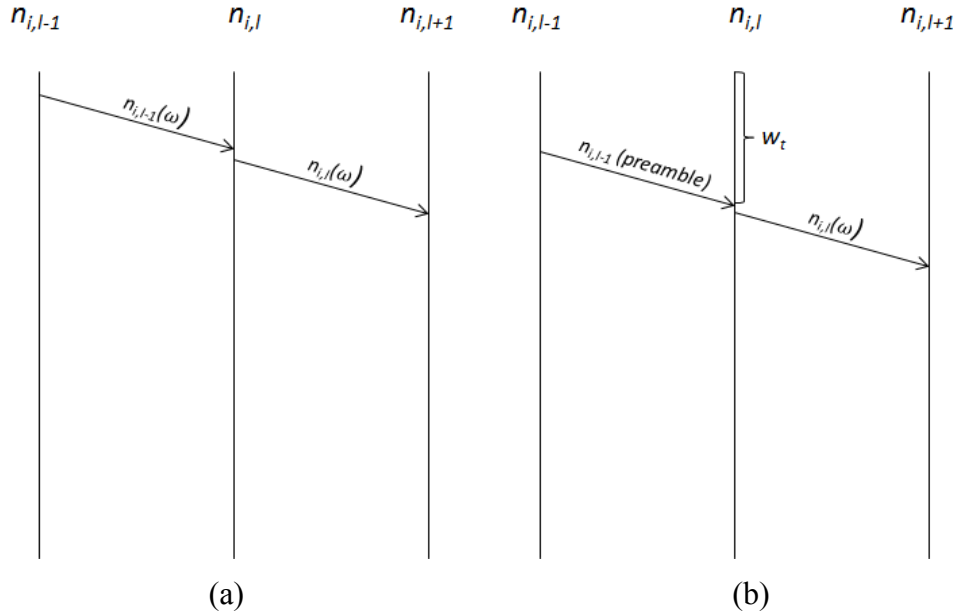


Figure 4.14: Message sequence for FS-Async report communications.

In the first scenario, the report time for the node (n) can be estimated according to its track number (l) and the total number of the network's tracks (N_{trk}). Since each active node waits for a report from other active nodes in outer tracks, the maximum wait time for node $n_{i,l}$ to receive node's $n_{i,l-1}$ report can be estimated as $w_t = (N_{trk} - l)(T_{send} + \Delta_t)$. That estimate is based on the assumption that all nodes are active and will send report frames. Consequently, the network's *core node* will have a maximum wait time of $(N_{trk} - 1)(T_{send} + \Delta_t)$. Since the core node will send a report to the S&I, the maximum reporting time for the network can be estimated as follows.

$$T_{report} = N_{trk}(T_{send} + \Delta_t) \quad (4.25)$$

The minimum report time can be estimated using the second scenario. In that scenario, if all outer track nodes of node $n_{i,l}$ are inactive, the waiting time for $n_{i,l}$ can be estimated as $w_t = (N_{trk} - l)(P_t + \Delta_t)$. And the minimum reporting time for the entire network can be estimated as follows.

$$T_{report} = N_{trk}(P_t + \Delta_t) \quad (4.26)$$

The total communication time (T_{comm}) of the network can be calculated as the summation of T_{report} and $T_{exchange}$. Accordingly, the maximum total communication time in such a model can be estimated as follows.

$$T_{comm} = (N_{trk} + 4)(T_{send} + \Delta_t) \quad (4.27)$$

And the minimum can be estimated as follows.

$$T_{comm} = (N_{trk} + 2)P_t + (N_{trk} + 4)4\Delta_t + 2T_{send} \quad (4.28)$$

4.7.2 Frame-slotted synchronous CwGN model

A synchronous communication model aims to guarantee message delivery using acknowledgement (*Ack*) messages. After sending a frame, a sending node waits a period of time to receive *Ack* from the receiving node. If no *Ack* is received, the sending node retransmits the message assuming that the sent message was lost. In this sub-section we discuss the synchronous model in terms of sending frames and acknowledgements without dealing with the retransmission process as it requires further discussion and research that may include the physical level. Similar to the FS-Async model described in the previous sub-section, a value active node starts exchange communications by sending its value (v) in a MAC frame to its next node in the track. After receiving the frame, the next node replies with an acknowledgement frame. Then it sends its own value frame if it is active. If not, the next node sends a not sending preamble. This is to inform the communicating node that there is no value to be sent, and to allow it to finalise the information exchange process. After an activated node performs a successful exchange communication with its next neighbour, it performs the same steps with its previous node. Figure 4.15 shows the sequence when all nodes are active. Figure 4.16 shows two possible scenarios for an active node and its adjacent nodes: (a) previous node ($n-1$) is inactive and (b) both previous and next nodes to node n are inactive.

From Figure 4.15 it can be seen that a full exchange communication of a node will involve sending and receiving 8 frames. This excludes the

retransmission. Hence, the total exchange time for a node in such a scenario can be estimated as $T_{exch} = 8T_{send} + 8\Delta_t$. In the scenario shown in Figure 4.16 (a) the previous node is inactive. Hence node n will either receive a not sending preamble from $n-1$ or wait for P_t to retransmit the sent frame. Consequently, the time of such an exchange scenario will be $T_{exch} = 5T_{send} + 6\Delta_t + P_t$. The same applies if the next node is inactive and the previous one is active. The last scenario shown in Figure 4.16 (b) requires both inactive adjacent nodes to send a not sending preamble to allow n to finish the exchange transmission process. In this case the time estimate will be $T_{exch} = 2T_{send} + 4\Delta_t + 2P_t$. Since P_t is less than T_{send} , and the network is functioning in parallel, the maximum exchange time for a network that runs a FS-Sync model can be estimated as follows.

$$T_{exch} = 8T_{send} + 8\Delta_t \quad (4.29)$$

And the minimum will be as follows.

$$T_{exch} = 2T_{send} + 4\Delta_t + 2P_t \quad (4.30)$$

That, assuming that there is at least one active node in the whole network.

To perform report communications, an edge activated node (as described in Definition 4.6) will firstly wait for reports from outer track nodes. Then, it will send an *Ack* frame to its reporting node. Then, the activated node will send a MAC frame that has a *sending* preamble, with its accumulated weight (ω) in the data field, to its assigned inner node in the inner track and receives an *Ack* to end the reporting communication. If the reporting or inner node is not an edge active node, the node sends a *not sending* preamble. This is

to minimise the wait time for the inner node as the preamble sending time is less than the frame sending time. Figure 4.17 shows the two possible scenarios for the reporting process for a node ($n_{i,l}$), its outer track node ($n_{i,l-1}$), and its assigned inner node ($n_{i,l+1}$) where i is the node's position in its track and l is the node's track number. Figure 4.17 (a) shows the reporting sequence if all nodes are activated and Figure 4.17 (b) shows the sequence when ($n_{i,l-1}$) is inactive.

In the first scenario, the report time for the node (n) can be estimated in a similar way to the first reporting scenario presented for the FS-Async in the previous sub-section. Taking the *Ack* frames into account, the maximum reporting time for the network can be estimated as follows.

$$T_{report} = 2N_{trk}(T_{send} + \Delta_t) \quad (4.31)$$

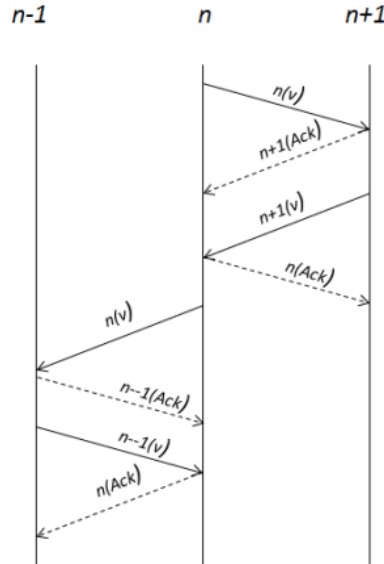


Figure 4.15: : CwGN Message sequence for FS-Sync exchange communicational model.

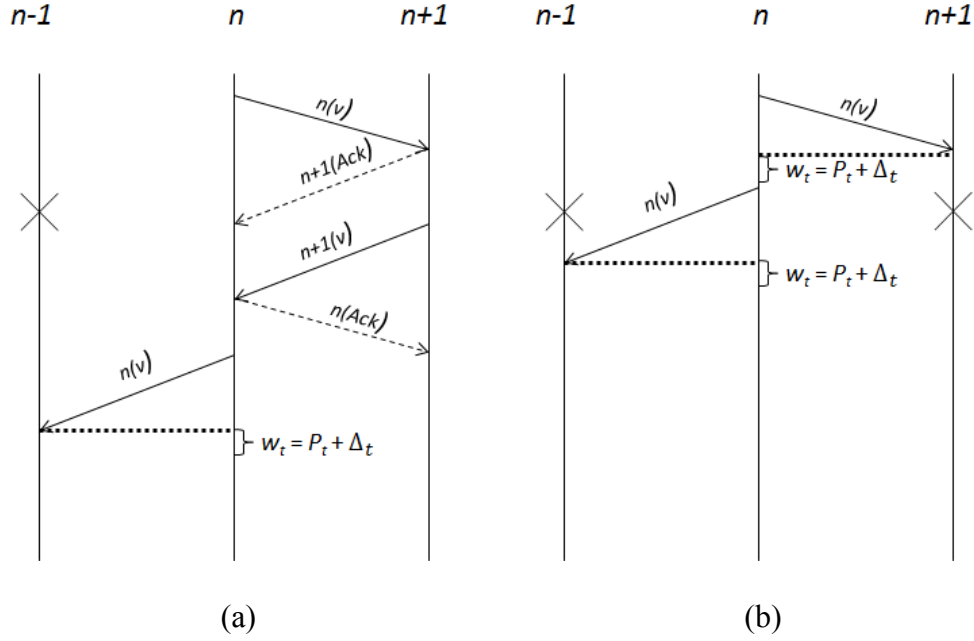


Figure 4.16: CwGN message sequence for FS-Sync exchange communicational model, (a) one adjacent node (previous) is inactive and (b) both adjacent nodes are inactive.

The minimum report time can be estimated using the second scenario. In that scenario, if all outer track nodes of node $n_{i,l}$ are inactive, the minimum reporting time of this model will be similar to the minimum report time of the FS-Async model which has been estimated according to Equation 4.25. The total communication time (T_{comm}) of the network can be calculated as the summation of T_{report} and $T_{exchange}$. Accordingly, the maximum total communication time in such a model can be estimated as follows.

$$T_{comm} = 2(N_{trk} + 4)(T_{send} + \Delta_t) \quad (4.32)$$

This is twice the maximum time of the FS-Async communication time presented in Equation 4.26. However, the minimum communication time for both models remains the same as that derived in Equation 4.27.

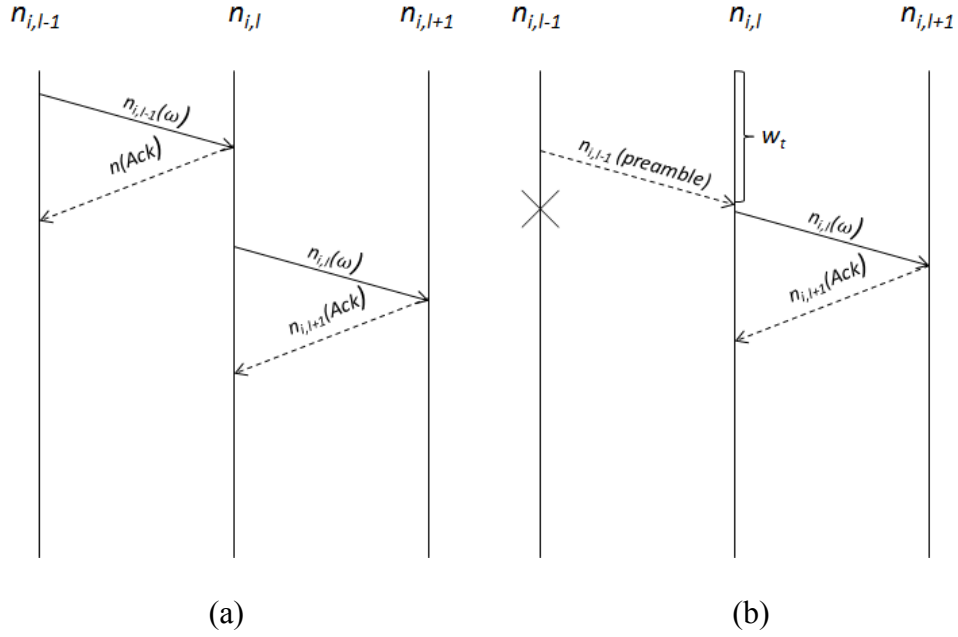


Figure 4.17: Message sequence for FS-Sync report communicational model.

4.7.3 Multi-channel CwGN models

In multi-channel (MC) models, a node is capable of handling multiple communications simultaneously. Similar to frame-slotted models, MC can work on asynchronous and synchronous modes. Figure 4.18 shows MC-Async exchange communications when all nodes are value activated. Figure 4.19

shows two possible scenarios: (a) one adjacent node is inactive, and (b) previous and next nodes are inactive.

From Figure 4.18, if all nodes are activated, all nodes will exchange frames at the same time and the time estimate can be determined as $T_{send} + \Delta_t$. However, when there is an inactive node such as in Figures 4.19 (a) and (b), the time estimate can be $T_{send} + 2\Delta_t + P_t$. This can be used as the maximum exchange time as it is higher than the first scenario time estimate. In report communications, the MC-Async model is similar to the FS-Async mode as there is only one report frame from one node to another. Hence, the communication scenarios shown in Figure 4.17 are applicable in this model. As a consequence, the report time estimates of FS-Async presented in Equations 4.26 and 4.27 represent the time estimates for this model. Thus the maximum communication time for a network running MC-Async can be estimated as follows.

$$T_{comm} = T_{send}(N_{trk} + 1) + \Delta_t(N_{trk} + 2) + P_t \quad (4.33)$$

And minimum time can be estimated as follows.

$$T_{comm} = N_{trk}(P_t + \Delta_t) + T_{send} + \Delta_t \quad (4.34)$$

In synchronous communication, each receiving node is expected to reply to the sending node with an *Ack* frame to confirm the receipt of the message. Figure 4.20 shows the exchange communication sequence in an MC-Sync model. Figure 4.21 shows two possible scenarios: (a) one adjacent node is inactive, and (b) previous and next nodes are inactive.

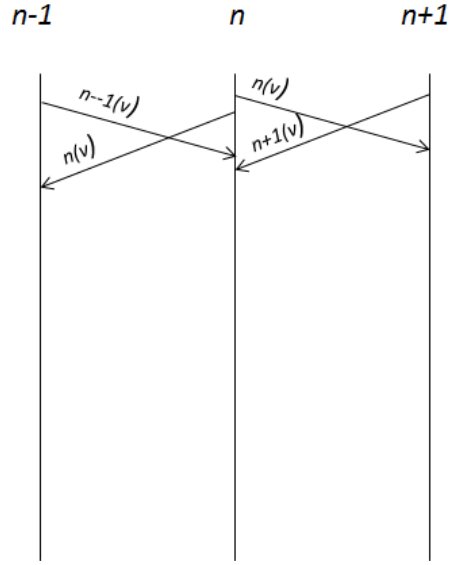
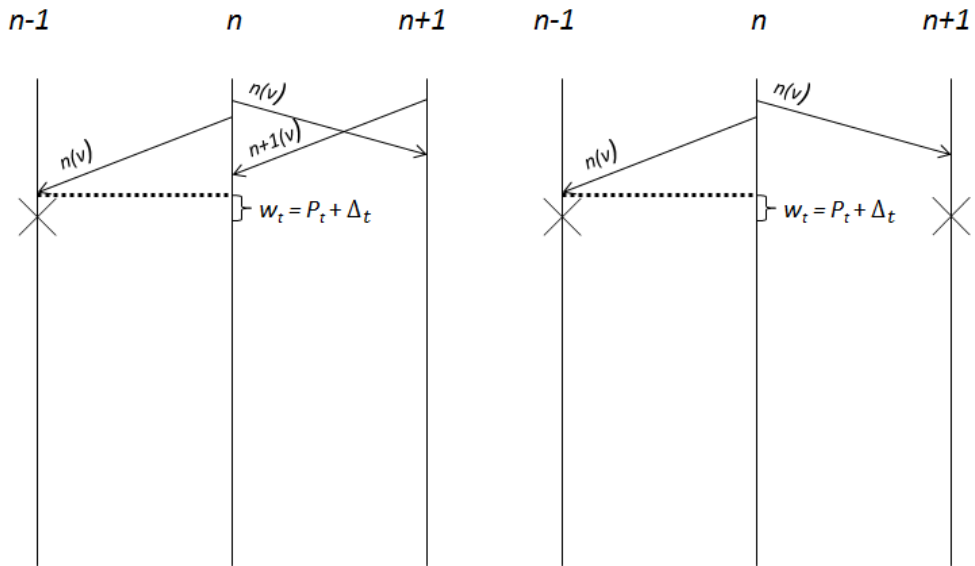


Figure 4.18: CwGN MC-Async message sequence for exchange communications when all adjacent nodes to a sending node are active.

(a)

(b)

(c)



(a)

(b)

Figure 4.19: CwGN Message sequence for MC-Async exchange communicational model. (a) One adjacent node (previous) is inactive and (b) both adjacent nodes are inactive.

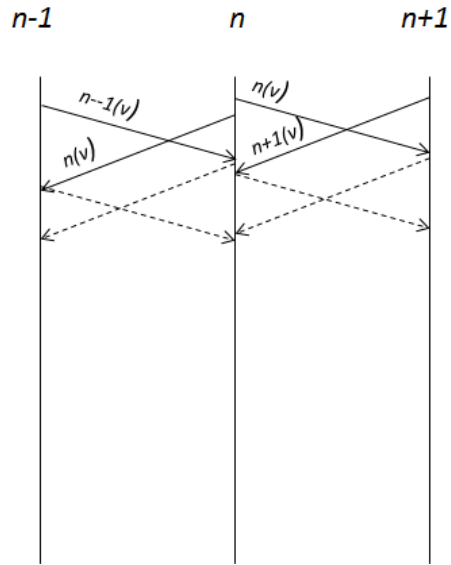


Figure 4.20: CwGN MC-Sync message sequence for exchange communications when all adjacent nodes to a sending node are active.

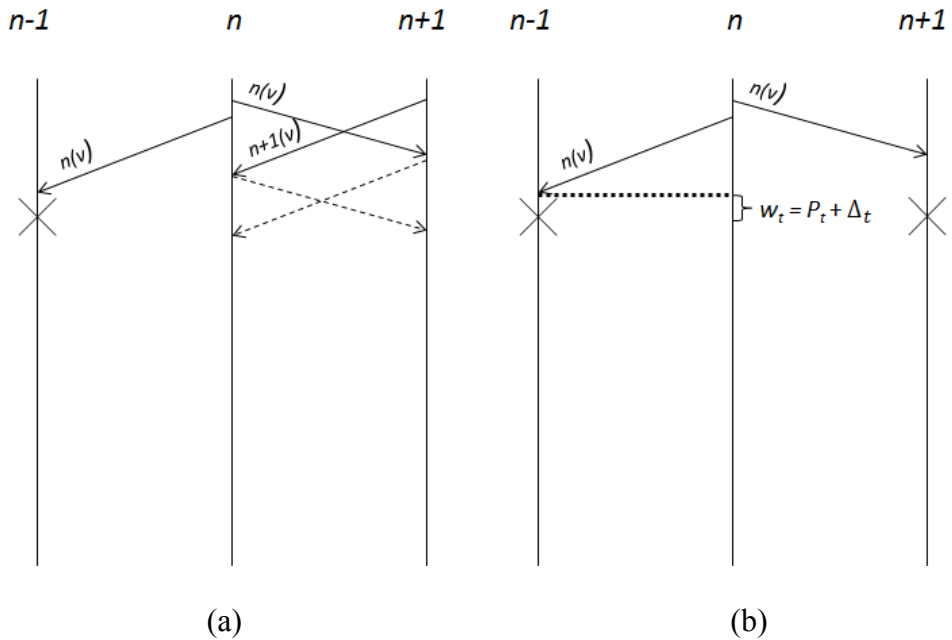


Figure 4.21: CwGN message sequence for MC-Sync exchange communicational model. (a) One adjacent node (previous) is inactive and (b) both adjacent nodes are inactive.

From Figure 4.20, exchange time can be estimated as $2(T_{send} + \Delta_t)$. This estimate includes *Ack* frames. The same estimate can be applied to the second scenario shown in Figure 4.21 (a) as the wait time for a response from node $n-1$ overlaps with exchange time with node $n+1$. In the third scenario shown in Figure 4.21 (b), time estimate will be $P_t + \Delta_t$ as node n will be waiting for a not sending preamble. A reporting sequence for this model is similar to the reporting sequence for the FS-Sync model shown in Figure 4.17 and has the same time estimates derived in Equations 4.31 and 4.32. Hence maximum network time in such a model can be estimated as follows.

$$T_{comm} = (2N_{trk} + 2)(T_{send} + \Delta_t) \quad (4.35)$$

Since the third scenarios of both MC-Sync and FS-Sync models are similar, minimum communication time can be estimated as presented, as in Equation 4.29.

Table 4.3 summarises the limits of communication time overhead estimates for each CwGN message sequence model. Experimental analysis on the four sequence models will be conducted later in Chapter 5 to evaluate the communication overhead of a CwGN network for each model using different types of patterns.

Table 4.3: Summary of communication time overhead (T_{comm}) limit estimates for each CwGN message sequence model.

Model	Minimum time estimate	Maximum time estimate
FS-Async	$(N_{trk} + 2)P_t + (N_{trk} + 4)4\Delta_t + 2T_{send}$	$(N_{trk} + 4)(T_{send} + \Delta_t)$
FS-Sync	$(N_{trk} + 2)P_t + (N_{trk} + 4)4\Delta_t + 2T_{send}$	$2(N_{trk} + 4)(T_{send} + \Delta_t)$
MC-Async	$N_{trk}(P_t + \Delta_t) + T_{send} + \Delta_t$	$T_{send}(N_{trk} + 1) + \Delta_t(N_{trk} + 2) + P_t$
MC-Sync	$(N_{trk} + 2)P_t + (N_{trk} + 4)4\Delta_t + 2T_{send}$	$(2N_{trk} + 2)(T_{send} + \Delta_t)$

4.8 Effects of Pattern Transformation on CwGN

Recognition

CwGN achieves invariant recognition by using weights rather than storing the pattern's information as is done in standard GN. The patterns' weights are stored in the S&I of a CwGN network as a vector, as described in Equation 4.1. Recognising an incoming pattern requires comparing the calculated accumulated weight with the stored patterns' weights in the S&I using Equation 4.2. This comparison can be described as follows.

$$\Delta\omega_{iC} = |\omega_i - \omega_C| \quad (4.36)$$

where $\Delta\omega_{iC}$ is the difference between the stored i^{th} pattern's weight and the network's calculated weight for an incoming pattern. Using Equation 4.7, 4.36 can be written as $\Delta\omega_{iC} = |\sum_{m=1}^S \omega_{im} - \sum_{m=1}^S \omega_{cm}|$ where i is the i^{th} pattern in the pattern vector, C is the calculated pattern, and m is the pattern element

number. The weight is calculated in accordance with the exchange of communications. Hence, each node's weight is determined as a function of its value and the values of its adjacent nodes such that $\omega_m = f(c, p, n)$, where c, p, n are the current, previous, and next nodes' values. As each node receives one pattern element, this function can be expressed as $\omega_m = f(\varepsilon_m, \varepsilon_{m-1}, \varepsilon_{m+1})$, assuming that a track's first node communicates with the track's last node. Accordingly, the weight difference can be calculated as follows.

$$\Delta\omega_{iC} = |\sum_{m=1}^S f(\varepsilon_{im}, \varepsilon_{im-1}, \varepsilon_{im+1}) - \sum_{m=1}^S f(\varepsilon_{Cm}, \varepsilon_{Cm-1}, \varepsilon_{Cm+1})| \quad (4.37)$$

To discuss the effect of pattern transformation on CwGN recognition, we will examine three types of changes below: translation, dilation and spatial rotation.

4.8.1 Pattern translation

In this research, pattern translation in 1-D space is formally defined as follows.

Definition 4.8: (Pattern translation) Given a pattern $\rho = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$ where ε_i is a pattern element in position i , S is the pattern size, and a translation value $(\alpha \in \mathbb{N}), (0 < \alpha < S)$, pattern translation involves shifting the position of each ε_i

by the value of α with the assumption that $(i + \alpha = i + \alpha - S), \forall (i + \alpha) > S, (\alpha > 0)$, and $(i + \alpha = i + \alpha + S), \forall (i + \alpha) \leq 0, (\alpha < 0)$.

Proposition 4.1: If γ_i is the translated pattern of the original pattern ρ_i by the value of α , the difference weight ($\omega_{\rho_i \gamma_i}$) will be equal to zero. The accumulated weight (ω_i) of the stored pattern ρ_i is equal to the accumulated weight of any translated pattern (γ_i).

Proof: Let $\gamma_i = \{\delta_1, \delta_2, \dots, \delta_S\}$ be the translated pattern of $\rho_i = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$ by the value of α . According to Definition 4.8, $\varepsilon_1 = \delta_{1+\alpha}, \varepsilon_2 = \delta_{2+\alpha}, \dots, \varepsilon_{S-2+\alpha} = \delta_{\alpha-2}, \varepsilon_{S-1+\alpha} = \delta_{\alpha-1}, \varepsilon_{S+\alpha} = \delta_\alpha, \varepsilon_{S+\alpha+1} = \delta_{\alpha+1} = \varepsilon_1, \varepsilon_{S+\alpha+2} = \delta_{\alpha+2} = \varepsilon_2$. This means that each element in the translated pattern γ_i maintains the same neighbouring elements that it had in the original pattern ρ_i .

As a consequence, the weight difference between any given pattern and its translated pattern will be equal to zero according to Equation 4.37. This implies that the CwGN weighting technique is translation invariant as any translated version of a pattern will result in the same cumulative weight.

Special considerations can lead to slight changes between a pattern's weight and its translation weight. The CwGN scheme assumes that the value of a node that has no reading will be equal to zero. This can happen if a node is turned off because it has run out of energy, lost communication, or for some other reason. Additionally, this can also happen in the core track in the standard communication model. In this case, the weight value of any adjacent

node to such lost nodes will change in accordance with the amount of the change to zero. In Equation 4.5, it has been assumed that the weight of the node in such a case will be the value of the node itself. Hence, the weight difference in such a special case will be equal to the value of the difference between the values of the original and translated elements in that position ($|\varepsilon_i - \varepsilon_{ai}|$). However, this amount of change will be reduced by using the normalising factor (Nf), which can be the pattern (or network) size (S). In contrast, this effect will increase if Nf is small or when there is a large number of defect (or non-deployed) nodes.

Another special case might occur when using the standard communication CwGN network. In this communicational scheme, the pattern is sub-divided into several sub-patterns and each sub-pattern's weight is calculated separately. When translating a pattern, parts of each sub-pattern could move to another part, causing changes in the neighbouring values, which may change the values of some sub-pattern weights. However, the CwGN weighting technique depends on determining the edges of patterns and sub-patterns. This means that this effect will be minimised if the edges of the translated pattern maintain the same adjacent node values as the original pattern. Additionally, division by Nf could also minimise this effect. In general, the CwGN weighting technique allows for its PR process to be invariant to pattern translation, especially when using the track linking communication scheme.

4.8.2 Pattern dilation

The dilated pattern is another form of pattern transformation that the CwGN scheme is capable of detecting. Dilation can occur in three forms: constant, average, and spatial. These types are defined as follows.

Definition 4.9: (Constant dilation) Given a pattern $\rho = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$, where ε_i is a pattern element in position i , S is the pattern size, and the value of dilation is $(\alpha \in \mathbb{R})$, in constant dilation the value of α is added to each pattern element. Hence, a dilated pattern $\gamma_\alpha = \{\varepsilon_1 + \alpha, \varepsilon_2 + \alpha, \dots, \varepsilon_S + \alpha\}$, $\forall \alpha \geq 0$.

Proposition 4.2: If γ_i is the constantly dilated pattern of the original pattern ρ_i by the value of α , then the variance value of any given node in the CwGN network for the dilated pattern is $VR_{cN\alpha} = |\frac{C_v - N_v}{N_v + \alpha}|$ and the difference value between (VR_{cN}) for the original pattern and dilated pattern for any node in the network is $\Delta VR_{cN} = |\frac{\alpha}{N_v + \alpha} \cdot V_{cN}|$.

Proof: Let $VR_{cN\alpha}$ be the variance value of any given node in the CwGN network for the constant dilated pattern γ_α of the original pattern ρ . Then, $VR_{cN\alpha}$ and ΔVR_{cN} can be calculated as follows.

$$VR_{cN\alpha} = |\frac{(C_v + \alpha) - (N_v + \alpha)}{N_v + \alpha}| \quad (4.38)$$

$$VR_{cN\alpha} = |\frac{C_v - N_v}{N_v + \alpha}| \quad (4.39)$$

$$\Delta VR_{cN} = |VR_{cN} - VR_{cN\alpha}| \quad (4.40)$$

$$\Delta VR_{cN} = \left| \frac{C_v - N_v}{N_v} - \frac{C_v - N_v}{N_v + \alpha} \right| \quad (4.41)$$

$$\Delta VR_{cN} = \left| \left(\frac{C_v}{N_v} - 1 \right) - \left(\frac{C_v + \alpha}{N_v + \alpha} - 1 \right) \right| \quad (4.42)$$

$$\Delta VR_{cN} = \left| \frac{\alpha(C_v - N_v)}{N_v(N_v + \alpha)} \right| \quad (4.43)$$

$$\Delta VR_{cN} = \left| \frac{\alpha}{(N_v + \alpha)} \cdot V_{cN} \right| \quad (4.44)$$

Note that the factor $\left| \frac{\alpha}{(N_v + \alpha)} \right|$ value ranges between 0 and 1 and gets closer to 1 as the value of α is increases. Thus, we can conclude that a node's weight can increase up to the value of V_{cN} for a constantly translated pattern. This predicts that the difference in weight $\Delta\omega_{iC}$ in Equation 4.37 for a constantly dilated pattern will range from 0 to V_{cN} .

Definition 4.8: (Average dilation) Given a pattern $\rho = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$, where ε_i is a pattern element in position i S is the pattern size and the value of dilation is $(\alpha \in \mathbb{R})$, in average dilation the value of difference average $\alpha\varepsilon_i$ is added to each pattern element. Hence, a dilated pattern $\gamma_\alpha = \{\varepsilon_1 + \alpha\varepsilon_1, \varepsilon_2 + \alpha\varepsilon_2, \dots, \varepsilon_S + \alpha\varepsilon_S\}$.

Proposition 4.3: If γ_i is the average dilated pattern of the original pattern ρ_i by the value of α , then the variance value for any given node in the CwGN network for the dilated pattern is $VR_{cN\alpha} = VR_{cN}$.

Proof: Let $VR_{cN\alpha}$ be the variance value of any given node in the CwGN network for the average dilated pattern γ_α of the original pattern ρ . Then, $VR_{cN\alpha}$ can be calculated as follows.

$$VR_{cN\alpha} = \left| \frac{(C_v + \alpha C_v) - (N_v + \alpha N_v)}{N_v + \alpha N_v} \right| \quad (4.45)$$

$$VR_{cN\alpha} = \left| \frac{C_v(1 + \alpha)}{N_v(1 + \alpha)} - \frac{N_v(1 + \alpha)}{N_v(1 + \alpha)} \right| \quad (4.46)$$

$$VR_{cN\alpha} = \left| \frac{C_v - N_v}{N_v} \right| = VR_{cN} \quad (4.47)$$

It can be concluded from Equation 4.47 that the difference in weight between a pattern and its average dilated pattern will equal zero. Consequently, the CwGN is invariant to average dilation. However, special cases may be excluded. Similar to the problem discussed in pattern translation, a change in the pattern weight might be encountered due to the assumption that a non-deployed or off node's value is zero. However, this effect is reduced by the use of N_f .

Spatial dilation can be described as the increase in the pattern's size when the pattern is modelled in 2-D space. Figure 4.22 shows a pattern and some of its possible transformations and Figure 4.22 (b) depicts the spatial dilation of a pattern. The figure shows that the dilated pattern has the same shape of the original pattern but has been spread out over the area. In this type of dilation, the main feature that a CwGN scheme is observing is the edges of the two-dimensional pattern, as it calculates weights based upon variances

between pattern elements. Consequently, the change in the number of elements that represent the edges of a pattern will have the greatest impact on changing the total weight value. The effect of such dilation can be minimised by the use of NF . If the NF is the pattern size, such an effect will be lower in large pattern sizes. This allows the CwGN scheme to be dilation invariant, especially for problems involving large pattern sizes.

4.8.3 Spatial rotation

Spatial rotation is another form of pattern change that can occur in 2-D space. In this type of change, the data location moves by an angle of θ in the field of interest. Figure 4.22 (c) depicts the spatial rotation of a pattern. Similar to pattern spatial dilation, the rotated pattern has the same original shape characteristics. However, the rotated pattern changes the location of data by an angle in the field of interest.

The main effect of spatial rotation on a pattern's weight calculation is to change the node's edge type. In such a pattern, a node's neighbouring value could change while the node's value remains the same. This will result in changing the edge type of the node, which will change the weight of the whole pattern. However, a horizontally flipped pattern will maintain similar neighbouring values.

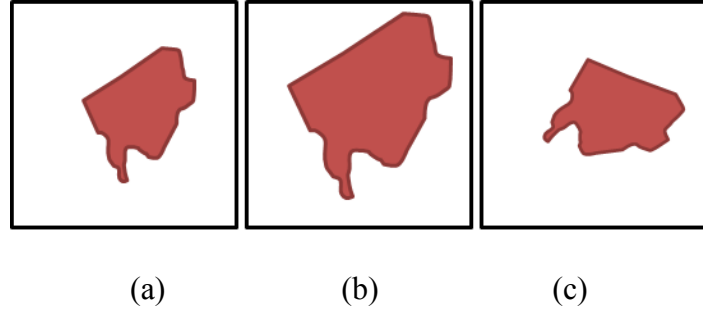


Figure 4.22: Possible types of pattern transformations. (a) Original pattern, (b) dilated pattern, and (c) rotated pattern.

Definition 4.9: (Flipped pattern) Given a pattern $\rho = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$, where ε_i is a pattern element in position i and S is the pattern size, the flipped pattern γ_θ with a value of rotation $\theta = 180^\circ$ such that $\gamma_\theta = \{\varepsilon_S, \varepsilon_{S-1}, \varepsilon_{S-2}, \dots, \varepsilon_1\}$.

Proposition 4.4: If γ_θ is the flipped pattern of the original pattern ρ , $\theta = 180^\circ$, then the weight difference $\omega_{\rho\gamma} = 0, \forall RE_v = LE_v$.

Proof: Let $\rho = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_S\}$, the flipped pattern will be $\gamma_\theta = \{\varepsilon_S, \varepsilon_{S-1}, \varepsilon_{S-2}, \dots, \varepsilon_1\}$. $\omega_{\rho\gamma}$ can be calculated according to Equation 4.24 as follows.

$$\omega_{\rho\gamma} = \left| \sum_{i=1}^S f(\varepsilon_{\rho i}, \varepsilon_{\rho i-1}, \varepsilon_{\rho i+1}) - \sum_{i=1}^S f(\varepsilon_{\gamma i}, \varepsilon_{\gamma i-1}, \varepsilon_{\gamma i+1}) \right| \quad (4.48)$$

where $\varepsilon_{\rho i}$ and $\varepsilon_{\gamma i}$ are the element values in position i in ρ and γ , respectively.

As the elements in the flipped pattern maintain the same neighbouring elements and $LE = RE$, then $f(\varepsilon_i, \varepsilon_{i-1}, \varepsilon_{i+1}) = f(\varepsilon_i, \varepsilon_{i+1}, \varepsilon_{i-1})$, and $\omega_{\rho\gamma} = 0$.

This means that the scheme is invariant to flip pattern effect.

The theoretical analysis of the effects of pattern transformations on a CwGN scheme has been conducted in this section. The analysis shows that the scheme is invariant to pattern translation, dilation and rotation translation effects. However, special cases can affect the robustness of the scheme, as discussed in this section. Experimental tests will be conducted in the next chapter to present further analysis of the scheme's transformation invariant features.

4.9 Summary

In this chapter, the CwGN scheme is presented and discussed in order address the problems of random and dynamic pattern changes. The goal of the CwGN scheme is to overcome the location sensitivity problem associated with the CGN scheme presented in Chapter 3 and provide transformation invariant recognition capabilities using limited resources to deal with network and patterns dynamics. The scheme adopts a weighting technique for pattern memorisation and recognition operations rather than storing information about the pattern itself. This eliminates the location sensitivity problem associated with a CGN scheme. The weighting technique is performed by distributing the computations amongst CwGN network nodes. Each node communicates with its adjacent nodes to perform its weight calculations and report the outcomes to another node. This allows the scheme to have the advantages of using distributed methods, which minimises the scheme's complexity and also reduces resource consumption.

Similar to the CGN scheme, CwGN involves limited communicational requirements as each node in the network communicates with its neighbouring nodes only. Additionally, these communications occur only once for each pattern. In other words, there is no need for an iterative process. In addition to the resource conservation provided by such features, it is also possible to predict the time needed to memorise or recall a pattern. Zoning structure is presented in order to provide online recognition capability to the CwGN network structure. This ability allows the scheme to be used in applications that require time deterministic operations such as real time operations.

The theoretical analysis of the effects of pattern transformations on a CwGN scheme was conducted in this chapter. Three types of transformations have been analysed: translation, dilation, and rotation. The theoretical analysis concluded that the scheme is invariant to these types of transformations. This concludes that the CwGN is a robust scheme that can detect various types of pattern transformation. Such recognition capabilities mean that the scheme requires a limited amount of training information in order to perform pattern recognition operations, which suits the nature of WSNs.

The capabilities provided by a CwGN scheme, including low complexity, limited time cycle and limited requirement of training samples make it the scheme best suited for WSNs, especially for real time and decision making applications, as discussed in Chapter 2. The next chapter will present experimental analysis and evaluation of the presented scheme in this chapter. This will include estimating the number of activated nodes, energy

consumption, and learning time required in a CwGN network. Translation recognition capabilities will be also demonstrated, using different types of patterns to evaluate the scheme's recognition accuracy. Additionally, the next chapter will compare the scheme's accuracy against existing pattern recognition schemes, using different types of problems. This involves testing the ability of the scheme to deal with real life problems.

Chapter 5

Experimental Evaluation of a CwGN Scheme

5.1 Introduction

In the previous chapter, a CwGN scheme was introduced as an efficient pattern recognition scheme for resource-constrained and large scale systems and networks such as WSNs. The scheme adopts a parallel and distributed in-network processing mechanism that is based on adjacency information exchange. Additionally, the scheme involves network node activation and de-activation processes to limit the number of participating nodes in the pattern recognition process. The in-network network structure and activation processes minimise communications and computations in the network, making the scheme light-weight and scalable so as to run on a limited resource network environment. Additionally, this allows the scheme to perform recognition operations within a single and predictable duration learning cycle, which makes the scheme suitable for online applications such as mission critical application types.

The scheme also provides pattern transformation detection capabilities by adopting a weighting technique that searches patterns' edges. The weighting

technique is location insensitive as weights are accumulated from all network nodes before a conclusion is reached about the detected pattern, which makes the scheme capable of dealing with topological changes in patterns. This gives the scheme the ability to efficiently recognise patterns while using minimal information about patterns. CwGN's scalability light-weight, single learning cycle, and efficient recognition features make the scheme the best option for large scale and limited resource networks such as WSNs.

Theoretical analysis has been conducted on the scheme to evaluate its time complexity, transformation recognition capabilities, and communicational overhead. The analysis showed that the scheme has high scalability for performing pattern transformation recognition efficiently within a single learning cycle. This chapter presents experimental analysis that evaluates the scheme's performance and resource consumption using different types of problems to confirm the theoretical findings presented in Chapter 4. The chapter evaluates the activation process of a CwGN scheme by estimating the number of participating nodes in the recognition process. The scheme's recognition accuracy will be evaluated using patterns that carry different types of transformations. This aims to perform sensitivity analysis of the scheme's recognition capabilities and determines its strengths and limitations. Energy resource consumption and the time required to perform recognition operations using a CwGN scheme will also be experimentally evaluated in this chapter. This involves evaluating these parameters using the different message sequence models presented in the previous chapter. This will also predict the

behaviour of CwGN networks running in different models. This chapter also presents an accuracy comparison between a CwGN scheme and other existing schemes using standard datasets and tools. Finally, in this chapter, the capability of the scheme to deal with real life sensory problems will be tested and compared with other schemes.

This chapter is organised as follows. Section 5.2 presents experimental accuracy analysis of a CwGN scheme. This involves introducing two types of datasets to evaluate the scheme's behaviour in the presence of different transformation levels. In section 5.3, the scheme will be compared with other recognition schemes using standard datasets. In addition, this section will examine the ability of the scheme to deal with real life problems that require sensory information. In section 5.4 the communicational overhead of a CwGN scheme in terms of energy and time will be experimentally evaluated. This will be presented in accordance with the message sequence models presented in the previous chapter. This will involve presenting a set of assumptions to find figures for the network's energy consumption behaviour, the network's lifetime, and the time required to perform one learning cycle. Section 5.5 summarises the chapter.

5.2 Accuracy Analysis of CwGN

In this section, the performance in terms of accuracy of a CwGN scheme will be evaluated using experimental tests. This includes testing the ability of the scheme to detect patterns with transformations such as translation,

rotation, and dilation. Additionally, this section will compare the recognition accuracy of a CwGN scheme with other schemes. We ran three test series to assess the accuracy of CwGN transformation recognition using different datasets. The first uniform shape patterns dataset was called *shapes dataset* and the second, using non-uniform map patterns, was called *contours dataset*. These datasets will also be used in evaluating the activation process, communicational overhead, energy consumption, and time requirements. The next section compares the scheme's accuracy with KNN, Naïve Bayes, and neural networks using standard datasets available in [112]. The datasets carry different types of transformations, as will be discussed.

5.2.1 CwGN Accuracy using uniform patterns (shapes dataset)

The first test aimed to estimate the limits of tolerance of the CwGN scheme to pattern rotation, dilation, and translation for uniform shape-like patterns. To perform this test, we constructed a dataset called *shapes dataset* that consisted of training and testing pattern datasets. The training dataset was constructed by creating and using four shapes modelled as a binary image of size 200x200 pixels, as shown in Figure 5.1. Two shapes from the training dataset were taken from the dataset presented in [129, 130]. These images were presented to the CwGN network for memorisation. To construct the testing dataset, altered versions of these images were produced for recall operations. Four sets of altered images were generated. In the first set, each image was rotated counter-clockwise from 1 to 360 degrees, with one degree for each

rotation level. In the second altered set, each image was spatially dilated by scaling the object size using 50 dilating levels. That is, the images were scaled from 1% to 100% scaling percentages in 2% steps. In the third set, each image was randomly translated 100 times by shifting the pattern's location. Figure 5.2 presents a sample of altered images using dilation, rotation, and translation effects. In the fourth set, each image was altered four times with a complex translation or a combination of rotation and translation, as shown in the example in Figure 5.3, to test the ability of the network to recognise such transformation types. Complex translation of an image involves taking parts of the image and translating these parts separately to different levels. Complex dilation involves dilating these parts to different levels as well. In this set, eight altered images were taken from the dataset presented in [129, 130]. The total number of altered images is 2052: 359 rotated images, 50 dilated images, 100 translated images, and 4 complex translated images for each original image. These images were used for testing to determine the boundaries of the CwGN's invariant recognition capabilities. These images were presented to the CwGN network to recall. We ran a simulated CwGN network of 40000 nodes assuming that the nodes are distributed as a grid and that each node detects one pixel reading. We set edge values to 0, 0.2, 0.2, and 1 for NE, RE, LE, and DE, respectively. The normalising factor is set to 40 000, which is the number of nodes. We first trained the network using the constructed training dataset. Then, we presented testing datasets for recall.

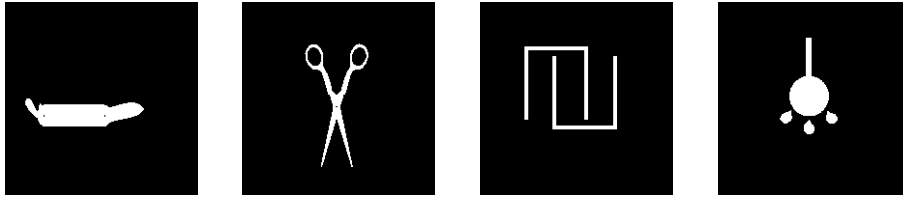


Figure 5.1: Shapes used as the training dataset for the first test series.



Figure 5.2: Sample of altered patterns used as the testing dataset for the shapes test series.

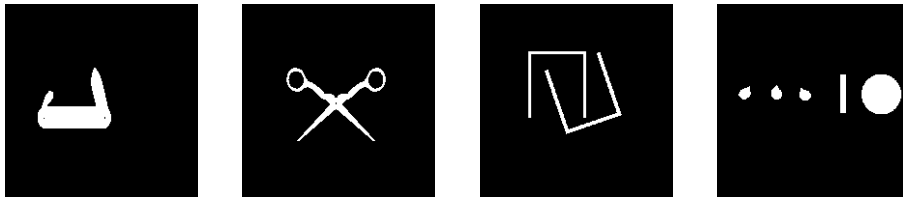


Figure 5.3: Sample of altered patterns used as the testing dataset for complex translation and combination of translation and rotation transformations.

The first set of recall images is the rotated samples. Figure 5.4 and Figure 5.5 show the accuracy of the system in recalling rotated images. The accuracy of the network in Figure 5.4 is calculated as the total number of correctly classified patterns as a percentage of the number of tested rotated images. The higher the score means the higher the accuracy. Alternatively, the accuracy shown in Figure 5.5 is calculated using the average weight difference

between a recalled (transformed) pattern's weights and the pattern's stored weight using Equation 4.36, as follows.

$$a_i = \left| \frac{\omega_c - \omega_i}{\omega_i} \right| \quad (5.1)$$

where a_i is the accuracy of pattern i , ω_c is the recall weight obtained by the network, and ω_i is the stored weight for pattern i in the network. The accuracy shown in Figure 5.5 is the average score for all recalled patterns for each rotational angle using the following equation.

$$a_j = \frac{\sum_{i=1}^{N_{pat}} a_i}{N_{pat}} \quad (5.2)$$

where j is the transformation level which is the rotation angle in this case, a_j is the accuracy of that transformation level, and N_{pat} is the number of stored patterns, which is equal to 4 in the shapes dataset. Calculating average distance to the stored weight means that lower scores indicate higher accuracy. The aim was to evaluate the scheme's behaviour in accordance with different transformation levels.

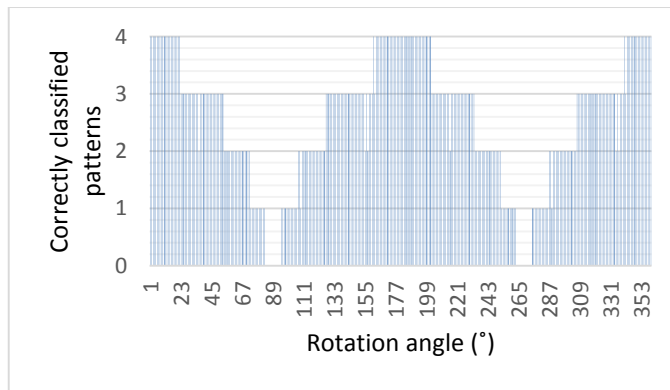


Figure 5.4: CwGN network accuracy in detecting spatially rotated patterns for the shapes dataset. Accuracy calculated as the number of correctly recalled patterns. Higher scores mean higher accuracy levels.

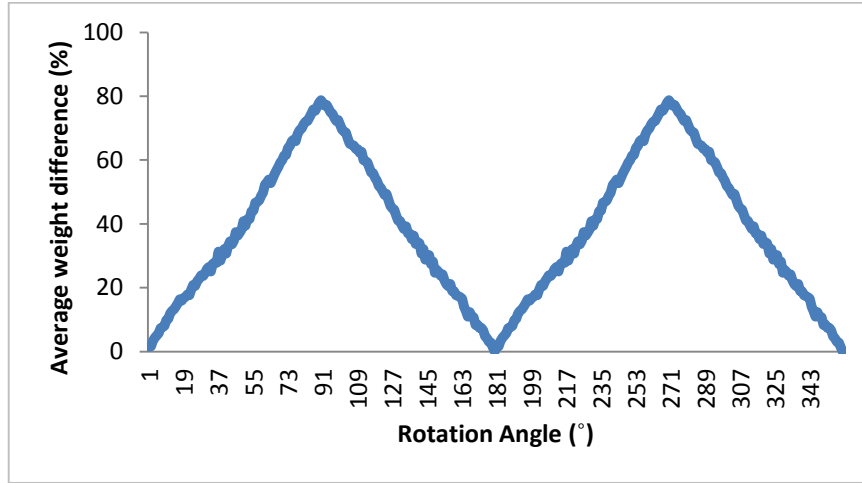


Figure 5.5: CwGN network accuracy in detecting spatially rotated patterns for the shapes dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

Figure 5.4 shows each rotation angle and the number of correctly recognised patterns at that angle. It is important to note that the total numbers of 4 samples were presented per rotational angle. The graph shows that the CwGN network is highly accurate (4 correctly classified samples) in three rotational regions. The first area is between 0 and 23 degrees, the second is between 161 and 202 degrees, the area where patterns are horizontally flipped or nearly flipped, and the third is between 341 and 360 degrees. In other words, the CwGN network is capable of efficiently detecting patterns rotated within these ranges, and could possibly detect higher rotational degrees. Figure 5.5 also shows the average distance to a stored pattern's weight according to Equations 5.1 and 5.2. From the figure it can be seen that the weight average difference increases linearly by increasing the rotation angle, reaching its highest value (78.77%) at 90°. The increase in weight difference is caused by

the change of a nodes' edge type. Vertical rotation causes some of the active nodes in the network to change their edge type from double edge (DE) to not an edge (NE) and vice versa. Since presented patterns are uniform shapes, the weight difference becomes more sensitive to this effect. After reaching the highest difference levels, the difference decreases until reaching to the value of 0% at 180° , which means that the stored weight is equal to its flipped pattern version weight. This confirms the conditions and results presented in Proposition 4.4. It can also be seen that the network presents the same behaviour for the rotation angles ranging between 180° and 360° . This means that a pattern's weight is equivalent to its flipped pattern's weight.

The second set of recall images is the translated and complex transformed samples. The scheme successfully recognised all patterns correctly. This shows how the scheme is capable of dealing with translation and combinations of transformations. Figure 5.6 shows the accuracy of the translated patterns using the average weight difference in Equations 5.1 and 5.2. The number of iterations is 100. Each iteration involves 4 randomly translated samples (one for each pattern), which gives a total of 400 samples. The figure shows the average weight difference for all patterns based on the iteration number. As can be seen from the figure, the average weight difference is very low, and in most iterations 0. This confirms the conditions and results presented in Proposition 4.1. The low difference values scored in some of the iterations resulted from the special cases discussed in section 4.8.1.

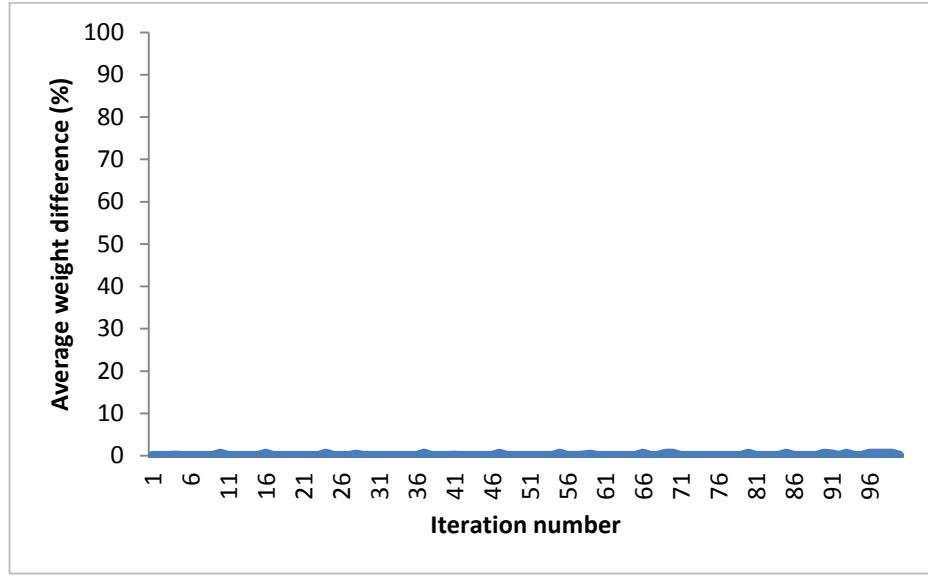


Figure 5.6: CwGN network accuracy in detecting translated patterns for the shapes dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

The third set of recall samples was the set of dilated images. Figure 5.7 shows the recall accuracy of this set (200 samples were presented for this test). The graph shows the number of correctly recognised patterns for each level of spatial dilation. Four samples were presented at each dilation level. The graph shows that the network is capable of providing perfect recognition accuracy for dilation levels up to 26%. The network is also capable of correctly classifying 3 patterns for dilation levels up to 58%. Note that increasing the level of dilation results in a decrease in recognition accuracy. This is due to the increase in the number of edges in the same area, which increases the weight value and leads to false recall. Figure 5.8 shows the average weight difference calculated according to Equations 5.1 and 5.2. The graph shows that the difference increases by increasing the dilation level, which is consistent with the

recognition results presented in Figure 5.7. This also confirms the concluded conditions and results in Proposition 4.2.

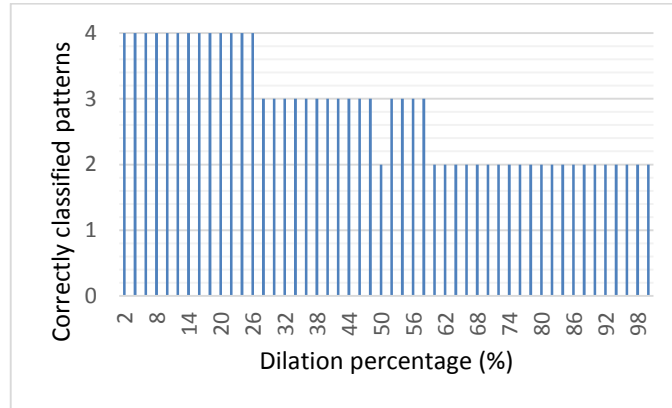


Figure 5.7: CwGN network accuracy in detecting dilated patterns for the shapes dataset. Accuracy calculated as the number of correctly recalled patterns. Higher scores mean higher accuracy levels.

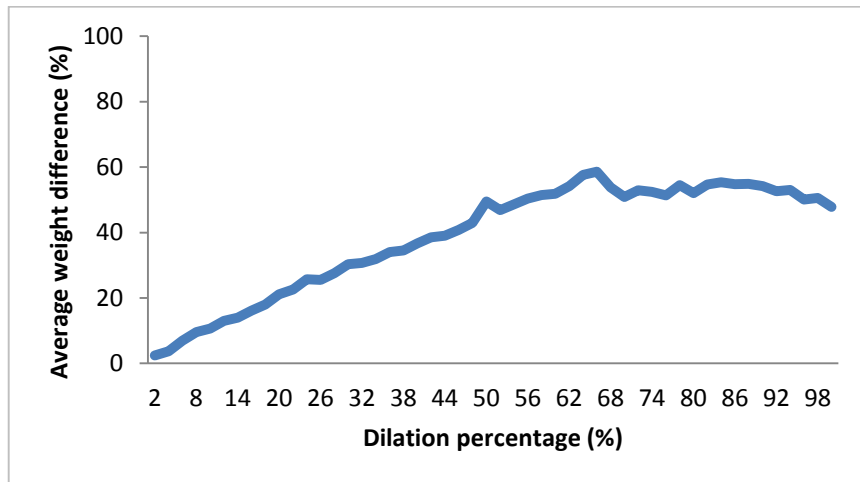


Figure 5.8: CwGN network accuracy in detecting dilated patterns for the shapes dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

5.2.2 CwGN Accuracy using non-uniform patterns (contours dataset)

The second test series aims to estimate the limits of the CwGN scheme's tolerance to pattern rotation, dilation, and translation for non-uniform patterns. For this purpose, we constructed a dataset called *contours dataset* that has training and testing datasets. We generated a training dataset of maps. We chose five raster map images from [131]. The size of each map is 100x100 pixels. These maps were transformed into contours using Matlab in order to obtain the heights of the maps. We then transformed the contour maps into binary map images to mark the highest pixels (peaks) on these maps. Figure 5.9 presents an example in which one such raster map is transformed into a contour map and then into a binary map. Figure 5.10 shows the rest of resulted maps from this operation. To create the test maps, we generated 360 rotated maps (rotating from 0 to 360 degrees), 50 dilated maps (from 2% to 200% dilation levels with 4% steps), and 100 randomly translated maps for each training map as the three recall sets. We also generated a fourth recall set in which each map was randomly altered 100 times with a combination of pattern dilation, translation, and rotation. The alterations were limited to 15% for dilation, 10x10 pixels for translation, and 10 degrees for rotation. A total of 2440 test maps were generated for recall. The accuracy of CwGN is calculated as the total number of correctly recalled patterns as a proportion of the number of altered images tested. Additionally, the accuracy of each type of

transformation was calculated as the average difference in weight values according to Equations 5.1 and 5.2.

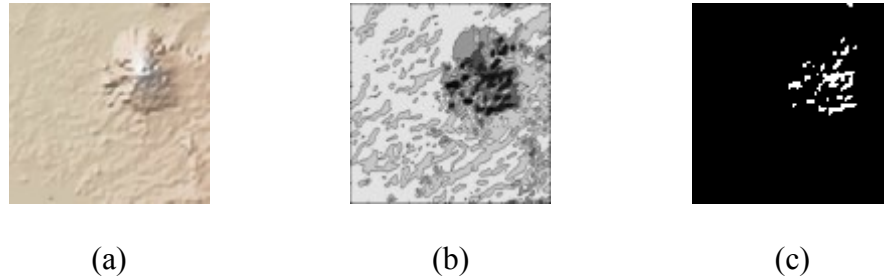


Figure 5.9: Process of producing contours training dataset. (a) A 100×100 pixel size raster map image. (b) Contour transformation of the raster map. (c) Binary transformation of the contour map. White pixels indicate the highest points in the map and are represented as ‘1’.



Figure 5.10: The rest of the non-uniform shape training patterns generated using the same steps as used in Figure 5.9.

Figure 5.11 shows each rotation angle and the number of correctly classified patterns at this angle. Five samples were tested per rotational angle. This graph shows the ability of the network to efficiently detect patterns rotated up to 12 degrees (in clockwise or counter-clockwise directions) or totally flipped horizontally within the same range of rotation. The system could possibly detect higher rotational degrees. Figure 5.12 shows the accuracy calculated in terms of weight difference. Similar to the rotation test conducted

on the shapes dataset, this figure shows that the scheme has high accuracy levels when the rotation value is close to 0° or 180° . This also confirms Proposition 4.4. However, this figure also shows that the contours dataset has lower average weight differences compared to the shapes dataset. This indicates that non-uniform patterns are less sensitive to rotation effects than uniform patterns. This also indicates that a lower number of active nodes change their edge type in non-uniform patterns.

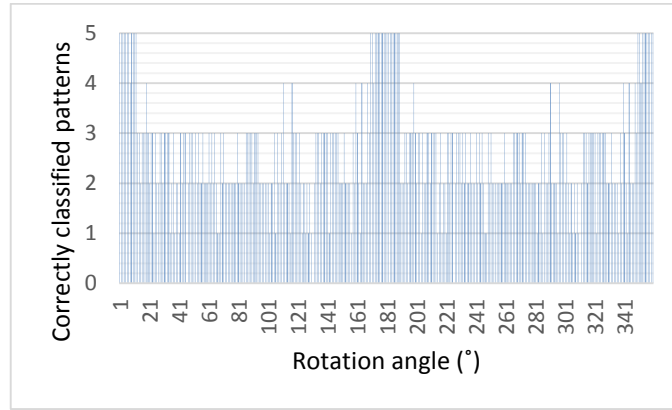


Figure 5.11: CwGN network accuracy in detecting spatially rotated patterns for the contours dataset. Accuracy calculated as the number of correctly recalled patterns. Higher scores mean higher accuracy levels.

Figure 5.13 shows the recall accuracy of the dilation recall maps in terms of correctly recognised patterns to the number of tested patterns. The graph shows that the network is capable of providing perfect recognition accuracy for dilation levels up to 24%. Additionally, the scheme can recall up to 4 maps correctly at up to 68% dilation. Figure 5.14 shows the accuracy levels in terms of weight difference average. The figure shows that accuracy

decreases with increased dilation level. These results are consistent with the findings of the theoretical analysis presented in Proposition 4.2.

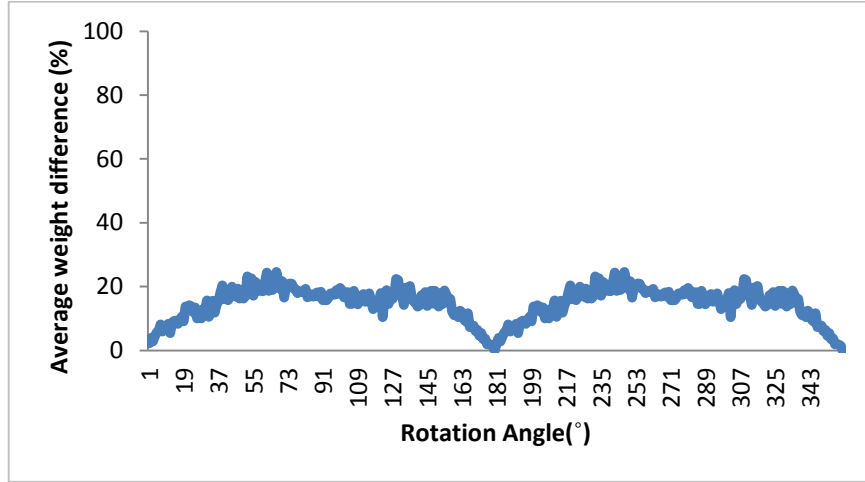


Figure 5.12: CwGN network accuracy in detecting spatially rotated patterns for the contours dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

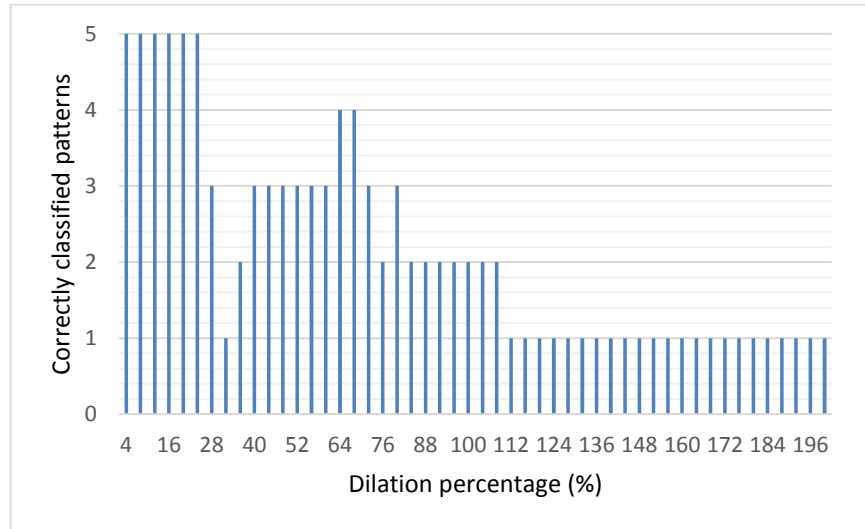


Figure 5.13: CwGN network accuracy in detecting dilated patterns for the contours dataset. Accuracy calculated as the number of correctly recalled patterns. Higher scores mean higher accuracy levels.

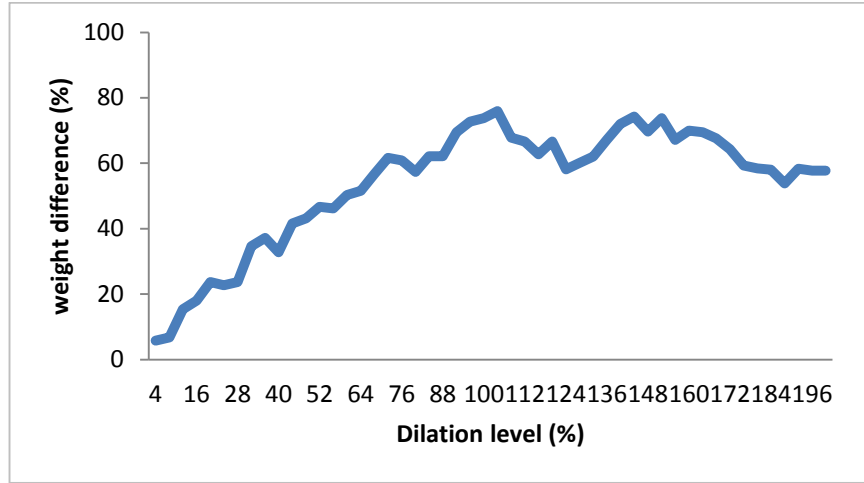


Figure 5.14: CwGN network accuracy in detecting dilated patterns for the contours dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

The scheme correctly recalled all translated maps. Figure 5.15 shows the accuracy levels for each translation iteration in terms of the average weight difference, calculated according Equations 5.1 and 5.2. Each iteration involved 5 randomly translated samples (one for each contour map). Similar to the shapes dataset translation test, the average weight difference is low and mostly equal to 0 in most iterations, meaning that the scheme is invariant to translation transformations. This also confirms Proposition 4.1. For the multiple transformation set of recall maps, which involved rotation, dilation, and translation levels, the network correctly recalled 452 out of 500 altered maps, representing a 90.4% success rate. This demonstrates the ability of the scheme to detect patterns in the presence of combinations of pattern transformations that include rotation, dilation, and translation.

In this section, the CwGN scheme's performance in terms of accuracy is experimentally evaluated. The evaluation shows that the scheme is capable of efficiently detecting translated patterns, rotated or even flipped rotated patterns of up to 23 degrees in any direction, and dilated patterns of up to 26% dilation level. In terms of translation, the tests show that the scheme is resilient to such types of pattern transformation as the scheme was capable of detecting all translated patterns. This confirms that the CwGN scheme is a transformation invariant recognition scheme. This also confirms the theoretical analysis findings presented in Chapter 4.

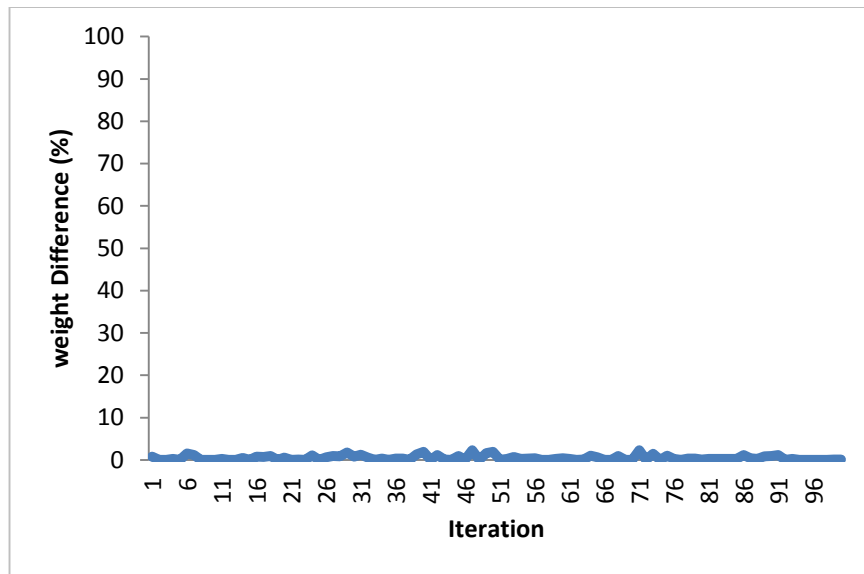


Figure 5.15: CwGN network accuracy in detecting translated patterns for the contours dataset. Accuracy calculated as the average scores of the average difference to stored weights. Lower scores mean higher accuracy levels.

5.3 Comparing CwGN Accuracy with Other Schemes

After evaluating the CwGN scheme's accuracy boundaries, this test attempts to compare the ability of CwGN to recognise transformed patterns with existing pattern recognition schemes using standard datasets. Additionally, this section will test the ability of the scheme to deal with real life sensory problems. The first test presented in this section uses a standard dataset that contains a set of pattern transformations. The use of this dataset is to compare the accuracy of a CwGN scheme with KNN, Naïve Bayes, and neural networks. The second test uses a sensory dataset that has been obtained from a real life problem. This test will compare the accuracy of a CwGN scheme with other schemes designed specifically to deal with such problems. By evaluating the accuracy of the scheme and comparing it with other existing schemes, the recognition capabilities of CwGN will be confirmed to be suitable for dealing with real life pattern recognition problems.

5.3.1 Hill-Valley problem

This sub-section aims to compare the accuracy of a CwGN scheme with iconic existing schemes using the hill-valley dataset [112]. The dataset contains 606 memorisation patterns and 606 recall patterns. Each pattern consists of 100 variables (i.e. elements). When plotting patterns in two-dimensional space, each pattern will create either a hill or a valley. Each hill or valley pattern

differs in its location and magnitude. This means that a hill or a valley is translated (shifted) and dilated in the recall dataset. Figure 5.16 and Figure 5.17 show examples of hill and valley patterns respectively. The task is to differentiate between hill and valley patterns.

A CwGN network of 100 nodes with a track-linking communication scheme was constructed to memorise and recall the patterns. Each node takes one value of the pattern. For this problem, we added a new edge type called the down edge (DW), as we are searching for peaks and troughs in the pattern field. The DW is activated when the current value of a node is less than the values of its adjacent nodes. We set the edge values to 0, 0.1, 0.1, 1, and 10 for NE, RE, LE, DE, and DW, respectively, and the N_f to 100. To compare this scheme with other schemes, the Weka tool [113, 114] was used to construct KNN ($k=1$), Naïve Bayes, and multi-layer perceptron neural networks to memorise and recall patterns. The percentage accuracy is calculated as the number of correctly classified patterns as a percentage of the total number of patterns presented to a network. Table 5.1 shows the accuracy percentage results obtained by CwGN and other schemes. The table shows that CwGN successfully differentiated between hill and valley patterns with a high accuracy compared to other schemes. However, this does not mean that the scheme is better than the other schemes, as this test only aimed to determine the accuracy of the CwGN scheme in this special case. Figure 5.18 shows the receiver operating characteristic (ROC) space and the plots for the two classes for the CwGN, KNN, Naïve Bayes, and Multi-layer NN schemes. A ROC

graph describes the performance of each network based on the false positive rate (FPR) and true positive rate (TPR). The figure shows that CwGN has higher detection accuracy and lower error rates compared to the other schemes in dealing with this problem.

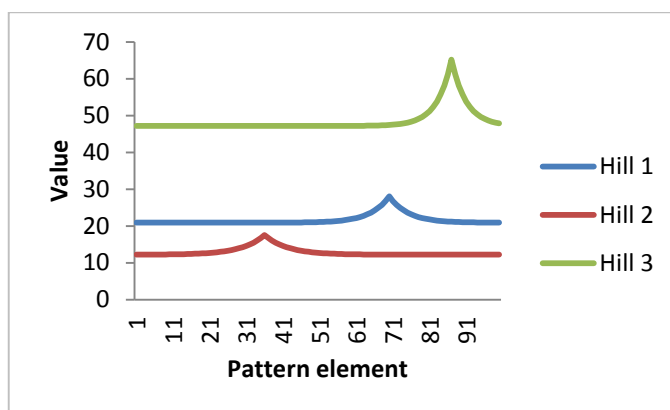


Figure 5.16: Three samples of hill pattern.

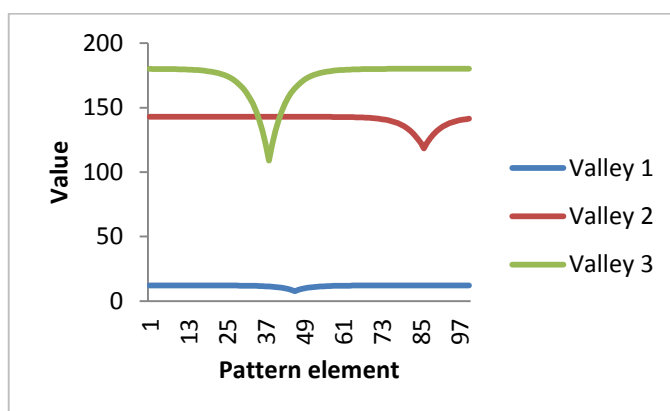


Figure 5.17: Three samples of valley pattern.

Table 5.1: Recognition accuracy results of different schemes for the hill and valley dataset

	CwGN	KNN (K=1)	Naïve Bayes	Multi-layered NN
Accuracy (%)	95.38	61.88	52.15	52.97

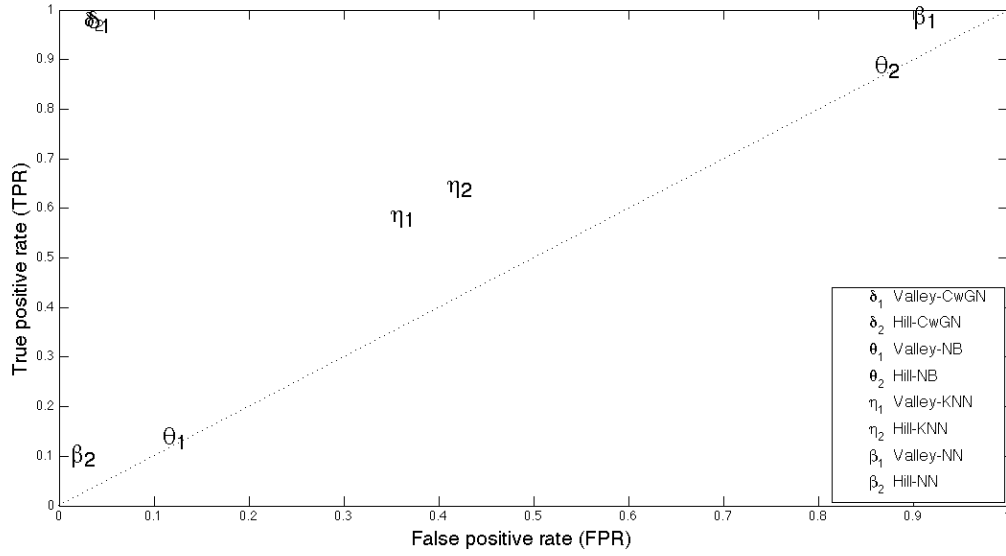


Figure 5.18: The ROC space and plots of the hill and valley classes for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes.

5.3.2 Wall following robot problem

This sub-section aims to confirm the ability of the CwGN scheme to deal with real life problems such as artificial intelligence (AI) robot navigation. Additionally, this test will be used to compare the accuracy of the scheme with existing recognition schemes and other schemes that are designed specifically to deal with such problems. In [132], the authors presented the problem of robot navigation in a closed area such as a room. The task was to allow a navigating robot to follow a wall (or obstacles) by finding a free-of-collision route. A robot mounted with 24 ultrasound sensors was used to obtain a training dataset according to Algorithm 5.1. The robot navigated through a closed room setting. In [112], the dataset obtained by applying the algorithm was presented as a training dataset named wall-following robot navigation data

set. According to the algorithm description, the dataset contains four classes: Move Forward (MF), Slight Right Turn (SRT), Sharp Right Turn (ShRT), and Slight Left Turn (SLT).

Algorithm 5.1: Collision-free rout finding for wall following robot

```
If Left Distance > 0.9
{
    If Front Distance <= 0.9
    {
        Stop and turn to the right
    } End If
else
{
    Slow down and turn to the left
} End else
} End If
else
{
    If Front Distance <= 0.9
    {
        Stop and turn to the right
    } End If
else If Left Distance < 0.55
{
    Slow down and turn to the right
} End else
else
{
    Move forward
} End else
} End else
```

For a CwGN scheme, this problem would need to take into account the problem of location-related sensing. The problem focuses on two main values: left and front sonar sensory readings. The relations between these readings decide the action that should be taken by the robot. Consequently, a CwGN has

been constructed that factorises the reading from left and front sensors. This means that front and left sensors are assigned different factor values to be multiplied by the obtained weight. This is to differentiate between left and front readings. This means the value of ω_c in Equation 4.6 is multiplied by a location factor depending on the sensor's assigned detection location (left or front).

To test the CwGN scheme, the two input datasets provided in [112] were used and two tests were conducted. The first test aimed to compare the accuracy levels that a CwGN network can achieve with KNN, Naïve Bayes, and multi-layered Perceptron neural networks. The second test compared the classification of CwGN with the schemes designed to handle this problem, as presented in [132]. To use this dataset in comparing a CwGN scheme with other classification methods, 50 instances out of each class were chosen for training and the rest (more than 5000) instances were used for testing. The results are shown in Table 5.2. Figure 5.19 shows ROC space graph for the classes: Slight-right-turn (SRT), Sharp-right-turn (ShRT), Move-forward (MF), and Slight-left-turn (SLT) used to navigate the robot.

Table 5.2: Recognition accuracy results of different schemes for the wall following robot dataset

Method	CWGN	KNN(K=1)	KNN(k=3)	Naïve Bayes	Multi-layer NN
Accuracy (%)	93.1	78.84	79.17	89.55	77.07

It can be seen from Table 5.2 that the accuracy of other techniques is higher than in the hill and valley test. This is due to the consistency of data in terms of location and magnitude in this dataset as compared to the test. It can also be seen from the table that the CwGN scheme achieved a higher classification accuracy level than the other schemes. This validates the ability of CwGN to be used as a robot navigation decision making scheme. ROC graph shown in Figure 5.19 shows that CwGN scheme has lower error rates compared to other schemes used in this test. This means that the scheme is capable of handling the problem more efficiently compared to the other schemes.

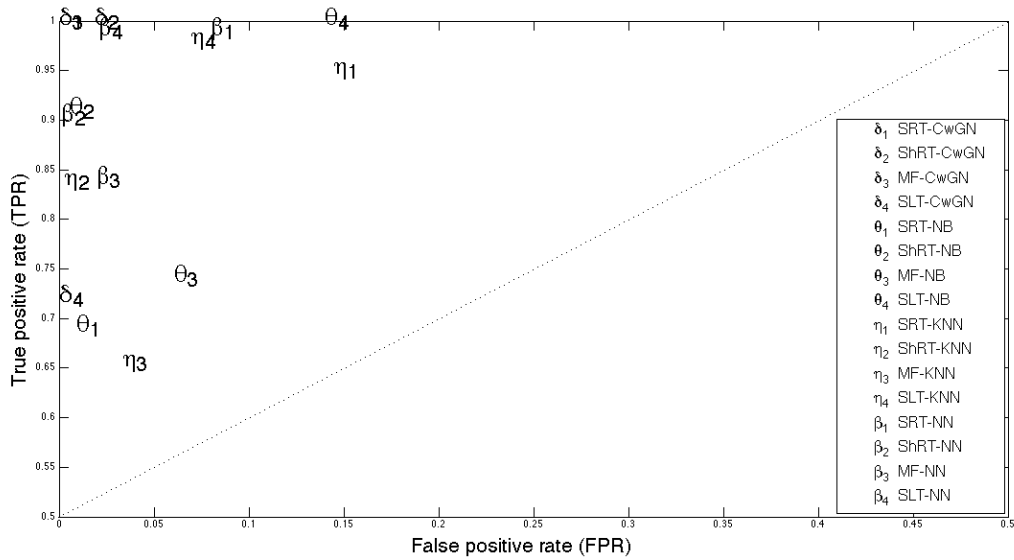


Figure 5.19: The ROC space and plots of the classes used in the wall following robot problem for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes.

In order to compare CwGN with the methods presented in [132], a robot simulation was designed. A room setting was created similar to the obstacle settings in [132], as shown in Figure 5.20. The CwGN network was trained using the provided training dataset. In this process, the robot in the simulation then senses the established environment and a decision was made based on the network's recognition. The robot moved for 2000 pixels and the resulting path is provided in Figure 5.21. The figure shows that the network successfully guided the robot to navigate in the room moving as close as possible to obstacles while maintaining a collision-free route. To compare CwGN with the presented schemes in [132], the sensed data was captured and compared with the results provided by Algorithm 5.1 to obtain the network's accuracy.

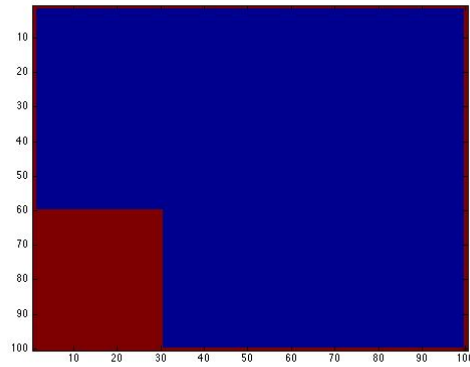


Figure 5.20: Room setting for the wall following robot navigation problem similar to the setting presented in [132]. Red pixels are walls and obstacles and blue pixels are free space.

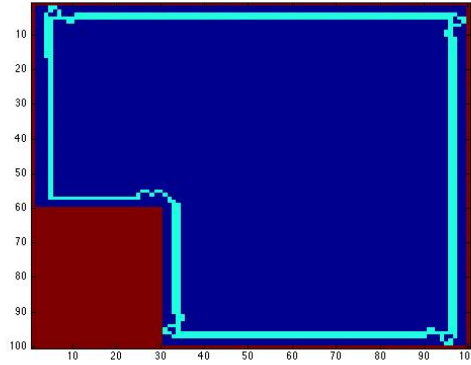


Figure 5.21: Robot route obtained by CwGN. Cyan (light line) represents the robot route.

The results of the network compared with the schemes presented in [132] are shown in Table 5.3. Accuracy is calculated as the number of correctly taken decisions to the total number of decisions. From the table, it can be seen that the CwGN network was capable of dealing with the problem of the wall following robot with higher accuracy levels compared to the schemes presented in [132]. This reflects the high accuracy levels that can be achieved by using a CwGN recognition scheme in complex sensory problems. This means that the scheme is capable of dealing with real life complex problems with high efficiency, which makes it a good candidate for use in various types of sensory problems.

Table 5.3: Recognition accuracy results of robot navigation simulation for different schemes.

Method	CWGN	Elman	ME	LP	MLP
Accuracy (%)	98.1	96.22	5.22	42.71	97.59

In this part of the experiment, the scheme was compared with other recognition schemes in terms of accuracy. The comparison included nearest neighbour, Naïve Bayes, and multi-layer neural network schemes using the hill and valley and wall following robot standard datasets. The first test showed that the CwGN scheme was capable of distinguishing between hill and valley patterns with a 95.38% accuracy level. On the other hand, nearest neighbour, Naïve Bayes, and multi-layered neural network scored 61.88%, 52.15%, and 52.97% accuracy levels, respectively. This shows the high capability of CwGN schemes to recognise transformed patterns compared to other existing iconic schemes. In the second test, CwGN was compared with the iconic recognition schemes presented in the first test using the wall following robot problem. CwGN was successful in determining the right actions to be taken by the robot with an accuracy of 93.1% compared to the other schemes. Additionally, the scheme was compared with other schemes that were designed to deal with this problem by implementing a simulation setting. The scheme was successful in guiding the robot through a closed room following obstacles and walls with a collision-free route and with an accuracy level of 98.1%, which was the highest of all the schemes. These results reflect the capability of the CwGN scheme in recognising transformed patterns and dealing with complex and real life problems with a high level of accuracy compared to other schemes.

5.4 CwGN Communicational Overhead Analysis

This section attempts to experimentally evaluate the CwGN scheme's communicational overhead in terms of time and energy requirements. As discussed in Chapter 2, network communications are considered to be the most time and energy consuming tasks in WSNs. Hence, this section experimentally estimates a CwGN network's learning time cycle duration and energy consumption based on communicational requirements. The section starts by analysing the activation process and its effect on communications in the network and then it presents a time and energy analysis.

5.4.1 Activation process analysis

The activation process of a CwGN network was presented in section 4.3.2. This process involves two types of activations, namely, node value, and node edge activations. These activations types were described in Definitions 4.5 and 4.6 respectively. The activation process determines the number of the network's communications, as only activated nodes in the network participate in the learning process. This sub-section uses the shapes and contours datasets presented in the previous section to estimate the number of activated nodes for each dataset.

Figure 5.22 shows two examples of node value activation. Figure 5.22 (a) shows the activation process of a binary pattern taken from the shapes dataset. The pattern size $S = 40000$ and the threshold φ was set to 1. The nodes

are assumed to be deployed in a grid to sense the pattern space. Since the pattern is binary, activated nodes will form the same shape that appears in the pattern. The number of activated nodes in this example is 1932 nodes out of 40000. In Figure 5.22 (b) a grey scale contour image is taken from the contours dataset as the pattern. The possible value of each element in this pattern is $V = \{0, 1, 2, \dots, 255\}$. In this example, the pattern size $S = 10000$ and the threshold was set to 200. The number of activated nodes in this example is 235. It can be seen from the examples shown in Figure 5.22 that the number of nodes that conduct exchange communications has been reduced from the pattern size to 1932 in the first example and to 235 in the second one instead of involving all of the network's nodes in the process (40000 and 10000 for the first and second examples respectively).

Figure 5.22 also shows the node edge activation of the patterns. The total number of edge activated nodes in Figure 5.22 (a) is 244 nodes, while it is 141 nodes for the example shown in Figure 5.22 (b). This limits the number of reporting nodes in the network to the edge activated nodes and relieves the rest of value activated nodes from sending report messages. This reduces the resource consumption of these nodes. Table 5.4 summarises the average number of activated nodes for both datasets.

From Table 5.4 it can be seen that the number of value activated nodes that will conduct exchange communications is small compared to the total number of nodes. The number of value activated nodes represents only 5.04% of the total number of nodes for the shapes dataset and 1.14% for the contours

dataset. Additionally, the number of edge activated nodes that will participate in report communications represents only 0.905% of the network size for the shapes dataset and 0.79% for the contours dataset. This reflects the amount of communicational overhead reduction that can be achieved by using the CwGN scheme.

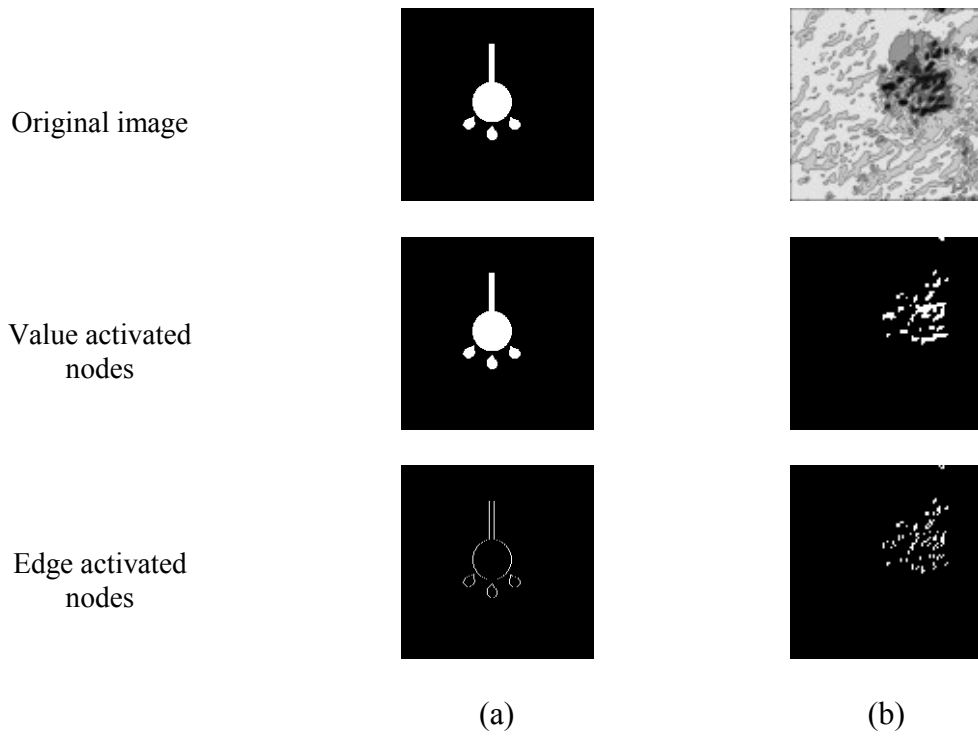


Figure 5.22: Example of CwGN nodes activation for (a) shapes dataset, and (b) contours dataset.

Table 5.4: Average number of activated nodes.

Dataset	Network size (nodes)	Average value activated nodes	Average edge activated nodes
Shapes	40,000	2016	362
Contours	10,000	114	79

5.4.2 Energy and time analysis

This sub-section aims to analyse a CwGN scheme from a practical perspective. The goal is to estimate the lifetime and execution duration of the network using the communicational models presented in section 4.7. To achieve this goal, we ran a simulation program that creates and runs a CwGN network on sensor nodes and derives energy and time readings. As noted, data transmission is the most time and energy consuming process in WSNs. Hence, the simulation evaluates these parameters based on the communications involved in running the network.

We ran the simulation on sensor nodes based on the following assumptions, as summarised in Table 5.5 [133-135]:

- i. The sensor nodes are Mica 2 type.
- ii. The frame size is 49 bytes. This includes preamble, addresses, data, control, checksum, flag, and other fields.
- iii. The preamble field is 8 bytes.
- iv. Communication data rate is 128 Kbps. This means that sending a full frame takes 3.0625 milliseconds (mS) and sending a preamble takes 0.5 ms.
- v. Transmitting one byte costs 59.2 micro joules (μJ) and receiving one byte costs 28.6 μJ . This means that sending a full frame costs 2.9 milli joules (mJ) ($49 \times 59.2 \mu\text{J}$) and receiving a full frame costs 1.4 mJ.

- vi. The maximum error in transmission is equal to one clock cycle and the clock cycle takes 32 microseconds (μs). This means that $\Delta_t = 32 \mu\text{S}$.
- vii. Each sensor is equipped with 3 volts (V) 30mAh battery. This battery is chosen as one of smallest commercial batteries. This means that a full sensor battery capacity is 324 joules.

To run the simulation, we used the shapes and contours datasets presented in section 5.2. The first dataset represents binary shape patterns of size 200×200 pixels. This *shapes dataset*, represents uniform patterns. For the second dataset, the *contours dataset*, we chose grey scale contour images of size 100×100 pixels, representing a non-uniform pattern type. We assume that the nodes are deployed in a grid, where each node obtains a pixel value. This requires 40000 nodes to represent the first network and 10000 nodes to represent the second. Four networks for each dataset were created. Each network runs in one of the models discussed in section 4.7: FS-Async, FS-Sync, MC-Async, or MC-Sync. We presented each dataset to its associated network to perform a learning process. Each dataset contains original and altered patterns (instances). The altered patterns can be a result of rotated, dilate, translated or a combination of alterations of the original pattern. The total number of instances for the first dataset is 2044 patterns and the second is 1805 patterns.

Table 5.5: Summary of assumptions used in the simulation.

Sensor type	Mica 2
Data transmission rate	128 Kbps
Frame length	49 Bytes
Time to send or receive a frame (T_{send})	3.0625 mS
Energy required to send a full frame	2.9 mJ
Energy required to send a full frame	1.4 mJ
Error rate (Δ_t)	32 μ S
Battery capacity	324 J

To evaluate CwGN communicational and energy overhead, we implemented the parallel KNN presented in [40] for comparison. We chose this scheme for its simplicity in its computations and time complexity. Figure 5.23 shows an example of the parallel KNN network. Each node in the network communicates directly with a central unit (i.e. base station). During memorisation, each node stores its associated input value. In recall, each node computes the nearest stored value to the incoming value and reports the difference to the base station, which concludes the final decision. Figure 5.24 shows the average number of communications involved in performing the learning cycle for each communicational type (frame slotted and multi-channel) for shapes and contours datasets. Additionally, the figure shows the number of communications required to implement parallel KNN for both datasets.

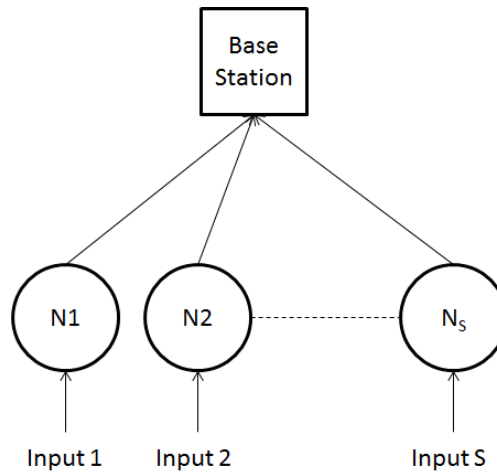


Figure 5.23. A simple parallel KNN network.

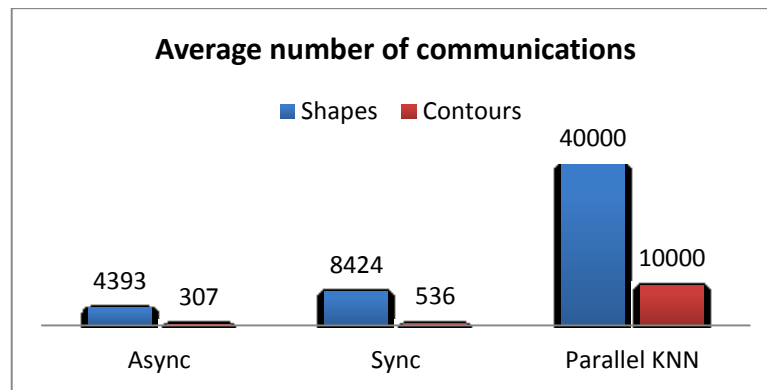


Figure 5.24. Average number of communications for CwGN Async, Sync, and parallel KNN for the shapes and contours datasets.

Since parallel KNN requires each node to conduct one communication to the base station, its communications average is equal to the pattern size (S), as shown in Figure 5.23. Figure 5.24 also shows that the average number of communications reflects the network size and the number of activated nodes. For example, the Async network involves 4393 communications for the 2016 value activated and 362 edge activated nodes for the shapes dataset. This includes both exchange and report communications. This is 10.99% of the

network size and the total communications required by parallel KNN. The Sync network involves 8424 communications for the same number of activated nodes for the shapes dataset. This is 22.06% of the network size and the total communications required by parallel KNN and almost double the average for the Async communication model. For the contours dataset, Async communication models recorded an average of 307 communications. This represents only 3.07% of the network size. The Sync model for the same dataset recorded an average of 536 communications, which is 5.36% of the network size.

The average number of communications is expected to have implications for the communicational overhead in terms of energy and time. Figure 5.25 shows the average energy consumption obtained by the simulation for each network type and each dataset. Figure 5.26 shows the average obtained from using both datasets. This is applied for both multi-channel and frame-slotted models, with the assumption that both models require the same amount of energy for each communication. The average values shown in Figures 5.25 and 5.26 represent the average energy required by a node to perform a full learning operation and include sending and receiving energy consumption.

Figure 5.25 confirms that the number of communications has implications for the average energy consumption. However, this is not applicable for the parallel KNN since each learning cycle requires all nodes to participate by sending one message to the base station. This means that the

average in this case is equal to the amount of energy required to send one message. From Figure 5.26 it can be seen that the Sync model requires almost twice the energy that the Async model does. This is caused by the *Ack* messages. However, it can be seen that it is not exactly the case as Sync average consumption is 188.08% of the Async average. This happens because of the activation process involved in a CwGN scheme, meaning that not all sent messages are received. The values shown in Figure 5.26 can be used to estimate the lifetime of the network. Figure 5.27 shows the lifetime in days for each network based on the assumption that the network receives one pattern every one minute.

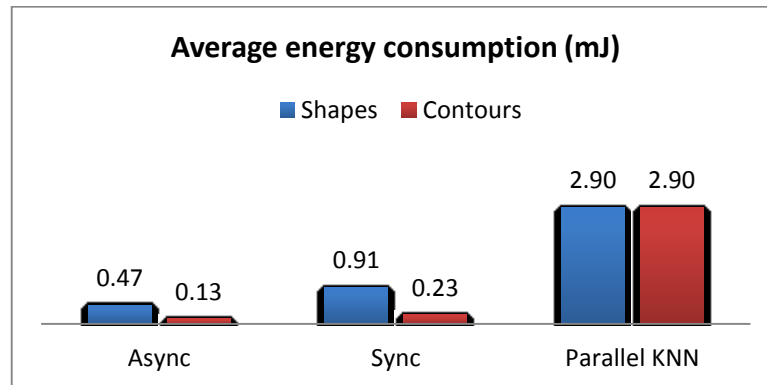


Figure 5.25: Average energy consumption for each network for each dataset in mJ. This represents the average energy required by a node to perform a full learning operation.

Figure 5.27 shows that a CwGN Async network can theoretically last for more than two years if it is used to obtain one pattern per minute using a small 3V 30mAh battery (324 Joules). However, other factors may affect that figure, such as physical sensory faults. Figure 5.27 also shows that the Sync

model could last for more than one year. In contrast, a parallel KNN will last for less than 3 months in similar conditions. According to Tanenbaum [27], a short WSN lifetime is almost 6 months. This analysis of a CwGN network shows that a large scale network can last for 1 or 2 years. This reflects the amount of communicational overhead reduction that can be achieved by using a CwGN scheme.

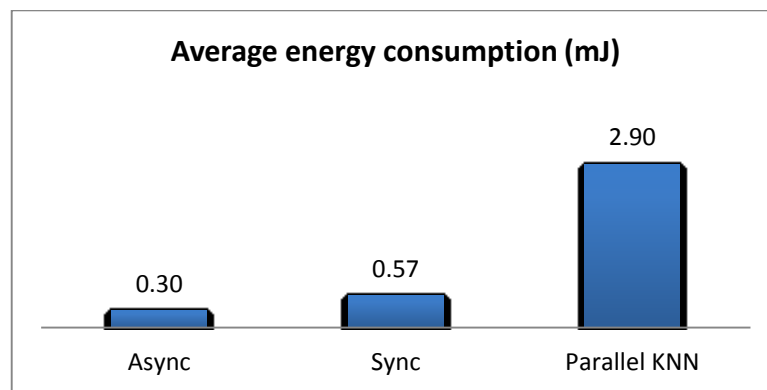


Figure 5.26: Average energy consumption for each network for both shapes and contours datasets in mJ. This represents the average energy required by a node to perform a full learning operation.

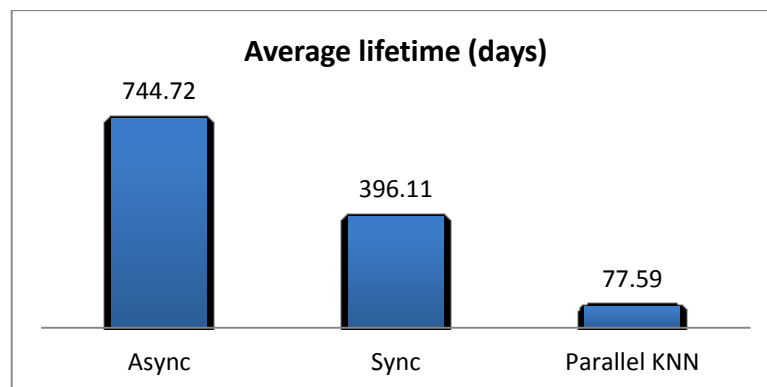


Figure 5.27: Average lifetime for CwGN and parallel KNN networks.

So far, the analysis has discussed average energy consumption. This means that all nodes are expected to encounter the same amount of communicational overhead. However, the network structure and the type of incoming patterns could affect the behaviour of the network and present different loads to its nodes. Evaluating the network's behaviour, Figure 5.28 shows the distribution of available energy in each sensor used in the simulation for Async and Sync communicational models for the shapes dataset. Figure 5.29 shows the distribution for the contours dataset.

From Figures 5.28 and 5.29, it can be noted that pattern shape can easily be seen through the energy distribution when presenting a small number of samples. However, energy distribution takes a cellular form after being presented with a large number of patterns. This cellular form reflects the cellular structure of a CwGN network. Both Async and Sync models have similar cellular distribution but with more consumption in the Sync model. It can also be seen that the energy distribution in the contours dataset is more scattered in the field, while it is concentrated in the middle for the shapes dataset. This is caused by the non-uniform pattern distribution of the contours dataset compared to the shapes dataset. Finally, both figures show that energy consumption increases going towards the core region. This is due to the network structure that involves reporting from outer tracks to inner tracks. Hence, it can be concluded that a node will have less lifetime as it gets closer to the core region.

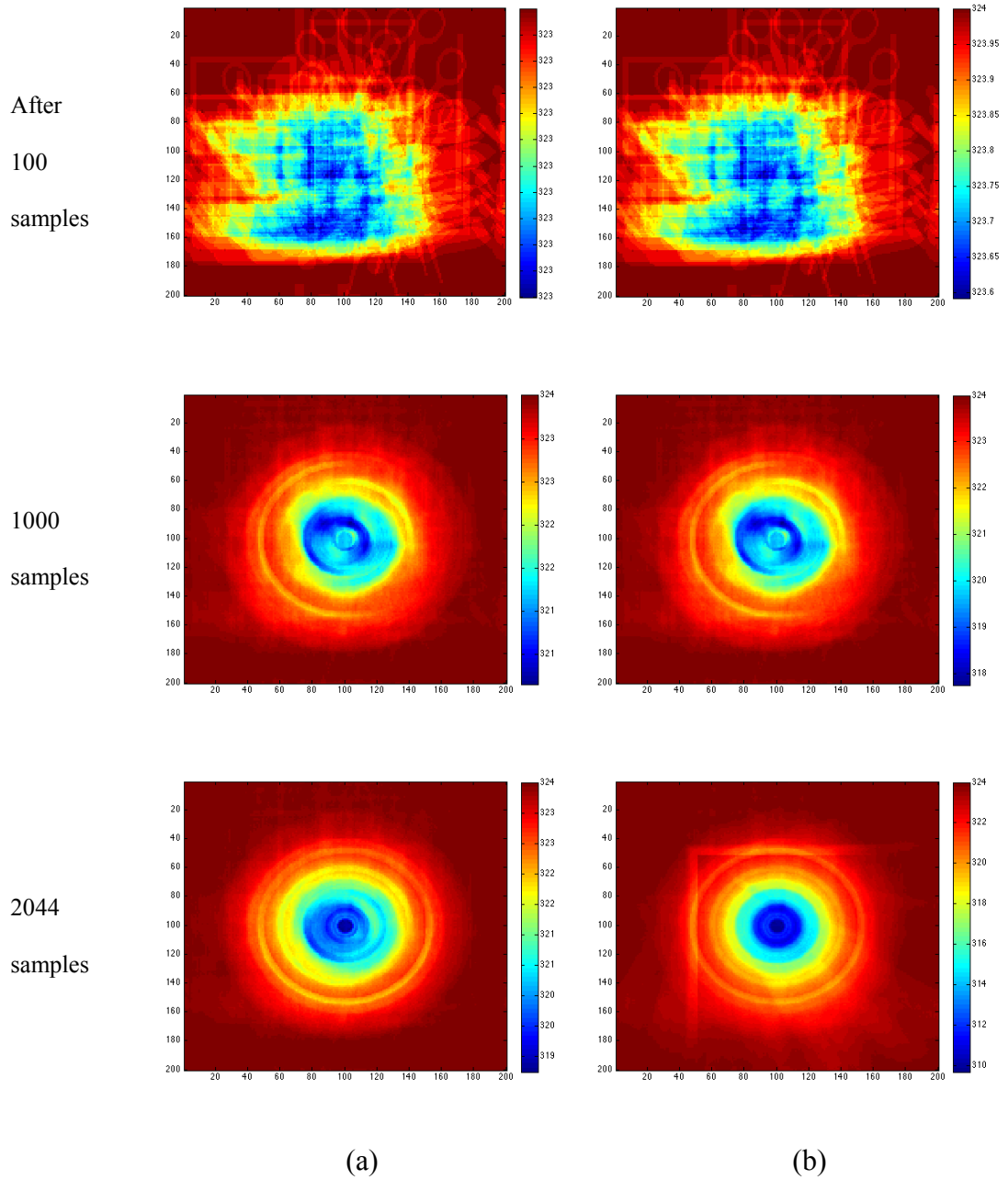


Figure 5.28: Available energy for each node in CwGN networks dealing with the shapes dataset. (a) Network applying Async model and (b) Network applying Sync model. The nodes are distributed in the field as a grid, where each pixel depicts one node's available energy. The colours are in the range between dark red and dark blue. Dark red indicates more energy resources left in the node. Dark blue indicates less energy available. Colours in between (such as yellow and green) indicate energy levels between dark red and dark blue. Colour bars show exact energy figures.

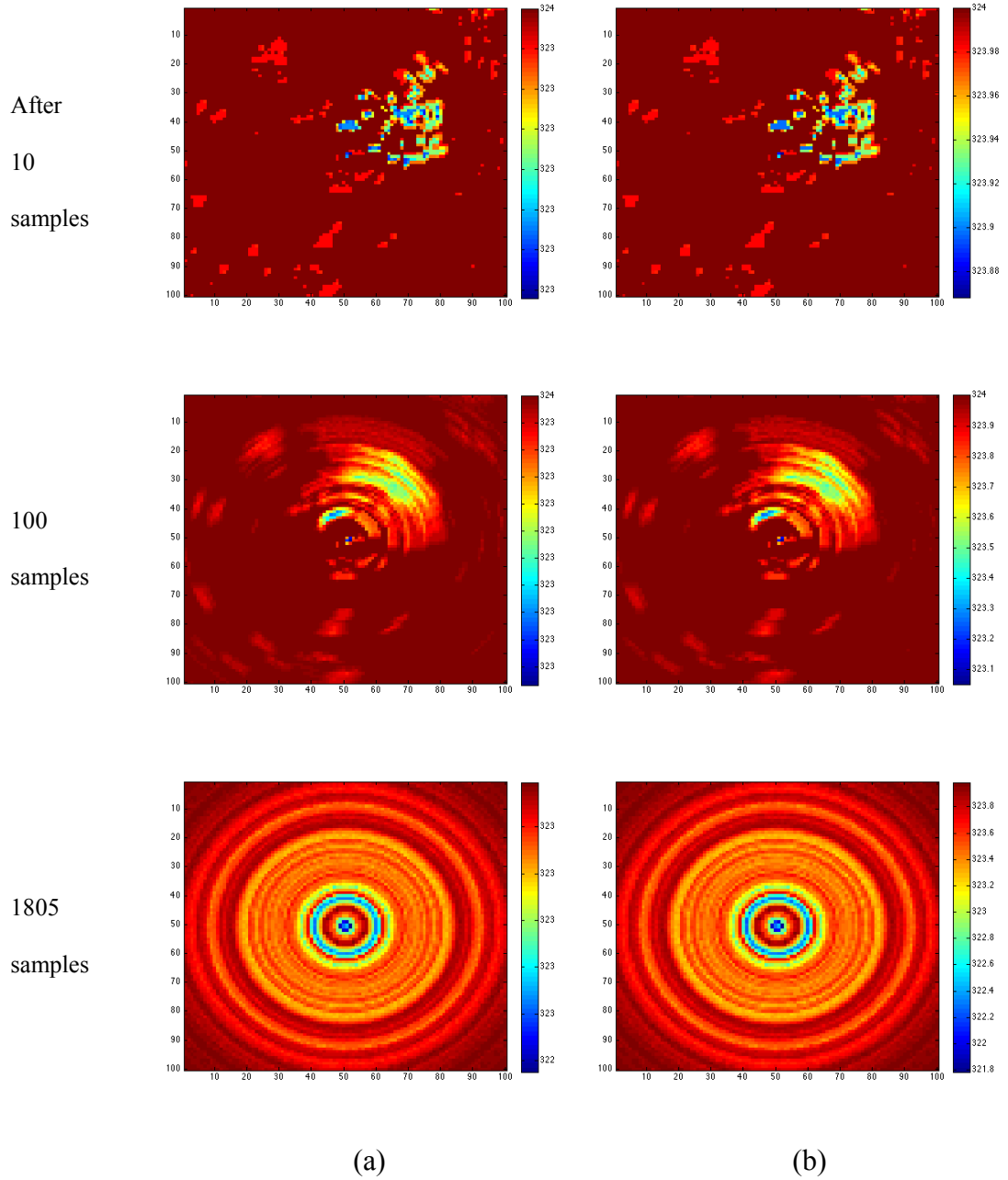


Figure 5.29: Available energy for each node in in CwGN networks dealing with the contours dataset. (a) Network applying Async model and (b) Network applying Sync model. The nodes are distributed in the field as a grid, where each pixel depicts one node's available energy. The colours are in the range between dark red and dark blue. Dark red indicates more energy resources left in the node. Dark blue indicates less energy available. Colours in between (such as yellow and green) indicate energy levels between dark red and dark blue. Colour bars show exact energy figures.

The second factor related to the communications overhead to be analysed in this section is network time. Learning cycle time (memorisation or recall operations) can be estimated in terms of computations and communications. However, it has been discussed theoretically and proven experimentally that communication time is much higher than other factors in such computations, as discussed earlier. Unlike energy analysis, the choice of communicational model (FS or MC) is expected to have an influence on cycle time. Figure 5.30 shows the average learning cycle time in milliseconds (ms) that was obtained from the simulation run.

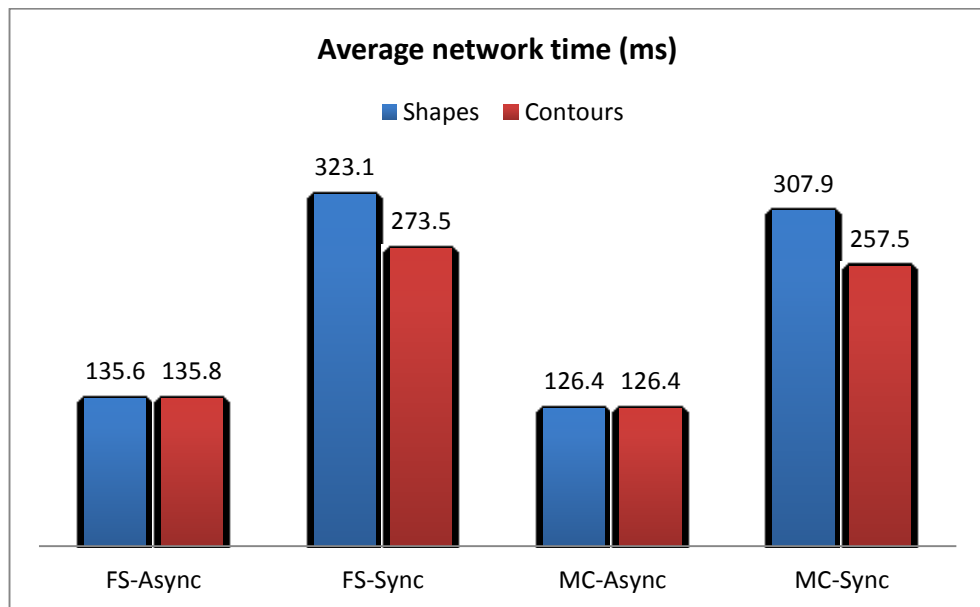


Figure 5.30: Average learning cycle time in milliseconds (ms) for a CwGN network that runs different communicational models.

From Figure 5.30, the average time ranges between 126.4 and 323.1 ms depending on the communicational model, network size, and dataset type. This means that a large scale network of a size between 10,000 and 40,000 nodes to memorise or recall a pattern with a rate of between 3 and 8 samples per second. It can be noticed from that both datasets involve an almost similar learning cycle despite the difference in network size. This is the result of two main factors. The first is that network time is mainly dependent on reporting step time, as exchange communications occurs in parallel. The second factor is the type of dataset. The shapes dataset represents uniform objects that are usually located in the centre of the field. This causes nodes to have condensed activation. In other words, more active nodes will come from the same track and hence less reporting messages will be required. Conversely, the contours dataset contains non-uniform patterns that are scattered all over the field. This will activate nodes in different tracks and will involve more reporting time.

Another important observation that can be made from Figure 5.30 is that the difference between frame-slotted and multi-channel models' time requirements is small. This is also the result of the fact that reporting time is the most time consuming part of learning cycle time. In an Async communicational model, reporting involves only one message from one node to another. In a Sync model, a sent message will be replied to with an Ack. However, the message and its acknowledgement cannot be sent simultaneously. As a consequence, in both models, the use of an MC model will not speed up the reporting time as no messages will be sent and received in

the same time. In contrast, the MC model will speed up exchange communications. However, such types of communications occur in parallel and their effect on the learning cycle is minimal.

This section presented a simulation analysis of CwGN network communications, including energy and time analysis. The energy analysis showed that a CwGN scheme is capable of limiting the number of communications and hence limiting the use of energy resources. It was shown that the network can have a lifetime of two years using one of the smallest batteries in terms of capacity (30 mAh). This is 8 times higher than other schemes, such as parallel KNN. The time analysis shows that a CwGN network can scale up, while having the ability to converge within a time range between 126.4 ms and 323.1 ms or a sample rate between 3 and 8 patterns per second. These results were obtained by implementing the scheme using different message sequence models and in accordance with the assumptions made in the beginning of the section. These results show that a CwGN scheme minimises communicational overhead, enabling WSNs to scale up efficiently and provide real-time learning capabilities.

5.5 Summary

In this chapter, the CwGN scheme has been experimentally evaluated and tested in three main areas: accuracy, comparison, and communicational overhead. Testing the scheme's accuracy involved constructing two datasets. One represented uniform patterns and the other non-uniform patterns. Three

main transformation types were tested: translation, rotation, and dilation. The tests showed that the scheme is capable of recognising patterns with the presence of these types of transformations. For both datasets, the scheme successfully recognised all translated patterns, while showing zero or very small weight differences in most cases between stored and recalled patterns. For rotation transformations, the scheme showed the ability to recognise rotated or even flipped rotated patterns of up to 23 degrees in any direction. The tests showed that any pattern has the same weight of its flipped version pattern. In terms of dilation, the tests also showed that the scheme is capable of detecting dilated patterns with up to a 26% dilation level. It was shown that recognition accuracy is proportional to the dilation level. These results confirm the theoretical analysis provided in the previous chapter.

Two tests using two standard datasets were conducted to compare a CwGN scheme with other schemes. The first test used the hill and valley dataset. The scheme showed a high level of detection accuracy compared to KNN, Naïve Bayes, and neural networks using this dataset, which confirms the ability of the scheme to recognise pattern transformations with much higher accuracy than other schemes. In the second test, the scheme was compared with a set of other existing and proposed schemes using the wall following robot dataset. This test demonstrated the recognition capabilities of the CwGN scheme in dealing with real life problems, achieving high accuracy levels compared to other schemes, even those designed specifically to deal with such problems.

The communicational overhead of the CwGN network and response time were experimentally evaluated in this chapter. To achieve this, simulation tools were constructed based on the activation process and the message sequence models presented in the previous chapter. Tests showed that the activation process of the CwGN network made it possible to minimise the number of required communications. By using the activation process, the required communications ranged between 3% and 11% of communications required by other schemes. The achievement of such minimisation is reflective of the network's energy consumption, the network having a lifetime that is eight times higher than other schemes, such as parallel KNN. The time analysis showed that a CwGN network can scale up to 40000 nodes while having the ability to converge within a time range of 126.4 ms and 323.1 ms or a sample rate between 3 and 8 patterns per second.

The experimental analysis presented in this chapter shows that a CwGN scheme can efficiently recognise transformed patterns, with the ability to scale up and minimise the network's communicational overhead as well as supporting online recognition operations. The comparison presented in this chapter shows that the scheme can provide higher accuracy levels than other schemes, even for complex and real life problems. These capabilities make the scheme the most suited for large scale, resource-constrained, and dynamic networks such as WSNs. These capabilities also make the scheme suitable for applications that require online operations and have limited prior information about problems and surrounding environments. This makes the scheme a good

candidate for use in for a variety of problems. The next chapter will discuss the possibility of using such promising capabilities in different disciplinary areas. Such a possibility would open the way for new research opportunities making it possible to involve pattern recognition in dealing with complex problems in different application domains.

Chapter 6

Using CwGN Schemes in Enhancing Optimisation and Pattern Matching Applications Performance

6.1 Introduction

In the preceding chapters, CwGN schemes were introduced and evaluated as light-weight and efficient pattern recognition schemes for resource-constrained and large scale systems and networks such as WSNs. The scheme minimises communications and computations by adopting a distributed network structure based on an adjacency computational mechanism. Additionally, the scheme involves an activation process that minimises the number of participating nodes in the recognition process in the network in order to further decrease in the computational and communicational overheads. The scheme's computational mechanism involves a weighting technique that is capable of describing patterns in terms of topological edges in order to find patterns' boundaries. This eliminates the scheme's dependency on location-based information as weights are calculated and accumulated by network nodes independently from their physical or logical location. This allows the scheme to have a transformation invariant feature that allows it to efficiently recognise

transformed patterns. These features make the scheme capable of performing online pattern recognition tasks with a high level of accuracy. Theoretical analysis and experimental results show that the scheme is capable of minimising computational and communicational overheads in the network to perform efficient recognition activities for patterns that involve transformations such as translation, dilation or rotation within a single learning cycle. Such features make the scheme best suited for large scale and limited resource networks such as WSNs.

According to Bishop and Nasrabadi [136], pattern recognition is one of the methods used for machine learning where a system uses pattern information for storing and recalling operations. However, machine learning applications could pose challenging requirements for traditional pattern recognition schemes. According to Bengio and Lecun [137], the goal of machine learning research and applications is to develop methods that are capable of learning complex problems such as behaviours and environments with minimal prior knowledge in order to support artificial intelligence and machine decision making processes. Fabisch et al. [138] highlight the most common characteristics of complex problems learning as large problem spaces, high levels of data noise and long learning time operations. Consequently, pattern recognition-based schemes should be able to deal with complex, noisy, and large scale problems if they are to serve machine learning applications and support intelligence and automated decision making processes. Achieving these goals requires pattern recognition schemes to model

problems into patterns and to use modelled problems and recognition capabilities to support decision making methods. However, the lack of prior knowledge about problems and the limited time requirements of machine learning applications are still challenging to pattern-based recognition methods.

CwGN schemes' features, discussed in Chapter 5, show that the type of scheme has promising capabilities that can be used to address machine learning application requirements. This chapter aims to use the promising features of CwGN, a pattern recognition-based scheme, in other machine learning disciplines that attempt to deal with complex problems. This can be achieved by involving the scheme in the steps of the problem solving process, where the scheme can improve the performance of the overall process. For this purpose two distinct disciplines have been chose as examples of how such promising schemes can be embedded in the process of dealing with complex problems. The first discipline example attempts to involve pattern recognition capabilities with optimisation techniques. The second example attempts to involve a CwGN scheme in dealing with complex classification problems such as human activity recognition. These examples pave the way for more innovative research opportunities in implementing such pattern recognition schemes in different application domains.

Optimisation techniques attempt to find optimal solutions for a given problem. Genetic Algorithms (GA) [139] is one example of such techniques. GA is a robust technique that attempts to find optimal solutions for unknown problems based on an evolution concept. The technique starts from total

randomness to produce a set of solutions and uses a set of operations to allow this set to evolve from one generation to another to reach the optimal solution. The survival of a solution in a generation depends on a predefined fitness function. Despite the technique's robustness and its capability in solving problems, it suffers from uncertainty in terms of convergence time. This is due to the randomness involved in generating solutions. In addition, GA suffers from the exponential relationship between the problem size and convergence time, as the search and fitness evaluating time increases by increasing the problem size [140].

In relation to the first example of using the scheme's recognition capabilities in different application domains, this chapter will propose a novel hybrid model that bolts a pattern recognition-based CwGN scheme to GA to minimise GA time complexity. The aim of such a hybrid scheme is to allow GA to learn from experience and solve problems in generating a set of proposed solutions for other similar problems. To achieve this, a CwGN network will model problems as patterns and store these patterns along with information about the GA solution of each problem. Then, when a new problem is encountered, the network will implement recognition operations on newly encountered problems to propose a set of possible solutions to the GA to start with. This means that the evolution process of the GA system will start from a knowledge-based set of solutions rather than complete randomness. The hypothesis is that the search time of a GA system will decrease if it is provided with a solution of a problem that is close to the optimal solution. In other

words, the closer the proposed solution to the optimal solution of a problem, the fewer number of steps a GA requires to converge to the optimal solution. This chapter will propose a hybrid model and test its feasibility by implementing a robot guidance problem to compare the performance in terms of speed and accuracy between the proposed model and the traditional GA model.

The second discipline that this chapter will present is the use of CwGN pattern recognition-based schemes in dealing with classification problems. In classification problems the task is to observe a set of attributes and identify to which class set these attributes belong. The classification process is based on training instances where each instance contains a set of attributes. One of the main issues related to existing classification schemes is that time complexity increases with the increase of the number of training instances. This is due to the search process, where an incoming set of attributes will be compared to the training instances. This chapter will show that a CwGN scheme is capable of translating attributes into patterns to perform classification operations. The problem of human activity recognition will be presented in this chapter as an example of how a CwGN scheme can be used to serve classification purposes. Such problems involve a high level of noisy data as human behaviour uncertainty is a key challenge in predicting activities. The scheme will be tested against one of the popular standard datasets available and will be compared with iconic schemes that are designed to deal with classification problems, such as nearest neighbour, Naïve Bayes, and neural networks, in

terms of accuracy. The comparison will be based on presenting a limited number of training samples to evaluate the accuracy of each scheme when using small quantities of training data to maintain high speed classification processes.

The chapter is organised as follows. In section 6.2, the hybrid CwGN-GA model will be presented as an example of using a CwGN scheme in enhancing the performance of optimisation algorithms. This will include an overview of GA algorithms, the approach of linking a CwGN scheme with GA, performance enhancement expectations, the implementation of the robot navigation problem, and experimental results. In section 6.3, a human activity recognition problem will be presented as an example of using a CwGN scheme for classification problems. This will include an overview of human activity problems, an overview of the testing dataset, and the implementation and experimental results. Section 6.4 concludes the chapter.

6.2 Hybrid CwGN-GA Schemes for Autonomous Robot Navigation Using WSN

This section discusses the use of CwGN schemes in enhancing the performance of GA in terms of accuracy and speed. One of the motives behind choosing such an application domain in this research is that GA systems have been used, presented and discussed in the literature to deal with sensory related problems. For example, Wu and Lui [141] propose a GA-based WSN routing

algorithm that creates optimal clusters that minimise routing energy consumption by balancing energy requirements among cluster heads. Their fitness function is based upon gathering information about WSN nodes, including distances and energy status. Martins et al. [142] propose the use of GA for dealing with WSN coverage problems that need to be addressed in the network deployment phase or for dealing with coverage problems resulting from network node failures. Such failures require dynamic connectivity mechanisms to maintain the intended coverage over an area of interest. The GA generates a connectivity model between sensor nodes based upon the energy consumption required for each node to reach its sink. Chin et al. [143] propose GA-based path planning for robotics based on obtained sensory information. The sensors collect information about obstacles in the problem space and provide the collected information to the GA in order to find an optimal collision-free path based on distance fitness functions.

As discussed earlier, GA systems offer to find optimal solutions for complex and unknown problems by implementing a set of evolutionary operations. These operations include crossover, mutation, and fitness evaluation in each produced generation of the evolutionary process. This process continues until the GA system reaches an optimal solution to the problem. Means such as limiting the number of generations may be employed to end a GA's computations [144]. The main benefits of using GA are the ability of finding solutions for complex problems and the ability of dealing with totally unknown problems [145, 146]. However, one of the main

challenges in using GA is the exponential relationship between the problem size and its time complexity [140]. Attempts to enhance a GA's performance have been studied in depth. These include parameters tuning, parallel GA, hybrid GA, and other approaches. However, predicting enhancements of GA systems in terms of time complexity is not always feasible [140, 145]. In some cases, performance enhancement of GA can be achieved by using previously solved problems. Hart [147] proves by experiment that seeding initial population to GA will lead to an optimal solution in fewer numbers of generations. This phenomenon is called evolutionary bias or GA drift. However, he also found that the total time of seeded GA, including obtaining seed solutions, is almost similar to the total time of running GA with a random initial population.

Motivated by Hart's [147] experimental findings, in this section we study the benefit of using GA-solved problems in enhancing and speeding up the process of finding solutions for other unknown problems. It is hypothesised that if a GA was given good initial solutions to start with, the scheme would find better optimal solutions for unknown problems in fewer numbers of iterations compared to a randomly initialised GA. The challenge here is how to know what good solutions can be provided to a GA system. This can be addressed by finding similarities between problems that lead to similarities between the solutions to these problems. Hence, the second hypothesis that this section discusses is that if there is a level of a similarity between two problems, then there is a similarity between the solutions to these problems.

Consequently, the proposed scheme in this section is based on these two hypotheses, where it attempts to find similarities between problems and propose solutions to the GA. By achieving this, the hybrid scheme will converge to a better optimal solution to a problem with fewer numbers of generations compared to a traditional, randomly operated GA. Hence, the task of the GA in the proposed hybrid model is to find optimal solutions to problems while the CwGN scheme stores the solutions and attempts to find similarities between solved problems and new incoming problems to suggest best stored solutions to the GA. The following sub-section describes the process of the hybrid scheme in detail.

6.2.1 Approach

The hybrid CwGN-GA model consists of two main components: a GA system and a CwGN network. The task of the network is to model problems into patterns, find similarities between patterns and suggest the best available solutions to the GA. The GA, on the other hand, performs its normal evolutionary operations to find optimal solutions. This can be achieved in two stages, namely, *memorisation* and *optimisation*. These phases are described in the following sub-sections.

Memorisation

In the memorisation stage, the aim is to store solutions for known or existing problems. To perform this stage there should be a set of training problems that the hybrid scheme can use to train the network. The GA

component finds solutions for existing problems and provides a CwGN network with its outcomes. The network stores information received by the GA to use it in the optimisation stage. The steps of the memorisation phase for storing information for a given problem can be described as follows.

- i. The GA component runs in random mode to find the optimal solution for the problem.
- ii. The GA sends the solution to the CwGN network.
- iii. The network models the problem as a pattern and calculates its weight according to the steps illustrated in section 4.3.
- iv. The network stores the pattern's weight associated with the optimal solution received by the GA in the S&I database.

Figure 6.1 shows the relationship between CwGN and GA systems for the memorisation stage.

Optimisation

In the optimisation stage, the aim is to find similarities between new problems and already stored problems. This is so as to provide GA with the best solution available in order to speed up the search process. In this stage, it is possible to provide GA with one or more solutions to include to its first generation. This means that the GA will include both suggested and randomly generated solutions in its first generation. The steps of this stage can be described as follows.

- i. The CwGN network models the problem as a pattern.

- ii. The network performs recall operations according to the process described in section 4.3.
- iii. The S&I obtains the associated solution to the recalled weight from its database and sends it to the GA.
- iv. The GA includes the provided solution to its first generation along with randomly produced solutions.
- v. The GA functions normally until finding the optimal solution.

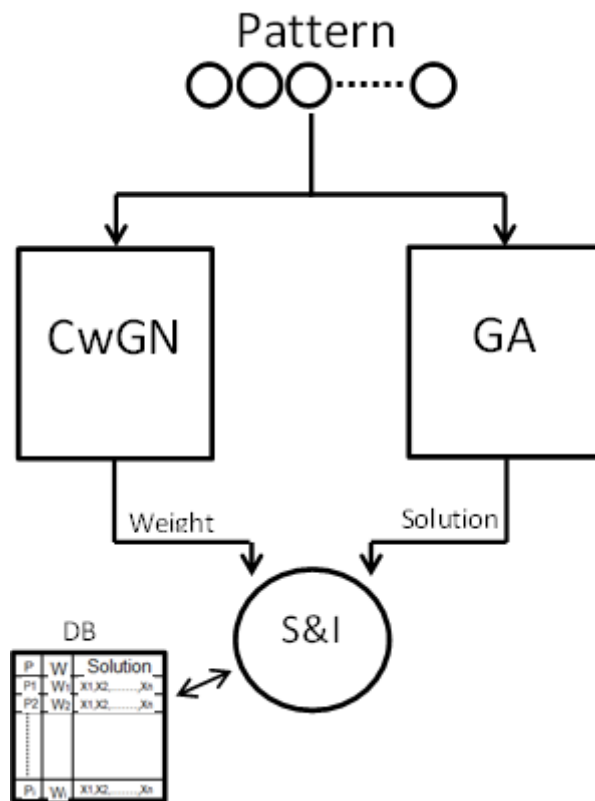


Figure 6.1: The memorisation phase of hybrid CwGN-GA scheme.

In case of multiple solutions to be provided to the GA, the S&I will provide the closest weights' associated solutions and the GA injects the received solutions into the initial population along with the randomly generated solutions. The GA continues its operations until the stop criterion is met. Figure 6.2 shows the process of the optimisation phase.

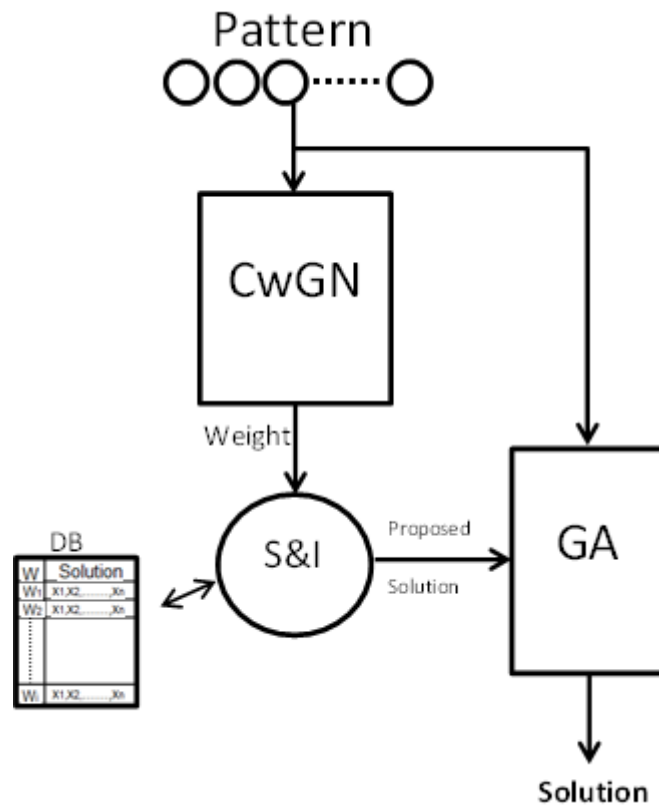


Figure 6.2: The optimisation phase of hybrid CwGN-GA scheme.

6.2.2 Performance enhancement

Several methods for estimating the time complexity of GA algorithms have been used in the literature. Drift analysis is one of the key methods that can be used in estimating the hybrid CwGN-GA scheme's computations of the

distance between the optimal solution (opt) and the set of solutions provided in each generation [148]. In this method, if $d(m_i)$ is the minimum distance between opt and set of population members, and $\Delta\varphi$ is the drift to opt , the estimated time step (τ) for a the GA system to reach the optimal solution is $\frac{d(m_i)}{\Delta\varphi}$ [140].

Definition 6.1: (Optimal distance) Let Z be the problem space for a GA problem, M_i is the set population i of the system, OP is the set of optimal solutions for the problem where $OP \in Z$, and $d(m_j, OP)$ is the distance between the member j in a given population to one of the OP members. The optimal distance $d(M_i) = \min\{d(m_j, OP): m_j \in M_i\}$.

The aim of the hybrid scheme is to minimise the optimal distance for the initial population ($d(M_0)$) in order to reduce the time step τ by injecting a set of learned solutions (M^*) into the population. If the solution is good enough, then $d(M_0) = d(M^*)$, and there is a high chance that the hybrid scheme will converge faster than a totally random GA system. The probability of a random GA system converging faster than the hybrid system occurs when the selection of M^* prevents the random selection from being one of the better solutions in M . Such probability can be estimated using the hypergeometric distribution as follows.

$$f(X = m^*) = \frac{\binom{L}{m^*} \binom{Z-L}{M-m^*}}{\binom{Z}{M}} \quad (6.1)$$

where Z is the problem size, M is the random population size, m^* is the number of injected solutions in the random population by the hybrid scheme, and L is the set of solutions that satisfy $\{d(l_j) < d(M^*) : l_j \in L\}$. The problem size can be computed in terms of pattern size (S) and possible values of each pattern's element (V) as $Z=S^V$.

6.2.3 Autonomous robot navigation using GA

As a proof of concept, we couple the CwGN scheme with a GA system, as presented in [149] and [150]. However, it is possible to couple the CwGN with any other type of GA. In [149], the GA system for autonomous robot navigation allows robots to find a collision-free path from the top left point to the right bottom point in a 2-D search space. The system assumes that robots use sensory systems to evaluate the problem space and then use GA to find the best candidate path. In order to find the best candidate path, the fitness function for the system evaluates each proposed path in terms of the path's length (Pl), number of collisions in the path (Nc), and the number of required robot turns for that path (Nt). The fitness function can be described as follows [149].

$$fp = \frac{f(Nc) \cdot [L \cdot f(Pl) + T \cdot f(Nt)] \cdot \frac{100}{L+T}}{Nc^2} \quad (6.2)$$

where L is the weighting factor of length and T is the weighting factor for turns. The equation calculates the number of turns and the path length then penalises the result by dividing the square of the number of collisions in that path. The authors' hypothesis is that these metrics can be used to find the

fastest free-of-collisions path from one point to another. The assumption here is that turning a robot is one of the most time consuming operations in robot movement. The navigation scheme's solution contains a set of points in the field representing the steps of the robot's movements. A solution includes two switching points to determine the robot's movement direction. The robot starts with horizontal movement and switches to vertical movement when it reaches the first switching point. Reaching the second switching point, the robot returns to horizontal movement again.

6.2.4 Simulation

To test this approach a simulation of autonomous path planning for mobile robots, presented in [149], was developed as the GA system. The training dataset generated in the second test in Chapter 4 is used as the original problem maps to simulate a robot that should find a collision-free-path to cross from the top left corner to the bottom right corner. It is assumed that sensor nodes are deployed in a grid over the map to sense heights. It is also assumed that each peak in a map represents a collision point that the robot should avoid. The fourth recall set in the second test series in the last chapter is used as the recall set in this simulation. In the recall set, each map is randomly altered 100 times with a combination of pattern dilation, translation, and rotation. The alterations were limited to 15% for dilation, 10x10 pixels for translation, and 10 degrees for rotation. A total of 500 test maps were generated for recall. Figure 6.3 shows one original map and a sample of altered maps of the original

one. The use of such maps ensures some level of similarity between training and testing problems.

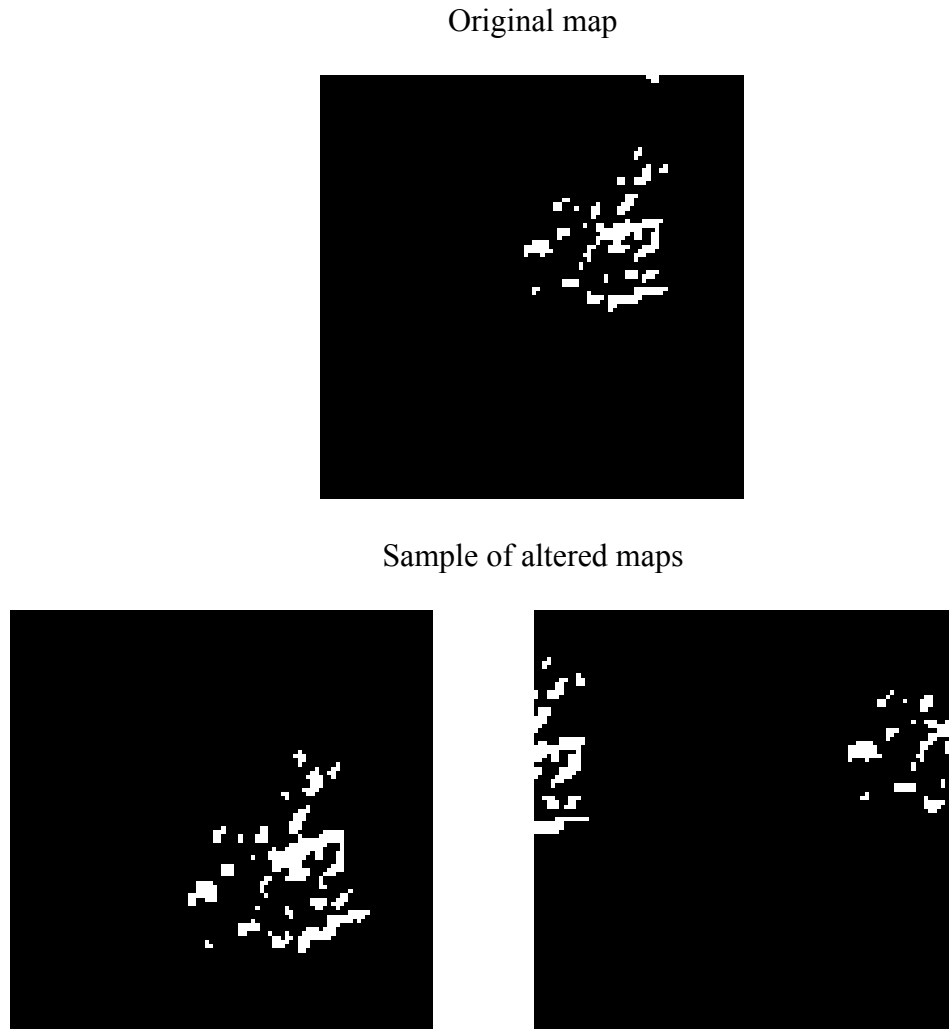


Figure 6.3: A training map from the contours dataset and a sample of altered maps used in the testing dataset.

The parameters of GA are: 0.033 mutation probability, 0.6 crossover probability, and a population size of 100. To limit GA search time, we set the maximum number of generations to 400. Mutation and crossover probability levels were chosen based on the ability of the GA to find optimal solutions for

the training binary maps within the maximum number of generations. An optimal solution in this test is defined as the solution that scores the highest fitness value in the last generation of the GA process. The fitness function in [149] takes the path length, the number of turns, and the number of collisions into account. We used the same fitness function with a solutions fitness value between 0 and 10. A solution with a fitness value of 10 is considered the best solution. For a map size of 100x100, a solution consists of 102 integers that can be used to guide a robot through the map. This includes two switching points.

To compare the hybrid CwGN-GA scheme with the autonomous GA, we ran the autonomous GA to find the optimal solutions for the five training maps used in the second test described section 5.2. The solutions were sent to the S&I for storing and association with the corresponding map. Both the coupled CwGN-GA scheme and the autonomous GA were run eight times with different maximum numbers of generations to find robot guidance solutions for the 500 test dataset. The maximum numbers of generations for each run time (for both schemes) were 50, 100, 150, 200, 250, 300, 350, and 400, respectively. This aimed to test the performance of both schemes when increasing the time limit. The schemes were run with different maximum numbers of generations to evaluate the degree of performance enhancement achieved by this approach. To ensure the accuracy of the comparison, we used the same pseudo-random number of generations for both schemes. Figure 6.4 shows the performance of the two schemes in terms of the average solution fitness values and the maximum number of generations. The figure shows that

the average fitness values resulting from the proposed coupled CwGN-GA scheme are always higher than those resulting from the autonomous GA. We also note that the average fitness value from the combined GA with a maximum of 250 generations is higher than the average value achieved by running an autonomous GA for 400 generations. This is due to the proposing of solutions provided by the CwGN network that drifts the GA search towards the proposed solution rather than starting from normal GA's complete randomness.

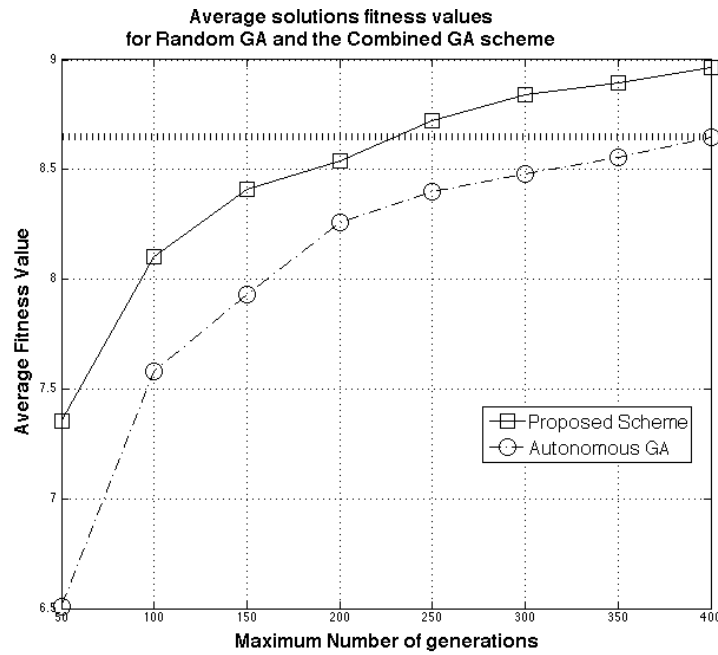


Figure 6.4: A comparison of the performance between the proposed combined CwGN-GA and autonomous GA.

Based on this experimental result, the total number of generations used to find the solutions for the 500 test maps using an autonomous GA can be calculated as 200000 generations. In contrast, the proposed scheme required

127000 generations (including the training phase) to find optimal solutions for the same set of maps. This represents a reduction of 73000 generations, or 36.5%. This shows that the use of CwGN scheme and limited number of training samples (5 samples in this experiment) allows GA to find optimal solutions with a cut off by 36.5% in time. In addition, this test shows that the proposed hybrid scheme is capable of reaching solutions that have higher fitness values than traditional GA when using the same number of generations as a stopping criterion. This means that the solutions obtained in the proposed hybrid scheme are more feasible for use in solving presented problems than solutions obtained by normal GA. In general, it can be concluded from this test that the transformation invariant recognition capability of CwGN can be used to improve the performance of AI systems such as GAs in terms of speed and accuracy.

6.3 Human Activity Recognition Using WSNs

This section discusses the feasibility of using a CwGN scheme in classification problems, the second example of how to use the features of the scheme in different application domains. In this section, a CwGN model will be presented that translates attributes into patterns and then uses these patterns to solve classification problems. It will be shown that such a model is capable of performing classification using a limited number of training instances with a high level of accuracy compared to some of the existing classification methods such as nearest neighbour, Naïve Bayes, and neural networks.

One of applications that can be used for this purpose is human activity recognition systems. Such systems analyse the physical behaviours of individuals in order to interpret the actual state, action or activity a human is performing at any given time [151]. These systems can be useful in numerous applications, such as medical monitoring, habitat monitoring, sports, security, and so forth. For such systems to be functional, two types of data collection means are generally used: camera-based means and sensor-based means [152]. Camera-based systems use visual equipment that observes an individual behaviour and attempts to use these observations for analysis. In sensor-based systems, a set of integrated sensors functions as a WSN to collect sensory information about the targeted individual. In recent research, sensor-based activity recognition systems are considered to have the most attraction compared to camera-based systems [153]. In this section, the focus is on sensor-based systems as a case study for pattern recognition using CwGN.

Different types of sensors are capable of collecting different types of measurements. For example, accelerometer sensors measure the acceleration of an object while gyroscope sensors measure the orientation [154]. These sensors can be wearable devices that are attached to an individual's body. Other types of sensors can be attached to physical objects such as drawers and doors. These sensors provide sensory information for analytical and pattern recognition systems that transform measures and readings into activities. Different sensors measure different readings that can be used for analysis and recognition. For

example, accelerometer sensors measure the acceleration rates of an object while gyroscope sensors measure orientation based on momentum [154].

The literature is rich in research that attempts to solve the problem of human activity recognition. For example, Parakka et al. [155] use sensor devices and PDAs to perform online daily life activity recognition based on a decision tree classifier. Zhu and Wihua [156] use markov models and neural networks to analyse wearable sensor readings in order to create a human-robot interaction approach for elderly and disabled people. Zhang et al. [157] present a sparse representation approach that leads to reducing human activity recognition complexity using wearable sensor devices. In general, classification methods such as SVM, neural networks, and Naïve Bayes algorithms are dominant in solving activity recognition problems in WSNs [151, 155]. However, no particular method was superior to other methods in dealing with the problem [155]. These methods could use probabilistic approaches such as Naïve Bayesian networks. However, such methods require huge amounts of data to be available and a large amount of analysis is required. Alternatively, other classification methods such as nearest neighbour can be used to create a model of relationships between collected.

The problem of human activity recognition using sensor networks encounters several challenges that limit the capabilities of such systems. These challenges can be human or technical. Human behaviour is one of the most important challenges to be addressed [158]. Several difficulties related to human behaviour emerge in conducting activities. For example, a person may

perform more than one action or activity at the same time. Here research needs to address the question of the means of distinguishing between one activity and another. Additionally, the behaviour of targeted monitored objects differ from person to person. For example, a person may perform a set of sequenced actions in order to complete an activity while another person may carry out a different sequence in performing the same activity. Other challenges are related to technical issues include those exposed in the design of sensor networks and the way these sensors are physically deployed. Activity recognition systems usually require small wearable devices that are attached to a target or mounted on surrounding objects. The main challenge in this case is to conserve battery consumption and reduce memory requirements [158].

It has been shown that a CwGN scheme reduces the communications required for performing recognition. Additionally, it has been shown that the scheme performs learning operations with no memory requirements placed on the network nodes. Consequently, CwGN is a good candidate for dealing with the problem of activity recognition as it deals with the major technical challenges that may be encountered in such applications. In this section, the CwGN will be compared with other existing techniques using limited collected data to demonstrate the capability of the scheme in addressing activity recognition problems.

6.3.1 Opportunity dataset

An opportunity dataset [159, 160] is a rich database of collected information from sensor devices that record the human activities of different subjects. Sensors are deployed on the body and on surrounding objects. The sensors deployed include inertial, accelerometer, and compass sensors. The sensors deployed on the body form a WSN and the ones deployed on objects form a wired network. The *challenge* dataset presented in [161] contains the reading measures of deployed sensors for four subjects. Our focus in this case study is on the body-worn sensors that represent the WSN part of the setting. In this case, there were 39 worn sensors that contained information about subjects' activities. These sensors were deployed in different parts of each subject's body. The types of sensors were as follows: 12 accelerometer sensors, 7 inertial sensors, and 2 compasses. Each accelerometer sensor provided 3D readings (x,y, and z). Five inertial sensors were deployed on the upper part of the body and each one contains three 3D readings, namely, acceleration, orientation, and magnetic field. The other two inertial sensors were deployed on the shoes of the body (left and right) and each one obtained five 3D readings, namely, acceleration, orientation, magnetic field, rate of turn, and angular velocity. Each of the two compasses provided a single reading that gave the direction of the object. This came to 113 attributes for each data instance.

The recorded activities were manually labelled. Two types of activities were targeted, namely, locomotion, and gestures. The locomotion activities

included four activities: standing, walking, sitting, and lying down, labelled as 101, 102, 104, and 105 respectively. Gestures included detailed activities such as opening or closing a drawer. In this case study, locomotion activities were studied as gesture activities, relying on the wired objects' mounted sensors. The case study attempted to classify the four locomotion activities based only on the measures provided by the body-worn sensors. This means that the time stamp of when an activity occurred was also neglected. This is to demonstrate the capability of the CwGN scheme in classifying activities without recording the historical information about a subject's behaviour. This is intended to minimise memory requirements so as to meet WSN resource constraints.

6.3.2 CwGN for activity recognition

An opportunity dataset is used in this case study to address the problem of activity recognition using CwGN. The first step in the case study was handling the dataset to represent valid readings to the network. The dataset provided sensory measures based on time. Each data instance represents the measures of the sensors at a given time. Some instances in the dataset contain faulty readings that are denoted as (NaN). In this experiment the instances containing more than three invalid readings were eliminated.

The second step was to address the relationship between sensors and S&I in the CwGN network. Since worn sensors give readings in 3D format, each sensor exchanges information with its adjacent sensors. The network is made up of 37 nodes that provide the 3D readings. The two compass readings

(1D) send information directly to the base station without exchanging. Each sensor in the network calculates three weights (x, y, and z) in accordance with Equations 4.3, 4.4, and 4.5. Each value in a certain dimension is considered to be adjacent to the neighbour's value of the same dimension. For example, the value of x in the first sensor is adjacent to the value of x in the second sensor. Figure 6.5 shows the connectivity relationship between adjacent sensors (nodes) in the same track of a CwGN network. Each weight is calculated as follows.

$$\omega_{cd} = ED_{vd} \cdot VR_{cnd} + ED_{vd} \cdot VR_{cpd} \quad (6.3)$$

where d is the dimension (x, y or z). The total current node's weight can be calculated as the summation of the three weights as follows.

$$\omega_c = \omega_{cx} + \omega_{cy} + \omega_{cz} \quad (6.4)$$

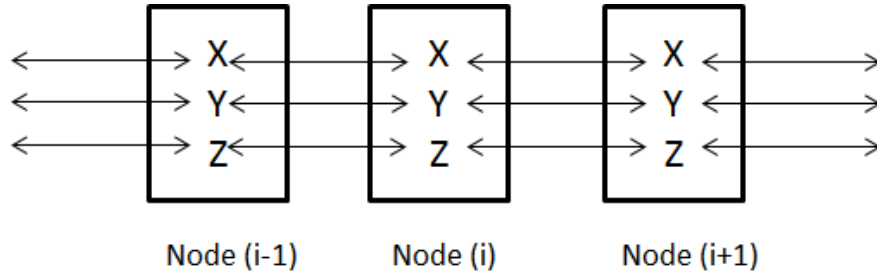


Figure 6.5: Connectivity between neighbouring nodes in a 3-D CwGN track.

The S&I (in the base station) receives the accumulated weight from the network along with the compass sensors' readings. According to Equation 4.6, the S&I normalises the accumulative received weight by the normalising factor

(Nf). In this experiment the Nf is a function of the number of participating sensors and the value of compass sensors as follows.

$$\omega = \frac{\sum_{i=1}^S \omega_i}{C1 * C2 * S} \quad (6.4)$$

where ω is the total weight, ω_i is the i^{th} sensor's weight, S is the network size, $C1$ is compass one sensor's value, and $C2$ is compass two sensor's value. The network was trained using 20 randomly selected data instances for each locomotion class. To compare with other schemes, the Weka [113, 114] tool was used to simulate three different schemes: KNN ($k=1$), mlti-layered NN, and Naïve Bayes. Figure 6.6 shows the receiver operating characteristic (ROC) space and the plots for the four activity classes for the CwGN, KNN, Naïve Bayes, and Multi-layer NN schemes. A ROC graph describes the performance of each network based on the false positive rate (FPR) and true positive rate (TPR). Table 6.1 shows the details of CwGN and other schemes' recognition accuracy levels obtained for each class. Accuracy in the table is calculated as the total number of correctly classified instances (patterns) to the total number of instances in that class. Overall accuracy is calculated as the total number of correctly classified instances compared to the total number of testing instances. The same training and testing datasets were used for all schemes.

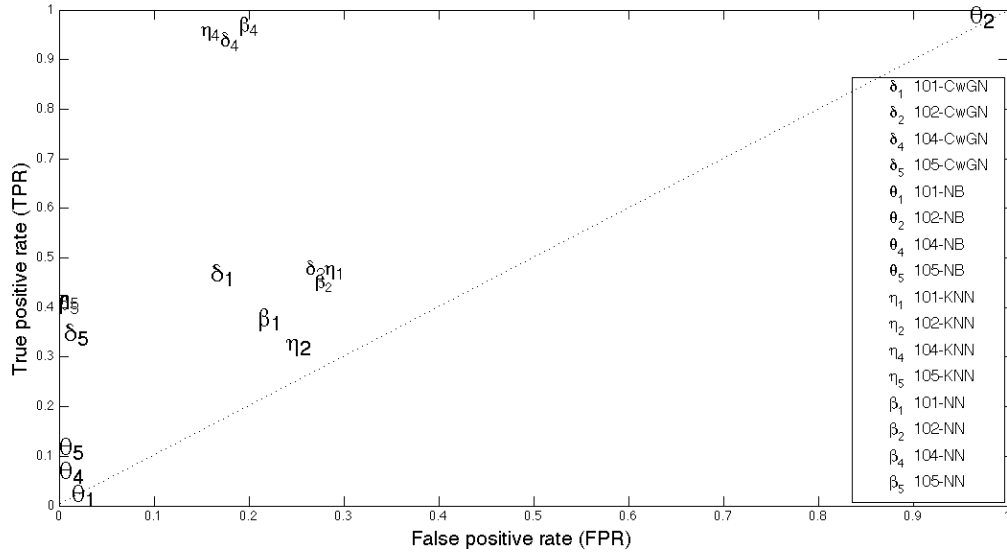


Figure 6.6: The ROC space and plots of the activity classes for CwGN, KNN, Naïve Bayes (NB), and Multi-layer NN schemes.

Table 6.1: Recognition accuracy results of opportunity challenge dataset for different schemes.

					Overall
Class	101	102	104	105	N/A
Training instances	20	20	20	20	80
Testing instances	154143	79589	80058	17557	331347
CwGN	45.86%	47.34%	92.48%	33.88%	56.96%
KNN (K=1)	47.24%	32.92%	95.21%	41.04%	55.06%
Naïve Bayes	1.42%	97.54%	6.24%	11.50%	26.21%
Multy-layer NN	37.26%	44.79%	96.26%	40.75%	53.50%

From ROC graph shown in Figure 6.6 and Table 6.1, it can be seen that Naïve Bayes schemes recorded the lowest overall average recognition accuracy compared to other schemes. More specifically, the Naïve Bayesian network classified most incoming instances as pattern 102. This was due to the small

number of training samples. As discussed in Chapter 2, Naïve Bayes attempt to create a probabilistic relationship between training samples and input variables. Since the number of training samples is limited, these probabilistic relationships cannot efficiently describe each pattern.

The other schemes produced comparable detection accuracy. However, both multi-layered NN and KNN schemes have their own requirements to reach such accuracy levels. Multi-layered NN involved input layers, one hidden layer that contained 71 nodes, and a four nodes output layer. The network structure required each node in each layer to communicate with each node in the higher layer. This means that each node in the input and the output layers had 71 connections to the hidden layer. Since the input layer contained 113 nodes (the pattern size) and the output layer contained 4 nodes, each node in the hidden layer had 117 connections to the input and output layers' nodes. Similarly, each node in the output layer required four connections to the hidden layers' nodes. The total number of connections in this case was 16950. Additionally, the network required 500 iterations for each incoming pattern in order to converge. In contrast, a CwGN node involves only three connections to its adjacent nodes. That includes exchange and report communications. Since the compass sensors report directly to the S&I, each one of these nodes requires only one communication to the base station. Consequently, the total number of communications was 335. Additionally, the CwGN network involves a single cycle to converge. Such reduction of communications and

iterations would have ramifications for the performance of the network in terms of resource consumption (e.g. energy) and convergence time.

The KNN (with $K=1$) scheme involves fewer communications and iterations compared to the multi-layered NN. Each node in a KNN in such network saves the input values of the training data instances and then compares incoming pattern values with the stored information. After calculating distances, the nodes report directly to the base station, which also holds information about training samples. This requires the memory resources available in each node to store such information. By increasing the number of training samples, the memory requirements and the time required to calculate distances for each node increase. Moreover, the direct communication to the base station affects the efficiency of the network and may cause a single point of failure threat. In contrast, the nodes in the CwGN network avoid storing information about training samples. Alternatively, each node reports the calculated weight to the S&I through one of its inner nodes in the network. This alleviates the need for memory resources in each node. Consequently, no search time is required by nodes to match patterns. Additionally, the S&I (i.e. base station) receives the accumulative weight from only one node in the network, which is the core node or its alternative node in case of failure. This assists in avoiding any single point of failure problem.

It can be seen that the CwGN scheme is capable of offering comparable and even higher recognition accuracy levels for the challenge dataset with minimal requirements in terms of the number of training samples, number of

communications and convergence time, compared to other iconic schemes. The problem of human activity recognition could involve using more complicated means such as filters and physical analysis, which could improve accuracy. This case study shows that the CwGN scheme is a good candidate to be integrated with such means in order to solve activity recognition problems in resource-constrained environments as it scores comparable accuracy levels thus minimising resources consumption.

6.4 Summary

In this chapter, the CwGN scheme was implemented in two different application domains. The first one looked at the capability of the scheme in enhancing the performance and speed of GA for optimisation purposes. More specifically, the scheme was combined with GA to solve the problem of autonomous robot navigation based on limited numbers of available samples. The experimental results show that the scheme was capable of enhancing the performance of GA in terms of fitness value and convergence time. The coupled CwGN-GA was capable of providing higher fitness values in the same number of generations for unknown problems compared to a random GA system. Additionally, the experimental results show that the coupled scheme speeds up finding an optimal solution by 36.5% for the same unknown problems compared to random GA.

The application domain implementation pointed to the ability of the scheme to use its pattern recognition features for solving classification

problems. The problem of human activity recognition using the challenge opportunity dataset was presented. The experimental results show that the CwGN scheme is capable of providing higher accuracy levels in detecting and classifying human activities such as walking, standing, sitting and lying down with minimal use of resources and with the presence of a high level of noise. The CwGN was compared with KNN, Naïve Bayes and multi-layered NN. It was shown that the CwGN is capable of achieving higher accuracy levels using a very limited number of training samples (i.e. 80 samples). Additionally, it was shown that the CwGN network involves considerably fewer communications and a high level of distribution in addressing the problem compared to the other schemes. Moreover, the scheme was able to successfully achieve these results in a single learning cycle technique while avoiding storing information in network nodes. These capabilities would reduce the resources and time requirements in resource-constrained environments such as WSNs.

The success of the scheme in dealing with such problems demonstrates its validity for use in optimisation and classification applications. It also suggests the scheme is a good candidate for implementation in other disciplines where pattern recognition can be used to enhance the process of problem solving. This suggests that the scheme will be part of new research areas and will play an important role in future research contributions. However, this would require proper analysis, design and modelling of the scheme to adapt to various problem solving applications. The next chapter of this thesis will

summarise the research contributions and findings and discuss the future research opportunities and limitations of the scheme.

Chapter 7

Conclusions and Future Work

7.1 Summary of the Research

This research proposes distributed and parallel pattern recognition schemes that minimise computations and communications by adopting local processing techniques to provide real time pattern recognition capabilities for complex problems such as real time event detection, artificial intelligence applications, and security applications. These features will be best suited to resource-constrained networks such as WSNs.

This research revised existing pattern recognition schemes for resource-constrained WSNs. In order to achieve a scalable scheme that meets the requirements of such networks, the scheme must combine limited communications and computations with using minimal memory resources. Additionally, the scheme must also involve low time complexity in order to serve online recognition applications. To deal with the real life sensory problems of WSNs, a good recognition scheme is expected to be capable of dealing with complex and noisy patterns with minimal available information. These requirements for good schemes derived from the challenges posed by WSN applications. Existing recognition schemes for WSNs have several

limitations in terms of these challenges such as iterative and centralised processing.

This research proposed light-weight, distributed and efficient pattern recognition schemes for resource-constrained and large scale systems and networks such as WSNs. The proposed schemes involve a distributed in-network processing paradigm that depends on local computations. The first proposed scheme is called CGN, which provides light-weight template matching capabilities. The proposed scheme adopts the GN distributed template matching technique in order to minimise communications and computations so as to perform recognition operations in a single learning cycle.

The design of CGN is extended in the second proposed scheme to present a more sophisticated approach that can provide more efficient recognition capabilities. In addition to the light-weight design, the second proposed scheme, called CwGN, adopts weighting technique that search for pattern edges and boundaries. The scheme is capable of detecting different types of pattern transformation such as translation, rotation, and dilation. The presented weighting technique depends on the change rate between direct adjacent nodes to minimise computations and communications. The scheme also adopts a two steps activation process to reduce the number of participating nodes in the recognition process so as to minimise communicational overheads and to increase network scalability. In order to support online operations, a zoning model was presented. The model addresses the constraints of timing requirements by allowing a number of CwGN networks to perform recognition

operations in a parallel paradigm. Required protocols that describe the network implementation and the nodes' message exchanges were also presented in this research.

This research presented theoretical and experimental analysis of both schemes, including comparing the proposed schemes with existing techniques. Theoretical analysis shows that both schemes are capable of converging to problem solutions within a predictable learning cycle duration that is proportional to the square root of the problem size. Network computational and communicational overheads can also be predicted on the basis of problem size. Additionally, theoretical analyses of CwGN scheme show that the scheme is capable of performing online operations as well as dealing with transformed patterns. Experimental evaluation confirmed the findings of the theoretical analysis. Experimental evaluations of CGN show that the scheme is capable of performing noisy pattern recognition. On the other hand, experimental evaluations of CwGN demonstrated the ability of the scheme to deal with transformed patterns. Moreover, these evaluations show how the network's communicational overheads in terms of time and energy can be reduced. Different experiments on the schemes presented in this research show that both schemes can perform recognition operations with higher accuracy levels as compared to other existing techniques.

Finally, the features of CwGN pattern recognition schemes were used to serve different application domains. Presenting two example models, the scheme was used in optimisation and classification problems. In the first

model, the CwGN scheme was used to enhance the performance of GA optimisation techniques in terms of fitness value and time. In the second model, the scheme was used to perform classification operations using its pattern recognition capabilities in the complex problem of human activity recognition to show that pattern recognition-based approaches can perform better than conventional classifiers.

7.2 Research Contributions and Outcomes

The outcomes of this research contribute mainly to the field of pattern recognition in limited resource networks and systems such as WSNs. More specifically, this research presents schemes that are capable of performing efficient online pattern recognition while maintaining minimal resource requirements that suit such limited systems. The significance of this research can be encapsulated as four major contributions: light-weight network design, online recognition performance, noisy, and transformation invariant recognition, and adaptability to different application domains.

Light-weight and distributed schemes design is the first key contribution of this research. Such design ensures high network scalability and increases its lifetime. Light-weight scheme design has been achieved by using distributed communicational and computational mechanisms, along with parallel cellular network design that minimises information exchange overheads. The network design of the schemes limits the number of messages required for each node to two exchange and one report messages. This relieves

network nodes from the tightly coupled connectivity requirements found in other schemes such as neural networks. If all network nodes participate in the recognition process, the total number of messages required can be calculated as $3S-2$ (S is the pattern size which is equal to the network size). However, the CwGN scheme's network design adopts activation mechanisms that minimise participating network nodes in the recognition process. This leads to further minimisation in the schemes' complexity, increases the network's lifetime, and increases network scalability. Experiments conducted on the CwGN scheme (in sub-section 5.4.1) show that the proposed activation process involved a range of only 1.14% to 5.04% nodes of the total number of the network's nodes in the information exchange process. The same experiments showed that a range of only 0.79% to 0.905% nodes were involved in the reporting process (see sub-section 5.4.2). In terms of communicational requirements, the experimental tests show that the communications required for a network of size 40000 nodes ranges between 4393 and 8424 nodes depending on the message sequence model. This is a range between 11% and 22.06% of the network size and of the number of communications required by parallel KNN. It was also shown experimentally that the number of required communications for a CwGN network of size 10000 nodes ranged between 307 and 536 nodes. That is a range between 3.07% and 5.36% of the network size, and of the required communications of other schemes (see sub-section 5.4.2).

Such minimisation of participating nodes also results in increasing the lifetime of the network. The tests conducted in this research show that the

average energy consumption scored, ranged between 0.3 and 0.57 mJ depending on the sequence model involved. Other schemes such as parallel KNN scored an average energy consumption of 2.9 mJ. By analysing these results, it was shown that a CwGN network can have a lifetime of two years with one of the smallest batteries in terms of capacity (i.e. 30 mAh or 324 joules). That is 8 times higher than other schemes such as parallel KNN (see sub-section 5.4.2).

The second significant contribution of this research is achieving pattern recognition within predictable single learning cycle duration. Such capability allows the scheme to support online and real time applications. The time complexity of presented schemes has been estimated as $O(\sqrt{S})$ in its worst case. That is when all nodes are activated and participate in the learning process. In other words, the required time to perform a single learning cycle is proportional to the square root of the pattern size. This complexity is minimal compared to other schemes that involve exponential iterative complexity such as neural networks. The parallel zoning CwGN model ensures a network's convergence within time restrictions for online applications support.

The third significant contribution of this research is efficient pattern recognition capabilities for noisy and transformed patterns. The CGN scheme presented in chapter 3 is mainly designed to deal with noisy patterns. Experimental evaluations on the scheme presented in section 3.5 show that the scheme is capable of dealing with noisy patterns with noise levels reaching 36.11% of the pattern size. In chapter 4, the transformation invariant

recognition capability is achieved by the CwGN weighting technique. This allows recognition of patterns while requiring minimal prior available information about these patterns. Theoretical and experimental analyses of the scheme show its ability to deal with translation, rotation, and dilation pattern transformation types. The results show that the scheme is capable of detecting translated patterns at any level, dealing with dilation levels up to 26%, and recognising rotated or even flipped rotated patterns with up to 23 degrees in any direction. The tests also showed that any pattern has the same weight as its flipped version pattern.

The fourth significant contribution of this research is the ability of the presented pattern recognition-based schemes to adapt to other techniques to serve different technological disciplines and application domains. This research presented a hybrid CwGN-GA model that contributed to enhancing the optimisation performance of GA in terms of time and accuracy. The model presented is always capable of finding better optimal solutions for given problems compared to traditional GA. The model is also able to provide similar fitness values to traditional GA for problems while cutting optimisation time by 36.5% (see sub-section 6.2.4).

The fifth contribution of this research is presenting the advantage of using pattern recognition based techniques in dealing with classification problems. In Chapter 6, a CwGN classification model was presented to deal with complex classification problems such as human activity recognition. Such a model shows the ability of the scheme to perform classification tasks with minimal

resource requirements using pattern recognition capabilities while maintaining high accuracy compared to other classification schemes. The examples presented in chapter 6 show the capability of the schemes presented here to adapt to different types of complex learning applications and systems. This opens the door for new research opportunities that involve recognition-based techniques in different research disciplines and application domains.

In general, the goal of the schemes presented in this thesis is to provide efficient pattern recognition capabilities. Both CGN and CwGN schemes were compared with other schemes in terms of accuracy. In Chapter 3, a CGN scheme was compared with Naïve Bayes and neural networks using a complex handwritten recognition problem. The test showed that the scheme is capable of dealing with the problem while providing higher accuracy levels compared to other scheme. In Chapter 5, the CwGN scheme was compared with several recognition schemes using two standard datasets. The first involved levels of pattern transformations. The scheme was compared with KNN, Naïve Bayes, and neural network schemes. The scheme was capable of achieving high an accuracy level of 95.38% compared to other schemes that scored accuracy levels ranging between 52.15% and 55.94%. The second test addressed the problem of robot guidance using a wall using a standard dataset. The scheme was firstly compared with traditional recognition schemes, then compared with schemes designed to deal with this specific problem. In both comparisons the scheme was capable of selecting the best routing decisions with the highest accuracy levels reaching up to 98.1% compared to other schemes.

7.3 Future Work

Future Work

This research has presented schemes mainly intended to provide online and efficient pattern recognition capabilities for large scale and resource-constrained networks such as WSNs. The features of the proposed schemes open the way for further enhancements and research opportunities. A useful extension of this research and further studies could involve the following:

- *New techniques for nodal computations outcomes reporting:* In subsection 4.3.2, reporting mechanisms were presented and discussed. However, attempts at parallelising nodes' reports will lead to further speed up in network performance and further minimisation of a network's time complexity. Implementing new techniques to find alternative reporting paths will also lead to higher network performance and efficiency. A zoning model was presented in this research that enhances and supports the online recognition capabilities of the proposed schemes. Hence, enhancing such models by researching possible ways to relate different network zones would extend recognition accuracy capabilities and ensure further online feature support.

- *Weighting mechanisms*: A weighting technique was adopted in the proposed CwGN scheme in this research (sub-section 4.3.3). The weighting technique adopted the change rate or the average change relationship between an active node and its direct adjacent nodes. This technique dealt with transformations of patterns such as translation, rotation, and dilation. A good extension of the research in this context would be implementing different weighting techniques that could lead to higher detection accuracy levels and could deal with other types of transformations and problems. The tradeoffs between implementing more complex weighting mechanisms and network performance would be a rich research area as more complex mechanisms could lead to more costs in terms of resource consumption and speed. The type of application and available resources are expected to be the main criteria that drive such tradeoffs.
- *Multi-dimensional design*: The proposed schemes in this thesis were modelled and designed to deal with 1-D, 2-D, and 3-D problems. For example, the wall following problem presented in sub-section 5.3.2 and the hybrid CwGN-GA model presented in section 6.2 both deal with 2-D problem types. The human activity classification model presented in section 6.3 deals with a 3-D problem space. However, other problems may involve higher dimensionality requirements. This research proposed the use of multi-track network

structures and zoning models to deal with multi-dimensional problems. However, network structure design constraints of the proposed schemes can be challenging. Hence, designing and analysing multi-dimensional network structures based upon the schemes proposed in this thesis would be a good extension of the research. New research would focus on design capabilities and prediction of the behaviour and performance of these schemes when dealing with multi-dimensional patterns.

- *Adaptation to different application domains:* The light-weight capabilities of the schemes proposed in this research can also be used in different types of application domain. How to use the recognition capabilities of a CwGN scheme in enhancing optimisation techniques and solving classification problems was discussed in Chapter 6. These points to further research opportunities that involve light-weight pattern recognition-based techniques such as CwGN in the steps and processes involved in solving complex real life problems. This could open up a wide area of research in fields such as artificial intelligence, optimisation, system security and network management.

In conclusion, this research has presented and demonstrated pattern recognition schemes that are capable of addressing the limitations associated with WSNs in serving mission critical and online applications. The schemes

presented in this thesis provide efficient recognition and high scalable capabilities while requiring minimal resources. Such capabilities motivate further research in the area of WSN pattern recognition and event detection. The proposed schemes also suggest research in other areas, such as optimisation and classification. Hence, the schemes proposed in this thesis are believed to have the potential to serve a wide range of applications beyond the fields of pattern recognition and WSNs.

References

- [1] A. Malinowski and Y. Hao, "Comparison of Embedded System Design for Industrial Applications," *IEEE Industrial Informatics*, vol. 7, pp. 244-254, 2011.
- [2] E. V. Krishnamurthy, "Algorithmic entropy and smart systems," in *Proceedings of International Conference on Intelligent Sensing and Information Processing*, pp. 175-180, 2004.
- [3] B. Chandrasekaran, "AI, knowledge, and the quest for smart systems," *IEEE Expert*, vol. 9, pp. 2-5, 1994.
- [4] W. Jie and I. Stojmenovic, "Ad hoc networks," *Computer*, vol. 37, pp. 29-31, 2004.
- [5] D. Puccinelli and M. Haenggi, "Wireless sensor networks: applications and challenges of ubiquitous sensing," *IEEE Circuits and Systems*, vol. 5, pp. 19-31, 2005.
- [6] M. Horton and J. Suh, "A vision for wireless sensor networks," in *Microwave Symposium Digest, 2005 IEEE MTT-S International*, vol. 4, pp.12-17, 2005
- [7] I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," *IEEE Communications*, vol. 40, pp. 102-114, 2002.

- [8] M. Tubaishat and S. Madria, "Sensor networks: An overview," *IEEE Potentials*, vol. 22, pp. 20-23, 2003.
- [9] P. Jiang, "A new method for node fault detection in wireless sensor networks," *Sensors*, vol. 9, pp. 1282-1294, 2009.
- [10] C. Farah, F. Schwaner, A. Abedi and M. Worboys, "Distributed homology algorithm to detect topological events via wireless sensor networks," *IET Wireless Sensor Systems*, vol. 1, pp. 151-160, 2011.
- [11] P. Ghosh, M. Mayo, V. Chaitankar, T. Habib, E. Perkins and S. K. Das, "Principles of genomic robustness inspire fault-tolerant WSN topologies: A network science based case study," in *IEEE Pervasive Computing and Communications Conference (PERCOM Workshops)*, pp. 160-165, 2011.
- [12] R. V. Kulkarni, A. Forster and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 13, pp. 68-96, 2011.
- [13] J. Yick, B. Mukherjee and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292-2330, 2008.
- [14] H. S. AbdelSalam and S. Olariu, "Toward efficient task management in wireless sensor networks," *IEEE Computers*, vol. 60, pp. 1638-1651, 2011.
- [15] P. Braca, S. Marano and V. Matta, "Enforcing consensus while monitoring the environment in wireless sensor networks," *IEEE Signal Processing*, vol. 56, pp. 3375-3380, 2008.

- [16] D. Chen and P. Varshney, "QoS support in wireless sensor networks: A survey," pp. 227-233, 2004.
- [17] X. Luo, M. Dong and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Computers*, vol. 55, pp. 58-70, 2006.
- [18] J. F. Chamberland and V. V. Veeravalli, "Wireless sensors in distributed detection applications," *IEEE Signal Processing*, vol. 24, pp. 16-25, 2007.
- [19] K. Chandy, "Event-driven applications: Costs, benefits and design approaches," *Gartner Application Integration and Web Services Summit*, California Institute of Technology, 2006.
- [20] G. Johansson, "Configurations in event perception," *Perceiving Events and Objects*, Hillsdale, NJ: Lawrence Erlbaum, pp. 29-122, 1994.
- [21] P. Boonma and J. Suzuki, "MONSOON: A coevolutionary multiobjective adaptation framework for dynamic wireless sensor networks," in *Proceedings of the 41st Annual International Conference on System Sciences, Hawaii*, pp. 497-497, 2008.
- [22] S. Ortmann, M. Maaser and P. Langendoerfer, "Adaptive pruning of event decision trees for energy efficient collaboration in event-driven WSN," in *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, MobiQuitous '09. 6th Annual International*, pp. 1-11, 2009.

- [23] M. Zoumboulakis and G. Roussos, "Complex Event Detection in Extremely Resource-Constrained Wireless Sensor Networks," *Mobile Networks and Applications*, pp. 1-20, 2010.
- [24] J. B. Predd, S. R. Kulkarni and H. V. Poor, "Consistency in models for distributed learning under communication constraints," *IEEE Information Theory*, vol. 52, pp. 52-63, 2006.
- [25] S. Watanabe, *Pattern Recognition: Human and Mechanical*. New York: John Wiley & Sons, Inc., 1985.
- [26] B. Catania, A. Maddalena and M. Mazza, "Psycho: A prototype system for pattern management," in *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 1346-1349, 2005.
- [27] A. S. Tanenbaum, C. Gamage and B. Crispo, "Taking sensor networks from the lab to the jungle," *Computer*, vol. 39, pp. 98-100, 2006.
- [28] V. Cantoni, L. Lombardi and P. Lombardi, "Challenges for data mining in distributed sensor networks," in *18th International Conference on Pattern Recognition, 2006 (ICPR)*, pp. 1000-1007, 2006.
- [29] A. Hac, *Wireless Sensor Network Designs*. Hoboken, NJ: J. Wiley, 2003.
- [30] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Chichester: Wiley-Interscience, 2007.
- [31] P. Santi, *Topology Control in Wireless ad hoc and Sensor Networks*. Hoboken, N.J.: Wiley, 2006.

- [32] S. Xingfa, W. Zhi and S. Youxian, "Wireless sensor networks for industrial applications," in *Fifth World Congress on Intelligent Control and Automation*, vol.4, pp. 3636-3640, 2004.
- [33] J. Lynch and K. Loh, "A summary review of wireless sensors and sensor networks for structural health monitoring," *Shock and Vibration Digest*, vol. 38, pp. 91-130, 2006.
- [34] A. Iyer, S. S. Kulkarni, V. Mhatre and C. P. Rosenberg, "A taxonomy-based approach to design of large-scale sensor networks," in *Wireless Sensor Networks and Applications*, Springer US, s. 1, pp. 3-33, 2008.
- [35] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Boston: Pearson Prentice Hall, 2011.
- [36] W. Charfi, M. Masmoudi and F. Derbel, "A layered model for wireless sensor networks," *SSD '09. 6th International Multi-Conference on Systems, Signals and Devices*, pp. 1-5, 2009.
- [37] W. Chonggang, K. Sohraby, L. Bo, M. Daneshmand and H. Yueming, "A survey of transport protocols for wireless sensor networks," *IEEE Network*, vol. 20, pp. 34-40, 2006.
- [38] A. Bachir, M. Dohler, T. Watteyne and K. K. Leung, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 12, pp. 222-248, 2010.
- [39] P. Huang, L. Xiao, S. Soltani, M. Mutka and N. Xi, "The evolution of MAC protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 101-120, 2013.

- [40] R. O. Duda, D. G. Stork and P. E. Hart, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
- [41] G. Wittenburg, N. Dziengel, C. Wartenburger and J. Schiller, "A system for distributed event detection in wireless sensor networks," *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 94-104, 2010.
- [42] J. B. Predd, S. B. Kulkarni and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, pp. 56-69, 2006.
- [43] E. F. Nakamura, A. A. F. Loureiro and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Computer Survey*, vol. 39, No. 9, pp.1-55, 2007.
- [44] Y. Kim, J. Kang, D. Kim, E. Kim, P. K. Chong and S. Seo, "Design of a fence surveillance system based on wireless sensor networks," in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, pp. 1-7, 2008.
- [45] S. Jabbar, A. E. Butt, N. us Sahar and A. A. Minhas, "Threshold based load balancing protocol for energy efficient routing in WSN," in *2011 13th International Conference on Advanced Communication Technology (ICACT)*, pp. 196-201, 2011.
- [46] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten and S. Jha, "Wireless sensor networks for battlefield

- surveillance," in *Proceedings of the Land Warfare Conference*, Brisbane, Australia, 2006.
- [47] R. Niu, P. Varshney, M. Moore and D. Klammer, "Decision fusion in a wireless sensor network with a large number of sensors," *Proceedings of the Seventh International Conference on Information Fusion*, pp. 21-27, 2004.
 - [48] S. R. Kulkarni, G. Lugosi and S. S. Venkatesh, "Learning pattern classification-a survey," *IEEE Information Theory*, vol. 44, pp. 2178-2206, 1998.
 - [49] J. Liangxiao, C. Zhihua, W. Dianhong and J. Siwei, "Survey of improving K-nearest-neighbor for classification," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD*, pp. 679-683, 2007.
 - [50] Z. Yang, N. Meratnia and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 12, pp. 159-170, 2010.
 - [51] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, pp. 1-58, 2009.
 - [52] K. Zhang, S. Shi, H. Gao and J. Li, "Unsupervised outlier detection in sensor networks using aggregation tree," in *Advanced Data Mining and Applications*, Springer Berlin / Heidelberg, vol. 4632, pp. 158-169, 2007.

- [53] L. Dan, K. D. Wong, H. Yu Hen and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, pp. 17-29, 2002.
- [54] S. Mittal, A. Aggarwal and S. L. Maskara, "Application of Bayesian belief networks for context extraction from wireless sensors data," in *14th International Conference on Advanced Communication Technology (ICACT)*, pp. 410-415, 2012.
- [55] E. Elnahrawy and B. Nath, "Context-aware sensors," *Proceedings of European Workshop on Wireless Sensor Networks*, pp. 77-93, 2004.
- [56] W. H. Wu, A. A. T. Bui, M. A. Batalin, L. K. Au, J. D. Binney and W. J. Kaiser, "MEDIC: Medical embedded device for individualized care," *Artificial Intelligence in Medicine*, vol. 42, pp. 137-152, 2008.
- [57] S. Xusheng and E. J. Coyle, "Low-complexity algorithms for event detection in wireless sensor networks," *IEEE Selected Areas in Communications*, vol. 28, pp. 1138-1148, 2010.
- [58] A. K. Jain, R. P. W. Duin and M. Jianchang, "Statistical pattern recognition: A review," *IEEE Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4-37, 2000.
- [59] D. Wang, "Pattern recognition: neural networks in perspective," *IEEE Expert*, vol. 8, pp. 52-60, 1993.
- [60] M. Hattori and M. Hagiwara, "Neural associative memory for intelligent information processing," in *Proceedings of Second*

International Conference on Knowledge-Based Intelligent Electronic Systems, KES '98, vol. 2, pp. 377-386, 1998.

- [61] G. Peng and L. Xiaolin, "Minimizing distribution cost of distributed neural networks in wireless sensor networks," in *Global Telecommunications Conference, GLOBECOM '07, IEEE*, pp. 790-794, 2007.
- [62] J. Nadal, "Study of a growth algorithm for a feedforward network," *International Journal of Neural Systems*, vol. 1, pp. 55-59, 1989.
- [63] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," *IEEE Neural Networks*, vol. 8, pp. 251-255, 1997.
- [64] A. Awad, T. Frunzke and F. Dressler, "Adaptive distance estimation and localization in WSN using RSSI measures," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD 2007*, pp. 471-478, 2007.
- [65] R. Rajkamal and P. Vanaja Ranjan, "Packet classification for Network processors in WSN traffic using ANN," in *6th IEEE International Conference on Industrial Informatics, INDIN 2008*, pp. 707-710, 2008.
- [66] X. Wang and S. Wang, "Collaborative signal processing for target tracking in distributed wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 67, pp. 501-515, 2007.
- [67] M. Ishizuka and M. Aida, "Achieving power-law placement in wireless sensor networks," *Proceedings of the 7th IEEE International*

- Symposium on Autonomous Decentralized Systems (ISADS'05)*, pp. 661-666, 2005.
- [68] D. A. Tran and T. Nguyen, "Localization in wireless sensor networks based on support vector machines," *IEEE Parallel and Distributed Systems*, vol. 19, pp. 981-994, 2008.
 - [69] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.
 - [70] G. Massini, "Hopfield neural network," *Substance Use & Misuse*, vol. 33, pp. 481-488, 1998.
 - [71] J. I. Z. Chen, Y. Chieh Chung, H. Meng Tsun and C. Yi Nung, "Employing CHNN to develop a data refining algorithm for wireless sensor networks," in *World Congress on Computer Science and Information Engineering, WRI*, pp. 24-31, 2009.
 - [72] D. Tisza, V. Peter, J. Levendovszky and A. Olah, "Multicast routing in wireless sensor networks with incomplete information," in *11th European Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless)*, pp. 1-5, 2011.
 - [73] J. Levendovszky, K. Tornai, G. Treplan and A. Olah, "Novel load balancing algorithms ensuring uniform packet loss probabilities for WSN," in *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pp. 1-5, 2011.

- [74] A. I. Moustapha and R. R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: application to fault detection," *IEEE Instrumentation and Measurement*, vol. 57, pp. 981-988, 2008.
- [75] J. T. Connor, R. D. Martin and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Neural Networks*, vol. 5, pp. 240-254, 1994.
- [76] J. W. Barron, A. I. Moustapha and R. R. Selmic, "Real-time implementation of fault detection in wireless sensor networks using neural networks," in *Fifth International Conference on Information Technology: New Generations, ITNG 2008*, pp. 378-383, 2008.
- [77] A. D. J. Raju and S. S. Manohar, "Recurrent neural network for faulty data identification in smart grid," in *International Conference on Recent Advancements in Electrical, Electronics and Control Engineering (ICONRAEECE)*, pp. 303-308, 2011.
- [78] G. Bartfai and R. White, "Adaptive resonance theory-based modular networks for incremental learning of hierarchical clusterings," *Connection Science*, vol. 9, pp. 87-112, 1997.
- [79] G. Carpenter, S. Grossberg and D. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991.
- [80] A. Kulakov and D. Davcev, "Tracking of unusual events in wireless sensor networks based on artificial neural-networks algorithms," in

International Conference on Information Technology: Coding and Computing, ITCC 2005, vol. 2, pp. 534-539, 2005.

- [81] L. YuanYuan and L. E. Parker, "Detecting and monitoring time-related abnormal events using a wireless sensor network and mobile robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008*, pp. 3292-3298, 2008.
- [82] M. Kumar, S. Verma and P. P. Singh, "Data clustering in sensor networks using ART," in *Fourth International Conference on Wireless Communication and Sensor Networks, WCSN 2008*, pp. 51-56, 2008.
- [83] M. Kumar, S. Verma and P. P. Singh, "Clustering approach to data aggregation in wireless sensor networks," in *16th IEEE International Conference on Networks, ICON 2008*, pp. 1-6, 2008.
- [84] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464-1480, 1990.
- [85] T. Kohonen, "Self-organizing maps," vol. 30, *Springer, Berlin, Heidelberg: Springer Series in Information Sciences*, 1995.
- [86] S. Wang, "Application of self-organising maps for data mining with incomplete data sets," *Neural Computing & Applications*, vol. 12, pp. 42-48, 2003.
- [87] G. Giorgetti, S. K. S. Gupta and G. Manes, "Wireless localization using self-organizing maps," 6th International Conference on Information Processing in Sensor Networks, Cambridge, Massachusetts, USA, pp. 293-302, 2007.

- [88] O. A. Postolache, P. M. B. S. Girao, J. M. D. Pereira and H. M. G. Ramos, "Self-organizing maps application in a remote water quality monitoring system," *IEEE Instrumentation and Measurement*, vol. 54, pp. 322-329, 2005.
- [89] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273-297, 1995.
- [90] D. Li, K. Wong, Y. Hu and A. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing*, vol. 19, pp. 17-29, 2002.
- [91] W. Xue, W. Sheng, B. Daowei, D. Liang and M. Junjie, "Collaborative peer-to-peer training and target classification in wireless sensor Networks," in *Future Generation Communication and Networking (FGCN 2007)*, vol. 1, pp. 208-213, 2007.
- [92] R. Abu Sajana, R. Subramanian, P. V. Kumar, S. Krishnan, B. Amrutur, J. Sebastian, M. Hegde and S. V. R. Anand, "A low-complexity algorithm for intrusion detection in a PIR-based wireless sensor network," in *5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 337-342, 2009.
- [93] A. Khan and V. Ramachandran, "A peer-to-peer associative memory network for intelligent information systems," presented at the 13th Australasian Conference on Information Systems, Melbourne, Australia, vol. 1, pp. 317-326, 2002.

- [94] A. I. Khan, M. Isreb and R. S. Spindler, "A parallel distributed application of the wireless sensor network," in *Proceedings of Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region*, pp. 81-88, 2004.
- [95] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Neural Networks*, vol. 19, pp. 212-229, 2008.
- [96] A. Khan and A. Amin, "One shot associative memory method for distorted pattern recognition," *AI 2007: Advances in Artificial Intelligence*, pp. 705-709, 2007.
- [97] R. Rengaswamy and V. Venkatasubramanian, "A syntactic pattern-recognition approach for process monitoring and fault diagnosis," *Engineering Applications of Artificial Intelligence*, vol. 8, pp. 35-51, 1995.
- [98] F. King-Sun and B. K. Bhargava, "Tree systems for syntactic pattern recognition," *IEEE Computers*, vol. C-22, pp. 1087-1099, 1973.
- [99] V. Latha, C. Subramaniam and S. Shanmugavel, "Fault tolerant wireless sensor network using case based reasoning with semantic tracking," in *Proceedings of the 5th WSEAS international conference on Communications and information technology*, pp. 240-246, 2011.
- [100] R. J. Hamilton, R. D. Pringle and P. M. Grant, "Syntactic techniques for pattern recognition on sampled data signals," *IEEE Computers and Digital Techniques*, vol. 139, pp. 156-164, 1992.

- [101] M. Marin-Perianu, C. Lombriser, O. Amft, P. Havinga and G. Tröster, "Distributed activity recognition with fuzzy-enabled wireless sensor networks," in *Distributed Computing in Sensor Systems*. vol. 5067, S. Nikolettseas, *et al.*, Eds. Berlin / Heidelberg: Springer, pp. 296-313, 2008.
- [102] J. S. R. Jang and S. Chuen-Tsai, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, pp. 378-406, 1995.
- [103] M. Zarei, A. M. Rahmani, A. Sasan and M. Teshnehlab, "Fuzzy based trust estimation for congestion control in wireless sensor networks," in *International Conference on Intelligent Networking and Collaborative Systems, INCOS '09*, pp. 233-236, 2009.
- [104] F. Xiufang, G. Zhanqiang, Y. Mian and X. Shibo, "Fuzzy distance measuring based on RSSI in Wireless Sensor Network," in *3rd International Conference on Intelligent System and Knowledge Engineering, ISKE 2008*, pp. 395-400, 2008.
- [105] M. Bahrepour, N. Meratnia, M. Poel, Z. Taghikhaki and P. J. M. Havinga, "Distributed event detection in wireless sensor networks for disaster management," in *2nd International Conference on Intelligent Networking and Collaborative Systems (INCOS)*, pp. 507-512, 2010.
- [106] F. Oldewurtel and P. Mahonen, "Neural wireless sensor networks," in *International Conference on Systems and Networks Communications, ICSNC '06*, pp. 28-28, 2006.

- [107] W. S. Hortos, "Unsupervised algorithms for intrusion detection and identification in wireless ad hoc sensor networks," in *SPIE Defense, Security, and Sensing*, vol. 7352, pp. 73520J, 2009.
- [108] G. Wittenburg, N. Dziengel, S. Adler, Z. Kasmi, M. Ziegert and J. Schiller, "Cooperative event detection in wireless sensor networks," *IEEE Communications Magazine*, vol. 50, pp. 124-131, 2012.
- [109] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Communications*, vol. 44, pp. 98-107, 2006.
- [110] W. M. Alfehaid and A. I. Khan, "Cellular microscopic pattern recogniser – A distributed computational approach for macroscopic event detection in WSN," in *Computational Science (ICCS), Proceedings of the 11th International Conference on Computational Science*, vol. 4, pp. 66-75, 2011.
- [111] S. Tactile, "Semeion Handwritten Digit Data Set", Semeion Research Center of Sciences of Communication [<http://www.semeion.it>], Rome, Italy, 1994, Access date: 23/01/2013.
- [112] A. Asuncion and D. J. Newman, "UCI machine learning repository," <http://archive.ics.uci.edu/ml>, Irvine, CA: University of California, School of Information and Computer Science, 2007, Access date: 11/03/2013.

- [113] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, pp. 10-18, 2009.
- [114] I. Witten, E. Frank, and M. Hall, *Data mining: Practical machine learning tools and techniques*, 3rd ed. San Francisco: Morgan Kaufman, 2011.
- [115] C. Farah, C. Zhong, M. Worboys and S. Nittel, "Detecting topological change using a wireless sensor network," in *Geographic Information Science*, Springer, 2008, pp. 55-69.
- [116] S. Bo, L. Osborne, X. Yang and S. Guizani, "Intrusion detection techniques in mobile ad hoc and wireless sensor networks," *IEEE Wireless Communications*, vol. 14, pp. 56-63, 2007.
- [117] B. Son, Y. Her and J. Kim, "A design and implementation of forest-fires surveillance system based on wireless sensor networks for South Korea mountains," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6, pp. 124-130, 2006.
- [118] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998.
- [119] C. Nebauer, "Evaluation of convolutional neural networks for visual recognition," *IEEE Neural Networks*, vol. 9, pp. 685-696, 1998.

- [120] F. Jialue, X. Wei, W. Ying and G. Yihong, "Human tracking using convolutional neural networks," *IEEE Neural Networks*, vol. 21, pp. 1610-1623, 2010.
- [121] K. Siddiqi and B. B. Kimia, "A shock grammar for recognition," in *Proceedings of the 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '96*, pp. 507-513, 1996.
- [122] T. B. Sebastian, P. N. Klein and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Pattern Analysis and Machine Intelligence*, vol. 26, pp. 550-571, 2004.
- [123] D. W. Arathorn, "Recognition under transformation using superposition ordering property," *Electronics Letters*, vol. 37, pp. 164-166, 2001.
- [124] T. Gedeon and D. Arathorn, "Convergence of map seeking circuits," *Journal of Mathematical Imaging and Vision*, vol. 29, pp. 235-248, 2007.
- [125] G. X. Ritter, P. Sussner and J. L. Diza-de-Leon, "Morphological associative memories," *IEEE Neural Networks*, vol. 9, pp. 281-293, 1998.
- [126] K. M. Iftekharruddin, "Transformation invariant on-line target recognition," *IEEE Neural Networks*, vol. 22, pp. 906-918, 2011.
- [127] W. M. Alfahaid, A. I. Khan and A. H. M. Amin, "A combined pattern recognition scheme with genetic algorithms for robot guidance using

- wireless sensor networks," in *12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pp. 759-764, 2012.
- [128] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current Opinion in Neurobiology*, vol. 14, pp. 481-487, 2004.
- [129] A. M. Bronstein, M. Bronstein, M. M. Bronstein and R. Kimmel, *Numerical Geometry of Non-rigid Shapes*, Springer-Verlag New York Inc, 2008.
- [130] A. Bronstein, M. Bronstein, A. Bruckstein and R. Kimmel, "Analysis of two-dimensional non-rigid shapes," *International Journal of Computer Vision*, vol. 78, pp. 67-88, 2008.
- [131] Made with Natural Earth. Free vector and raster map data @ [naturalearthdata.com](http://www.naturalearthdata.com/downloads/10m-raster-data/10m-cross-blend-hypso/). Available: <http://www.naturalearthdata.com/downloads/10m-raster-data/10m-cross-blend-hypso/>, Access date: 15/01/2013.
- [132] A. L. Freire, G. A. Barreto, M. Veloso and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in *6th Latin American Robotics Symposium (LARS)*, pp. 1-6, 2009.
- [133] A. S. Wander, N. Gura, H. Eberle, V. Gupta and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Third IEEE International Conference on Pervasive Computing and Communications, PerCom 2005*, pp. 324-328, 2005.

- [134] G. Guimaraes, E. Souto, D. Sadok and J. Kelner, "Evaluation of security mechanisms in wireless sensor networks," in *Proceedings of Systems Communications, 2005 Conference*, pp. 428-433, 2005.
- [135] S. Ping, "Delay measurement time synchronization for wireless sensor networks," *Intelligent Research Berkeley Lab*, Intel Research, IRB-TR-03-013, 2003.
- [136] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning* vol. 1. New York: Springer, 2006.
- [137] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Machines*, Cambridge, MA: MIT Press, 2007.
- [138] A. Fabisch, Y. Kassahun, H. Wöhrle and F. Kirchner, "Learning in compressed space," *Neural Networks*, vol. 42, pp. 83-93, 2013.
- [139] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, pp. 66-72, 1992.
- [140] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial Intelligence*, vol. 127, pp. 57-85, 2001.
- [141] Y. Wu and W. Liu, "Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks," *Wireless Sensor Systems, IET*, vol. 3, pp. 112-118, 2013.
- [142] F. V. C. Martins, E. G. Carrano, E. F. Wanner, R. H. C. Takahashi and G. R. Mateus, "A hybrid multiobjective evolutionary approach for

- improving the performance of wireless sensor networks," *Sensors Journal, IEEE*, vol. 11, pp. 545-554, 2011.
- [143] Y. Soh Chin, V. Ganapathy and C. Lim Ooi, "Improved genetic algorithms based optimum path planning for mobile robot," in *11th International Conference on Control Automation Robotics & Vision (ICARCV)*, pp. 1565-1570, 2010.
- [144] T. A. El-Mihoub, A. A. Hopgood, L. Nolle and A. Battersby, "Hybrid genetic algorithms: A review," *Engineering Letters*, vol. 13, pp. 124-137, 2006.
- [145] K. F. Man, K. S. Tang and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Industrial Electronics*, vol. 43, pp. 519-534, 1996.
- [146] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *4th Student Conference on Research and Development, SCOReD 2006*, pp. 183-188, 2006.
- [147] W. E. Hart, T. E. Kammeyer and R. K. Belew, "The role of development in genetic algorithms," *Foundations of Genetic Algorithms*, vol. 3, pp. 315-332, 1994.
- [148] P. Oliveto, J. He and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, pp. 281-293, 2007.

- [149] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright and H. M. Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway*, vol. 2, pp. 1338-1345, 2004.
- [150] T. W. Manikas, K. Ashenayi and R. L. Wainwright, "Genetic algorithms for autonomous robot navigation," *IEEE Instrumentation & Measurement*, vol. 10, pp. 26-31, 2007.
- [151] C. Liming, J. Hoey, C. D. Nugent, D. J. Cook and Y. Zhiwen, "Sensor-based activity recognition," *IEEE Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, pp. 790-808, 2012.
- [152] C. Liming, C. D. Nugent and W. Hui, "A knowledge-driven approach to activity recognition in smart homes," *IEEE Knowledge and Data Engineering*, vol. 24, pp. 961-974, 2012.
- [153] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu and P. Havinga, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," in *23rd International Conference on Architecture of Computing Systems (ARCS)*, pp. 1-10, 2010.
- [154] C. Huasong, V. Leung, C. Chow and H. Chan, "Enabling technologies for wireless body area networks: A survey and outlook," *IEEE Communications* vol. 47, pp. 84-93, 2009.
- [155] J. Parkka, L. Cluitmans and M. Ermes, "Personalization algorithm for real-time activity recognition using PDA, wireless motion bands, and

- binary decision tree," *IEEE Information Technology in Biomedicine*, vol. 14, pp. 1211-1215, 2010.
- [156] C. Zhu and S. Weihua, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *IEEE Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, pp. 569-573, 2011.
- [157] M. Zhang and A. A. Sawchuk, "Human daily activity recognition with sparse representation using wearable sensors," *IEEE Biomedical and Health Informatics*, vol. 17, pp. 553-560, 2013.
- [158] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu and P. Havinga, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," *23rd International Conference on Architecture of Computing Systems (ARCS)*, pp. 1-10, 2010.
- [159] D. Roggen, A. Calatroni, M. Rossi, T. Holleczeck, K. Forster, G. Troster, P. Lukowicz, D. Bannach, G. Pirkel, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, J. del R Millan, "Collecting complex activity datasets in highly rich networked sensor environments," in *Seventh International Conference on Networked Sensing Systems (INSS)*, pp. 233-240, 2010.
- [160] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, pp. 2033-2042, 2013

- [161] *Opportunistic activity recognition systems*. Available:
<http://www.opportunity-project.eu>, 2011, Access date:12/7/2012.