DESIGN FOR NOVEL ENHANCED WEIGHTLESS NEURAL NETWORK AND MULTI-CLASSIFIER

A Thesis Submitted to the University of Kent
For The Degree of Doctor of Philosophy
In Electronic Engineering

By
Pierre Lorrentz
November 2009

TABLE OF CONTENT

Abstract

Pierre Lorrentz, PhD 2009

Weightless neural systems have often struggles in terms of speed, performances, and memory issues. There is also lack of sufficient interfacing of weightless neural systems to others systems. Addressing these issues motivates and forms the aims and objectives of this thesis. In addressing these issues, algorithms are formulated, classifiers, and multi-classifiers are designed, and hardware design of classifier are also reported. Specifically, the purpose of this thesis is to report on the algorithms and designs of weightless neural systems.

A background material for the research is a weightless neural network known as Probabilistic Convergent Network (PCN). By introducing two new and different interfacing method, the word "Enhanced" is added to PCN thereby giving it the name Enhanced Probabilistic Convergent Network (EPCN). To solve the problem of speed and performances when large-class databases are employed in data analysis, multi-classifiers are designed whose composition vary depending on problem complexity. It also leads to the introduction of a novel gating function with application of EPCN as an intelligent combiner. For databases which are not very large, single classifiers suffices. Speed and ease of application in adverse condition were considered as improvement which has led to the design of EPCN in hardware. A novel *hashing function* is implemented and tested on hardware-based EPCN.

Results obtained have indicated the utility of employing weightless neural systems. The results obtained also indicate significant new possible areas of application of weightless neural systems.

Acknowledgement

Thanks to:

Prof. K. McDonald-Maier
Department of Computing and Electronic Systems
University of Essex
Wivenhoe Park, Colchester,
CO4 3SQ, UK.

and

Dr. W. G. Howells
Department of Electronics
University of Kent, Canterbury
Kent CT2 7NT, UK.

for their support.

LIST OF FIGURES

LIST OF TABLES

List of Publications

1. Lorrentz P., Howells W.G.J., McDonald-Maier K.D., Enhanced Probabilistic Convergent Network, RASC 2006, K. Sirlantzis (Ed.), pp. 267 – 272, 2006.

2. Lorrentz P., Howells W.G.J., McDonald-Maier K.D., Design and analysis of a novel weightless neural based Multi-classifier, World Congress on Engineering, July 2007, Vol. 1, pp. 65-70.

3. Lorrentz P., Howells W.G.H., and McDonald-Maier K.D.: A novel weightless artificial neural based Multi-classifier for large classification, Neural Processing Letters: Volume 31, Issue 1 (2010), Page 25.

4. P. Lorrentz, W.G. Howells, K.D. McDonald-Maier: An FPGA based adaptive weightless Neural Network Hardware, IEEE, NASA/ESA 2008, AHS-2008, Noordwijk, The Netherlands, June 22-25 2008, pp. 220-227.

5. P. Lorrentz, et al., An advanced combination strategy for multi-classifiers employed in large multi-class problem domains, Appl. Soft Computing J. (March 2011), Volume 11, Issue 2, ISSN 1568 – 4946, 2151–2163, doi:10.1016/j.asoc.2010.07.014.

6. Lorrentz P., Howells W.G.J., McDonald-Maier K.D., An Analysis of Hardware Configurations for an Adaptive Weightless Neural Network, World Congress on Engineering, July 2008, pp. Vol. 1, pp. 66-71.

1. THESIS INTRODUCTION

1.0 Introduction

A neural network attempts to solve a particular problem with which it is identified as an expert. The methods with which neural networks provide solution vary. Any of the method usually relies on mathematical calculations. The definition of neural network assumed in this thesis is due to Haykins [49], and it states:-

A neural network is a massively parallel distributed processor made up of simple processing unit, which has natural propensity for storing experiential knowledge and making it available or use. It resembles the brain in two respects:

1) Knowledge is acquired by the network from its environment through a learning process.

2) *Inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

It is noteworthy that the term "neural network" and "classifier" may refer to the same network loosely in this thesis sometimes. Most pattern recognition problems can however be solved by performing operations, such as one-to-one or one-to-many mappings on input patterns to output. When an input pattern is binary, the problem is reduced to a simple logic problem. Under this condition, Random Access Memory (RAM) based weightless Neural Networks (classifier) are well suited. Pattern recognition problems become attractive given that it is a simple case of transforming problems to its logical equivalent and supplying it to a RAM-based Network for solution. An early variation of RAM-based classifier known as n-tuple recognition system was introduced in the 1950s by Bledsoe and Browning [10].

Most customary weighted neural networks, during training, passes through many epochs. Epochs refers simply as the number of times the input data will be accessed by the network, before the error rate decreases appreciably. The epochs of training are often quite large, ranging from the order of tens to order of thousands sometimes. During the epochs, it is expected that the input data does not change value or structure, i.e. it is typically required that input dataset be static during learning. This period could last for minutes and hours. This is disadvantageous for dynamic and real-time systems and systems from which fast and intelligent response is required. In such situation as these, a neural network from which one epoch of learning is required is more beneficial. One epoch of learning is also

referred to as one-time pass over the input database; which is also referred to as one-shot learning for weightless nets. One epoch of learning implies time reduction as compared to many epochs of learning. It also implies a reduction in time required for input pattern to be stationary. Stationary in the sense that input pattern must not change between sampling intervals. Although it is possible for RAM-based classifier to make several passes over an input space, this may not be required for an increased performance. Quite often, one-pass over the input data may be sufficient. Austin [7], Howells [57], and some other authors have experimentally confirmed this on RAM-based classifiers. RAM-based classifiers identify itself with binary number. So that both input and output data are inherently binary. Because of this and the reasons in subsequent paragraphs, the decision to work in areas of weightless neural system may be beneficial and significant.

Examples of RAM-based classifier that has combined advantages of one-pass over input data-set, sometimes called one-shot learning, and the ability to process probabilistic reasoning have not been successfully produced. Recall that probability is usually expressed as a number lying between 0 and 1. Secondly, associated with it is the frequency of occurrence of events. These characteristics are specific to RAM-based classifiers reported in this thesis. For this reason, the study of weightless classifier is attractive.

All neural systems presented in this thesis require mainly reading from and writing to RAM-memory locations for their functionality. For this reason, they are commonly called RAM-based classifiers, since almost all mathematical functions are converted to their Boolean equivalent. For example, Boolean addition does not demand a high memory requirement as does floating-point calculations. Thus the amount of mathematical calculation performed is relative to the amount of data supplied to the classifier. Based on this fact, the decision to implement weightless neural network is motivated. Secondly, floating-point and continuous mathematics require a relatively long time to complete. The long completion time of processes also implies long training and recognition time of neural network. Thus the elimination of time required for traditional mathematical calculation, by replacing many forms of mathematical calculation with their equivalent Boolean logic, will be beneficial, and it is a significant venture. It also constitutes one of the main reasons for deciding to consider mainly weightless neural networks in this thesis.

In order to increase the speed of weightless neural networks, methods such as pipelining and parallelism are introduced. Parallel execution of processes means different and many events occur concurrently. Pipelining is a group of parallel processes with a shift in time. But pipelining and parallelism has no software equivalent, they apply almost only

to hardware design. Employment of pipeline and parallelism in hardware design is required when confronted with an implementation of a complex system such as addressed in this thesis. To enable the employment of weightless classifier on large-class database, the implementation of a pipelined system is significant. Now and in future, neural networks will have an increasing number of large-class databases to classify. For this reason, it is decided to implement RAM-based neural network in hardware.

The Probabilistic Logic Neuron (PLN) [71] and Generalised Convergent Network (GCN) [57] are notable examples of weightless neural networks. Probabilistic Convergent Network (PCN) is introduced due to its ability to process probabilistic reasoning. Probabilistic reasoning finds itself in PCN and is beneficial in many respects. One of the most important benefits is the confidence measure obtainable at the output of PCN.

1.1 Aims and Objectives

When PCN was first developed, connectivity are formed as specified in [57]. This method fails to consider other attributes and features of patterns well suited to connectivity formation. Secondly, it does not consider alternate method of forming connectivity [65], [76]. These motivate the intention to introduce other methods of connectivity formation algorithms to PCN. Introduction of new connectivity methods to PCN facilitates its application to many other problem domains such that the word “enhanced” is added to the PCN, and the new neural network is called Enhanced Probabilistic Convergent Network (EPCN). When other connectivity formation methods are introduced, different types of PCN may result as different types of connectivity methods are utilised, and thus able to solve different types of problems.

Multi-classifier (MCS) design is motivated by the need to improve performance on “difficult” patterns such as found in some handwritten characters. Weightless MCS have an added advantage of reducing problems to simple logic problem. A pattern is termed “difficult” if it could not be classified, or if it may be classified wrongly by a neural network. A Multi-classifier System consists of a number of single classifier arranged in such a way as to decrease classification errors beyond that which is possible for a single neural network. The need to design an MCS consisting wholly of weightless classifier is motivated by computation overhead, speed, and memory requirement. These are much reduced in a random access memory based MCS as compared to a customary (other

alternative) MCS. Weightless multi-classifier design is motivated by very low computation overhead, high speed, and low memory processing requirement.

Currently, most traditional MCS often struggles with large-scaled multi-class databases such as found in biometric domain. For this reason the state-of-the-art MCS may utilise very many base classifiers when classifying very large-scaled multi-class databases. There are other problems associated with traditional MCS such as bias, and saturation effects. Bias and saturation effects may affect weightless MCS also. Providing solution to these problems constitute the motives for specific weightless MCS design in this thesis. Such that, it may be demonstrated that a weightless MCS consisting of few base classifiers is capable of classifying biometric database without performance trade-offs. In short, solutions to bias and saturation effects in weightless MCS will be addressed. The potential industrial benefit of each MCS will be demonstrated by their application to databases such as handwritten character (such as are found in filling of forms), and in fingerprint verification.

One of the aims of this thesis is to research, in practical terms, the possibility of implementation in hardware (possibly FPGA) of a weightless classifier. Currently, a hardware implementation of a complex weightless NN is very rare. The possibility of a hardware implementation offers many benefits such as ease of application (e.g. a PC is not required), higher speed of execution of processes etc. An extension of the hardware implementation may be enabling and enhancing a variance of the neural network to portable devices. Thus a hardware implementation is significant and beneficial when time and ease of application are considered. For these reasons, this thesis examines the implementation in hardware of weightless classifier, and RAM-based multi-classifiers. Hardware implementation of EPCN also shortens the time requires for learning and recognition. Additionally, hardware implementation of EPCN is motivated by the scarcity of a digital classifier hardware that classifies a pattern, and at the same time provides a level of confidence that a pattern meant for recognition belongs to each class. Hardware implementation of EPCN is attractive because of its speed and portability which are comparable to (and may surpass) that of existing neural networks. This means that it is a good alternative (and even a better alternative) to a state-of-the-art neural system. Software implementation of weightless multi-classifier systems and Hardware implementation of EPCN has memory consumption overhead, and also the problem of mapping all probabilistic values (between 0 and 1) to positive whole number values. Research efforts in these implementations should place adequate consideration on these problems.

This research is of academic significance since; the experimentation with different types of connectivity usage on PCN has exhibited different types of characteristics of the network which has not been encountered. Secondly, these implementations have made possible using weightless NN in parallel and hierarchical design of MCS. Works in this thesis may be useful in providing a hardware and software prototype of EPCN, and also a software prototype of a MCS, for industrial and academic benefits. Since the MCS and EPCN, when implemented, may also be used in Banks and Hospitals for recognition on handwritten character, therefore for "good" pattern target performance may be expected to be very high, whereas for "difficult" patters target performance may be expected to be high. False recognition and low % recognition of handwritten characters may be intolerable in banks and hospitals, because of aftermath consequences.

The work is also motivated by the intension to produce intelligent neural systems which have applications in industries and also in, land, sea, and air based exploration systems. Production of this system concerns mainly the production of sub-systems that carry out the perception of data, interpretation of these data, pattern recognition, and control signal. This will be achieved by using real-world data, and external industrial application. The projects could be grouped as follows:

1. Input data sensing, and input data pre-processing.
2. Weightless Neural Networks (NN):- RAM-based and using formal logic.
3. Weightless Multi-Expert System.
4. Hardware development.

The rest of this chapter is organised as follows. The projects contained in this thesis employ the work in [57] as background materials so that, in section 1.2, the content of [57] with respect to PCN is reviewed as a background material for the thesis. The organisation of the projects reported in this thesis is explained in section 1.3. In executing these plans, the areas where difficulties might arise are explained in section 1.4. This is followed by conclusion in section 1.5.

1.2 Weightless Neural Networks

Weightless neural network is formally defined as a neural network whose functionality does not explicitly depend on activation function and weights. One of the advantages of this class of neural network is that it neither requires high mathematical computation nor high linguist demand. But require only two values "1" and "0" for its functionality. Since only two values are involved, they may be referred to as binary (dependent) neural

networks.

Most problems which may be difficult for other classes of NN or require high resource demand are solved easily by binary neural network, thus making this class of neural network an important class. The binary neural network is also called RAM-based or weightless NN. They depend, for their functionality, on Boolean logic. So that, mapping problems to their Boolean equivalent is the only requirement. Mapping problems to their Boolean equivalent signifies a mapping to binary domain.

The necessity of design and application of weightless NN could thus be seen from the much less resource demand, high speed, and comparable performance. A simple and quick illustration of how problems may be mapped to logic domain is the recognition of the character "0". The character "0" is shown in figure 1.0. The problems become very simple to resolve when they are threshold-binarised to give binary pattern. Thus pattern recognition problems could be expressed as Boolean or Logic function involving only two values, 1 and 0.

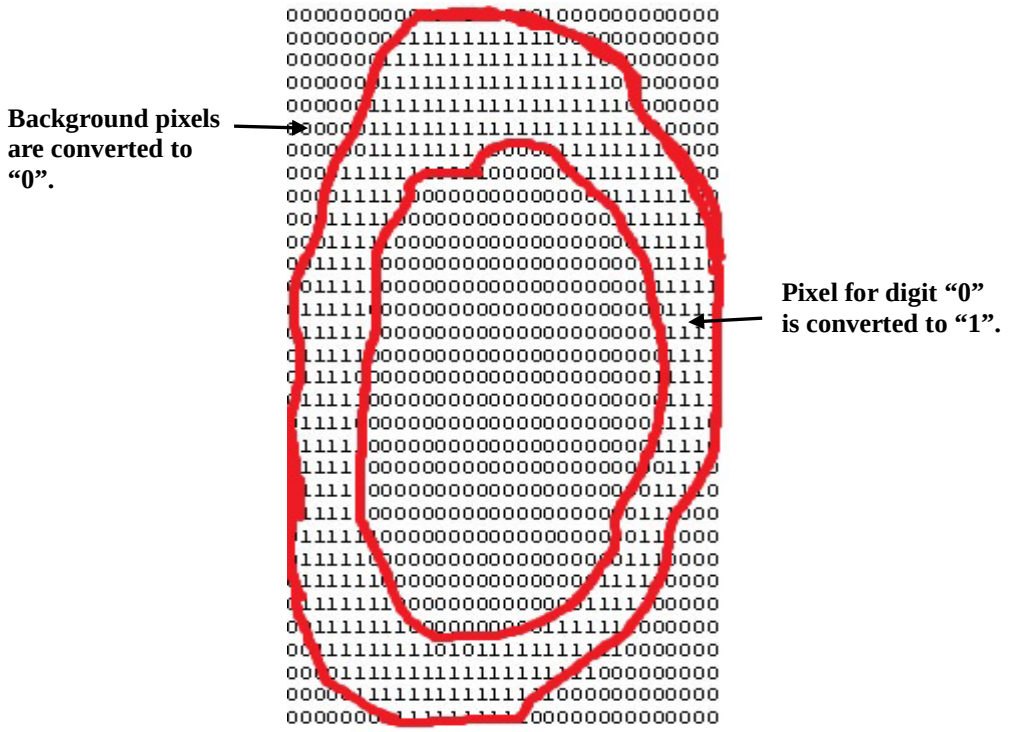


Figure 1.0: Example of input pattern.

RAM-based (N-tuple) weightless NN is normally employed to implement and map these logic functions. There is one-to-one mapping between the recognition problems and the logic functions. RAM-based NN provide solution to these problems by mappings to these logic functions. Besides, weightless NN offers considerable advantages over

weighted ones; these are:-

- One-shot learning: - This refers to one-time pass over the input database. RAM-based NNs does not go through many epochs of learning as the weighted ones.
- It is fast: - They are often arranged in look-up Tables (LUTs). And learning is a process of modifying the content of these LUTs. No complex mathematical computation involved.
- RAM-based NN will attempt to provide input-output Boolean logic mapping for any arbitrary problem.

The operation of RAM-based classifier is analogous to that of conventional RAM chip, namely, it consist of address-line(s) and memory locations. Because RAM-based NN is binary in nature, for n number of address-line, 2^n memory locations can be addressed. The relationship between the address-lines and memory location is

$$y = 2^n$$

Where:

n = number of address lines; here, this may be referred to as connectivity of the neuron.

y = number of memory location; this is referred to as a neuron with n connectivity.

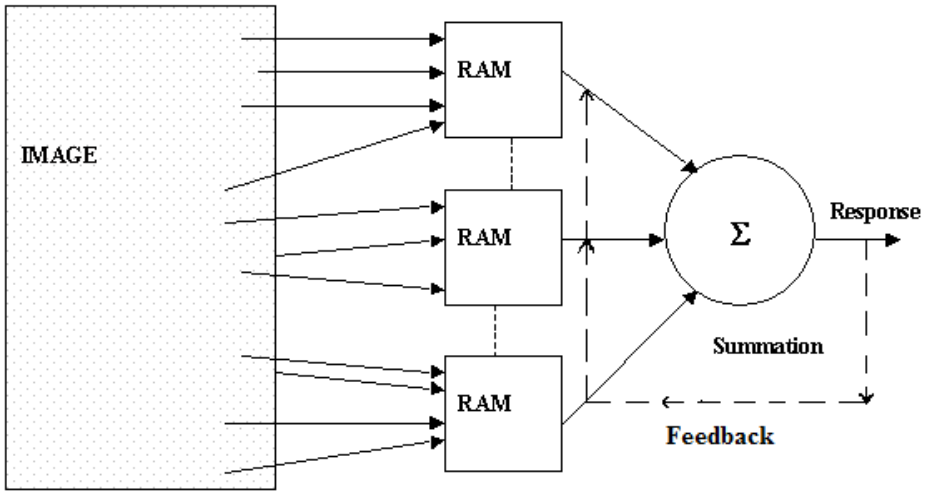


Figure 1.1: A schematic of a general weightless artificial neural network.

Of a particular interest is the architecture espoused in [110] which will be explained here.

Figure 1.1 shows a diagram of weightless classifier whereby the input pattern is a threshold image. An algorithm often exist, known as learning algorithm, which connects the image features to the neurons located in the RAM. Within each neuron are 2^n numbers of locations where n = number of address-line (the connectivity). A good example of weightless classifier is PCN. For this reason PCN is also referred as *n-tuple*

classifier. The process whereby the RAM accesses the input pattern, and some locations within the neurons in each layer are assigned non-zero values, is called *learning process.*

Similarly, the process whereby the RAM accesses both the input pattern and the learned RAMs, and some locations within the neurons in the RAM are assigned non-zero values, is called *recognition process*. These RAMs (the "recognition" RAMs) are combined to the output. This represents the output of the neuron. The output may be feed-back iteratively until convergence or a fixed number of times (see figure 1.1) during a recognition process.

1.3 Organisation of the Research Projects in the Thesis

This section introduces the projects that constitute the research. It then explains the planning of the projects logically. Some terminologies will be introduced as follows.

Difficult pattern: A "difficult" pattern is a patter class that cannot be classified by weightless NNs when they generalise. The weightless NN require an auxiliary system and/or be put into a special behavioural mode before they are able to classify "difficult" pattern. Example of methods used to classify difficult pattern are Boosting and Bagging. The "good" pattern class can always be classified correctly by weightless NN when they generalise well.

Ambiguous cases: Ambiguous case of pattern recognition occurs when a pattern is said to belong to more than one class with equal probability during recognition. Thus the ambiguous pattern cannot be assigned to a specific class by the weightless NN during classification. Ambiguous case may be event-driven (behavioural) within the NN structure such that in some cases it may be correctly classified. Bias (see below) may also give rise to ambiguous cases. These have nothing to do with a pattern (a "difficult" pattern) that is consistently classified wrongly. When a specific pattern is consistently ambiguous, it may also be called a "difficult" pattern.

Saturation effect: as patter class becomes large and also for increasing large number of classes, the weightless NN distinguishing features for inter-class classification diminishes, this phenomenon is known as saturation effect.

Bias: The weightless NN is said to be biased when the performances depend on a specific arrangement of the training classes.

By formulating some problem areas associated with these projects and explaining method(s) which shall provide solution to them, the project plans will become clearer.

A) **Problem:** How to optimise PCN/MCS to achieve better recognition performance? Secondly, what effect will system parameter modification have on performances?

Proposed Solution: Paramount in this area is the address formation methods. It is intended to perform both literature research and practical research into optimisation techniques. The introduction of novel weightless NN will be attempted. If the novel NNs give promising result, then improvement of the NN for performances and robustness is required. In such situation, Genetic Algorithm (GA) may be considered for parameter optimisation. A better solution is an intrinsic improvement to the weightless NN which does not warrant an auxiliary system like GA. One or two of these techniques may be used on PCN/MCS. And their effect on performance will be considered.

B) **Problem:** What modification to what aspect of the PCN/MCS will increase generalisation, most especially on "difficult" patterns? There are other problems such as: (1.) ambiguous cases (2.) saturation effects. (3) bias

Proposed Solution: It is to be expected that performance depends, to some extent, on type of input database also. Large number of inputs may also lead to saturation. These projects may also consider:

1) Investigate effects of tuple-size.
2) Consider using filter.
3) Consider using Particle swarm/Ant-colony optimisation.
4) Genetic Algorithm.
5) Reconfiguration: Dynamic and static adaptation techniques.

To extend the functionality of PCN (i.e. increase the situations where it may be utilized), reviewing the input mapping method is in order. The first part of this work will focus on PCN and its enhancement, whereby two novel types of connectivity will be introduced. The first part of this work focuses on two types of PCN. The main differences between these PCNs are in their input mapping methods. It is envisaged that the address formation method be substantially different from that of [57].

Secondly, since PCN expects its input to be binary and most real-life input varies greatly, tools for processing input to give binary patterns are developed.

Howells [57] explains the advantages of RAM-based classifier but does not point out serious limitations to its areas of application. Of a particular interest is a large multi-class problem that is databases which consist of large classes. With this type of databases there

are two major problems. One is known as *bias* and the other is called *saturation effects.* Bias toward a class occurs when the probability of recognition of a class is unusually high. Saturation occurs when distinguishing features of classes are no longer represented within the network. The network is said to saturate. For these reasons single neural network becomes incapable of classification of large databases. It is to be noted that all intended neural networks designed depends on logging and retrieval of information from RAM-locations in a neuron. By "good" pattern is meant those patterns easily recognisable by any classifier, while "difficult" patterns are those patterns that are classifiable only by special techniques such as Bootstrapping, Bagging and Boosting. Weightless classifiers capable of classifying large-scaled multi-class databases are very few. The aim therefore is to implement multi-classifier using RAM-based neural networks as component (base) classifier. The proposed MCS (see paragraphs below) may not require a special technique, such as Bootstrapping, on "difficult" pattern (see section 4). The proposed Multi-classifier system has some advantages over the contemporary MCS. For more details see section 2.2.

It is worthy of note that the input format of Enhanced Probabilistic Convergent Network (EPCN) *demands an encoding* of its input if it is intended as an intelligent combiner. In order to map the output of the base classifiers to the input of the intelligent combiner, none of the existing gating functions were found suitable. The unsuitability of existing gating function motivates a new gating function. The new gating function introduced constitutes a combination strategy and is known as a combiner unit. Thus the design of a RAM-based MCS automatically entails a design of a novel combining strategy – the combiner unit. To fully test the impact of the combination scheme, a parallel

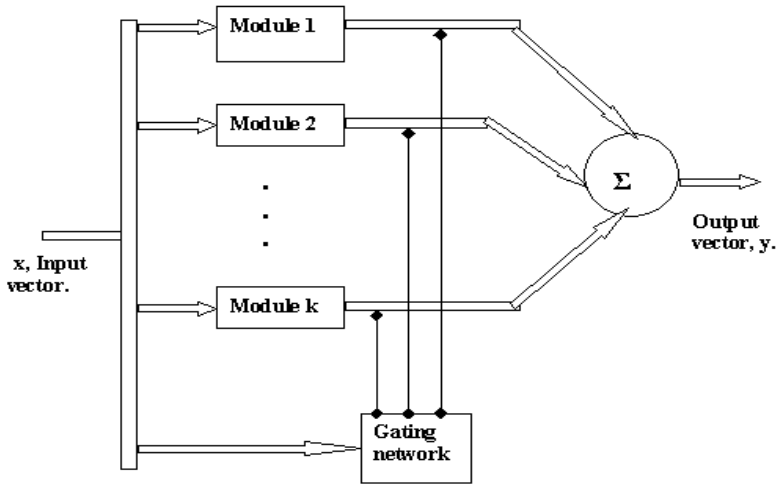


Figure 1.2: Schematic diagram of a multi-classifier. It consists of different types of base network in parallel. **[40]**

arrangement of component classifiers is preferred, of the type shown in figure 1.2.

When the proposed multi-classifier have been designed, it becomes appropriate to explore its usefulness by experimentations. Experimentations on some databases then follow. Experimentation on the MCS employs fingerprint databases. Areas of automatic template-free fingerprint verification have been subject of intensive research. It is also noteworthy that biometric databases are characterized with large classes. This large-class criterion makes it a suitable candidate database for the MCS reported in the thesis. If the MCS is able to generalise, it may found application as a biometric template-free fingerprint verifier.

There are situations which demands autonomous operations such as in automated machines, Robots, etc. There is also electronic harsh surrounding in which an intelligent weightless neural network might operate, e.g.; sea exploration. To enhance the capability of the presented software-designed EPCN and weightless multi-classifier in such environment, the implementation platform needs to be changed. Secondly, a quick and accelerated response is required in cases of emergencies. These conditions suggest a hardware implementation, because a hardware-based classifier operates very fast as compared to a classifier designed in software. High speed of hardware based design is very suitable for cases of emergency and adverse conditions. It suggests a digital hardware which is reconfigurable. Virtex-II pro is an advance Field Programmable Gate Array (FPGA), and belong to a group reconfigurable (see paragraphs below) FPGA. The high level of integration possible with FPGA means it lends itself easily to implementation of complex electronic systems. Reconfigurable FPGA, like Virtex II pro, offers rapid design process and reprogrammable functions.

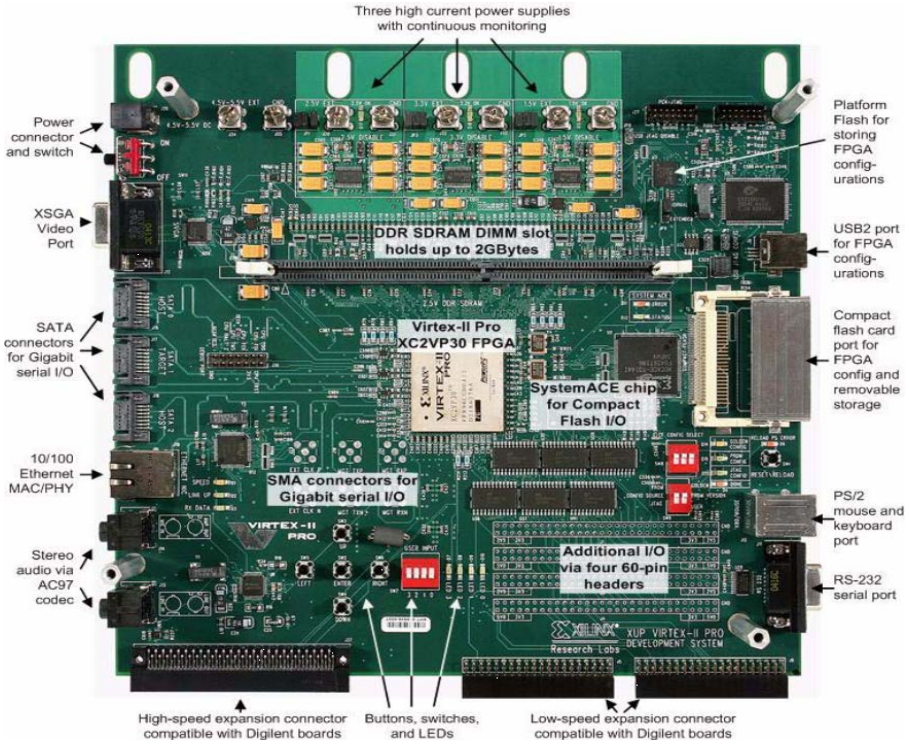


Figure 1.3: Annotated Diagram of Virtex-II pro which will be used in EPCN prototyping. **[134]**

This is in contrast to micro-processing whereby functions are not reprogrammable and a long time is required to produce working silicon. Also, since EPCN is adaptive in nature, to adequately represent its characteristic, a reconfigurable FPGA is required. The Virtex-II pro development board, Figure 1.3, is an advanced digital prototyping board, it is reconfigurable and found suitable for EPCN implementation. The proposed hardware is expected to operate autonomously, thus suitable for electronically harsh surrounding, autonomous machines and robots.

ORGANISATION: - The researches investigated have been organised into chapters. The chapters are organised as follows.

In chapter 3, other forms of connectivity were successfully introduced, followed by additional enhancement in what is now called Enhanced Probability Convergent Network (EPCN). For example, PCN cannot train/recognize objects of type shown in figure 1.1 (section 3.1) while EPCN can. Two types of EPCN are designed in software and they are employed on a benchmark of unconstrained handwritten numbers. It is expected that the classifiers, taken singly, may be unsuitable for applications that demands higher accuracy.

Secondly, a single classifier will often struggle with large-class classification tasks. These motivate the design of multi-classifier.

In chapter 4, and 5, various classifiers are designed and used for various purposes. The multi-classifiers designed in chapter 4 introduces a combination strategy (i.e. classifier fusion methods) suitable for weightless multi-classifiers. This combination strategy together with a weightless multi-classifier is tested on unconstrained handwritten numerals. The multi-classifiers are unsuitable for very large classification tasks due to bias, and saturation.

In chapter 4, a multi-classifier is designed, and solutions to bias and saturation problems are provided. The target application in this chapter (i.e. chapter 4) is biometric problem domain. Biometric database classification tasks demand very high performance accuracy, and are usually very large databases. High accuracy is required because it deals with issues such as authentication, and identification of individuals. The biometric database utilised here is fingerprint databases. The input, and the training strategy of the optimised multi-classifier utilised is specially planned so as to minimise problems of saturation and biases. Excellent performances are achieved. But very poor performances are also achieved.

This motivates a consideration for, possibly, a new combiner or optimisation of the existing one.

So that in chapter 4, the coding scheme of the combiner is replaced by a better encoding scheme, and again tested on larger classes for which the multi-classifier of chapter 4 fails. In the same chapter, comparisons are drawn between classifier fusion using EPCN and majority voting method. No 0% performances are observed, rather all performances are above 60%.

The speed of software implemented EPCN is determined by sequential execution of function calls, refresh rates, etc. This is to such an extent that, it is difficult to employ EPCN in certain professional areas. This speed constraint might be minimised by considering parallel execution of function calls, pipeline of certain processes, and memory management within EPCN. These scenarios motivate the hardware design of EPCN in chapter 6. Reconfigurable field programmable gate array (FPGA) of advanced type is considered as a suitable hardware. Secondly, since EPCN is adaptive, its re-configurability and/or adaptability are investigated also in chapter 6.

The conclusion, in chapter 7, of the thesis summarises all the works accomplished, and reflect on their merits, demerits, and their application potentials.

There are problems envisaged in the design of these systems as presented below. These problems are however surmountable.

1.4 Major Challenges

The current PCN could only process data of known size explicitly specified to it. And it is required that the number of classes in one training session not to be too large. This makes possible a one-shot learning since most system parameters and database parameters are explicitly specified a priori. In this thesis however, the removal of these constraints is desirable. Since the aim of the thesis is also a design which is generic, adaptive, and reconfigurable, the removal of explicit specification of both systems and database parameters represent a right step in the direction of adaptability and wider applicability. It is challenging to design such a system that identifies an "optimal" database and system parameters to use in learning and recognition. More so of a challenge as to add to this the possibility of high recognition rate in one pass over the input database.

Currently, most weightless classifier converts any mathematical calculation to its equivalent Boolean calculation automatically, and fails completely where the classifier cannot convert the problem to its Boolean equivalent. To widen the scope of mathematical calculation possible for conversion to its equivalent Boolean logic and wherever the weightless neural system fails to convert a given function to its Boolean equivalent, a coding scheme is employed. The coding algorithm is a conversion, a priori, of a given mathematical function to a simpler form suitable for conversion to its Boolean equivalent. Deciding on the best possible coding algorithm to replace a given mathematical function posses a significant challenge.

The advanced nature of the classifier planned demands reduction in execution time as and where possible within the algorithms. The nature of the classifier allows training and recognition in one period (one epoch) of learning, the so-called one-shot learning. Recognition in one epoch is beneficial and posses a significant challenge to this project. The benefit and significance of one-shot learning become pronounced when dealing with large classes and large-class databases. This is lacking in the present-day weighted classifiers.

Pipeline processes is desirable within the functionality of the classifiers, so also is parallelizing of processes as both saves time. The structural impact and significance of pipelining and parallelism is the reduction in overall size of each classifier. It is believed

that utilisation of pipeline and parallel processes may enable the classifier to fit wholly onto the FPGA. It is a big challenge to determine when and where, within the neural networks' functionality and structure, is a pipeline or parallel process required, and if so does it lead to an overall reduction in size of the classifier?.

Other difficulties encountered are the processing of large amount of data, and real-time processing. Real-time processing of data involve a multi-classifier system capable of "online" training, rapid adjustment of its parameters, and capable of handling data stream in a timely manner. Multi-classifier shell is best suited to pattern recognition and interpretation of data. A multi-classifier has (for its input) results from the subsystems and combine these result, in order to produce appropriate responses. The multi-classifier section often starts by introducing classifier in general, whereby specific types of classifier are described briefly. In conclusion, programming challenges in the software and the hardware design are as follows. The major areas where these challenges arise are stated below.

1. Index and indexing: These problems occur mainly at the boundaries.
2. Encoding method to the MCS combiner: The problem here is deciding which encoding method is suitable for a given application.

Programming challenging areas to the Hardware design of EPCN are:-

1) Glitches and race conditions.
2) Memory issues.
3) Time and timing.

It is also a big challenge trying to write such portable code as to fit into Virtex II pro or Single-chip-microprocessor. It seems that testing and verification demands more time and expertise.

1.5 Summary

Boolean logic is simpler and execute faster so that if the conversion is either automatically done by a tool or by algorithmic programming, there will be a gain in time.

Considering that the learning is a one-shot learning and that the epoch of learning can be limited to one without appreciable error, the system thus designed will be very fast as compared to a typical classifier. Projects reported in subsequent chapters will determine if the systems designed based on these concepts are usable and useful both industrially and academically.

In previous sub-section, the decision to parallelize and pipeline as many of the procedures as possible, were explained. This decision should result in design which are economical (monetary or otherwise) and portable. The feasibility of implementation of parallel processes in NNs and MCSs, and resultant benefits are elucidated in subsequent chapters.

2. INTRODUCTION TO ARTIFICIAL NEURAL SYSTEMS

The aims and objectives of this thesis are explained in chapter one. By means of introduction to subsequent chapters, this chapter aims to present current methods in design and application of neural networks.

Neural networks aims to mimic human experts. A human expert is a person very intelligent and knowledgeable in a specific area, and is based on certain number of characteristic outward behaviours. Intelligence is neither proportional to the size nor number of neuron in the brain, nor the biochemical activities going on in a neuron. But all these contribute to intelligence, including the biochemical activities which are all the same in all neurons (names of minerals and extent of activity may differ) of every being. Thirdly, though one neuron is no more intelligent than any other neurons [49], they are nevertheless responsible for sensory perception of beings.

The interaction of neurons within themselves, and with their surrounding, play major roles in acquisition and processing of knowledge and knowledge related information. Although, current research efforts have not given a definition of intelligence, it is widely agreed that an intelligent system possesses one or more of the following characteristics:-

- Ability to invent
- Common sense
- Sensory perception
- Learning and inductive reasoning.
- Pattern recognition and classification.
- Inference from incomplete or approximate information.
- Adaptability to new or unfamiliar situations.
- Display of emotions.

Neural Networks belong to a group of machines called *intelligent machine*. Currently, hardware based artificial neural network (ANN) are also referred to as *Neuro-computers, Parallel distributed processors, connectionist network,* etc. [49]. Neural network, being an intelligent machine possesses many of aforementioned characteristics. The definition of neural network assumed in this thesis is due to Haykins [49], and it re-states:-

A neural network is a massively parallel distributed processor made up of simple processing unit, which has natural propensity for storing experiential knowledge and making it available or use. It resembles the brain in two respects:

1) Knowledge is acquired by the network from its environment through a learning process.

2) *Inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

The procedure followed by classifiers to acquire the knowledge is called *learning algorithm*. In case of weightless (also called RAM-based or N-tuple) neural network, "*synaptic weights*" may be replaced by "*neuron connectivity*". Referring to the characteristics of intelligent, explained in previous paragraph, classifier concerned in this thesis shares many of these characteristics [49], the relevant ones are briefly explained below:-

Sensory perception: This is made by deriving some parameters or values from input pattern or data-set.

Learning (also called *training*).: See section 2.1.

Pattern recognition and classification: Classification refers to how best a given pattern is said to belong to certain class. Recognition concerns a pattern being known. The recognition algorithms of supervised learning involve minimising the error function derived from the actual output and the desired output. An output is accepted after a certain number of epochs or iteration, or when the error has fallen below a set minimum value. In unsupervised learning, classification/recognition is achieved when a winning node achieves a certain value or by simply observing the output after a fixed number of iteration.

Adaptability to new or unfamiliar situation: This is achieved through the specialisation/generalisation capability of neural networks [73].

Ability to acquire knowledge: This is done by high-level processing of information. Enhanced acquisition of knowledge in a specific field is termed *expertise*, hence the term *multi-experts (or multi-classifier)* [49].

Since specialisation and generalisation capability is the backbone of every neural network, a more elaborate explanation of specialisation and generalisation is in order, due to their importance.

Specialisation and Generalisation: Generalisation refers to the ability of neural networks to produce reasonable result for input patterns not encountered during training. Specialisation refers to high-level processing of knowledge acquired in a specific area. Many neural networks having the ability to generalise or specialise [49, 73], are often combined in numbers of forms and termed *multi-classifier (expert) shell*.

This chapter is organised as follows. Section 2.1 introduces weighted neural network, their learning and recognition algorithm, and application areas. Section 2.2 presents an introduction to multi-classifier systems, while section 2.3 introduces many other types of neural networks. Section 2.4 explains methods of hardware implementations of neural systems. While their testing and validation methods are explained in section 2.5. A summary, which concludes this chapter, is found in section 2.6

2.1 Weighted Neural Network

The human brain has great capabilities in processing information and making immediate decisions. This is as a result of a massive network of parallel and distributed computational element called neurons. The linking and interaction of these neurons provides living organisms a very powerful capability to learn. This is very much unlike computers that only implements specific algorithms. Neural Networks (or classifiers) are designed to model these neurons, their linkages, and their interactions. These could be achieved by using electronic components or software. The general procedures involved in modelling of neurons are:-

- Network architecture [73]
- Learning [73]
- Recognition/classification [73]

In early 1940s, W.S. McCulloch and W. Pitts were the first to make a serious attempt at modelling the neuron [106]. This ultimately sparked a series of research into neuron models. The result is the existence of a set of weighted classifiers whose neuronal interconnection, i.e. the synapses, are modelled by weights. The first attempt at neuron models were made by Bledsoe and Browning [10], which does not involve weights or weight adjustment. But rather, certain logic functions will be derived from the problem set, evaluated by the classifier, stored in RAM-memory. This type of classifier is called RAM-based classifier or N-tuple classifier. The main aim of a learning algorithm is to combine the major features of a computer with those of human expertise. A system capable of learning without a guide could acquire and gain knowledge of its own. Two main features are desirable in a classifier.

1) Adjustment of their parameter in response to unpredictable changes in their dynamics.
2) Ability to adjust to a new operating environment.

To (1), Evolutionary methods e.g. Genetic Algorithm (GA) and the like [11] is often used to evolve optimum parameters to suit the changes in dynamics. GA is also used to evolve a new configuration whenever a new operating environment is encountered. Thus learning could always proceed unsupervised and autonomous in any environment. Learning of classifier normally proceeds in one of the following ways:-

- **Symbolic Learning:-** Symbolic learning refers to maintenance of a knowledge base from an operating set of rules. These rules are derived from input data-set and data relating to the performance of the system. A good example is the self-organising fuzzy logic system. [73]
- **Numerical learning:-** This often involve minimising the cumulative sum of errors between the desired output vector and the NNs' output vector. E.g. Back-propagation algorithm.
- **RAM-Based learning:-** Some features of the input pattern is converted to numbers and stored in RAM-memory, or these features are used to form addresses to memory location. A classifier that implements RAM-based learning is simple and learns very fast. It neither depends on guided rules nor is any rigorous numerical analysis involved [7].

The types of learning introduced in this sub-section constitute supervised learning i.e. learning from examples. But in sub-section 2.1.1, unsupervised learning will be introduced.

2.1.1. Unsupervised Learning

All neural networks pass through a process of learning. Having introduced supervised learning in section 2.1.1, it becomes important to introduce the counterpart, which is known as unsupervised learning. Unsupervised learning is a type of learning where the classifier is left to discover pattern regularity within classes and organise these pattern into clusters or categories depending on collective properties discovered. There is no comparison with a target pattern hence it is sometimes called *open-loop adaptation learning.* The result produced at the output is as a result of competition between the nodes of the output layers. At any point in time, the node with the highest value is the winner. The connection weight between the input layer and the output layer are often adjusted in favour of the winning node. Some unsupervised learning schemes exist where, at the output layer, winning nodes and neighbouring nodes' connectivity are

strengthened. This is achieved by employing a neighbouring parameter.

The fuzzy classifiers possess knowledge acquisition system which is capable of deriving *knowledge base rules* from historical data. The fuzzy classifiers also possess the ability to modify its knowledge base and change its configuration without an external teacher – thus an unsupervised learning. Self-organizing fuzzy logic systems represent a good example of unsupervised learning [77]. RAM-based classifier exhibit unsupervised learning when the RAM-locations are modified by the learning algorithm with respect to intrinsic regularities discovered in the input pattern. In most cases, with RAM-based classifiers, features are converted into connectivity and used as addresses for the RAM-location.

2.1.2. Matched-based Learning and other Learning Algorithms

Matched-based learning is based on similarity between the input, and target pattern (or desired output). Match-based learning may be regarded as a template-matching with integrated learning and generalisation capability, and thus able to override noise in pattern. Also once it is trained, a classifier using match-based learning possesses the capability of detecting an incomplete version of pattern or a modified version, and thus a desirable class of classifier. Other advantages over error-based learning are:-

- Easy knowledge extraction
- No catastrophic forgetting
- Fast (one-shot) learning.[19]

Implementation of matched-based learning algorithm is network dependent. It may be feed-forward or recurrent Network. An example is a Hopefield network. A Hopefield network is a topology of recurrent network within which a certain associative (content addressable) memory is formed. The process of storage goes through a learning algorithm called *Hebbian learning rule*. In memory, locally stable states are formed by the outer product of adjacent nodes, hence the memory is called associative memory (association of adjacent weights (nodes)). Thus the memory forms series of locally stable states from any input pattern. These stable states become centres of attraction for any pattern meant for recognition. Hence the associative memory is capable of overwriting a noisy or an incomplete pattern presented to it, by using these stable states. Hopefield statement explain that: “Any physical system whose dynamics in phase space is dominated by a substantial number of locally stable states to which it is attracted can

therefore be regarded as a general content-addressable memory." [54]. Consider a set q of pattern p_k (k=1,…q) presented to the classifier with n number of neuron, then the weight is expressed as

$$w_{ij} = \begin{cases} \frac{1}{n}\sum_{k=1}^{q} P_{kj}P_{ki} \text{ if } i \neq j \\ 0 \text{ if } i = j \end{cases} \tag{2.1}$$

$$w = \{w_{ij}\} = \frac{1}{n}\sum_{k=1}^{q}\left[P_k (P_k)^T - \frac{q}{n} I \right] \tag{2.2}$$

where w = synaptic weight update

$$O_j = sign\left(\sum_{\substack{i \neq j \\ i=1}} w_{ij} o_i - \theta_j \right) \tag{2.3}$$

$$O_i = \begin{cases} 1 \text{ if } \sum_{i \neq j} w_{ij} o_i > \theta_i \\ -1 \text{ if } \sum_{i \neq j} w_{ij} o_i < \theta_i \end{cases} \tag{2.4}$$

After weights initialisation and weight adjustment as in equation (2.2) and (2.3), the activation rule is applied to produce the output o_i (which gives +1 or –1) depending on the value of the threshold θ_i .Hopefield network found extensive application in information retrieval, pattern and speech recognition, and optimisation problems, e.g. the travelling salesman problem [63].

Another NN that employs match-based learning is Adaptive Resonance Theory Map (ARTMAP), and its derivative Reward/Punishement Adaptive Resonance Theory (RePART). ARTMAP consist of two modules, ARTa and ARTb. Input patterns are addressed by ARTa and the target or desired output is addressed by ARTb. These patterns are linked by an algorithm called *outstar learning* [19], in a module called the map-field. A *vigilance parameter* is used to adjust the minimum level of similarity before a pattern is accepted as belonging to a certain class. This is the scope of advantages which this type of neural network has as compared to other types of neural systems. The fuzzy ARTMAP extract its rule in the form of "if-then" from the patterns and match this with its knowledge base. An added advantage of the fuzzy

ARTMAP (variance of ARTMAP) is that it is capable of autonomous learning in a non-stationary surrounding. Fuzzy ARTMAP employs the winner-take-all in its decision. The RePART NN [19] uses the reward/punishment strategy for its decision. These NNs learns by creating a new set of neurons, or RAM-locations are created to store information about new patterns. These new neurons created are called *category neurons*. In case of fuzzy ARTMAP, new rules are created and added to the knowledge base. The ARTMAP NN and its variance have the following disadvantages:

- They are sensitive to noise which may cause category (neuron) proliferation.
- Misclassification of pattern during recall process.

When results from ARTMAP family of NN have been compared with those of error-based learning NNs, it was found that the ARTMAP NNs out-performed those of error-based learning [19] in many cases. ARTMAP NN found application in medical diagnosis. Other forms of learning Algorithms are:

- Conventional algorithm. [39]
- Deterministic algorithm. [14]
- Lazy conventional algorithm. [32]
- Lazy deterministic algorithm. [32]
- Progressive algorithm. [33]

The Goal seeking neuron (GSN) network is a good candidate for these algorithms.
The question as to when does a neuron starts learning and when does it stop is answered in the concept of activation function. This is a topic of sub-section 2.1.3.

Reinforcement learning: - This is a type of learning that includes effects from its surrounding. Reinforcement learning is hereby explained with respect to RAM-based classifier as follows. Weights are set to initial random values. In RAM-based classifier, the RAM-locations are initialised to u or zero (0). The weight connection, or the connectivity (in case of RAM-based classifier) are adjusted according to the feedback from the surrounding and the classifier's output [73]. This is a positive feedback system, since the output nodes with highest value have their connectivity strengthened while those of lower values have their connectivity weakened. This form of learning is also called *graded learning* because the adjustment of connectivity or the update of weights is regulated by the feedback from the surrounding. When excited by its surrounding, a *random search method* is used by the classifier to reach a correct output. Examples of reinforcement learning are Adaptive heuristic critic and the Q learning. Reinforcement

learning algorithm differs from supervised learning in that there is no target pattern present a priori. It also differs from unsupervised learning because it involves feedback from its surrounding.

2.1.3. Activation Functions

An Activation function is an important function required by a neuron for turning on and off its activity. The study of this function is important for the full understanding of learning procedure of classifiers. An activation function is a mapping applied to a weighted sum of inputs. These inputs are supposed to come from other nodes or neuron, and the output of this mapping is delivered to the next neuron. The mapping could also be regarded as *transfer functions.* Prior to application of activation function f, to every incoming signal x, is applied a weight w_{jk}. The weights are then summed, and a threshold θ_K is applied as shown in equation (2.5).

$$O_k = f\left[\sum_{i=1}^{l}\left(w_{ij}x_i - \theta_k\right)\right] \tag{2.5}$$

$$H(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.6}$$

The output o_k (equation (2.5)) of application of activation function goes to the next node. Activation functions have the following characteristics:

- Ability to model the density of joint probability P(X|Y) for X input and Y output.
- Ability to approximate an arbitrary continuous function on a compact domain with arbitrary precision.
- Ability to, in conjunction with other processes, classifies images.

The function used as activation function depends on any of the aforementioned characteristics, and also on dimension. Some names of activation functions are step (Heaviside) function, equation (2.6) above, sigmoid function, bi-radial function etc. These are not multidimensional functions. For multidimensional purposes, activation functions like arc tangent, hyperbolic tangent or multi-quadratic function could be employed. Sums of one-dimensional activation functions have been reported to yield

good performance [49] e.g. Gaussian bar functions, sigmoid bar function. Products of activation functions have also been used. The multivariate Gaussian gives hyper-ellipsoidal output densities. A good activation function should not be trapped in local minima, caught in plateau or oscillate excessively. Activation functions are used mainly in NNs that involve weights and weight adjustment.

In RAM-based NNs, activation functions are not used. But it could be argued that activation function, the Heaviside function, is used since this result in equation (2.5) which is compatible with digital systems. The RAM-location and the LUTs will be modified, read from, or written to according to the result or combinations of result of H(x). This is referred to as address formation or connectivity formation. Otherwise it is the value of the H(x) or its combination that is written to LUTs or RAM-locations directly.

Activation functions are also employed in the process of recognition. The process of recognition is introduced in sub-section 2.1.4

2.1.4. Recognition and Classification

Classification refers to how best a given pattern is said to belong to certain class. Recognition concerns a pattern being known. The recognition procedure that follows reinforcement learning involves the strengthening of connections between a winning node and input layer. Reinforcement recognition procedure involves a method of querying the environment in order to validate its output. A reinforcement learning and subsequent recognition procedure has proved to be closest to human reasoning procedures by interacting with its physical environment.

In match-based learning, the actual output is compared with the desired output for each pattern. Acceptance depends on similarity between the target output and the actual output. Hopefield neural network for example employs content-addressable memory to form locally stable states from any input pattern which is used to overwrite an incomplete or inaccurate version of that pattern when presented to it for recognition. This is useful in error-correction tasks. The RePART neural network [19] on the other hand employs reward/punishment strategy during recognition. The winner node will be rewarded, that is, have the connectivity between them and the input layer strengthened. While the “looser” nodes will have the strength of their connectivity reduced (i.e. punished).

2.1.5. Architecture

Most weighted NNs are arranged in layers which are input, hidden, and output layers each layer ends in a series of nodes, and within each node are a summation, threshold, and activation function. Figure 2.1 shows generalized schematics of a node of neural system.

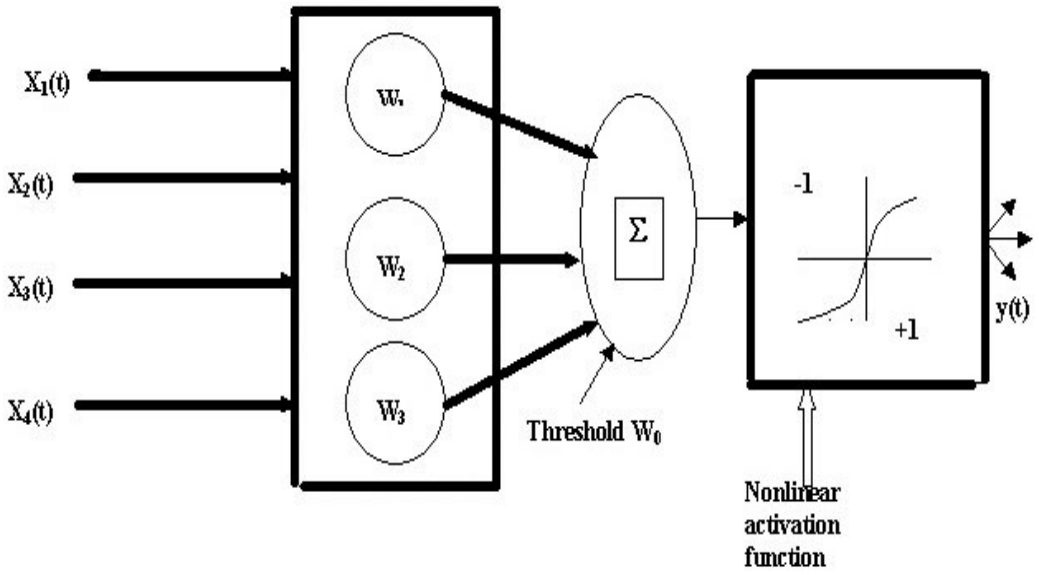


Figure 2.1: The operation at a node of a neural network. Xi(t) – Neural input; Wi – Synaptic weights; i – 1,2,3.... y(t) = Nodal output**. [3]**

The diagram (Figure 2.1) shows an input layer $X_i(t)$ i =1,2,3,… connected to the input pattern. The layer connected to the output layer is called output layer. All layers in between are called hidden layers. The layers are interconnected (either unidirectional or bi-directional) by means of synaptic weights, w_i, classifiers with a unidirectional synaptic weights connection are called feed-forward (FF) or open-loop network while classifiers with bi-directional synaptic weight connection are called recurrent network (RN). The output of feed-forward network is independent of previous output while the outputs of RN are fed back and thus depends on the previous output (or state). Examples of RN are Hopefield network, and time delayed classifier. The supervised learning is also called active learning [49]. Figure 2.2 shows schematics of typical supervised learning. Supervised learning requires the input and desired output pattern being presented to the classifier a priori. During training, the classifier output is continuously

compared with the desired output pattern. The measure of discrepancies between the classifier output and the desired output, the error, is used to adjust the weight connections between the nodes.

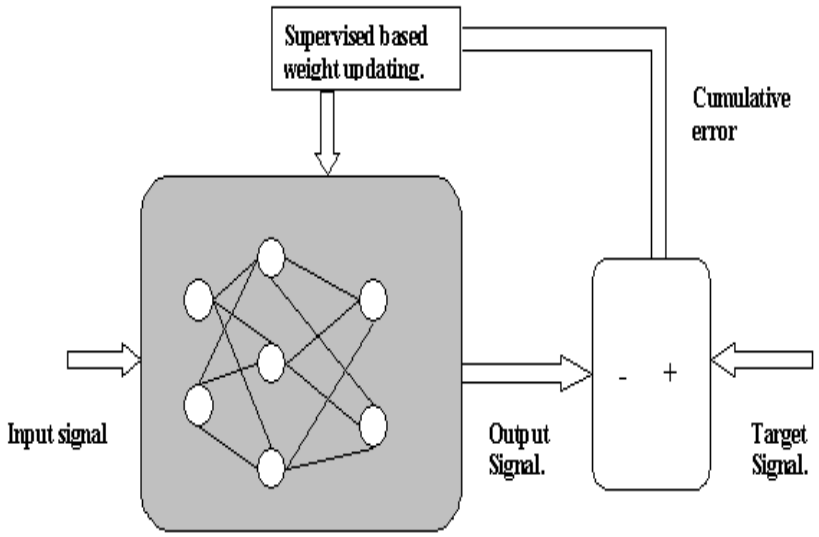


Figure 2.2: A schematic of supervised learning rule.**[49]**

The most commonly used error measure is the mean square error called the error function. Good examples of supervised learning algorithm are the back-propagation algorithm and least mean square (LMS) algorithm [49]. These algorithms use mean square x^2 error, (see equation (2.7) below) to updates the connection weights. Supervised learning is used often in feed-forward NN topology, such as Multilayer perceptron (MLP), and Radial basis function (RBF). Back - propagation algorithms find application in MLP while Least-mean-square method is used in RBF. Though supervised learning exists in fuzzy system and RAM-based NN, it is not a common practice to use error function to adjust weights.

$$x^2 = \sum_{p=1}^{m} \left(y^p - t^p \right)^2 \qquad (2.7)$$

Where y^2 = output of Neural Network;

t^p = desired (target t) output;

and x^2 = mean-squared error.

In some RAM-based and fuzzy NN, features of input patterns are compared with a knowledge base in look-up-Tables (LUTs), or some values stored in RAM-neurons. The

content of the RAM and LUTs are then modified with respect to the target pattern e.g. Goal Seeking Neuron (GSN) [19].

ADVANCES IN ARTIFICIAL NEURAL NETWORK RESEARCH: - research publications in weighted neural network extend both the architecture and the learning algorithms of this section and prevous sections. Such that neural network could now be grouped into the following groups:-

Support Vector Machines (SVM): These are kernel based, and the learning algorithms depend often on distribution such as Gaussian distribution. The most notable and industrially applied example is the Radial Basis Function. Significant research publication in SVM includes Bernad [8], Lopez [80], and these has, in various ways, increase the application arreas. RBF is a good example of support vector machine.

Fuzzy dependent Neural Network: Fuzzy algorithms are often applied independent of a neural network, but current research results have changed the trend to include fuzzy neural network. In Canuto [30], fuzzy neural netwok is designed and applied to character recognition. In Karray [73], fuzzy neural network is applied to PID control scheme.

Bayesian neural networks: These are neural networks whose learning/classification algorithms are based on conditional probability. Though the concept of conditional probability has been around for decades, it is only grace to the current research that great numbers of neural networks has been developed based on Bayesian rule. Notable examples are Beyers [9] and Bocsi [13]. The variation in the individual algorithm and implementation are significant and thus the grouping into a distinct group is necessary.

Percepteron: Backpropagation and percepteron learning is one of the oldest learning method in history. But current research has introduces very many variations and hybrids of Multi-layered percepteron. Some notable publications includes; Sukanesh [127], Lahoz [78], and Nahid [98].

2.2 Weightless Neural Networks

2.2.1 Introduction

Previous sections have introduced weighted neural networks. Weighted neural networks utilises high mathematical functions. Accompanying the utilisation of mathematical function is high memory demand and high resource utilisation. The

fundamental principle behind *n-tuple* network is that pattern recognition may be assumed to be the process of building a set of Boolean logic functions which describe the problem. A standard weightless classifier may be regarded as a discriminator composed of m RAM-based neurons. A discriminator is a 1-to-N decoder followed by storage cells. A summing device following the storage cell of a discriminator completes a weightless neuron.

At initialization, all storage cells may be set to zero. For each training pattern, a "1" is stored in the memory location addressed by the pattern. When learning completes, some memory locations will have been set to "1" by corresponding training patterns, while other locations may remain at "0" or "u" (where "u" denote unknown). The learning record of the RAM-memory will be used to solve previous unseen problems when one is given as pattern. During recognition stage, RAM-memory content addressed by the input pattern are read and summed by the summing device to obtain what is called the discriminator response.

Illustration of a basic weightless neural network follows. When input data is presented to WNN for classification, sub-sets of the Boolean logic function will evaluate to true for a specific pattern class whereas other sub-sets will evaluate true for other pattern classes; thus solving a classification task. A simple example is given below for illustration.

Example Basic Look-Up Table (LUT)
Consider a 3x3 LUT below;

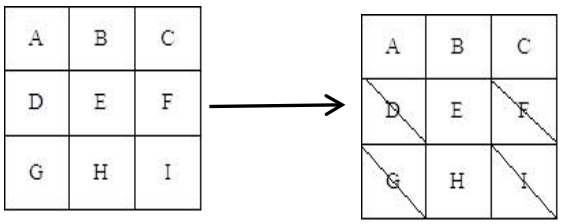


Figure 2.3: This (3x3) LUT can recognize the letter "T"

Figure 2.3 is a LUT of tuple-size = 3. Each class of patterns has a set of Boolean logic function that evaluates to true to indicate the recognition of that pattern class. For "T", the Boolean logic function that may express "T" recognition is given by:

$$f(T) = ABC + \overline{D}E\overline{F} + \overline{G}H\overline{I} \tag{2.8}$$

Equation (2.8) is shown schematically in figure 2.3. Thus f(T) may classify "T" and all

letters that resemble “T”. Similarly f(K) may classify “K” and all letters written as “K”.

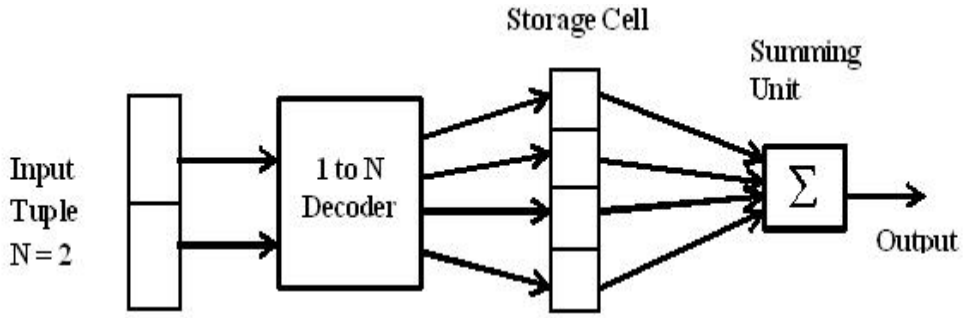


Figure 2.4: A basic weightless neural network.

By using a logical 1-to-N decoder followed by a set of binary storage memory for each term of the Boolean logic equation, each tuple (e.g.; ABC or $\overline{D}E\overline{F}$ of equation (2.8)) require one storage unit. The Boolean logic decoder is able to calculate all possible Boolean function of the N inputs. So that when presented with learning input data, various decoder will indicate which function they have derived from the input learning data. To classify the test/validation data, the test data is presented to the NN of figure 2.4. The NN will access the storage cells and evaluate the Boolean functions that are true for each pattern class. Evaluated values are summed to output. Thus the basic Boolean neural network learns and classifies patterns.

Multiples of discriminators are often employed in parallel to learn/classify patterns, example of which Shefa [116] called m-RAM weightless neural network for handwritten digit recognition. This is the same as in figure 2.4 except that N = m and m may be any number greater than 2. When figure 2.4 is implemented in a microcontroller of a robot for sensor control and/or monitor, we have the scenario of Siti Nurmaini [123]. Though the basic principle may appear simple, it forms the building block of WiSARD [5] architecture with a recent application to change detection by Massimo [92] and deformable objects by Massimo [93].

2.2.2 Probabilistic Convergent Network (PCN)

Many prediction problems and pattern recognition problems can be solved by performing Boolean logic on them. In situations whereby prediction or recognition problems can be interpreted in terms of Boolean logic, a type of random access memory (RAM) based network called Probabilistic Convergent Network (PCN) becomes suitable. An added

advantage of PCN over existing RAM-based network is the inclusion of confidence measure.

To perform a logic mapping, it is expected that all inputs be reduced to binary pattern. Due to the complexity of architecture and function of PCN, it is worth introducing some terminologies which will be used throughout this thesis. They are explained below

Binary inputs: - The Probabilistic Convergent Neuron (PCN) accepts as input, binary images only. Any input data is threshold-binarized and appropriately resized so that PCN may make sense of the data.

Compound symbol: - Symbols are used to denote the neuron output of PCN. The architecture of PCN is shown in figure 2.5 below. Neuron outputs are inherently restricted to a small set of symbols often only "1" and "0". These are set of symbols often called base symbols. To increase the set of base symbols for a neuron, other symbols may be introduced. For example, for numbered classes 0 – 9, one may allow the same number 0 – 9 as possible symbols. This permits the storage and retrieval of multiple symbols consistent with input classes. The symbols thus employed are known as *compound symbols*. Thus for example, a *compound symbol* consistent with class "5" and "6" will be "56". This means that class "5" and class "6" has been presented to the NN. The main difference in neuron output of PCN, as compared to other weightless nets, is the indication of frequency. The frequency at which one class addresses a given location is indicated in PCN and EPCN. For example, for two classes addressing a location, if class "1" addresses this location 75 times and class "2" addresses this location 25 times, the output of the RAM-neuron will be: [75 25]. If class "2" addresses the location 25 times and class "1" addresses the location 75 times, the output will be: [25 75]. Thus PCN and EPCN give the "probability" of occurrence of each pattern.

Adjustment: - For N training pattern and x division, a number "a" occurring in a memory location will be adjusted as in equation (2.9):

$$\hat{a} = a\left(\frac{D}{N}\right)$$

$$\tilde{a} = round(\hat{a}) \qquad (2.9)$$

$\tilde{a} =$ the new value replacing a in that location

This adjustment is necessary to restrict the probability measure of all classes to the number of division that has been set a priori. If the number of training pattern per class varies, classes with large training set would have large probability even when they are

not many in the test set or validation set. Adjustment reduces this large probability to the real value present during recognition. Adjustment is also used to remove rounding errors and truncation errors.

Division: - This is a value set as the sum of all the scaled probability of the classes. The probability of occurrence of the classes is proportional to division. For example, if there are 3 classes and the division is set to 100, then if the output of PCN is, say [50 25 25], the sum of this should equal 100. This result will be interpreted as: The pattern presented to the NN belongs to class "1" with probability 0.5, to class "2" with probability 0.25, and to class "3" with probability 0.25. Notice that the length of vector output from PCN is always equal to the number of classes under consideration.

Merging: - The term merging referred to a group of layers, composed together so as to form one layer. Merging of main-group layers consist of averaging the values in the memory locations with respect to class to form a single compound symbol.

Neuron: - The smallest complete functional information-processing unit in the PCN and EPCN is known as a neuron (see section 3.2 for a detailed discussion).

2.2.3 PCN Network Architecture

The PCN consist of a pre-group, a merge layer for the pre-group, the main-group, and merge-layer for the main-group. A feedback path from the merge layer of the main-group to the main group layers is included in the design. Each group is arranged in layers. Each layer consists of neurons. Each neuron consists of storage locations called the N-tuple locations.

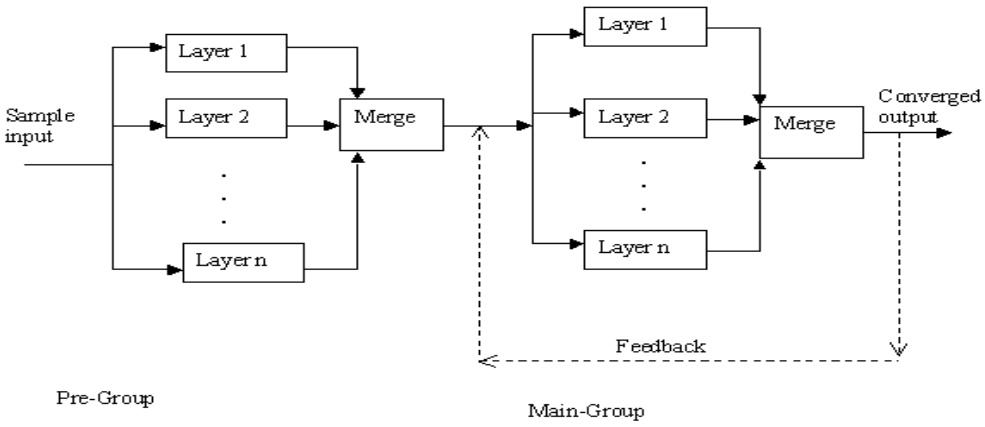


Figure 2.5: A schematic representation of Probabilistic Convergent Network (PCN). This is an example of a RAM-based Neural Networks (NNs). **[55]**

An alternative view is to regard each layer as a look-up table (LUT). The neurons are arranged in (x * y)-matrices where (x* y) represents input pattern dimension. Every element in an input pattern is associated with a neuron in each layer. . A feedback path from the merge layer of the main-group to the main group layers is included in the design as shown in figure 2.5. The pre-group layers are meant for learning while the main-group layers are meant for patter recognition. Pattern learnt during training are merged and stored in pre-group merged-layer such that the main-group layer connectivity can be formed. Results of the main-group layers are merged such that they can be sent to output and feed back unmodified.

2.2.4 Learning or Training

Learning starts when a new pattern is presented to the NN. It is assumed the pattern is thresholded (binary). The procedures are as follows:

- Addresses are formed from input pattern. These addresses (also called connectivity) are used to access neurons within a layer and location within neuron.
- The locations within a neuron are relative to number of classes.
- The size of a layer is relative to the size of pattern (Further information are found in chapter 3).
- Depending on which pattern class an address is formed from, a corresponding location will have its value incremented [83]. Otherwise zero will be stored in the location.
- A normalisation phase followed. This consist of dividing the value in each neuron location by the number of training pattern of a corresponding class, this is multiplied by the number of division. The result is rounded to the nearest whole number as in equation (2.9) of section 2.2.2.
- These whole numbers will be stored in neuron locations of the pre-group.

2.2.5 Recognition or Classification

A recognition procedure is as follows:

- The pre-group layers will be merged into a single layer; this is called pre- group merge-layer.
- Values in the neuron location of the merge layer will be adjusted to make the

"sum-of probabilities" [57] equal to the number of division.

- The connectivity and pre-group merge-layer will be employed in the formation of the main-group layers.
- Adjustment of values in the RAM-location follows.
- Merging of main-group layers gives main-group merge-layer.
- Values in the merge layer will be summed and adjusted to become neural network output. The output may be feed back iteratively into the main-group layers.

ADVANCES IN CURRENT RESEARCH: After the invention of WIZARD [5], and AURA [7], there has been extension and variation of application of pyramidal neuron and Correlation Matrix Memory (CMM). Major researches into weightless neural networks have been mainly extension of the previous one, and application based. Few exceptions do exist as in Howells [55]. Most current research efforts has aimed to expand the areas of applications of weightless neural networks as evidenced in Sirlantzis [122], and Lorrentz [84], where weightless classifiers are applied to biometric and other databases. A future quantum weightless neural network is also "available" in W. De Oliveira [131].

2.3 Multi-Classifier Systems

Multi-classifier systems begin in the early 18th century. A notable invention in the 18th century is the Borda Counts, for combining multiple rankings, named after the inventor Jean-Charles de Borda. Subsequently, a pandemonium was invented in about 1958 by Selfridge. A pandemonium is a learning paradigm whereby a head-demon would select a demon that performs best. Thereafter follows several publications about multi-classifier systems, in which the most notable among these are Kanal [72] and Minsky [94]. Early works on multi-classifier centred on combining multiple models of the same problem.

In the late seventies emerges distinction between models; those which are heuristic and/or statistical, and those that are not. Many more and differing approaches evolved. It has now been discovered that the concept of integrating multiple data sources and/or multiple intelligent system models occurs naturally e.g. combining of estimators in econometrics, combining of evidences in rule-based systems, multi-sensor data fusion, and combining of senses in the human central nervous system. It

has been explained that neural networks are expert in one area and not in other areas. This means that many neural networks are experts in different areas. Could a single system be designed which, given different and varied problem domain, is capable of providing reasonable solution? The design of such a system is significant in that it saves time and resources. The answer to this question is the design of multi-classifiers and is the subject of this sub-section. The development of classifier shell demands human experts in the field of interest. One set of human experts deals with problem representation; they could be engineers, managers, and programmers. They define and model the domain of problems to be solved. Another set of engineers will be involved in design of appropriate expert shell. A "shell" in the sense that the experts should not contain a specific prior solution to a specific class of problem but rather capable of providing solution to various problem sets. Ideally, an expert system should have the capability to learn and continuously update itself [73]. Expert shells are combined in various form to form Multi-classifier Shell (multi = many). There are four main architectural category of multi-classifier shell, they are:

- Parallel system.
- Modular (Hierarchical) system.
- Sequential (serial) system.
- Hybrid system.

These architectures will be explained below.

2.3.1 Parallel System

A parallel system here refers to a case whereby a machine and one or more NNs are arranged to accept input simultaneously, and their output are combined concurrently. A parallel system is sometimes called ensemble-based system. An example of ensemble based system is shown in Figure 2.6 which shows the arrangement of NNs in parallel. It is to advantage if these NNs are as different as possible. The same input signal may be used to excite these parallel NNs.

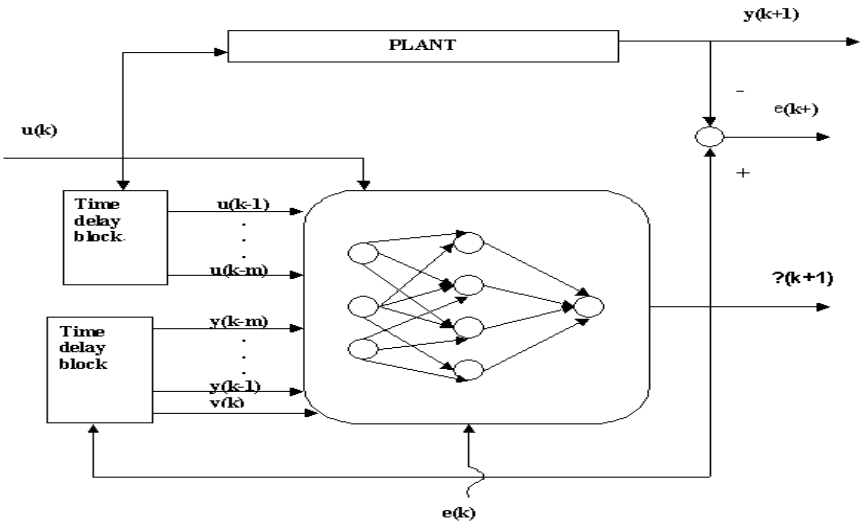


Figure 2.6: Schematic of a parallel Neural Network system. [73]

The output may be combined by:

- Summation and averaging. [20]
- Summation and weighted averaging.
- Winner-take-all approach may be employed.

In a variant called the blackboard system, various inputs go into a global database system called the blackboard. This database is then made visible to all the NNs. Since the NNs are different, different decision is to be expected at their output layers.

2.3.2 Hierarchical System

This is an approach in which input is divided into many tasks, and experts divided into modules or clusters. One cluster of expert is assigned to one task. Modular network is formally defined as follows:

"A neural network is said to be modular if the computation performed by the network can be decomposed into two or more modules (subsystems) that operates on distinct inputs without communicating with each other. The outputs of the modules are mediated by an integrating unit which both (1) decide how the output of the modules should be combined to form the final output of the system, and (2) decides which modules should learn which training patterns" [49]. A schematic of hierarchical system is shown in

figure 2.7. It is possible for one cluster of experts to learn, supervised, while the others learn, unsupervised. A simple form of modular network consists of single NN as a module and a gating network.

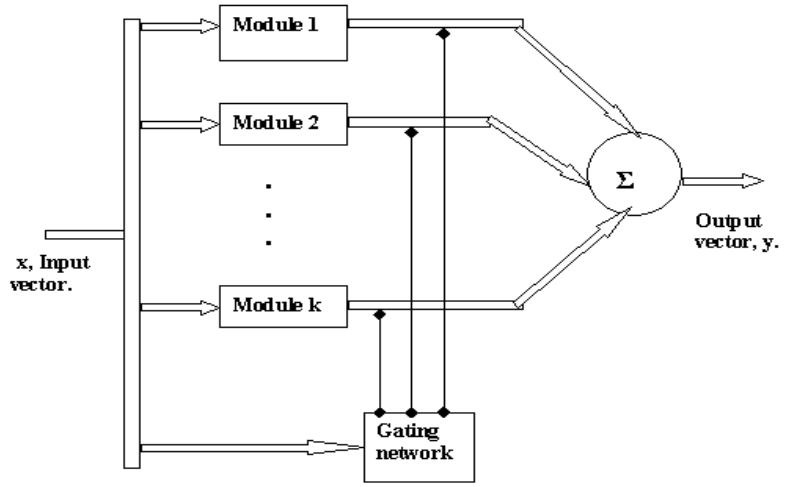


Figure 2.7: Modular network; the output of the expert networks (modules) are mediated by a gating network.**[49]**

Before combination of NNs as modules, experiments are performed to determine their area of expert. And portion of tasks are allocated to each modules of neural network, by the gating network, according to their capability. Just as synapse work by getting the right information at the right time for optimum performance so does the gating network works by getting the right type and amount of training data to the right module at the right time. The gating network receives the error between the actual output and the desired output of the neural network; use this in a feedback system to decide which module of neural network learns which task. Thus the *gating network performs the role of a mediator*. It is also responsible for implementing a combination strategy to the output of the ensemble.

In a complex system, the modules are arranged into hierarchy. And the amount of information to each hierarchy is graded, with the module of the highest hierarchy having the least information resolution. Generally, the NNs in higher hierarchy are “more intelligent” than the ones at lower levels. [73]

2.3.3 Serial System

The serial system, also called sequential system, comprises of linking the output of one

neural network to the input of the other. One type of neural network is used as input layer, one or more types of neural network processes the fan-in from the input nodes, and one type of neural network may be used as the output layer. Serial system is common in Neuro-fuzzy system in what is called *cooperative neuro-fuzzy system*. As shown in figure 2.8

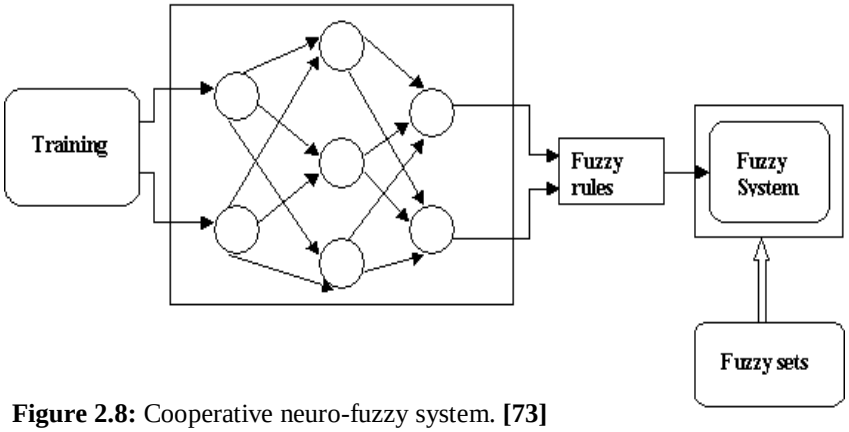


Figure 2.8: Cooperative neuro-fuzzy system. **[73]**

This is a Multi-expert shell whereby the conventional neural network extracts fuzzy sets (membership function) from training data. Fuzzy neural network accepts fuzzy sets as input from conventional neural network; compare it with the fuzzy rules in a rule base (figure 2.8). Other possibility exist whereby the conventional neural network (e.g. MLP) extract fuzzy rules from training data or where the neural network is used for parameter tuning before the input is piped to the fuzzy system. Data clustering techniques is used by neural network to identify and extract fuzzy rules from training data, which is then transferred to the fuzzy system. The fuzzy system implements the membership function and updates its knowledge base. One important sequential system is Neural Network-driven Fuzzy Reasoning (NNDF) designed by Takagi and Hayashi. [128]

2.3.4 Hybrid System

All other forms of combination of neural networks (the neural networks must be independent) that is neither, sequential, parallel nor hierarchical (modular) is hybrid. Various forms exist and could be grouped as follows:

1. Those that have the conventional topology but uses fuzzy neurones at their

nodes.

2. The conventional fuzzy system that employs classical neural network for numerical computations either during derivation of membership function or during derivation of fuzzy rules. This is different from the sequential system, since the classical neural network could be located anywhere within the fuzzy system.
3. There exist a group of classical neural network that employs fuzzy methods to update their weights instead of a learning parameter and sensitivity function (i.e. differential of log-likelihood function with respect to weights).
4. A group consist of fuzzy systems and classical neural network, working independently, and synchronised.
5. A group involve one or more mixtures from the above-mentioned neural networks.

One example of Neuro-fuzzy hybrid NN was designed by Canuto [20]. In this, a fuzzy neural network called RePART, a fuzzy multi-layer perceptron (F-MLP) and radial RAM was used. The RePART neural network is a normal ARTMAP but with reward/punishment process. The fuzzy MLP is a normal MLP but with fuzzy nodes at the output nodes. Radial RAM is a normal neuron but employs a radial region, defined by its Hamming distance from a reference point in its training and recall phase. The final output is then compared with a radial region defined by Gaussian distribution.

Another important example is the Adaptive-Network-based Fuzzy inference System (ANFIS), proposed by Jang [67]. ANFIS is a Sugeno-type fuzzy system. The commonest ANFIS system is a first- or zero-order Sugeno system.

CURRENT RESEARCH MILESTONE IN MCS: - Research publications on multi-classifier are increasing with notable methods of combinations. Breiman [16], [17] utilizes decision tree as base “classifiers” with boosting, and refers to decision tree multi-classifier system as the most significant development in classifier design in this decade. Refering to classifier diversity and biases, Gemam et al. [44], and Mitchell [95] maintained that different types of classifiers have different types of “inductive biases”. The combination of base classifiers has witness sequences of development - from averaging [108], to majority voting by Bodgan [12], to using special techniques and/or function by Gunter [119]. The most advanced stage is the usage of other classifiers for combination [109]. Using classifier for classifier fusion is termed *intelligent combination*. The methods used in this thesis belong to *intelligent combination* method.

2.4 Hardware Implementation

Neuron was first implemented by McCullogh and Pitts in 1943 when it was represented by a threshold-logic unit. There after begin hardware development when, in 1959, Bledsoe and Browning developed the first weightless neural network. Following it is the RAM-neuron developed by Aleksander in about 1979. There was a brief silence in hardware neural system development until the mid-eighties, when it was realised that the massive parallelism inherent in neural network models could be utilized to profit only by implementation in hardware. This led to industrial development of various neural systems in hardware. For example WISARD was developed from RAM-neuron and in 1986 marketed; AURA was developed by Austin, etc. The early hardware development also spread to other neural systems such as MLP, RBF, etc. The hardware platforms utilized also vary from digital, analog electronics, optical, to hybrids of these platforms. Neural network research became widespread in the mid-eighties when it was realized that the massive parallelism inherent in neural network models promised great advantages which is realizable only when implemented in hardware. This has given rise to variety of hardware implementations ranging from digital and analog electronics, optical, to hybrid techniques.

Most ANNs are implemented in software. However a hardware implementation offers considerable advantages over software implementation. These are:

Speed: - Pipelining and parallel execution of instructions is faster than sequential execution. Pipeline instructions are more associated with hardware implementation of ANN. Concurrent implementation of both pipeline and parallel instructions are possible in hardware but scarce in software.

Cost: - In high-volume applications, hardware implementation will provide overall reduction in system cost, by reducing total component count. Total component count is reduced in integrated systems.

Reliability: - The decision (output) from hardware neural network is more reliable when concerned with large input data that involve large amount of computation. Building of fault tolerance into a neural network system is easier done in hardware. The performance of software neural network (e.g.; speed) is dependent on hardware computer on which it is installed.

Property protection: - Hardware offers protection against "reverse Engineering" which could be made use of by competitors. The protection offered against "reverse Engineering" may or may not be effective. Decryption and decoding is always possible with software based neural networks.

As components, hardware neural networks are available in different forms. These include embedded microcomputers, Neuro-computers, Cell libraries, chips and PC accelerators. Hardware NN implementations are divided into three main categories. These are:-

1. Digital implementation.
2. Analogue implementation.
3. Hybrid implementation.

The advantage of software NN is:-

Flexibility: - Software neural network could be implemented on any general purpose computer. And is generally advisable to do so during experimentation of a new technology or/and a new neural network. Low-volume applications software neural network offers (1) considerable advantage in terms of consumption of resources; (2) more possibility of parameter tuning and dynamic reconfiguration. For high-volume application however, software neural network is unsuitable due to decreased precision and long execution time.

2.4.1 Digital Implementation

Digital implementation of NN are characterised by having all values represented in binary word length. Exact precision values and operations on values are made easy. Weights and coefficients stored in RAM do not need to be refreshed and are free from noise. Since inputs from the real world are analogue in nature, converting this to digital signal may lead to distortion, and loss, e.g. during quantisation. The followings are different method of digital implementation of NN:-

A) **Random Access Memory (RAM) based NN: -** When used to classify patterns not used during learning, neural networks tend to generalise. Depending on a specific NN, there are various variations of the learning process. The Probabilistic Convergent Neuron PCN, outputs a graded response due to decision reached from the main group. Sample hardware implementations of

weightless NN include WISARD, designed by Alexander and Stonham [3]. A specialised processor, the C-NNAP, has also been designed to implement Advanced Distributed Associative Memory (ADAM) in parallel [74]. ADAM hardware implemented, applies the Generalised Hough transform (GHT) to inputs, e.g. document images. Other weightless hardware implementation includes pRAM, GSN, etc.

B) Slice Architecture: - These are building blocks for NN of arbitrary word length and size e.g. Neuralogic NLX-420, Neural Processor, MicroDevices MD 1220, Philips LneuroChip.

C) Multiprocessor Chips: - This involves having many simple processors on a single chip. Multiprocessors have two method of operation; one is SIMD (Single Instruction, Multiple Data). Here all processors work in parallel, executing the same instruction but on different input data. The other is systolic arrays, the processors operate sequentially, and one step of an instruction is performed by a processor before passing it to the next in the array. Examples of SIMD chip are Inova N64000, HNC 100NAP, Siemens MA-16.

D) Radial Basis Function (RBF):- This involves defining and storing regions of influence or attractions around training input data using basis functions. RBF will often define hyper-surface around points of influence. Commercial RBF are IBM ZISC (Zero Instruction Set Computer) chip and Nestor Ni1000 Chip.

E) Other digital designs: - Other digital designs are those that could not be grouped as belonging to any of the groups above. Examples are: Micro Circuit Engineering MT19003 NISP (A multilayer Perceptron), Hitachi wafer Integration Chips (Hopefield Network).

2.4.2 Analogue Implementation

Analogue neural networks are those neural networks, in hardware implemented, which employ other alternative means for storage apart from random access memory. Information is not explicitly stored in 1's and 0's. Information is stored in charged capacitances most of the time. For optical neural networks, information is stored in light intensities.

The commonest problem associated with analogue neural network hardware is system noise. System noise is more pronounced in analogue neural network as compared to any other hardware alternatives, and it causes limited accuracy for the network. Secondly, the components of analogue neural network (electronic or optical) are non-uniform. This

arises mainly from the fabrication process and the operating condition. Most learning algorithms are implementable in analogue, and does give reasonable results. The algorithms are, in most cases, discretised, and derivatives are replaced by a suitable approximate equivalent. Most difficulty encountered in implementation of algorithms, in analogue, is due to representation of non-linear functions. A good example is the use of Heavyside function in place of sigmoid function. Another example is the replacement of the Gaussian function in radial basis function network by a triangular function.

The implementation of neural networks in analogue has beneficial effects. Beneficial effects may motivate design of neural networks. Of these benefits, the most important for hardware analogue implementation of neural networks over all other alternatives are:-

1. Real-time processing: - The processing of information is real-time. Intermediate storage is not essential for its functionality per see during information exchange between the neural network and its surrounding, but is an advantage. Thus real-time processing of information by this type implementation is inherent.
2. High density of NN: - Many components are multi-functional e.g. filters. Multi-functional utilisation of component saves resources. If there is little or no intermediate storage required, then the system could be very compact.
3. High speed: - The possibility of real-time processing and parallel processing increases its speed considerably.

But the difficulties to be surmounted are:-

1. Problems of reliability and accuracy: - Variation in operating conditions, such as changes in temperature, thermal noise, etc., changes the tolerance of circuit components. This in turn makes many components unstable and may change "weights" stored in capacitors.
2. Problems of consistency of weights: - It is difficult to store charges in a capacitance without changes. Charges stored in capacitances represent the weights. The capacitances need to be refreshed periodically to avoid loss of weights.

2.4.3 Neuromorphic design

Neuromorphic refers to circuitry designed which closely emulates biological neuron. The function ranges from classification to being used as sensor e.g. silicon retina, synaptic touchpad. The Pulse Coupled Neural Network (PCNN) is an example of a neuromorphic neural network. A common biological model of neuromorphic neural

network is a model of the cortical column. This follows from the fact that the brains' cortical column is mainly responsible for information processing in the cerebral cortex. The cerebral cortex consists of neurons which vary slightly in anatomy. It is the interconnection between the neurons that plays a vital role in learning. A neuron model often used in Engineering is known as the Hodgkin and Huxley model [52]. The Hodgkin and Huxley model of a neuron is characterised by membrane potential V_{mem}, potassium ionic current i_k, sodium ionic current i_{Na}, leakage current i_{leak}, and a modulating current i_m. These currents are voltage V dependent. The time dependent equivalence of events at a synapse is described by a concept of spike timing-dependent plasticity (STDP). This describe the spike train (or the waveform against time) of an event at a synapse.

Hardware *neuromorphic design* of neural networks in analogue implementation is very promising because this has the capability to mimic the biological neurons and synapses. E.g. Intel 8017 ETANN [101]. In a proton-antiproton collider at Fermilab Tevatron, Intell ETANN chip is employed in the classification of energy deposited in a calorimeter as either from electron or from gamma rays.

2.4.4 Hybrid Design

Hybrid design aims to combine digital and analogue methods. External communications (excluding input sources) and weight storage may be done digitally (apart from source) while signal processing is in analogue domain. Bellcore CLNN-32 Chip performs *simulated annealing* using analogue circuitry. The *simulated annealing* schedules store weights coefficient in the digital domain. Other examples are Neural Semiconductor Chip set comprising SU3232 synapse unit, the NU32 Neurone unit, and Ricoh RN-100.

2.4.5 Comparison of Implementation Practises

In hardware, NNs are implemented in analogue or digital. The analogue implementation demands for reference voltage. Reference voltages are difficult to maintain. Analogue implementation has a very good performance and low cost. Once built, the architecture is fixed, therefore suited only to one type of target task. Whereas the embedded system is more robust and reconfigurable, this involves the use of

software and hardware. The digital system is divided into ASIC and FPGA. Like the analogue, the digital ASIC is fast to implement, has fixed architecture and very good performance. The disadvantage being that it is only suitable for one type of problem. The reconfigurable FPGA has lots of attractions. In addition to having all the advantages of analogue counterpart, its architecture and system parameters are reconfigurable at any time. A reconfigurable FPGA may be slower than a "corresponding" ASIC because of extra time required for modification of system parameters. Digital implementation of NN does not support floating-point arithmetic, thus runs the risk of non-convergence, and wrong output. Software implementation is low cost, possibility of high precision, compact, less tedious, and the problem of non-convergence, and inaccurate output could be adequately addressed. NNs could be implemented using any programming language; among them are Matlab and C.

2.5 Methods of Testing and Validation

By images we mean any picture, character, or number written, painted or captured (in camera) by man or machine. All images are processed by some functions to scale them to manageable size. Afterwards follows the binarisation procedure that renders a binary image. Once they have been converted to binary images, which are regarded as patterns (or set). Group of similar patterns are grouped into one class e.g. unconstrained handwritten character "2" written by different people all fall in the same class. For neural network that depends on weight adjustment during training, conversion to binary images may not be required. Generally, patterns are divided into three parts:-

A) Training set: - These are patterns used for training. Training sets are often selected as representative of a class of object.

B) Test set: - These are pattern which were not used during training and which the NN is expected to generalise to. Test set is often used, during recognition phase to obtain an unbiased estimate of the generalisation error. Generally, this set will be chosen from a population of a class randomly.

C) Validation set: - Validation set may be the same as test set or be different. This set is often used to determine the suitability of NN for any task. And is therefore used to test how robust the NN is.

D) Measure: - For patterns without cross-correlation, the percentage of correct classification could be used as performance measure. This is the case in most of the

projects treated in the thesis. But generally, for an output y, and a target output t, the sum of squared error is often used as measure of performance. The sum of squared error is defined as:

$$\xi = \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 \tag{2.10}$$

where y_n = actual output;

t_n = desired output;

and ξ =sum of squared error.

If the target variable is binary, for a single output y, we use *Bernoulli random* variable to represent the conditional probability of equation (2.11):

$$P(t \mid x) = y^t (t - y)^{1-t} \tag{2.11}$$

where P(t|x) = posterior probability;

y = output;

and t = target coding scheme.

Taking the negative logarithm and summing yields the *cross-entropy error* function given in equation (2.12):

$$E = -\sum_{n}\left[t^n - \ln(y^n) + (1 - t^n) - \ln(1 - y^n)\right] \tag{2.12}$$

Where n = number of pattern;

y_n = actual output;

t_n = desired output;

E = sum of errors raised to power of n.

For example, for a target class one t = 1, for class one, and t = 0 for the rest classes.

E) Model Testing: - Model testing aims to investigate how *system parameters* affect the performance of NNs. For NNs with weight adjustment, this may refer to the learning rate, biases and decay terms. For weightless NN, specifically for PCN, this refers to the number of layer in the pre-group, the number of layer in the main group, the division, and the connectivity pattern. Here, the graph of percentage recognition versus pre-group, main-group number of division, and connectivity may be used as performance criteria.

F) Test for Application: - The aim of testing for application is to obtain an unbiased estimate of the generalisation error. Cross-validation and Bootstrapping are both methods used to obtain an unbiased estimate of the generalisation error. The process is

as follows:-

- Divide the pattern into m subset.
- Train the net m times. Then, after every training period, leave one or more patterns out.
- Use the omitted pattern to compute the generalisation error.

Since the number of unclassified and misclassified patterns is under consideration, leave-v-out cross-validation is more suitable, where v is an integer greater than one. A suitable error function should be capable of processing discontinuous cases. Cross-validation v, is approximately;

$$v = n\left[1 - \frac{1}{\log(n) - 1}\right] \tag{2.13}$$

where n = number of training times;
v = number of patterns to be left out.

Sub-samples for training will be selected randomly without replacement. Few patterns per class may be employed for learning.

2.6 Summary

The current state of the art in classical neural systems has been introduced in general terms. This is followed by weighted neural systems. Their implementation and application in a multi-classifier is introduced. Also a class of neural network, the fuzzy neural network, is introduced as currently and industrially been used. An introduction to their functionality and application were provided.

Afterwards, the other type of neural network called the weightless neural network is introduced. The state-of-the-art learning algorithm, implementation, and applications were explained. This is followed by introduction to PCN. The current state of work on PCN and its current functionality is introduced.

In the next chapter however, novelty (ies) in the architecture and functionality of PCN will be identified. The novelties may lead to realization of PCN potentials, thereby making it more beneficial than other existing or similar networks.

3. THE ENHANCED PROBABILISTIC CONVERGENT NETWORK - EPCN

This chapter presents a novel adaptation of a weightless neural network entitled the Enhanced Probabilistic Convergent Network (EPCN). This work is motivated by the need of PCN to improve its performance and widen the problem domain on which it may be applied. One of the problem domains is handwritten characters. For this reason, the input mapping methods of EPCN will be enhanced and tested on handwritten numerals. The EPCN possesses the ability to associate a relative probability with each candidate class when a test pattern is presented for classification. The relative probability measures the certainty that a pattern meant for recognition belongs to a class with the highest probability measure. Two distinct types of EPCN are presented; one is termed rand-EPCN and the other fix-EPCN. The rand-EPCN employs random selection of bits within the input patterns to form connectivity, while fix-EPCN uses consecutive bits within the input patterns during connectivity formation. These EPCNs are contrasted.

3.1 Introduction

This chapter proposes some major modifications to the customary PCN. These modifications concern the input mapping method, the introduction of input scaling of patterns, and image processing possibilities. The customary PCN employs a static method for formation of connectivities, while the methods of connectivity formation here is dynamic. Possibility of input scaling has been introduced which enhances the portability of the whole system. Programs for image conversion to binary image have been introduced. They are employed on figure 3.1(a) resulting in figure 3.1(b) as an example of its usefulness. Figure 3.1(b) is the form acceptable to EPCN. Thus input methods to PCN has been modified and incorporated into EPCN to support other sources and types of data. Any other forms, e.g. JPEG, MPEG etc is automatically converted (thresholded) to binary image before being presented to EPCN's input. A sample image, e.g. hurricane Rita (Figure 3.1(a)) will be compressed and converted, using a function, to a binary image as in figure 3.1(b).

Figure 3.1(a) A satellite image shows Hurricane Rita taken at 10.15 a.m. CDT Sept. 24, 2005, as it plowed through Texas and Louisiana

The sea body (marked “S” in both figures 3.1(a) and 3.1(b)) is converted to “1” essentially in figure 3.1(b). Land and green vegetation (Example is marked as “L” in both figures 3.1(a) and 3.1(b)) is represented by”0”. There are regions of mixed “1” and “0” in figure 3.1(b) representing mixed vegetation in figure 3.1(a).

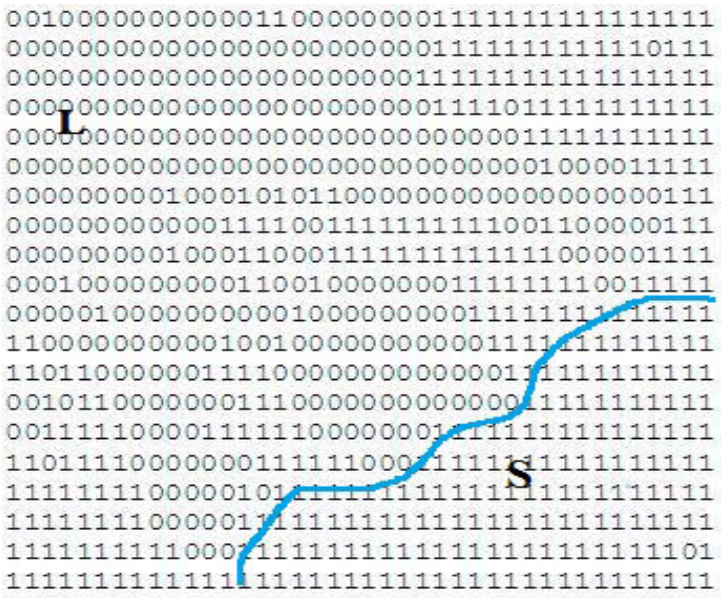


Figure 3.1(b): A binary image of Hurricane Rita.

The EPCN is a classifier which gives a confidence measure to all classes, based on supervised learning, when a pattern is presented to it for classification. Two types of EPCNs are implemented; one is called rand-EPCN and the other termed fix-EPCN. The major differences between the EPCN designed by the thesis author, and the customary PCN [55] are:-

- The possibility of adjusting and rescaling any input pattern.
- Formation of connectivity by using consecutive bits within input pattern coupled with rejection criteria.
- Random selection method of address formation. This method is the customary method of connectivity formation but with the exception that coordinates are functionally initialized and dynamic.
- Improved system interfacing: For example, EPCN can learn/recognize pattern of type shown in figure 3.1(b), while PCN cannot. This is because the input interface of EPCN is adaptive and can sense which pre-processing steps may be required on the input pattern.

From henceforth, and because of these modifications (enhancements), the PCNs are known as EPCN – Enhanced Probabilistic Convergent Networks.

3.2 The Input Pre-processing

The input pattern pre-processing into binary pattern will be explained in general perspective because pre-processing procedures follow similar sequence for most data sources. A holistic processing method is hereby presented.

1) Noise filter is applied to minimize the effect of noise.
2) Edge enhancing filter id applied
3) A threshold is determined for binarization
4) Pixels in patter below the threshold will be converted to “0” whereas those equal to or above the threshold will be converted to “1”.

Figure 3.1(a) is an aerial picture of advancing hurricane. As the composition of water-to-land aerial image changes, figure 3.1(b), the counterpart to figure 3.1(a) also changes in unison. The pre-processing procedure is applied to benchmark databases (known as

CEDAR and NIST). Example of handwritten digits from CEDAR is shown in figure 3.2(a) while samples of its binary pattern are shown in figure 3.2(b). CEDAR and NIST databases are employed in the experimental section of this chapter.

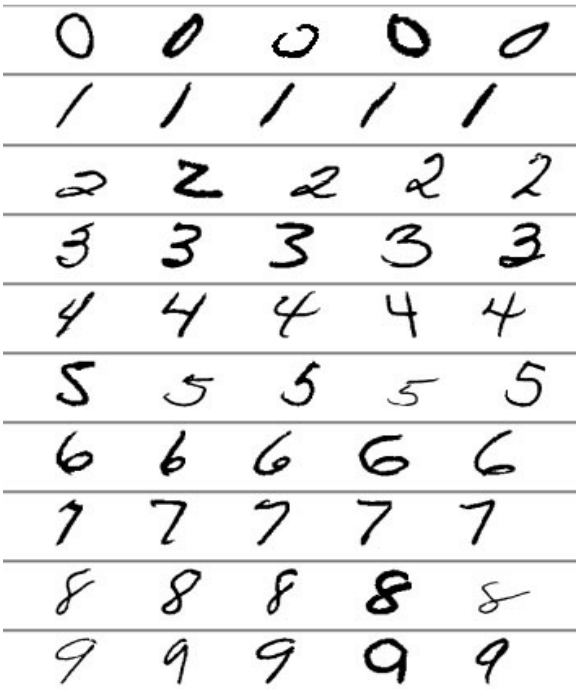


Figure 3.2(a): An extract of handwritten digits from a benchmark database known as CEDAR.

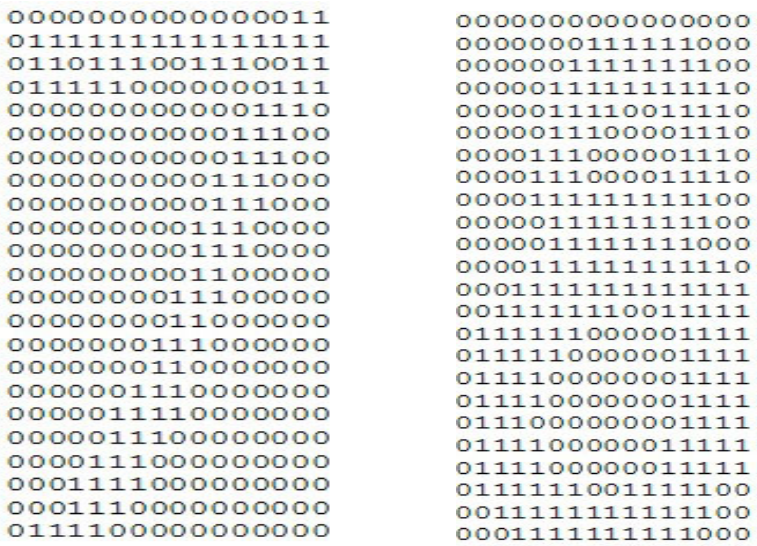


Figure 3.2(b): A binarised handwritten digits from a benchmark database known as CEDAR.

3.3 EPCN – The Enhanced Probabilistic Convergent Network

Weighted Neural Networks are those Neural Networks whose modification to system parameters and performance depend on weights and weight adjustment. On the other hand, Neural Networks whose performance and system parameters are independent of weights (and their adjustments) are called weightless Neural Networks or RAM-based Neural Networks [7]. One of the advantages of a weightless Neural Network is its fast learning algorithm, of which the EPCN is an example. The EPCN consist of neurons which are arranged into layer. The architecture of EPCN consists of two groups of layers. The group of layers utilized during the training process is known as pre-group layer. The group of layer utilized mainly during the recognition process is called the main-group layers. The architecture of EPCN consists primarily of these four component layers termed the *pre-group*, a *merge-layer* for the *pre-group*, the *main-group*, and *merge-layer* for the *main-group*. It incorporates a feedback path from the merge layer of the main group to the main-group. Each group of the layer is made up of a number of layers with each constituent layer consisting of component neurons (defined in section 2.2) which themselves consist of a number of storage locations known as RAM-locations, as shown in figure 3.3.

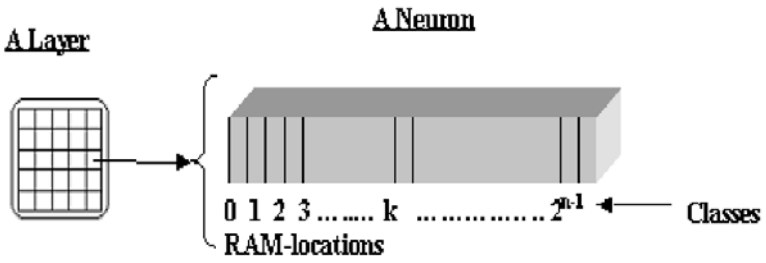


Figure 3.3: An EPCN neuron.

Each storage location itself is divided into separate values for each pattern class under consideration for the neuron. An alternative view is to regard each layer as a look-up table (LUT). The neurons are arranged in (x * y)-matrices where (x* y) represents the input image dimensions. Every element in an input image is associated with a neuron in each layer. The EPCN possesses a learning algorithm which percolates the pre-group layer and end in a merged layer for the pre-group as depicted in figure 3.4. It also has a recognition algorithm which percolates both the pre and the main-group layers, ending in a

merged layer for the main-group. During learning and recognition, an integer number called *division* is required for *adjustment* purposes. The term *adjustment* refers to multiplying values in a RAM-location by *division* and dividing by the number of pattern per class. Two types of EPCN are implemented; one is called rand-EPCN and the other termed fix-EPCN. A comparison is presented below:

Table 3.1: Comparison of fix-EPCN and rand-EPCN.

fix-EPCN	rand-EPCN
During connectivity formation, random number generator is not used.	During connectivity formation, random number generator is used. Hence the name rand-EPCN.
Consecutive bits from input pattern (number of bits depends on pattern) are used for connectivity formation.	Random selection of bits within input pattern is used during connectivity formation.
Exactly same connectivity can be reproduced i.e. for the same image and same system parameters, the connectivity is fixed. Hence the name fix-EPCN.	For the same image and system parameter, different connectivity will be produced. Formation of same connectivity is probable.

The functionality of the architecture depicted in Figure 3.4 is divided into two procedures called *Learning* and *Recognition* procedure.

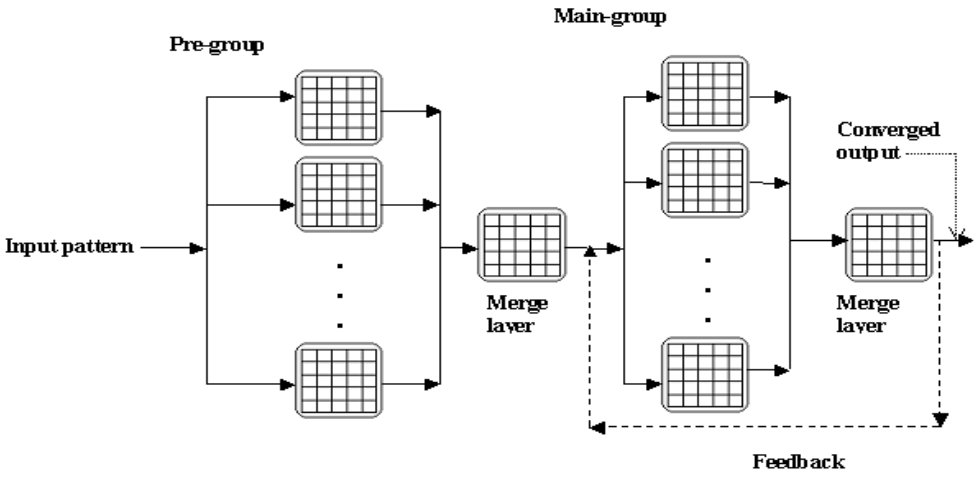


Figure 3.4: A schematic representation of EPCN.

Both learning and recognition algorithms are now presented.

3.3.1 Learning procedure

1) Only the *pre-group* layers will be trained for a given pattern class.

2) For each neuron in the *pre-group* layer, an address is formed from binary threshold input pattern, governed by the given connectivity pattern for the layer.

3) Depending on the address so formed, the respective RAM-location is incremented for the given pattern class. Let the input pattern be $X_i = x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}$ Let the set of addresses [m, n][p, q][r, ..., s] be required; where [r, ..., s] depends on size and dimension of input pattern. The set of addresses are derived from input pattern X_i feature vectors, or pattern attributes. It is employed to access a given memory location, M. Depending on the address, A, formed, (where A = [m, n][p, q][r, ..., s]) a corresponding memory location will be modified as given by equations (3.1) to (3.3):

$$\left(M_A\right)^{tc+1} = \left(M_A\right)^{tc} + 1 \tag{3.1}$$

when A is activated.

$$\left(M_A\right)^{tc+1} = \left(M_A\right)^{tc} \tag{3.2}$$

when A is not active

The equation above is an iterative sequence of tc = 1, 2, 3 , ..., ∞; where tc is the learning cycle; t = time, and c = a constant. In practice, the iteration is limited by the constraint given by:

$$\left(M_A\right)^{tc+1} \leq D \tag{3.3}$$

where D = the number called *division*.

4) Subsequent to the completion of training, an *adjustment* phase occurs to normalize the natural number in each memory location. For N training patterns and D division, a frequency of occurrence of a value “a” in a memory location will be adjusted as:

$$\hat{a} = a\left(\frac{D}{N}\right) \tag{3.4}$$

$$\tilde{a} = round(\hat{a}) \tag{3.5}$$

This adjustment is necessary to restrict the probability measure of all classes to the number of division that has been set a priori. Also if the number of training pattern per class varies, classes with large training set would have large probability even when there are not many examples in the test set or validation set. The learning algorithm utilizes the pre-group layers which are merged to produce the merge layer for the pre-group. Similarly, the recognition procedure utilizes mainly the main-group layers which are then merged to produce the main-group merge layer. After the Learning procedures terminates, EPCN is able to recognise similar objects and patterns. This is done by employing Recognition procedures. Other views about the training procedure may be found in [57]. The recognition procedure is now presented.

3.3.2 Recognition procedure

The Recognition procedures for EPCN are as follows:-

1) An Address is formed for each neuron, within the *pre-group* as for training. The address formed is from input pattern and corresponds to the connectivity patterns of various layers.
2) The *main-group* layers will be merged to form one layer. Locations within the neuron of this *merge-layer* contain independently calculated averages from corresponding locations of the *main-group* layers.
3) After merging, an adjustment is required to make the “sum of probabilities” [57] equal to the number of division.
4) The output of the *main-group merge-layer* is fed back iteratively, a fix number of time, or until the solution stabilises, whichever happens first.

3.3.3 The PCN Software Implementation

In this sub-section, the PCN functions, PC configuration, and Matlab configuration on which EPCN was prototyped will be introduced. The software modelling of EPCN employs Matlab because of its availability, costum functions (e.g. sin, cos, plot, etc. functions) exist already, and because it is more suitable for engineering prototyping. As compared to alternative modelling software, matlab require less effort in order to produce simulations. The Matlab is the medium of software implementation. In Matlab, help about a PCN functions is obtainable for individual function by typing:

>>help function_name

on the command-line. Here, function_name is the name of an EPCN function under inquisition. These functions are written in Matlab, with the usual Matlab's function naming method i.e.

function ans = function_name(variables)

as the first line in the M-file. After that, important constants are specified, which followed by the algorithm which the function implement when called. It is often the case that one function calls another. This maintains interrelationship between PCN functions, analogous to the synapse between neurons. The Matlab used in this case was installed on a PC situated in the Digital Research Laboratory.

3.4 Experiments and Analysis

Experimental Data

For EPCN to be able to learn and recognise objects and images by following the procedures above, pattern and images are first binarised. A binarised image serves EPCN as experimental data for training and recognition. These experimental data used here are handwritten digits "0" to "9", and letter "T" and "I", binarised, and come from sources listed below. The sources are:-

- The Centre of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York, Department of Computer Science. Handwritten numbers from CEDAR were resized and binarised to 16-by-24 in dimension.

- National Institute of Standards and Technology (NIST) in Gaithersburg USA. NIST provide the handwritten simple form (HFS) of numerals, which were binarised and resize to 32-by-32.

The pattern used from CEDAR and NIST are handwritten number “0” to “9”. These are thus named class 0 to class 9. The number of patterns in each class varies from 200 to 1000 depending on class. These numerals are divided into training patterns and test patterns. Training patterns and test patterns are stored in different directory. Test patterns do not form part of training patterns and vice versa.

Statistical analysis reveals that the errors incurred are negligibly small when sample size [96] equal to or greater than 30 patterns were used from each pattern class. This applies to both the test and training classes. Thus in this work, sample size between 30 – 50 patterns per class will be used.

--- --- ---

The aims of this experiment are to evaluate the networks, and to investigate the effects of changes in system parameters with respect to performances. The first two databases mentioned are more relevant to these tests since they are complete, large, and benchmark databases. These databases are independently collected from the society. They represent unconstrained handwriting of various individuals. Each hand writing is independently collected. Thus these databases correspond to real and natural tests for EPCN on unconstrained handwriting of numerals. Simulations were performed on EPCN using the experimental data detailed above. The training patterns from experimental data are supplied to both EPCNs at its input during learning while the test data, also from experimental data, are supplied to the EPCNs at its input during recognition. This is done during all the experiments. Three types of simulations were performed on fix-EPCN in order to determine dependence of percentage (%) recognition on *pre-group* layers, *main-group* layers, and division. For rand-EPCN, variation in connectivity pattern is not measurable when division is varied. For this reason, two types of simulations were performed on rand-EPCN; these are variation of % recognition with respect to *pre-group* layers, and variation of % recognition with respect to *main-group* layers.

Previous works on EPCN detailed in [55], hypothesised that the major causes of variation in performance of EPCN depends on the group layers, number of division, and

the connectivity pattern of layers concerned. The experiments detailed here aims to verify these hypotheses.

1. In the first experiment, the number of divisions and the number of *pre-group* layers is held constant while the number of *main-group* layer is increased from approximately 2 to 9. Percentage (%) recognition is used as performance measure. The percentage recognition is recorded after every increase in the number of *main-group* layer. This is done both for fix-EPCN and rand-EPCN.
2. In the second experiment, the number of divisions, and the number of *main-group* layers is held constant while the number of *pre-group* layer is increased from approximately 1 to 11. Percentage (%) recognition is used as performance measure. The percentage recognition is recorded after every increase in the number of *pre-group* layer. This is done both for fix-EPCN and rand-EPCN.
3. To investigate the dependence of performance on division, the number of *pre-* and *main-group* layers is made constant while the division is varied from approximately 100 to 1000. Values of % recognition are recorded after every change in division. This experiment is performed on fix-EPCN only. Results obtained are recorded in the table 3.2 and 3.3.

Table 3.2: rand-EPCN; Record of percentage recognition when system parameters are varied. Numbers of pre- and main-group layers, and numbers of division constitute system parameters.

Experiment 1		Experiment 2	
% recg.	main-gp	pre-gp layers	% recg.
80.63	3	1	59
84.33	4	2	63
85.67	5	3	64
84.33	6	4	65
82.66	7	5	65
		6	64
gp=Group		7	64
% =Percentage		8	66
recg =Recognition		9	67
		10	67
		11	66

Considering Figures 3.5 and 3.6 below, we note that as the number of layers increases, more class representative features are extracted from input patterns, and the performance increases until the maximum is reached.

Table 3.3: Fix-EPCN; Record of percentage recognition when system parameters are varied. Numbers of pre- and main-group layers, and number of division are system parameters.

Experiment 1		Experiment 2		Experiment 3		
Main-gp	% recg	pre-gp layers	%recg	Division	recg.	% recg
1	63	1	59	80	55	68.75
2	71	2	63	100	69	69.00
3	76	3	67	200	138	69.00
4	79	4	65	300	208	69.33
5	74	5	65	400	277	69.25
6	71	6	64	500	346	69.20
7	73	7	64	600	415	69.17
8	73	8	66	700	485	69.29
gp=Group		9	67	800	554	69.25
% =percent		10	67	900	623	69.22
recg =recognit		11	66	1000	692	69.20

Maximum performance occurs at 85.7% in figure 3.5, and 67% in figure 3.6. From point (85.7,5) in figure 3.5, and point (67,9) in figure 3.6 onward, a decrease in performance is observed. This is because the number of iteration is not sufficient to account for other additional features extracted from input patterns.

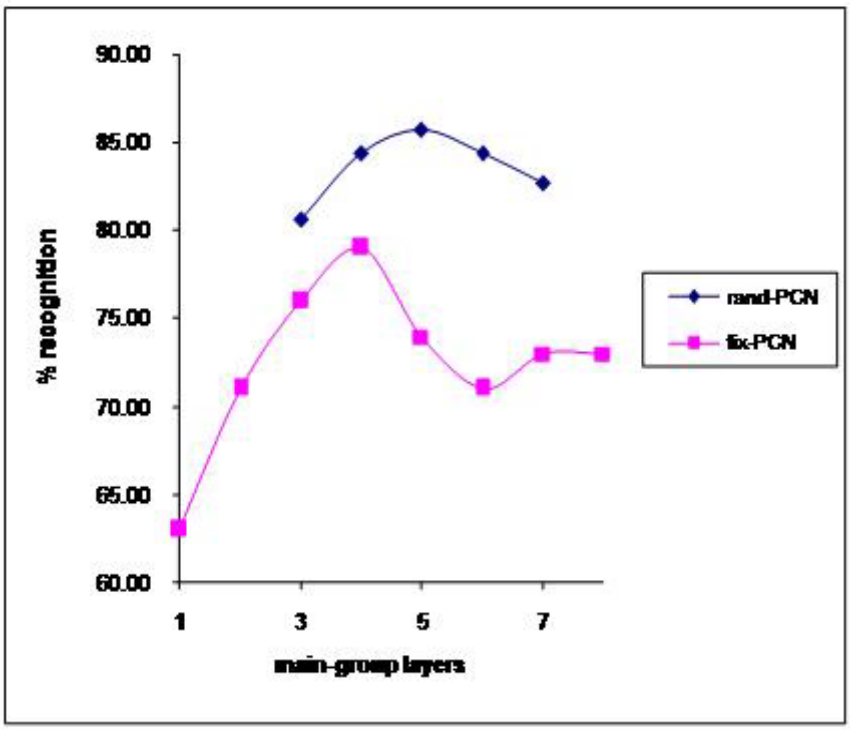


Figure 3.5: A graph showing the effect of the main-group layer on performances. This is a plot of table 3.2 and 3.3 column 1 vs.2.

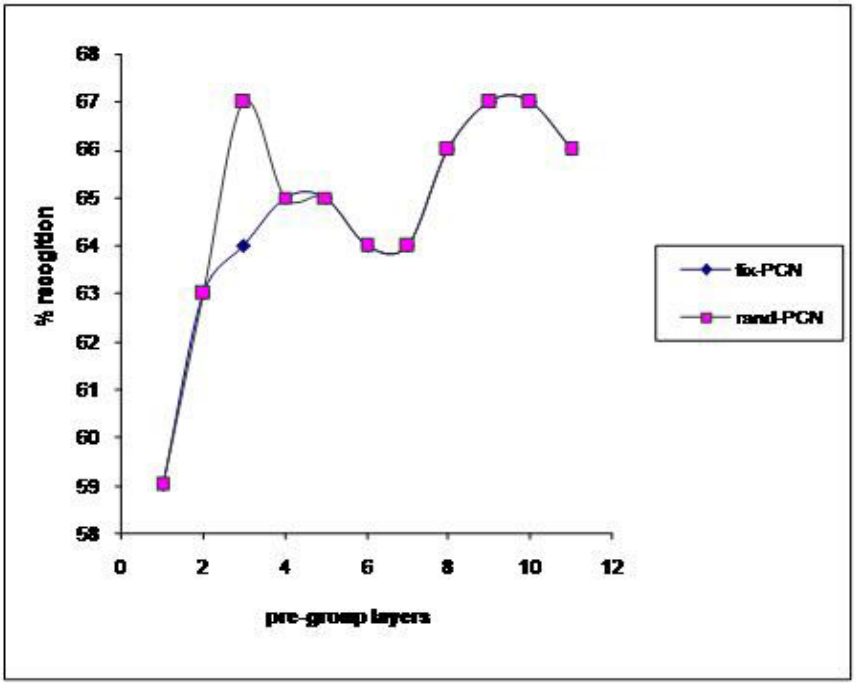


Figure 3.6: A graph showing the effect of the pre-group layers on the performances. This is a plot of table 3.2 and 3.3 column 3 vs. 4.

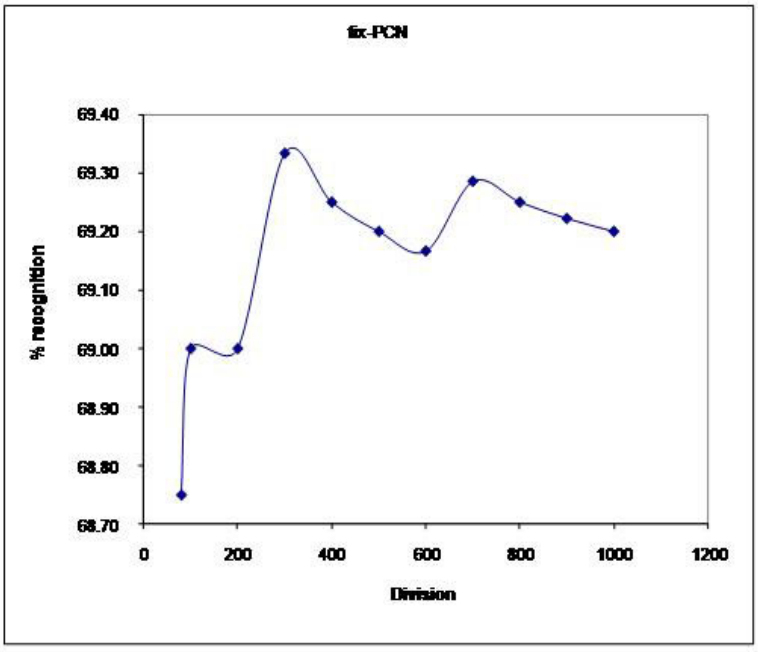


Figure 3.7: A graph showing the effect of the Division on performance. This is a plot of table 3.3 column 5 vs. 7.

Additional features of EPCN is its ability to leave areas of local maximum as shown at point (65,5) in figure 3.6. It is not unusual to obtain a local maximum solution for a problem. As long as the network is able to leave the local maximum and achieve a global maximum in quest, the aim has been achieved, and local maximum may be ignored. Saturation of RAM-neurons can also prevent performance from increasing. From figure 3.6, there is no (except at point pre-gp = 3) difference between the performance of fix- and rand-EPCN. But from figure 3.5, the performance of fix-EPCN (maximum at 79%) is lower than that of rand-EPCN (maximum at 86%). Since the difference between fix-EPCN and rand-EPCN is in their method of connectivity formation, it would be concluded that their methods of connectivity formation has led to this performance difference. The division value is employed in the adjustment phase to limit the probability measure. Percentage recognition is related to the probability measure through an averaging procedure. This means that changes in the value assigned to division is directly related to changes in performance. As the value of division increases, more bits become available to enumerate the features of input patterns. This leads to stabilisation of probability measure for the different classes as the neurones reaches a consensus concerning the different classes. This leads, in turn, to increase in percentage recognition. And thus the percentage recognition increases to 69.35% in figure 3.7. As the value of division becomes very large, same sets of features are enumerated repeatedly and this forces EPCN to reach a repeating sequence of state. This event is witnessed, in figure 3.7 when the division is 500 and 1000.

3.4.1 Benefits of weightless Neural System-EPCN

Benefits derived from using the implemented neural network are:

- One-shot learning (as explained in chapter 2)
- Easier to make fault tolerant because of its binary nature.
- More amenable to digital implementation
- Minimum of mathematical (floating-point) computation
- Increased speed

Though these benefits are not exclusive to weightless neural networks only, only weightless neural system possesses all the mentioned characteristics concurrently. These points might be exemplified by comparing EPCN with state of the art neural networks such as MLP.

3.5 Comparison of EPCN with other Neural Networks

A learning cycle is known as (one) epoch in weighted neural system while in weightless neural system referred to as one-pass or one-shot. Weighted neural network requires more than one epoch of learning cycle, while the weightless network require a one-pass over the input data.

Digital system is known to be resistant to noise because of its binary nature. Since weightless system does not require weights and activation function, spectral functions (e.g. Fourier) are known and discrete. Thus noise sources are easily identified and, when necessary, removed. A great deal of effort is required to maintain an analogue weights or the charge on an ion for a long time. Thus the weightless neuron is more amenable to digital implementation.

Table 3.4 shows a comparison of EPCN with other similar weightless neural networks.

Table 3.4: Comparison of EPCN with other neural networks.

Neural Networks	% performances
GCN	82.5
Fix-PCN	75.0
Rand-PCN	85.0
CMM	94.5
k-NN	95.4
MLPN	92.3
RBFN	94.1

It is possible, either automatically or otherwise, to map almost all algebraic computation to its Boolean logic equivalent. Mapping problems to its Boolean logic equivalent avoids any computation (most especially floating-point) overheads that might be required in, for example, a weighted neural system or any other alternative systems. Mapping problems to a Boolean logic equivalent is not a trivial task.

An increased speed may be inferred from lower computation demand and a learning/recognition cycle of one-pass over the input data.

Inferring from the result obtained, for practical purposes, EPCN may be useful in preliminary decision making and routine classification purposes. Both EPCNs are unsuitable in their present form (they require optimisation) for sensitive application, for

example, in security issues and in life-death issues. In these areas, one expects error of less than 1%.

3.6 Summary

Novel methods of connectivity formation have been introduced to EPCN. The random selection of input bits within the input pattern to form connectivity has been shown to improve percentage recognition. The dependence of performance on values of division and the number of *pre-* and *main-group* layers has been verified. In this experiment, all inputs were static during learning and recognition. Cases of moving object and/or moving surrounding were not investigated. This experiment was designed to investigate performances of EPCN with respect to their system parameters and emphasis was not placed on state of database. This may be considered as areas of future research and development.

The results of this chapter call for improvement in implementation as the performances of both EPCNs are generally low.

Two types of EPCN have been implemented. Combination of these two novel classifiers may form a multiple classifier structure (with or without addition of other NNs) which may find application in Automated Control and Guidance Systems, Robot visual guidance systems, etc. In the next chapter however, use will be made of a multi-classifier derived from these two EPCNs.

4. A WEIGHTLESS ARTIFICIAL NEURAL BASED MULTI-CLASSIFIER

Recent years have witnessed intense research in the general area of Multi-Classifier systems (MCS), but this has rarely incorporated any utilisation of weightless neural systems (WNS) as the combiner of an MCS ensemble. This chapter explores the application of weightless networks within the multi-classifier environment by introducing an intelligent multi-classifier system using a WNS called the Enhanced Probabilistic Convergent Neural Networks (EPCN). The chapter explores the use of EPCN by illustrating its major features, such as the specification of disjoint or overlapping input subset to the MCS, and the parallel nature of the design. Within the proposed system, the number of base classifiers per MCS could be specified manually or automatically. The proposed MCS is problem-domain independent and, our investigation is performed on handwritten characters. The proposed MCS is adaptive; its combiner is capable of extracting absolute or weighted classification decision (output) from base classifier. Diversity is increased in the base classifier by injecting randomness into the system parameters. Two types of EPCN classifiers are employed: fix-EPCN and rand-EPCN. These EPCNs are independent and orthogonal in behaviour because one uses a fixed method of forming connectivity while the other uses random method of forming connectivity.

In order to verify the performance of the recognition system, tests were performed, off-line, on benchmark datasets of unconstrained handwritten numerals. Experimental results suggest that MCS outperforms single EPCN in classification of handwritten characters.

Artificial Neural Systems in general and Weightless systems in particular, have traditionally struggled in performance terms when confronted with problem domains possessing a large number of independent pattern classes. The overloading and saturation experienced by traditional networks is addressed by training the base classifiers on differing subsets of the required pattern classes and allowing the combiner classifier to derive a solution based on the whole ensemble. The system is demonstrated on the exemplar of fingerprint identification and utilises a Weightless Neural System called the Enhanced Probabilistic Convergent Neural Networks (EPCN).

4.1 Introduction

Motivated by the performance of each of the EPCN in chapter 3, the implementation of a multi-classifier is conceived. The aim of the multi-classifier design is to improve recognition on handwritten characters; most especially an improvement on classes with low performance (i.e. the "difficult" pattern) rate is desired. Over recent years, a significant research effort has been devoted to the development of multi-classifier systems (MCS) [109]. MCS consist of component classifiers, possibly of an artificial neural configuration, called base classifiers, arranged in a specific fashion so as to carry out a specific task which would otherwise yield a poorer performance should such a task be performed by a single neural network or classifier. The specific arrangement of this classifier s is commonly referred to as a classifier selection. R. Ranawana [109] summarises various methods used in classifier selection but does not significantly include weightless classifier. Weighted classifiers are that classifier whose performance and system parameters depend on weights and weight adjustment. In contrast, classifier s whose performance and system parameters do not depend on weights (and their adjustments) is called weightless classifier. It is highly suitable for implementation in portable embedded systems and its ability to efficiently learn with a reduced number of training iteration. In a weightless classifier, binary weights are stored and retrieved from RAM. To date, most significant research in classifier used in MCS has involved weights, for example [36] uses classifier selection based on weights. This chapter presents an MCS employing weightless classifiers. The base classifiers employed in this work are derived from *The Enhanced Probabilistic Convergent Network, EPCN*. Details of the base classifiers were published in [85].

But the multi-classifier designed is useful only if the number of classes is large, ten (10) classes and more will be employed. A multi-classifier for large-scaled multi-class classification is motivated by the fact that most state-of-the-art multi-classifiers has been shown to fail, in performance term, when the number of pattern classes becomes very large. This scenario forms the motivation of this chapter. The issue of applying a RAM-based multi-classifier to large-scaled multi-class classification tasks is addressed here. As the number of pattern classes required to be recognised by an artificial neural systems are increased, problems associated with the overloading and saturation of the network begin to manifest themselves. This chapter presents a novel method which not only aims to solve these problems, but is also able to produce an appreciable recognition performance when a

large classification is required. The solution that this chapter presents comprises of partitioning the pattern classes into disjoint sub-sets, and employing a multi-classifier system (MCS). The component classifiers are derived from The Enhanced Probabilistic Convergent Network, EPCN.

This chapter is organised as follows. Sub-section 4.1.1 introduces multi-classifier system. The design of the MCS commences in section 4.2 where an MCS implemented using EPCN is introduced, and then experimented on in section 4.4. The results obtained were recorded and analysed in section 4.5 and 4.6. The chapter completes with a summary and areas of further possible experimentation in section 4.7 and 4.8.

4.1.1 Introduction to RAM-based Multi-classifier

Two types of EPCN were introduced in chapter 3. They were tested on handwritten numerals. Here, base (component) classifiers are derived from these EPCNs. The base classifiers derived from EPCN (Chapter 3) will be employed in a multi-classifier framework. In order to facilitate the employment of EPCN in sector such as security and health, an improvement on the performance of chapter 3 is required. Whilst maintaining the benefits of section 3.4.1, it is considered significant to achieve a performance in excess of 90% on handwritten characters and numerals such as that EPCN are further useful. It is expected that the design and employment of Multi-classifier on handwritten character will improve performances as compared to when a single EPCN classifier is employed. The EPCN is a classifier which allocates a confidence measure to each candidate class, based on supervised learning, when a pattern is presented to it for classification. An interested reader could consult [85] for more detail on EPCN.

A significant component within the design process of an MCS system is the selection of the base classifiers to employ. The most common selection methods used for base classifiers are: input data [109], Genetic Algorithm (GA) [2], Objective functions [112], [119], Random selection [42], Boosting [42], and Bagging [119]. Some designers [42], [55] make classifier selection to depend on certain diversity measures.

One of the most successful ensemble creation methods is the random subspace method [42]. Here input space is partitioned by random selection into subspaces of equal length and a classifier is assigned to each subspace.

The most common arrangement of base NN used in an MCS is the parallel method. Other topologies are the cascading and hierarchical topology [34]. Cristian Dima [26] proposes the implementation of a hierarchical mixture of experts and the employment of dynamic reconfiguration to analyse robot dynamics.

It is essential that for classifier to be included in an MCS, either the performance must be significantly above average (50%), or the classifier must make other significant and positive contribution to the ensemble after combination, which may not be expressed in terms of percentage performance. Lam [79] states that orthogonality; complementarities and independence of a base classifier determine its inclusion in an MCS. During training and recognition, each base classifier utilises its normal training and recognition algorithm. The combination of base classifier output is called *classifier fusion*. Various techniques for classifier fusion are broadly divided into: *objective functions [120]; Qualitative combination [11]; intelligent combiners [35]; Fixed combiners or balanced classifiers [112]*. Significantly, EPCN, when used as a combiner, is a novel weightless *intelligent combiner* since it possesses its own learning and recognition algorithms.

A. Krzyzak [135] categorizes combiners of MCS into two, namely, feature-vector-based method (i.e. using neural network) and syntactic-and-structural (i.e. fuzzy-rule based) method. [119] categorises them as: Linear, Non-linear, Statistical, and Computational Intelligent combiners.

The overall performance of an MCS is often compared to a single base classifier. At present, it is difficult to quantify how diversity measure affects performance, most especially for MCS comprising large number of base classifiers. Gabrys and Ruta [12] maintain that diversity measure has limited correlation with MCS performance. It should be emphasised that MCS performance depends on careful selection of base classifiers. Min [77] uses a Rejection criterion and reliability to measure performance. The rejection criterion and reliability are numerical quantities derived from a fuzzy integral. A performance improvement has been made on isolated handwritten characters [126], whole words [121], postal addresses [70], [77], and bank cheques [63]. It is difficult to achieve a high recognition rate using a set of features and a single classifier. This is because totally unconstrained handwritten numerals, as is the case in this work, contain an appreciable level of pattern variation which mainly depends upon individual writing style.

The design of MCS using EPCN as an *intelligent combiner* will be the subject of the next section.

4.2 The Design of an MCS from EPCN

Multi-classifier utilising weightless classifiers are currently rare. This section presents an MCS that utilises weightless NN called *Enhanced Probabilistic Convergent Network (EPCN)* [85]. MCS may be grouped according to their output. A formal grouping of such classifiers is: abstract form, rank level, and measurement level [109]. Of these, the measurement level group is relevant.

- *Measurement level:* - No attempt is made to arrange the output of a base classifier in any order, since the order of values in itself has meaning. Each class is assigned a belief of the classifier about the input. The result is an array of belief values. These classifiers are also called *probabilistic classifiers*. Fix-EPCN and rand-EPCN are novel weightless *probabilistic classifiers*.

Previous studies have shown the performance of both EPCNs to be well above 50% [85]. Fix-EPCN is orthogonal to rand-EPCN due to its inherent method of forming connectivity. The rand-EPCN uses random method while fix-EPCN uses a pre-defined or "fixed" method, a systematic method which is reproducible.

These EPCNs are designed to be independent and without correlation with regards to their errors, giving no consideration for any future input dataset. Varying the system parameter of each EPCN has a profound effect on their decision making. These decisions (outputs) do not give rise to error correlation, for disjoint input dataset. Thus EPCNs are good candidates for MCS production.

In this work, the input space is partitioned into overlapping subsets and a classifier is assigned to each subspace. This allows for a clear comparison with a standalone single classifier. Since this MCS uses EPCN, it will henceforth be denoted by MCSPCN for short. MCSPCN is designed with the possibility for dynamic reconfiguration, and the parallel scheme is employed. In a changing environment, system parameters could be made dependent on changes in the environment.

Diversity is increased in MCSPCN by incorporating diversity within the training algorithm [106], [109] of all EPCNs. This influences their behaviour during training and recognition. For example, if F classifier is required, and N_i classes each, this will be specified as:

>> mcspcn(F,N_i,r,c); i = 1,2,3, ... (4.1)

where,

r = number of rows in pattern.

c = number of column in pattern.

For each F, the size of N_i may vary or overlap.

This work utilises the *Computational intelligent* method for classifiers fusion by employing another EPCN as the combiner.

4.2.1 Combiner Unit

The term *Combiner Unit* refers to the EPCN combiner [P_c, M_c], and the gating function *f (.)* (See fig. 4.1). The gating function consists of a *decision make*r and a *converter*. The *Decision maker* is required for the following reasons:-

- If the same character is classified or assigned by different NN to differing classes and these classifications are correct, without the *decision maker*, these two interpretation will be converted to different images by the *converter*. A correct classification of a pattern by different NN should produce similar pattern for the EPCN-combiner to train.
- The combiner EPCN does not know if the input space overlaps or not. The *decision maker* is also required to monitor overlap and to reflect this it its output by weighting.

Decision Maker: - The *decision maker* considers the performance of the component classifiers with respect to the classes, and passes its decision to the *converter*. It utilises a weighting strategy on the output of the base classifiers when inputs overlap. This weighting strategy affects only those outputs corresponding to the region of input overlaps. A zero weight switches off an output of a base NN with respect to a given class, while a weight greater than zero switches it on. The decision maker does not eliminate a base classifier, but only inhibits certain outputs with respect to certain classes. This inhibition depends on input space overlap and performance on that class. For example, consider a character "a" is trained to one NN as class 1, and trained to another NN as class 2. During recognition, correct classification requires the first NN to classify "a" as class 1 and the second NN should classify it as class 2. The *decision maker* is responsible for informing the converter that the two output are the same i.e. are correct classifications of "a".

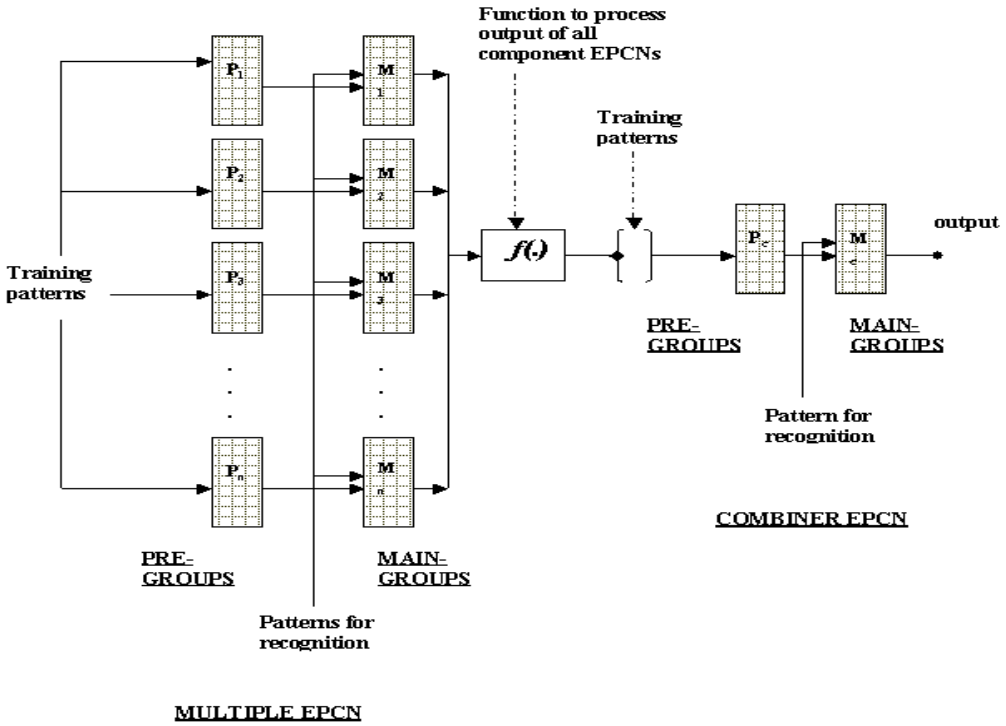


Figure 4.1: The MCS unit is divided into multiple EPCN group and combiner EPCN group. The multiple EPCN group consist of EPCNs in parallel. P_i = *pre-group*; M_i = *main-group*; i = 1,2,3,… *f(.)* = gating function. P_c = combiner's *pre-group*. M_c = combiner's *main-group*.

Converter: - This converts the *Decision maker*'s integer output into binary, e.g. for division = 1000, [0, 0, 65, 45, 0, 0] will be converted to:

[0000000000
0000000000
0001000001
0000101101
0000000000
0000000000];

EPCN-combiner configuration: - An example configuration of EPCN combiner is shown in figure 4.2.

```
>> con

con =

    nost: [9 10 8 9 10 10 13 10 6 10]
    nclas: 10
    nlay: 2
    nosnn: 1
    dimz: 10
    dimr: 10
    ntuple: 3
```

Figure 4.2: An EPCN configuration.

In the first field, nost, each number represents the number of training patterns per class. The second field, nclas, represents the total number of classes. The third field is the number of layers in the pre-group. The 4^{th} field is the number of columns in the image while the 5^{th} is the number of rows in the image. The last field, ntuple, is the tuple-size. The combiner's *main-group's* configuration is the same, except the field “nlay" is replaced by "mglay", where "mglay" is the number of layers for the *main-group*. Thus we have a MCS that looks like fig. 4.1, where $[M_i, P_i]$ is a base classifier; i = 1, 2, 3 … This multi-classifier will be tested in experiment section.

Advantages of weightless classifier are their fast learning algorithms, ease of implementation in digital hardware, and ease of implementation in a portable embedded system. It could be argued that in a weightless classifier, binary weights are stored and retrieved from RAM. An important component within the design process of an MCS system is the selection of the base classifiers to use. The combination of base classifier output is called *classifier fusion.* The design of MCS in this section is input independent and it uses EPCN as an *intelligent combiner*.

4.3 Multi-Classifier System for Biometric Databases

Much research is currently based on biometric identification, but most of these research efforts have utilized other means, not involving weightless neural networks. This is because of problems of biases and saturation which accompanies such venture when weightless neural networks are employed [7]. For these reasons, a multi-classifier is proposed in this chapter in which problems such as biases and saturation is specifically considered. Fix-EPCN and rand-EPCN are novel weightless *probabilistic classifiers.*

From the fact that no error correlation exists between their outputs for disjoint input, and varying the system parameter of each EPCN has a profound effect on their decision, these make them diverse. Thus, by varying their configuration, EPCNs are very good candidates for MCS production.

Since this work is concerned with large classification domain, the pattern classes are partitioned into disjoint subset and a classifier is assigned to each subspace. This also allows for clear comparison with a standalone single classifier. The system parameters of a component EPCN are shown in a structure of figure 4.4, and an extract of fingerprint from FVC2004 database is shown in figure 4.3. The system parameters influence their behaviour of the base classifiers during training and recognition. A desired number of classifier required and number of class per classifier is usually specified to MCSPCN (equation (4.1)). For example, if F classifier is required and with N_i classes each, these values will be utilised for multi-classifier initialisation. For each F, the size of N_i may vary, overlap, or be disjoint as in this case.

In this work, the classifiers fusion method used is the *Computational intelligent* method. This is a case where another EPCN acts as the combiner.

Figure 4.3: A sample Fingerprints from database DBA_1

```
>> con

con =

    nost: [9 10 8 9 10 10 13 10 6 10]
    nclas: 10
    nlay: 3
    nosmn: 1
    dimz: 24
    dimr: 16
    ntuple: 3
```

Figure 4.4: A component (base) neural network's configuration.

In figure 4.1, [P_i, M_i] are the component classifiers, i = 1, 2, 3…, *f(.)* is the gating function, while [P_c, M_c] is the combiner.

The Input Pre-processing

The pre-processing steps are experimentally determined so as to minimise distortion of local features. Global features of the fingerprints are pre-processed as this will leave local features as they are. The proposed pre-processing steps are summarised below.

• Geometric alignment: The smallest bounding box for each fingerprint is found. This may require rigid (uniform) rotation and/or translation of the fingerprint involved. The alignment serves to isolate only relevant region of the fingerprint for pre-processing.

• Effects of uneven illumination are removed by morphological element (window).

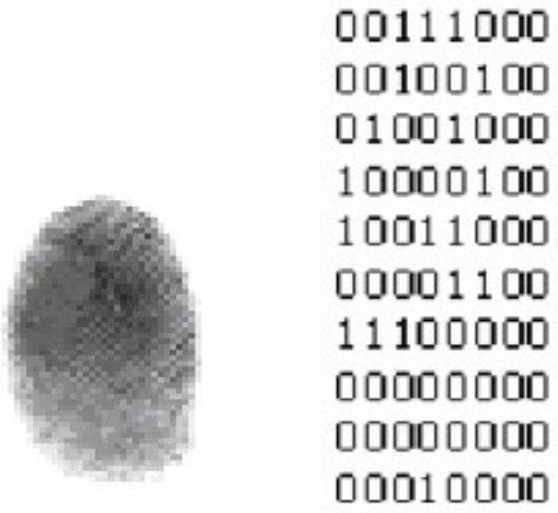
• Noise is removed by filtering.

• The edges of ridges and valleys are emphasized by edge enhancing filters.

• Pinches and punches are corrected for by interpolation.

• downsampled fingerprints are binarized (see figure 4.5) using a one-value thresholding.

The binarised fingerprints are of the form accepted by EPCN. The resulting fingerprint is downsampled. Though the downsampling is uniform, no adverse effects were observed on the local features.



. **(a)** **(b)**

Figure 4.5: (a) Processed fingerprint, (b) binarised picture of fingerprint.

4.4 Experimentation

Chapter 3 demonstrate the difficulty experienced by a single classifier when a high performance rate is required. For this reason, the same CEDAR database is employed in this experiment. The aim is to achieve higher performance rate on databases of unconstrained handwritten numerals, a performance such as may not be possible for a single classifier. Off-line handwritten characters and numerals recognition has been a topic of intensive research for many years. The performance of EPCN as combiner should equal or surpass that of feature vector based classifiers or syntactic/structural based classifiers. The MCSPCN is problem-domain independent and as such should perform well on handwritten characters. The source of totally unconstrained numerals used in this work is:-

- The centre of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York, Department of Computer Science. Handwritten numbers from CEDAR were resized and binarised to 16-by-24 in dimension.

The pattern used from CEDAR is handwritten numerals “0” to “9”. These are thus labelled class 0 to class 9. The number of patterns in each class varies from 200 to 1000 depending on class. These numerals are divided into training patterns and test patterns. Training

patterns and test patterns are treated independently. Test patterns do not form part of training patterns and vice versa.

Experiment 1 (five NN; five classes each)

The aim of this experiment is to determine if the combiner can successfully interpret result and ignore individual erroneous result from the component classifiers. In this experiment, the input space was partitioned as shown in Table 4.1. In this experiment, the training set of the classes overlaps. Cases of disjointed training set of classes are discussed in experiment 3. Where, for example, classifier NTW1 is only trained on classes 0 through 4. The component base classifiers, NTW# (where # = 1, 2, 3,…), are assigned to be trained on the subset of classes depicted in each row of Table 4.1. This sub-setting strategy has been employed in order to artificially lower the performance of each of the base classifiers to observe if the PCN combiner, $[P_c, M_c]$, is able to allow for the poor performances and give a good overall result – this is significant.

During recognition, each network is required to classify patterns belonging to all the ten classes. All patterns that result were collected in a directory. These were afterwards separated into training set and test set. The training set is used to train the combiner while the test set was employed during recognition. The performance metric used is the percentage (%) of patterns recognised. All results obtained were processed and important results recorded in Table 4.2.

Experiment 2 (five NN, ten classes each)

Obviously in practice, base classifiers would be trained on the entire pattern class set. The second experiment is therefore aimed at determining if the MCS performs better than any of the component classifiers alone.

Experiment 1 is thus repeated with each of the five component classifiers trained on all 10 classes. In practice, this is done by setting F_i =5 and N_i = 10 in MCSPCN (equation 4.1 of section 4.2). During recognition, each network is required to classify patterns belonging to all the ten classes. The results were later collected and processed by the gating function *f (.)*. Again all patterns that result were collected in a directory. These are afterwards separated into training set and test set. Training set is used to train the combiner while the test set was employed during recognition. The performance metric used is the percentage (%) of patterns recognised. This follows because all numerals were written independently

by distinct writers, and no correlations were found between them. All results obtained were processed and important results recorded in Table 4.3.

Table 4.I: Partitioning of the input space in experiment 1; NTW = Base classifier; # = number.

	CLASSES				
NTW1	0	1	2	3	4
NTW2	2	3	4	5	6
NTW3	4	5	6	7	8
NTW4	6	7	8	9	0
NTW5	8	9	0	1	2

Table 4.2: Comparison of a combiner with base neural networks when F = 5; N_i = 5. Clasf. = classifier; NTW# = Network, where # = a number. % = percentage.

Average Performance Comparison (%)			
Component clasf.		Combiner	Difference
NTW1	50	93.37	43.37
NTW2	80	93.37	13.37
NTW3	36	93.37	57.37
NTW4	74	93.37	19.37
NTW5	52	93.37	41.37

Table 4.3: Comparison of a combiner with base neural networks when F = 5; N_i = 10. Clasf. = classifier; NTW# = Network, where # = a number. % = percentage.

Average Performance Comparison (%)			
Component clasf.		Combiner	Difference
NTW1	74	95.14	21.14
NTW2	80	95.14	15.14
NTW3	63	95.14	32.14
NTW4	79	95.14	16.14
NTW5	69	95.14	26.14

Experiment (3) on Large-scale Multi-class database

The aim of this experiment is to explore how EPCNs in a Multi-expert configuration perform when exposed to large classification problems with few patterns per class.

For the experiment, the input space is partitioned into disjoint subset and a

classifier is assigned to each subspace. This allows for a clear comparison with a standalone single classifier. The performance of the NN-based classifiers, EPCN as combiner, should equal or surpass that of feature vector based classifier or syntactic/structural based classifier. MCSPCN is problem-domain independent MCS and as such should perform well on fingerprint databases.

The database used for this experiment is DBA1 from the Third International Fingerprint Verification Competition 2004, FVC2004. "NOTE: FVC2004 databases are markedly more difficult than FVC2002 and FVC2000 ones, due to the perturbations deliberately introduced…" [43]. Most experimentation methods rely heavily on minutiae and template matching of minutiae. Holignum [53] employs the graphical method, Jain [65], [66] uses point pattern matching, Wahab [132] employ structural matching techniques to minutiae, and Tico [129] uses transformation operation. To improve on these methods, Maio and Maltoni [90] introduce the detection of false positive. [60], [106], [133] provides methods aimed at removing false minutiae, and [85] uses NN for minutiae filtering. The advantage of using an artificial neural network (ANN) instead of minutiae analysis [85], [106], is that a global operation on the images is less sensitive to local distortion that normally occur during extraction of local features. Fingerprints in this database, of the type shown in figure 4.5, are extracted into a directory (see figure 4.5(a)). They are then filtered, centred, and then binarised (see figure 4.5(b)). Each finger printed in various forms represents a class. The binarised fingerprints are divided into two sets, the training set and the test set. Each set consist of 100 classes. This motivates the initialisation of MCS consisting of 10 classifiers, and 10 classes per classifiers. In practice, F, the number of NN, is set to 10, and N_i, the number of class/NN is set to 10, these are passed to the program MCSPCN (in equation 4.1 of section 4.2).

4.5 Results and Analysis on Large-scale Multi-class database

Table 4.4 is obtained with an MCS of ten classifier and ten classes per classifier.
During the experimentation, the input space is partitioned into disjoint subset and a classifier is assigned to each subspace, and little corrections were made for the following deformations:

- Shifts

- Rotations
- Intensity changes.

Table 4.4: This table shows the performance (in % of patterns recognised) of MCS with respect to large pattern recognition problems. In this case fingerprints.

classes	%	classes	%	classes	%	classes	%	classes	%	classes	%
class1	62.5	class19	62	class37	37.5	class55	25	class73	62.5	class91	62.5
class2	87.5	class20	0	class38	25	class56	37.5	class754	50	class92	12.5
class3	87.5	class21	87.5	class39	25	class57	50	class755	50	class93	37.5
class4	87.5	class22	100	class40	62.5	class58	25	class756	12.5	class94	25
class5	100	class23	100	class41	75	class59	50	class757	37.5	class95	25
class6	100	class24	100	class42	100	class60	50	class758	25	class96	25
class7	75	class25	87.5	class43	87.5	class61	87.5	class759	50	class97	50
class8	100	class26	100	class44	75	class62	87.5	class760	37.5	class98	50
class9	75	class27	87.5	class45	87.5	class63	87.5	class761	87.5	class99	25
class10	87.5	class28	100	class46	100	class64	100	class762	87.5	class100	75
class11	12.5	class29	100	class47	62.5	class65	87.5	class763	100		
class12	62.5	class30	62.5	class48	87.5	class66	100	class764	87.5		
class13	37.5	class31	37.5	class49	62.5	class67	87.5	class765	87.5		
class14	37.5	class32	25	class50	62.5	class68	87.5	class766	87.5		
class15	62.6	class33	37.5	class51	0	class69	87.5	class767	87.5		
class16	50	class34	50	class52	12.5	class70	87.5	class768	100		
class17	37	class35	50	class53	25	class71	62.5	class769	87.5		
class18	0	class36	62.5	class54	0	class72	50	class770	100		

- Occlusion
- Pinch
- Punch.

The conditions under which these fingerprints are collected are as specified in [43]. Edge-enhancing filter is applied, and this is followed by binarisation of the fingerprints. The reason for avoiding intensive pre-processing is to prevent artificially adding to local distortions already present, and leave all decision making processes to the NNs. This makes the processes close to real-life recognition system, and also decreases processing time.

Performance Measures: **-** On databases such as fingerprints, the most commonly used performance measures are: True Acceptance rate (TAR), true rejection rate (TRR), Predictive value positive (Pos), and Predictive value negative (Neg). These are quantitative measures of the trustworthiness of results obtained. However every output of EPCN, in

recognition mode, includes confidence measure. The confidence measure or the trustworthiness measure is a *scaled probability measure*, scaled by a value called the *Division*. For example, an output of EPCN is as shown in figure 4.6.

```
>> ans=

   desout: [0 118 0 0 0 50 69 50 264 450]
     sor: 1
     soc: 10
     pat: ([10x8 char])
  desmatc: [63 75 89 73 72 76 86 84 92 101]
    socd: 10
    sord: 1
  nositer: 1
  ntuple: 3
    matc: [63 75 89 73 72 76 86 84 92 101]
    matr: [87 78 106 93 85 86 92 88 107 115]
   mglay: 5
  noplay: 5
 division: 1000
```

Figure 4.6: An Output of EPCN.

The important field in this structure is “desout”. This states that the network is trained on ten classes. When a pattern is presented to it for recognition, it is (450/1000)% (the confidence measure) likely to belong to class ten, (118/1000)% likely to belong to class two, etc. The summation of the numbers (variables in the field “desout”) should equal the variable in the field division. Thus results from EPCNs, and Multi-classifiers dependent on EPCN, are inherently with trustworthiness measure. From table 1, the results show recognition performances ranging from 0% to 100%. In the MCS configuration utilised, no self-reconfiguration was employed, the component classifiers employed are feed-forward supervised EPCNs. It may also be noted that most classes has their recognition well above 50%.

Because of zero cross-correlation and independent update of classes, percentage of pattern correctly classified may be regarded as optimum performance metric.

4.6 Result and Analysis on experiments 1 and 2

Table 4.2 illustrates the result of experiment 1 and is obtained when F = 5; N_i = 5 is specified to MCSPCN (function 4.1 of section 4.2) with the databases specified in section 4.3 is employed. Table 4.3 represents the result of experiment 2 and is obtained when F =

5; N_i = 10 is specified to MCSPCN (function 4.1 of section 4.2) with the databases specified in section 4.3 were employed. Averages were calculated with respect to training set.

The first column of both tables shows the component classifiers, the second column shows their respective performances, and the third column shows the overall performance of the MCSPCN. In table 4.2, NTW1 shows an average (50%) recognition rate while NTW2 shows a high percentage recognition rate (80%). NTW3 shows a poor recognition rate (36%) while NTW4 shows a high percentage recognition rate (74%). In table 4.3, NTW1 shows an average (74%) recognition rate while NTW2 shows a high percentage recognition rate (80%). NTW3 shows a fairly good recognition rate (63%) while NTW4 shows a high percentage recognition rate (79%). From this trend, it could be inferred that when some base classifier performs fairly well on a database, others perform very well on the same database. This shows the inherent orthogonal properties of fix-PCN and rand-PCN.

Comparing the second column of both tables, the classifiers are seen to perform better when trained on all ten classes than when trained on sub-section of the classes. This affects the combiner positively with an average improvement of about 2%.

In table 4.2, the performance of the combiner (at 93.37%) was well above that of the component classifiers and shows that the combiner is able to filter out poor component classifier results. In table 4.3, the performance of the combiner (at 95.14%) was also well above that of the component classifiers. From this we may deduce that the gating function *f (.)* considers only the merits of the base classifiers. The individual entries in the difference column (in %) show the performance of the combiner over their corresponding base classifiers.

4.7 Summary on Large-scale Multi-class database

In this work, we have focused on utilisation of EPCN in an MCS framework on large class problem domain - an instance of this is fingerprint identification. The multi-classifier shows performances ranging from 0% to 100%. This is to be expected because little corrections were made for deformations, same pre-processing steps are applied equally to all fingerprints, and also because the level of noise and deformations varied and are distinct from one fingerprint to the other.

Given the fact that 100 classes are presented for classification, the possibility of high recognition rate (up to 100%) shows that this configuration provides a solution to overloading and saturation that result when very large class problem is given to a single neural network.

To further this research, utilisation of adaptive pre-processing techniques to make pre-processing each image dependent on level of noise and deformation in them, is in order. Recall that no NN selection strategy were employed to remove, modify system parameters, or replace the less performing NNs, these are also considered to be subjects of next research possibility. It is not possible, due to various reasons that include resource and time, to conduct all suggested future research possibilities in this single thesis. For this reason, a key research point – that of implementation of a novel combination strategy is conceived. The next chapter explores the possibility of a new combination strategy for a multi-classifier system.

4.8 Summary

In this chapter, we have focused on a multi-classifier combining strategy using a novel RAM-based artificial neural network EPCN. The combiner of the multi-classifier has been shown capable of interpreting results from component classifier and ignoring individual erroneous results. Significantly, the multi-classifier is shown to have achieved a high performance rate (93.37% in Table 4.2, and 95.14% in Table 4.3) compared to the component classifiers. It is to be noted also that this performance compares favourably well with other multi-classifiers derived from weighted base classifier or neural network, using other techniques, e.g. [77]. Experimental results suggest that MCS outperformed single EPCN [85] in classification of handwritten characters.

The problems associated with the multi-classifier designed in this chapter are:

- The input may be biased.
- The network may be easily saturated.
- Its support for large-scaled multi-class databases poor.

Some of these problems are addressed in chapter 5, most especially the problems of classifying large-class databases.

Other areas of further investigation may include other configuration methods, such as Boosting, Bagging, or using performance criteria to initialise and choose base classifiers.

As this is likely to have the effect of eliminating such network as NTW3 (at 36 % in Table 4.2) from the WNS since its performance is sometimes below 50%.

Following a very good performance in this chapter, the classification of very large-class databases is considered in the next chapter.

5. AN ADVANCED COMBINATION STRATEGY FOR MULTI-CLASSIFIERS

An advanced combination strategy is hereby introduced which addresses the scale problems exhibited by traditional artificial neural networks. The encoding scheme introduced here produces a different and significant approach to solving the problems of memory demand, execution time, and performances.

A sub-setting strategy of the required input pattern classes is introduced in this chapter which provide a more robust solution to the problems of overloading and saturation experienced by traditional neural networks.

Current Multi-classifier Systems faces the problem of bias when classes are arranged and maintained in a specific fixed pattern. A novel statistical arrangement method is hereby presented which aims to solve the bias problem. This statistical arrangement method also enhances independence of component classifiers.

The system is demonstrated on the exemplar of fingerprint identification and utilises a Weightless Neural System called the Enhanced Probabilistic Convergent Neural Network (EPCN) in a Multi-Classifier System.

5.1 Introduction

Most combination methods for multi-classifiers are meant for weighted neural networks. Attempts to utilise EPCN as a multi-classifiers *combiner* has till now failed. However, there are successful attempts made in chapters 4. *But the combination method implemented in chapters 4 could not combine the output of very large classes*. There is clearly lack of RAM-based solution to the problems of combination of RAM-based component classifier. So that, the objectives of this chapter is to implement a combination strategy. Implementation of the component classifier combination method is thus motivated by the need to encode the output of the base classifier, so that an EPCN combiner is able to combine large-classes.

Multi-classifier systems have traditionally struggled in performance terms when confronted with problem domains possessing a large number of independent classes and containing few patterns per class. Such Multi-classifier systems (MCS), consist of component classifiers, called base classifier, arranged in a specific fashion so as to carry

out a specific task which would typically yield a poorer performance should such a task be performed by a single classifier.

Currently, large number of distinct pattern classes is a classification bottle-neck for typical MCS because they suffer from large storage requirement, and long execution time. This is mainly due to the fact that floating point mathematical calculations take longer to complete as compared to simple Boolean logic. For this reason, only weightless (also called Random Access Memory (RAM) based) neural networks are used as component classifier in this chapter. The decision to use only RAM-based neural networks as components to solve the aforementioned problems is faced with yet another problem, which is that of combining the output of base neural networks.

This chapter presents a solution to the above problems via a novel combination method. The combination method consists of a *gating function* which implements an encoded pattern for the combiner. The combiner is a neural network entitled the Enhanced Probabilistic Convergent Network (EPCN) [81], [85]. The difference between an ordinary EPCN and the EPCN used for the combination, hereby termed EPCN-Combiner, is its input and configuration as shown in figure 5.1. An encoding system is required in the *gating function* because the combiner neural networks expect a binary threshold pattern as input, and the encoded pattern is binary. The encoding system is hereby termed *engine encoding*. The term *gating function* and combination method will be used interchangeably. There are two types of EPCN utilised in this work. Their difference lies chiefly in their method of connectivity

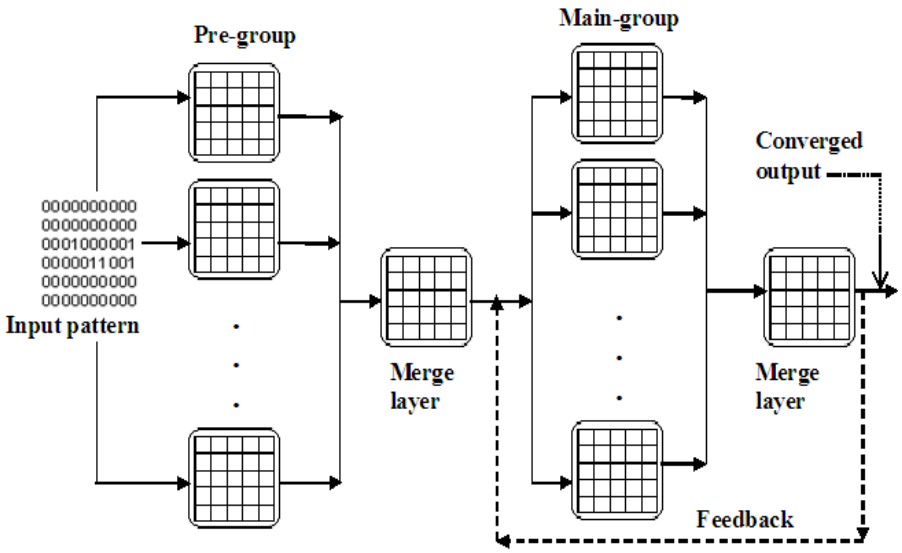


Figure 5.1: A schematic diagram of combiner-EPCN

formation [85], details of which are found in chapter 3. The base classifiers derived from the two types of EPCN has been found, by experimentation, to be error independent. Their advantages, over weighted Neural Networks are that their employment in a parallel scheme does not incur additional high mathematical computations.

Overloading and saturation are often associated with decrease in distinguishing features between classes, and limited means of enumerating these features. The problems of overloading and saturation experienced by traditional networks [57] are solved for EPCN based MCS by assigning the base classifiers on differing subsets of the required input pattern classes.

When a class is trained to more of the base classifiers than other classes, this class will be more recognise than other classes. Equal chances are not been given to all classes for classification or miss-classification; this is called *bias.* To solve this problem, a novel randomisation technique is introduced in section 5.2 which produces an arrangement of input patterns suitable for the removal of Classifier *bias.* When this randomisation technique is employed, a base classifier sees a sub-set of the classes with approximately equal probability. It is expected that when input pattern classes are randomised and evenly distributed, *bias* within the network will be removed.

5.2 Multi-classifier Systems

5.2.1 Selection

The decision to include a Neural Networks (NN) in a MCS is commonly referred to as *classifier selection*. Two selection strategies are currently widespread. They are:

- The direct method.
- The “pool of network” method.

The direct method consists of selection of NN which are error-independent of one another. This selection is often a static selection method. The “pool of network” method is a case whereby an initial large set of neural networks are available and various methods such as error diversity measure is employed to select an error independent set from this large set of neural networks [117].

The direct strategy is employed in this chapter as the “pool of network” method requires a very high computing resources and time.

The selection of base classifiers may broadly be divided into static selection and dynamic selection. Static selection methods are those methods used to select base classifiers before learning/recognition and can not change the composition of the ensemble in an experiment. Dynamic selection method on the other hand can modify the composition of the ensemble in an experiment. An appropriate feedback mechanism of error correlation, for example, may change a static method to a dynamic method.

Common selection methods used to select base classifiers can be grouped as static or dynamic selection methods depending on the presence or absence of a feedback system.

Given $n = \sum_{i=1}^{r} n_i$ classes of r distinct types, where n_i are of type i and are otherwise not distinguishable, the number of permutation without repetition, of all n classes is:

$$M_n(n_1,...,n_r) = \frac{n!}{\prod_{i=1}^{r} n_i} \tag{5.1}$$

The number M_n is known as a multinormial coefficient. A special case is when r = 2. This is a case of binormial coefficient and is denoted by $M_n(n_1,...,n_r) = {}_nC_r$ where

$${}_nC_r = \frac{n!}{r!(n-r)!} \tag{5.2}$$

$$f(x,y) = f(x)_X \cup f(y)_Y = f(x)_X + f(y)_Y + f(x)_X \cap f(y)_Y$$

$$f(x,y) = f(x)_X \cup f(y)_Y = f(x)_X + f(y)_Y + f(x)_X . f(y)_Y \tag{5.3}$$

When events f(x) and f(y) are independent, the last item may be zero

$$f(x,y) = f(x)_X + f(y)_Y + 0$$

where f = the experimental outcome.

f_X = experimental outcome of X;

f_Y = experimental outcome of Y;

x = an element of X;

y = an element of Y.

The statistical selection method [34] is employed in this work as it possess the potential to alleviate the problem of bias. An outline of the method is as follows.

In this statistical selection method, n is the total number of class used and r is the amount of class picked without replacement. Since the aim here is to minimise *bias*, randomisation is done according to equation (5.2). Two independent randomisations are performed and the results multiplied according to equation (5.3). Setting n =50 and r = 1 in equation (5.2);

$$f(x)_X == \frac{n!}{r!(n-r)!} = \frac{50!}{1!(50-1)!} = \frac{50!}{49!} = 50$$

Similarly,

$$f(y)_Y = \frac{50!}{49!} = 50$$

and we have

$$f(x,y) = 50 + 50 = 100$$

In f(x,y) (Table 5.1) we have 100 classes in which each class is repeated twice. In Table 5.1, each row represents pattern classes for one base classifier. The component classifiers are named NTW# (where # = 1, 2, 3, …). The arrangement and the reoccurrence of a class is random and independent.

Table 5.1: Randomised input classes. NTW# = Network, where # = 1,2,3,…,n.

	CLASSES									
NTW1	40	14	49	50	2	36	12	3	30	16
NTW2	45	26	13	1	39	17	18	47	11	9
NTW3	48	20	41	19	5	32	10	46	34	24
NTW4	25	22	42	33	6	28	27	23	44	29
NTW5	8	37	4	43	21	7	31	15	35	38
NTW6	49	48	22	24	2	39	44	47	3	32
NTW7	42	4	7	14	38	30	45	11	26	29
NTW8	5	16	33	6	10	1	9	50	18	25
NTW9	35	31	12	19	37	23	20	43	34	8
NTW10	15	41	13	36	28	21	27	40	46	17

The component classifier, named NTW# (where # = 1,2,3,…), are trained following the scheme depicted in Table 5.1. An entry in a row is a number that represent a class. As an

illustration, NTW2 refers to the second neural network. All numbers in the row corresponding to NTW2 refers to the classes on which NTW2 is trained, first class being 45, second class being 26, third class being 13, etc. The numbers, e.g. 45, 26, are instances of fingerprints. They are files containing real-life finger-prints [91] printed in various forms. This method of arrangement is known as the *statistical arrangement* method. The *statistical arrangement* is employed in the MCS to alleviate bias and maintain independence of base classifiers. An optimal arrangement of the conmponent classifiers has been found, experimentally, to be 10 classifiers and 10 classes per classifier.

For component base classifiers to make a significant contribution to an MCS, they must be as diverse and independent as possible. Dependence of performance on diversity and correlation measures decreases rapidly as the number of Neural Networks increases. Lam [79] states that orthogonality, complementarities and independence of a base classifier determine its inclusion in an MCS. Some designers like Mladenic [36], and Zouari [50] employ certain diversity measures in Neural Networks classifier selection. A criterion for Neural Networks selection is lack of error correlation among selected Neural Networks [109]. Dynamic methods such as Bagging, is employed by Gunter [119] while Boosting is employed by Freund [42]. The term Bagging (Bootstrapping and AGGregatING) refers to a selection mechanism for ensemble creation implemented by randomly drawing N training sample from a training set S of size n with replacement, and assigning a classifier to each group of samples drawn. The probability of being drawn is equally distributed over the training sample. Boosting is a different selection method from Bagging in that the probability of selection increases in favour of “difficult” pattern and decreases for “easy” patterns. The most widely used variant of Boosting is AdaBoost. AdaBoost.M1 [42] is a variation of AdaBoost for multi-class problems. In AdaBoost.M1 a classifier is assigned to each subset of training pattern drawn.

The next step that follows the selection of base classifiers according to the conditions specified here the arrangement of these classifiers.

5.2.2 Topology

Topology refers to the arrangement of base classifiers in an ensemble. Common arrangement methods are serial, parallel, and hierarchical arrangements. The parallel method is the most common arrangement of base Neural Networks in MCS. The introduction of dynamic self reconfiguration may enable the MCS to switch between these

topologies, depending on new environment, new tasks or both. During training and recognition, each base classifier utilises its normal training and recognition algorithm.

The decision to employ a parallel combination method in this work follows from the fact that the output of all component neural networks will be combined. All output of the base NN will be combined because the base neural networks are error uncorrelated and diverse. Parallel method of combination is known to incur high computation costs [109]. This problem is solved here due to the decision to employ RAM-based (weightless) neural networks. RAM-based neural network performs mainly logical mathematical computation and thus does not involve high computation costs such as would a floating-point computations for example.

The parallel method is employed in [109], while the serial method is employed by Austin [7]. Dima [26] proposes a hierarchical mixture of experts and the employment of dynamic reconfiguration to analyse robot dynamics.

5.2.3 Combination methods

The combination of component classifier output is called *classifier fusion*. Multi-classifier (MCS) design is usually problem dependent, which may imply data-dependency. The idea of flowchart figure 5.2 is to represent in a diagram a non-exhaustive combination strategy of MCS. When an MCS developer chooses an explicit data-dependent combination method of MCS, large prior experimentation is usually required to determine the composition of the ensemble. Each of these methods of MCS combination is very extensive and beyond the scope of this chapter with one exception. The one exception is the data-independent combination method which consists mainly of trained combiner. MCS with trained combiner usually require no prior experimentations because the component classifiers can adapt themselves to match the given problem. The MCS designed in this chapter with EPCN as a trained combiner is an example.

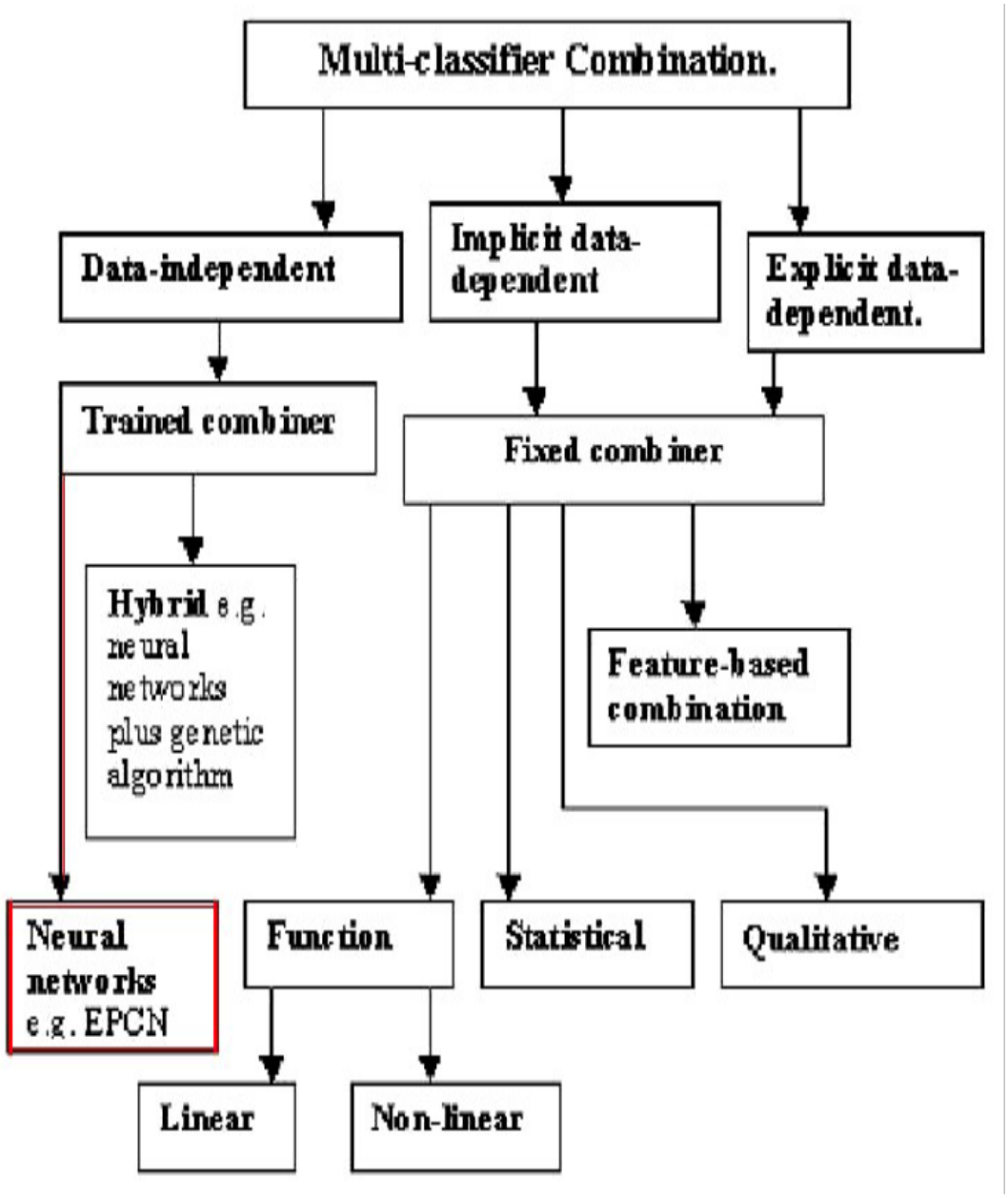


Figure 5.2: Multi-classifier combination scheme with respect to database.

Every output of each base classifier is a one dimensional vector of positive whole numbers. The one-dimensional vector output of all base classifiers are arranged into a multi-dimensional matrix, as shown in equation (5.4), at any time during combination.

$$CM(x) := \begin{pmatrix} n(x)_{1,1} & .. & n(x)_{1,j} & .. & n(x)_{1,c} \\ .. & .. & .. & .. & .. \\ n(x)_{i,1} & .. & n(x)_{i,j} & .. & n(x)_{i,c} \\ .. & .. & .. & .. & .. \\ n(x)_{r,1} & .. & n(x)_{r,j} & .. & n(x)_{r,c} \end{pmatrix} \quad (5.4)$$

Each entry in equation (5.4), $n(x)_{i,j}$, is derived from the output of EPCN of the type shown in figure 5.4. The resulting multi-dimensional matrix requires an encoding scheme which speaks the language of the trained neural network combiner. We require an encoding scheme such that equation (5.4) will convey knowledge of input space to the trained combiner when encoded appropriately. This is one of the main themes of this chapter.

Areas of multi-classifier combination has attracted intense research lately, most [110], [114], [76] of which maintains that a benchmark combiner is non-existent. The majority voting is suitable in situations where common consensus is required. But it ignores the fact that some neural network, though in minority, do produce the correct result [114], especially in area of their specialisation. Secondly, the existence of diversity is ignored by majority voting as one of the motivation for ensemble creation, but favours common consensus. Hansen and Salamon [48] showed that only when the nets make independent

errors does majority voting provide increased accuracy. Tumer and Ghosh [130] maintain that error-independence leads to better accuracy than a specific combination method.

Gunter [119] called *objective functions* the functions used in combination. *Qualitative combination* is used by Blue [11]. Duin [35] refers to intelligent combiners as *trained combiner*, and maintains that trained combiners outperformed fixed combiners. Roli and Giacinto [112] calls component classifiers *balanced classifiers* provided they are combined by any of the *fixed combination* method, and have zero or negative correlation. De Carvallo *et al.* [31] combined two Boolean neural networks in series. Prabhakar [105] grouped Multi-classifiers according to their output.

5.2.4 Experimentation

There are MCS designed for specific purposes, and they make explicit use of features of their database, for which they are required to classify, for the design. An MCS, derived from weighted Neural Networks, is specifically designed for fingerprint classification in Cappelli [107]. There Cappelli *et al.* identify two types of fingerprint classification: the exclusive classification and continuous classification. Continuous classification is specific to fingerprint classification, and refers to a multi-dimensional numerical feature vector which is obtained from a fingerprint. The vector is used in nearest neighbour (or similar distance measure) search to map close finger prints into cluster. This method inhibits "ambiguous" classification of a fingerprint from being exclusively classified as belonging to a cluster. Exclusive classification (also specific to finger prints) is a partitioning of fingerprint database into a given number of classes according to their macro-features. A class can only belong to one partition.

Most other experimentation methods on fingerprints rely heavily on minutiae and template matching of minutiae. Hollignum [53] employs the graphical method, Jain [66] uses point pattern matching, Wahab [132] employs structural matching techniques to minutiae, and Tico [129] uses transformation operation. To improve on these methods, Maio and Maltoni [90] introduce the detection of false positive. Hung [60], Prabhakar [106], Xiao [133] provides methods aimed at removing false minutiae, and Maio [90], [91], uses Neural Networks for minutiae filtering.

In this chapter, the MCS, derived from weightless Neural Networks, is adaptive and assumes no knowledge of the databases employed a priori. It is expected that the base Neural Networks are capable of detecting features necessary for their classification. Thus

this configuration should work for any large multi-class problem domain. A typical large multi-class problem domain is a biometric database such as fingerprint database. The proposed MCS configuration will be applied to biometric databases, specifically fingerprint databases that comprise of large classes with relatively few patterns per class, in an experimental set-up.

5.2.5 Performance measure

When inputs to the component classifier are randomly permuted and approximately equally distributed, it makes the performance independent of a specific arrangement of input to the component classifier.

The performance of a MCS is often compared to that of a single component Neural Networks that forms part of the MCS. Gabrys and Ruta [12] maintain that diversity measure has limited correlation with MCS performance. The paper states that MCS performance depends on careful selection of component classifiers. Generalisation performance of MCS should equal or exceed that of base classifiers. The most commonly used performance metrics are sensitivity and specificity. Sensitivity, $S_{n,}$ is defined as

$$S_n = \frac{T_p}{T_p + F_n} \tag{5.5}$$

where T_p = true positive;

and F_n = false negative.

It measures the ratio of positive patterns being correctly classified as positive to the whole pattern classes. Specificity, S_p, is defined as

$$S_p = \frac{T_n}{T_n + F_p} \tag{5.6}$$

where T_n = true negative;

and F_p = false positive.

It is a measure of the ratio of negative pattern being correctly recognised as being negative. Min *et al.* [77] uses a rejection criterion and reliability to measure performance. The rejection criterion and reliability are numerical quantities derived from fuzzy integral.

On databases such as fingerprints, the most commonly used performance measures are: True Acceptance Rate (TAR), True Rejection Rate (TRR), Predictive value positive (Pos),

and Predictive value negative (Neg). While performance measures such as False Acceptance Rate (FAR), False Rejection Rate (FRR), TERR, [106], [107], [1], and Equal Error Rate (EER), [96], [83], are commonly associated with biometric authentication of fingerprints. These are quantitative measures of the trustworthiness of results obtained. However every output of EPCN (see chapter 3), in recognition mode, includes a confidence measure. The confidence measure or the trustworthiness measure is a *scaled probability measure*, scaled by a value called the *division*.

5.3 Implementation of the Multi-classifier

As stated above, a Multi-classifier (MCS) is a system that fuses several base classifiers into one. In this context, a base classifier is a neural network that is known to be very good at a certain classification task but may be poor at other tasks. The MCS implemented here is data-independent in the sense explained in section 5.3. The direct method of base classifier selection is used and these base classifiers will be arranged in parallel since every output of all base classifiers will be combined. These decisions suggest the MCS architecture shown in Figure 5.3.

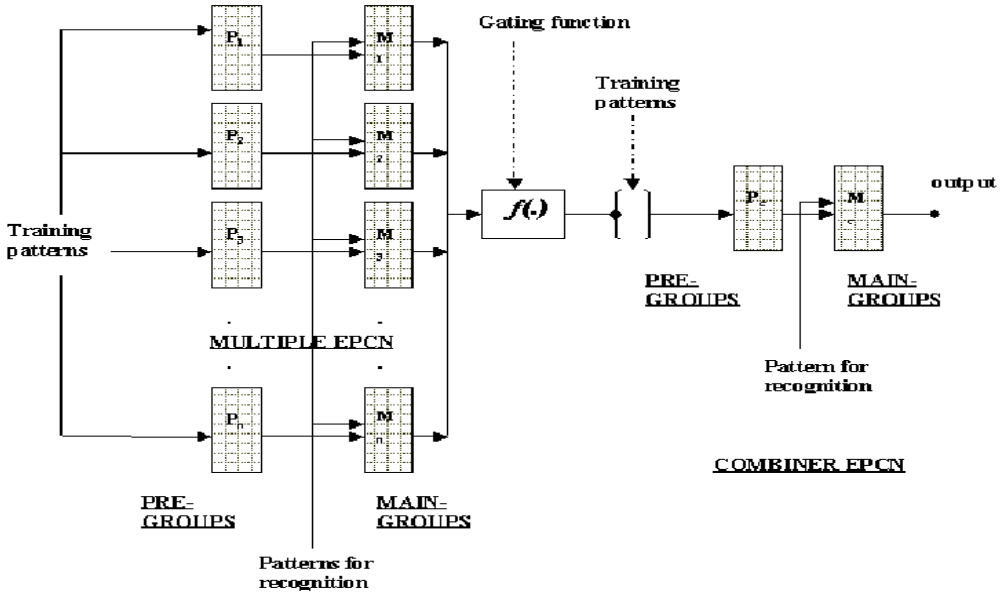


Figure 5.3. The MCS unit is divided into multiple EPCN group and combiner EPCN group. The multiple EPCN group consist of EPCNs in parallel. P_i = *pre-group*; M_i = *main-group*; i = 1,2,3,… *f(.)* = gating function. P_c = combiner's *pre-group*. M_c = combiner's *main-group*.

The specific component classifiers used, Fix-EPCN and rand-EPCN, are weightless *probabilistic classifiers* by reason of the nature of their output. Previous studies have shown the performance of both EPCNs to be well above 50% [84]. In the proposed system, each component classifier is trained upon, and thus only able to recognise, a subset of the available pattern classes. It is thus a challenge for the trained combiner to successfully combine the matrix of classifier outputs, i.e. equation (5.4), many of which will necessarily have produced incorrect classifications. The input pattern classes are randomly permuted, and evenly distributed according to equation (5.1) – (5.3) of section 5.2. A component classifier is assigned to each sub-set of input pattern classes. This ensures that there is no bias within any component EPCN. The selection of base classifier does not depend on input pattern classes. Rather a parallel method of arrangement of component classifier is made a priori, with randomization injected into the system parameter [77]. This ensures independence and de-correlation of the base classifiers. The address formation method of both EPCNs is their distinguishing features [85]. Their learning and recognition algorithm are equivalent as described in sub-section 3.2.1 and 3.2.2 respectively. Coupled with the fact that the system parameters of base classifiers are distinct and uncorrelated, all base classifiers are expected to produce distinct results during recognition.

Figure 5.3 is a schematic representation of the proposed MCS. The $[P_i, M_i]$ represent the component classifiers, i = 1, 2, 3…, and *f(.)* represents the *gating function*, while $[P_c, M_c]$ is the combiner. The $[P_i, M_i]$ component classifiers are fix-EPCN and rand-EPCN with varied system parameters, so is $[P_c, M_c]$ the combiner. A gating function is a function used for weighting, encoding, and synchronizing the output of base classifiers before combination. The novel component of this MCS is, *f(.)*, the gating function. The *gating function* in turn consists of various components. The relevant components of the *gating function* are the *combiner engine.* The *combiner engine* consists of the interpreter and converter, and is the subjects of the following sub-sections. The functional operation of the *interpreter* and the *converter* constitute an *encoding scheme* for the EPCN-combiner. The output of all component classifiers is combined using an *intelligent combiner* $[P_c, M_c]$, i.e. a neural network which in this instance is an EPCN with alternative configuration. Thus the combination method consists of an *interpreter*, a *converter* and an EPCN combiner.

5.3.1 Combiner engine – the coding scheme

The term *Combiner engine* refers to the EPCN combiner [P_c, M_c], and the gating function *f(.)* (see Figure 5.3). The gating function is made up of an *interpreter* and a *converter* which together produce the encoding scheme. To illustrate the function of the *combiner engine*, if the same character, e.g. "b", is trained to different Neural Networks as belonging to different classes. This means that the first class for classifier 1 is different from the first class for classifier 2. During the recognition phase, a correct classification by these Neural Networks requires that "b" be classified to their corresponding (respective classes to which it is trained) classes. But in classifier fusion stage, and without the *interpreter*, this same character will be converted to false and true pattern classes by the *converter*. For these reasons an *interpreter* is essential.

The intermediate outputs are weighted by the gating function only in special circumstances of input space overlap. Weighted results are treated as patterns to be processed and not results. In the very special circumstance when intermediate results are weighted, they are no longer results but class patterns meant to be binarised and served as input to the combiner-EPCN. Traditionally, an MCS is usually named after the constituent neural networks and/or with respect to the arrangement of the network. The thesis follows this tradition of naming the MCS with respect to the constituting neural networks. Because all component classifiers are weightless neural networks, it naturally follows that the MCS which is hierarchical composition of the weightless NNs, is a weightless MCS. Employment of an external gating function (any type) for input/out pattern pre-processing does not affect the weightless MCS both in composition and in learning/generalization behaviour, or in any other way.

When a Genetic Algorithm (GA) is applied to MLP to modify its parameter (for example), it does not change the name of MLP to something else, and it may also not change the main behaviours of MLP. But it only enhances the performances of MLP in the special circumstances. It is the same analogy here. The gating function is even more restricted in this architecture because it applies itself only to the (fan-in) input pattern of combiner EPCN. The gating function is usually not an integral part of the MCS and it can also be employed with other types of MCS. The weightless MCS implemented here can also employ an alternative (any other) gating function. The author recommends that an MCS utilising the combiner engine as a gating function may not change its name simply because the combiner engine is being used since another gating function may yield

equivalent result. The thesis will adopt to use standard nomenclature in which when all component NNs are weightless NN, the MCS that is formed is a weightless MCS. In other situations such as Lorrentz [81] it may be called a hybrid MCS.

5.3.2 Interpreter

The function of the *interpreter*, as the name implies, is to accept equation (5.4) at its input, and make sense of it to the converter. The configuration information, the output, and confusion matrix of the base classifiers are also accessible to the *interpreter*. Based on these, a decision per pattern is made by the *interpreter* as follows. A weighting strategy is employed on the output of the base classifiers when inputs overlap. This weighting strategy affects only those outputs corresponding to the region of input overlaps. A zero weight turns off an output of a base neural networks with respect to a given pattern class, while a weight greater than zero turns it on. The interpreter is incapable of eliminating a base classifier; rather it inhibits undesirable outputs with respect to certain classes. This inhibition depends on input space overlap, configuration, and performance on that class. As an illustration, if character "b" is trained to one neural networks as class 1, and trained to another neural networks as class 2. During the recognition phase of the component classifiers, correct classification by the base classifiers requires the first neural networks to classify "b" as class 1 and the second neural networks should classify it as class 2. The *interpreter* informs the converter that the outputs from the two base classifiers are correct classifications of "b", and will be weighted by their respective probabilities.

5.3.3 Converter

The converter encodes the *Interpreter*'s integer output into binary. It makes use of an integer value, called *division*, to adjust its output. For example, consider the output of a component classifier to be [10, 10, 65, 25, 10]. The vector [10, 10, 65, 25, 10] represents a row vector of equation (5.4), and analogous to variable field “decision output” of Table 5.2. The vector is converted by *combiner engine* to Figure 5.4(b).

(a) Unit encoding

(b) Engine encoding

Figure 5.4: Encoded information by the gating function *f(.)*, (a) Unit encoding from the combiner unit; (b) Engine encoding from the *combiner engine* .

A similar encoding scheme in [73] yields Figure 5.4(a). The general methodology is as follows. Any decimal number, N, is expressible in the form:

$$N = \left[\left(d_n . d_{n-1} ... d_0 \right)_{10} \right] \tag{5.7}$$

Equation (5.7) is expandable in a polynomial P(d)in equation (5.8);

$$P(d) = d_n . 10^n + d_{n-1} . 10^{n-1} + ... + d_0 . 10^0 \tag{5.8}$$

The following algorithm (5.9) converts equation (5.8) to its binary equivalent.

$$\begin{aligned} b_n &= d_n \\ b_{n-1} &= d_n + d_n . \beta \\ b_{n-2} &= d_{n-2} + d_{n-1} . \beta \\ &... \\ b_0 &= d_0 + d_i . \beta \end{aligned} \tag{5.9}$$

$i = 0,1,2,...n$

$d_i =$ constituent integer.

The least probable classes are indicated with low values, 10, in the vector [10,10,65,25,10]. The low values are omitted by the *combiner engine*. In the vector [10,10,65,25,10], 65 occurs in the 3rd position. It is binarised according to equation (7) to (9), and occurs in the 3rd row in Figure 5.4(a), while the same number is binarised and occurs in the 1st row in Figure 5.4(b). The position of 65 in the vector [10, 10, 65, 25, 10] is 3. This position number 3 is binarised to 00000000011 and occurs as the 3rd row in Figure 5.4(b) by the *combiner engine*. The pattern concerned here is most probably (the highest probability 0.65) classes three and more probably class four (with probability

0.25). For this reason, 25 is binarised, and it occurs in row 4 in Figure 5.4(a), while in Figure 5.4(b) it occurs in the second position. Both Figures (i.e. 5.4(a) and 5.4(b)) indicate that the most probable class to be the 3rd class and following this is the 4th class. Secondly, a reversed bit of the most probable class, 3, is also passed on, only by *combiner engine*, to the *converter*. Thus, it is included in Figure 5.4(b) in the 4th row. The reversed bit serves to make the information detectable by the EPCN-combiner. Figure 5.4 is the form of pattern accepted by EPCN-combiner. The functional activity of the *interpreter* and the *converter* constitute what might be referred to as the *coding scheme* to the EPCN-combiner. The engine encoding is deployed in the experiment of section 6.4.

5.3.4 EPCN-combiner configuration

An example configuration of EPCN combiner is shown in Table 5.2. In the field, *ntppc* (i.e. *n*umber of *t*raining *p*attern *p*er *c*lass), each number represents the number of training patterns per class. The field, *pg-layer* (*pg* stands for *p*re-*g*roup), is the number of layers in the pre-group.

Table 5.2: An output of EPCN

Structural component	Field elements
decision output:	[1188 154 174 1485 165 464 164 287 726 193]
n-tuple:	2
division:	5000
ntppc:	[5 5 5 5 5 5 5 5 5 5]
mg-layer:	2
pg-layer:	3

Other important fields in this structure are division and *n-tuple*. *Division* is used by the converter to adjust its output, and used during learning and recognition for other scaling purposes. When summing out the output, the *division* is also used to scale output probabilities. The field, *n-tuple*, specifies the tuple-size. The combiner's *main-group's* number of layer is specified in field mg-*layer* (where *mg* stands for *m*ain-*g*roup).

5.3.5 Comparison to other similar coding scheme for multi-class problems

The combination strategy addressed here is comparable to Bayes combination strategy, and to majority voting. The combination method introduced is much similar to Bayes combination strategy than to majority-voting method. The combination strategy in [86]

employs a *combiner unit* which also bear many similarities to the combination strategy introduced in this section. For this reason, comparisons will be drawn between *combiner engine, combiner unit,* and majority voting as and when possible.

The *combiner engine* has the following advantages over *combiner unit* in [82]:-

The combiner engine encoding produce a reduced pattern size, as compared to combiner unit encoding, and thus has the following implications:-

- Leads to a reduced storage requirement.
- A reduced size of layers in the pre- and main-group (recall that the size of a layer equals the size of input pattern). This leads to a reduced amount of data being processed at a point in time.
- An increase in speed of execution since less data will be processed at any given time.

A consequence of using the *combiner engine* instead of *combiner unit* is that a single EPCN will now be able to combine large class sets and this combination possibility no longer depends, to a large extent, on a specific configuration.

The structural difference between the *combiner unit* and the *combiner engine* is the utilisation of an *interpreter* in place of *decision maker*. This leads further to the following advantages of the *combiner engine* over the *combiner unit*:-

- The interpreter considers more information with respect to individual base classifier.
- More efficient synchronisation between base classifiers.

We claim that the performance of the *combiner engine* will supersede other similar combiner methods both in speed and in percentage of pattern recognized when employed in the same experimentation (section 5.4).

5.4 Experimentation on the MCS

In this section, the MCS designed according to sections 5.3 from weightless Neural Networks (EPCN), is tested. It is adaptive and assumes no knowledge of the databases employed a priori. It is expected that the base neural networks are capable of detecting features necessary for their classification. Thus this configuration should work for any large multi-class problem domain. A typical large multi-class problem domain is a biometric database such as fingerprint database. The MCS configuration will be applied to fingerprint databases, which comprise of large classes with relatively few patterns per class, in an experimental set-up.

Specifically, the aims of this experiment are to investigate the effect of the *combiner engine*, and the effect of approximately equally distribution of input data (sub-section 5.3.1) on recognition performance of EPCN based MCS when a large class database with few pattern per class are utilized. The database used for this experiment is DBA1 from the Third International Fingerprint Verification Competition 2004 (FVC'2004) [43]. It is available at http://biometrics.cse.msu.edu/fvc04db/index.html. These fingerprints are real (as opposed to synthetic) fingerprints.

5.4.1 Pre-processing

Intensive pre-processing is avoided to prevent local distortions which such procedures will add. Instead, global pre-processing steps (see section 4.3) are employed in this work. It is a test for this network to determine its own capability to recognise and classify the fingerprints amid the noises, distortions, and deformations already present in the source. Thus no corrections were made for the following deformations present in the source fingerprints:

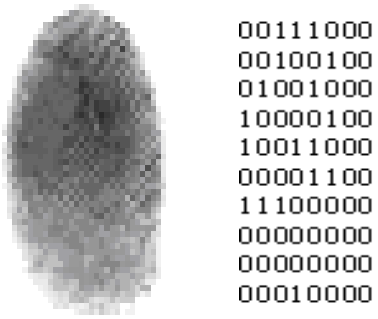
(1) Shifts: Change of position or direction
(2) Rotations: Angular shift of an object with respect to a fixed point.
(3) Intensity changes: Irregular or changes in illumination.
(4) Occlusion: Covering of part or preventing of part of fingerprint from being processed.
(5) Pinch: This is squeezing or nipping of parts of a fingerprint.
(6) Punch: May be a hole or missing parts of a finger print.

The conditions under which these fingerprints are collected are as specified in [43], [58]. However, the following pre-processing operations are performed. This method of pre-processing avoids the aforementioned pre-processing problems.

- The fingerprints are filtered to remove noises. The same level of noise-filtering was applied equally to all patterns, and the level of noise varies from pattern to pattern.
- An edge-enhancing filter is then applied.
- This is followed by the binarisation of the fingerprints.

5.4.2 Experiment on the MCS

The advantage of using an artificial neural network (ANN) instead of minutiae analysis [87], [106], is that a global operation on the images is less sensitive to local distortion that normally occur during extraction of local features. Fingerprints in this database are extracted into a directory, an example of which is shown in Figure 5.5(a). The fingerprints are then filtered, centred, edge-enhanced, and then binarised;



(a) (b)

Figure 5.5: (a) Fingerprint, (b) filtered and binarised picture of (a)

(See Figure 5.5(b)). Each finger printed in various forms represents a class. The following points represent the main experimental processes:-

- Prior to the commencement of learning, input classes are randomly permuted as stated in equations (1) to (3), and evenly distributed as shown in Table 1, to remove any input bias from the network. Training set is sampled randomly from the whole fingerprint classes without replacement.
- Fifty (50) classes are employed in this experiment. Ten base classifiers are initialised in parallel, and each network is trained on ten classes arranged as shown in Table 1. In Table 1, each row represents pattern classes for one base classifier. The component classifiers are named NTW# (where # = 1, 2, 3 …). An entry in a row is a number that represent a class. This arrangement of input patterns ensures that each class is trained on (at least) two networks.
- During the recognition phase each network is required to classify *all test patterns* belonging to all classes that participated in the learning phase.
- The outputs of these base networks are combined as in equation (4), and then encoded to the trained combiner by the *combiner engine.*

- For comparison, the outputs of these base networks are also combined using the *combiner engine* in majority voting mode.
- *This experiment could not be performed on combiner unit due to memory issues* (when the same memory size is used).

Performance measure:- Every output of EPCN, in recognition mode, includes a confidence measure. The confidence measure, or the trustworthiness measure, is a *scaled probability measure*, scaled by a value called the *division*. We ensure that inputs to the component classifier are randomly permuted and evenly distributed as explained in sub-section 5.2.1. This ensures that performance is independent of a specific arrangement of input to the component classifier.

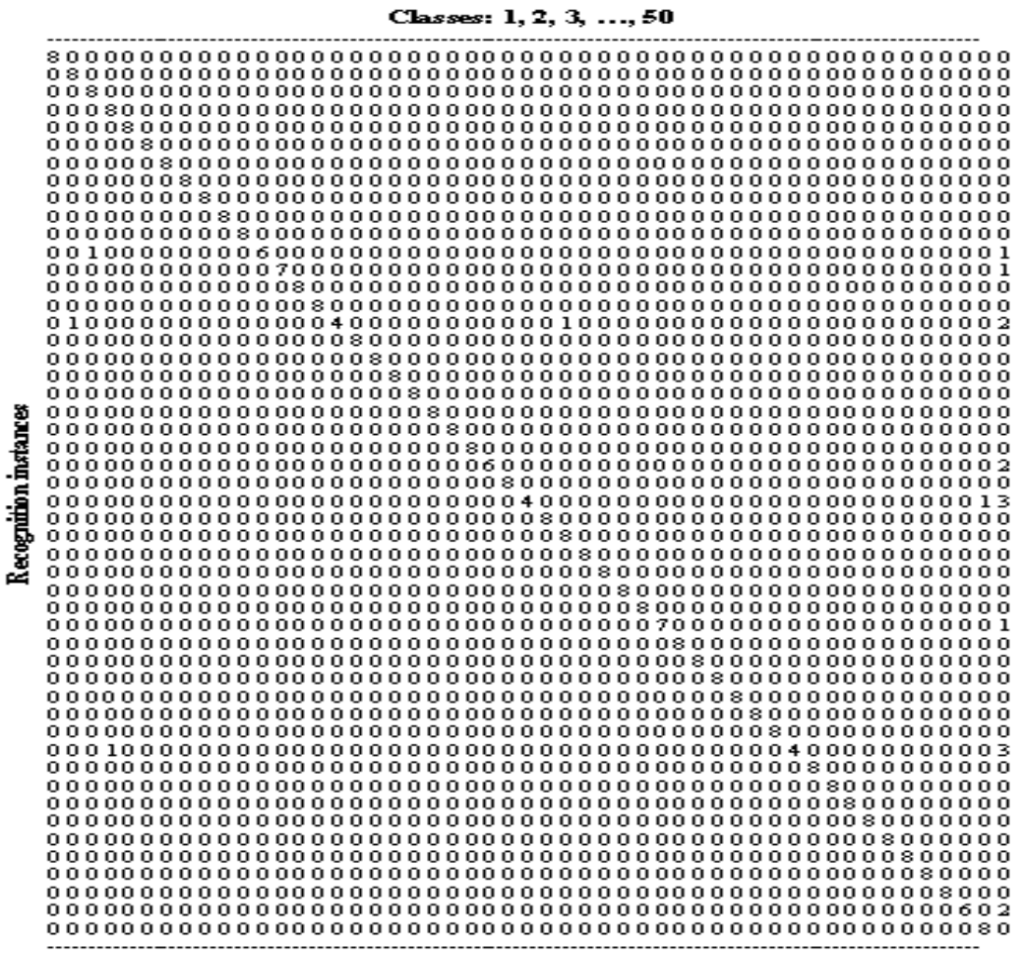


Figure 5.6: Normal combination mode: The confusion matrix from EPCN combiner. Columns 1 to 50 represent classes. The last column is unclassifiable patterns.

An example of an output of EPCN is as shown in Table 5.2 of section 5.3.4. The important field in this structure is the *decision output* of the neural networks. This states that the network is trained on ten classes. When a pattern is presented to it for recognition, it is

(1188/5000) % (the confidence measure) likely to belong to class one, (154/5000)% likely to belong to class two, etc. The summation of the numbers in the field *decision output* should equal the variable in the field *division*, in this case 5000.When a pattern is presented to it for recognition, it is (1188/5000) % (the confidence measure) likely to belong to class one, (154/5000) % likely to belong to class two, etc. The summation of the numbers in the field *decision output* should equal the variable in the field *division*, in this case 5000. Thus results from EPCNs, and Multi-experts dependent on EPCN, are inherently with trustworthiness measure.

5.5 Results

When the experiments are performed, each component classifier produces a structure of type shown in Table 5.2 for every given pattern meant for recognition.

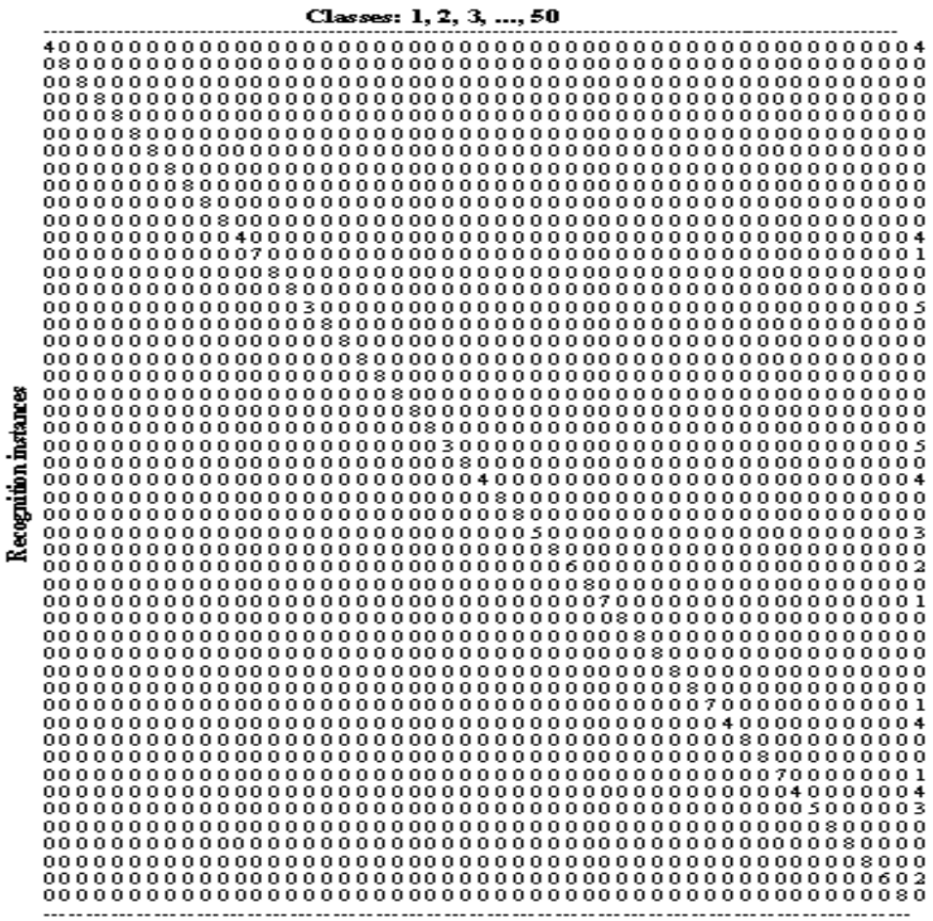


Figure 5.7: Majority Voting mode: The confusion matrix from EPCN combiner. Columns 1 to 50 represent classes. The last column is unclassifiable patterns.

Based on these and other configuration information the *combiner engine* is able to produce the confusion matrices of figure 5.6 and 5.7. Figure 5.6 is produced by the combiner unit in normal operation mode, while figure 5.7 is produced by the *combiner engine* in majority voting mode. Figures 5.6 and 5.7 are known as confusion matrices. In figure 5.6 and 5.7, each row represents recognition instances, while each column represents the classes to which the patterns are classified. The last column represents ambiguous cases. The values along the diagonals represent the number of patterns that are correctly classified, while the off-diagonal elements represent the number of patterns that are wrongly classified.

5.6 Analysis

The percentage recognition is calculated from the combiner's confusion matrix, figures 5.6 and 5.7, and recorded in table 5.3.

Table 5.3: Summary of results obtained when the experiments in sub-section 5.4.2 were performed. Column 1 and 3 represents class numbers, while column 2 and 4 represents the percentage (%) of patterns recognised in a test set.

Classes	% Recg.	Classes	% Recg.
1	100	26	50
2	100	27	100
3	100	28	100
4	100	29	100
5	100	30	100
6	100	31	100
7	100	32	100
8	100	33	87.5
9	100	34	100
10	100	35	100
11	100	36	100
12	75	37	100
13	87.5	38	100
14	100	39	100
15	100	40	50
16	50	41	100
17	100	42	100
18	100	43	100
19	100	44	100
20	100	45	100
21	100	46	100
22	100	47	100
23	100	48	100
24	75	49	75
25	100	50	100
average	93.250		90.988
average		92.119	

5.3(a)

Classes	% Recg.	Classes	% Recg.
1	50	26	50
2	100	27	100
3	100	28	100
4	100	29	62.5
5	100	30	100
6	100	31	75
7	100	32	100
8	100	33	87.5
9	100	34	100
10	100	35	100
11	100	36	100
12	50	37	100
13	87.5	38	100
14	100	39	87.5
15	100	40	50
16	37.5	41	100
17	100	42	100
18	100	43	87.5
19	100	44	50
20	100	45	62.5
21	100	46	100
22	100	47	100
23	100	48	100
24	37.5	49	75
25	100	50	100
average	82.031		82.521
average		82.276	

5.3(b)

Table 5.3 shows the performance on most pattern classes to be 100%. A row consists of two numbers; the 1st number is the class while the 2nd number represents the % of pattern recognised. The 1st average quoted is the average of the corresponding column, while the second average represents the average of the two. Table 3(a) represents the result of the combination strategy in majority-voting mode while 3(b) represents the result of the combination strategy in normal mode. From Table 5.3, when EPCN-combiner is switched

to majority voting mode, the average performance is 82.276%, while EPCN-combiner in normal combination mode gives an average of 92.119%, a difference of about 10%.

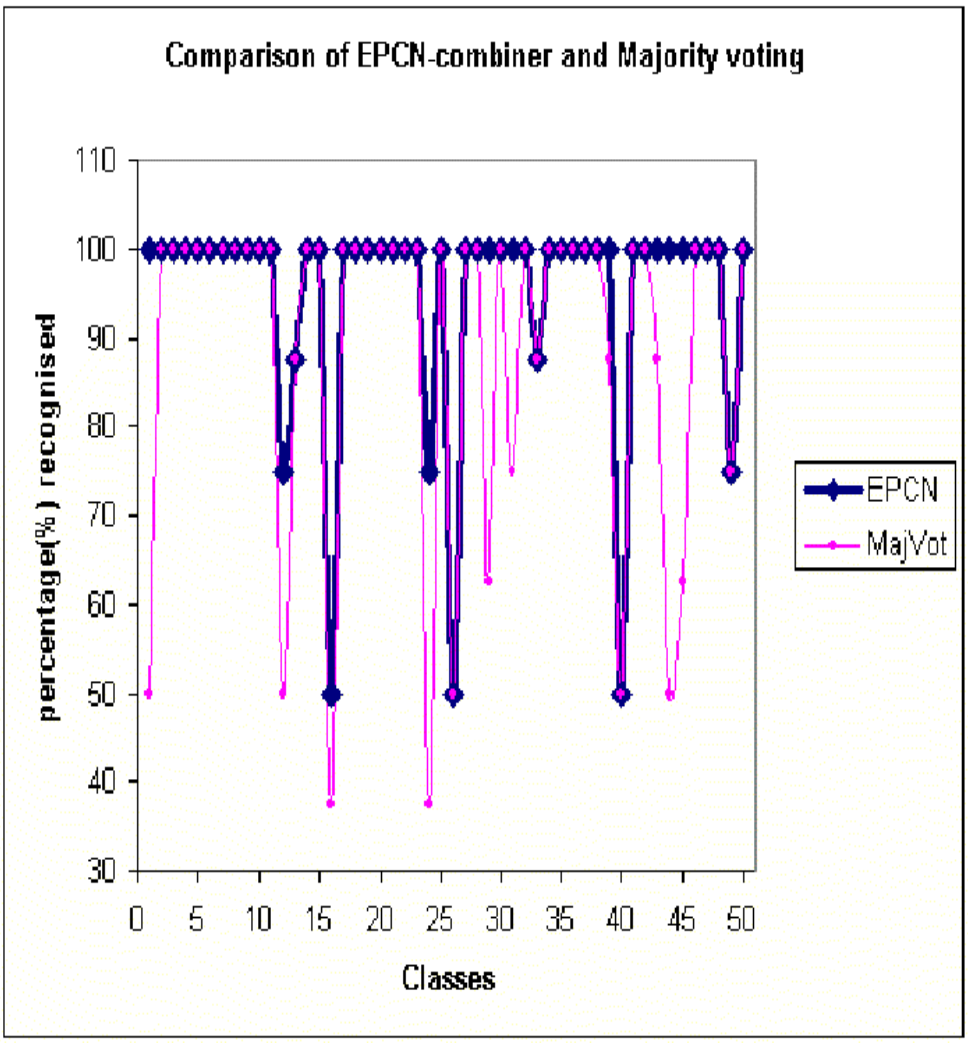


Figure 5.8: Comparison of EPCN combiner and Majority Voting (Majvot) combination method when applied to base neural networks.

That these aims:

- A reduced pattern size for the EPCN-combiner:- This leads to a reduced storage requirement;
- A reduced size of layers in the pre- and main-group (recall that the size of a layer equals the size of input pattern) of the EPCN-combiner:- This leads to a reduced amount of data being processed at a point in time;
- An increase in speed of execution;

has been achieved could be inferred from the output of the *gating function*, Figure 5.3. From Table 5.3, it could be deduced that EPCN as combiner outperformed Majority voting method where a large class database, with few patterns per class, is concerned.

Since the *combiner unit* is faced with difficulties when confronted with experiments of sub-section 5.4.1, this indicates that *combiner engine* is applicable in situations where *combiner unit* is not. As seen in figure 5.8, from class 35 to 50, the variation from 100% of percentage recognition decreases. Thus the problem of saturation and overloading has been solved.

5.7 Comparison of FVC2004 with MCSPCN

The databases employed in Fingerprint Verification Competition (FVC'2004 competition) are divided into two categories, the “light” category and the “open” category. The light database is required for algorithms characterized by low computing resources, limited memory usage, and small sized fingerprints. The open category database is meant for all other algorithms. All participating algorithms are independently developed by various academia and industries. The databases are benchmark databases [58] for real-life fingerprint identification, and verification algorithms. In the competition, all participating algorithms have the same input-output format, and they are tested in the same environment. Most participating algorithms employ *fingerprint matching* techniques. All results emanating from these algorithms are similarly formatted and quantified to enable direct comparison between them. Methods used in quantifying results rely on Receivers Operating Characteristics (ROC) of certain parameters. The choices of units mainly used are False Match Rate (FMR), and False Non-Match Rate (FNMR). These matchings refers to matching of minutiae, ridges, or some other features characteristic of fingerprints. The point at which FMR equals FNMR is known as equal error rate (EER). The rate employed in FMR, FNMR, and EER refers to percentage of fingerprints matched. The ROC analysis originates from statistical decision theory, and was originally introduced during World War II. Thereafter, in the 1960s, ROC analyses become prominent in medical analysis/diagnosis. Though ROC has gained popularity in other disciplines, it has not been used in neural system analysis. But since the database hereby processed originates from Biometry, it enhances direct comparison to employ ROC analysis on the result of EPCN-combiner, and majority-voting when they have been trained on biometric databases. It is noteworthy that ROC analysis does not indicate, with confidence, how good the MCS performs. Similar situation has been noticed Yager [97] among others. Figure 5.9(a) and

5.9(b) shows a summary of ROC from FVC2004 while Figure 5.10 shows the ROC of EPCN combiner, and figure 5.11 shows the ROC of Majority Voting combination method.

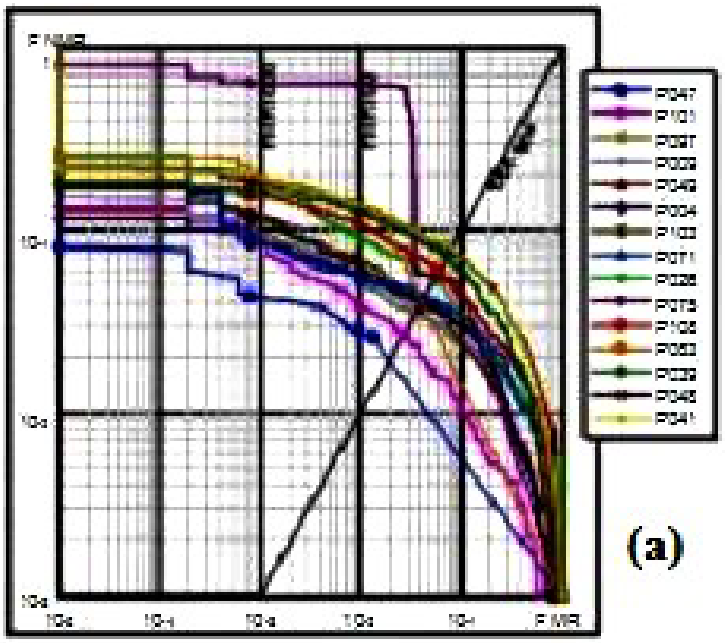


Figure 5.9(a): Open category; ROC curves from FVC2004 n DB1 (only top 15 algorithms are shown) [27].

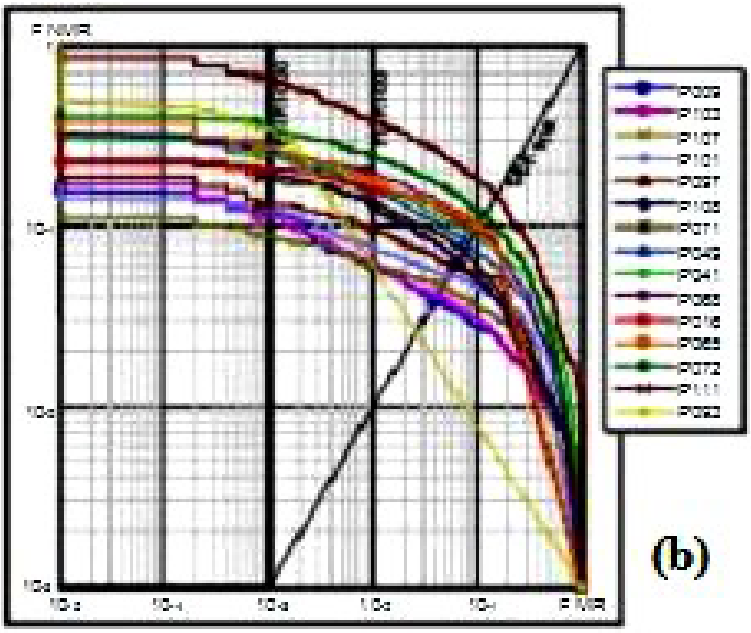


Figure 5.9(b): Light category; ROC curves from FVC2004 n DB1 (only top 15 algorithms are shown) [27].

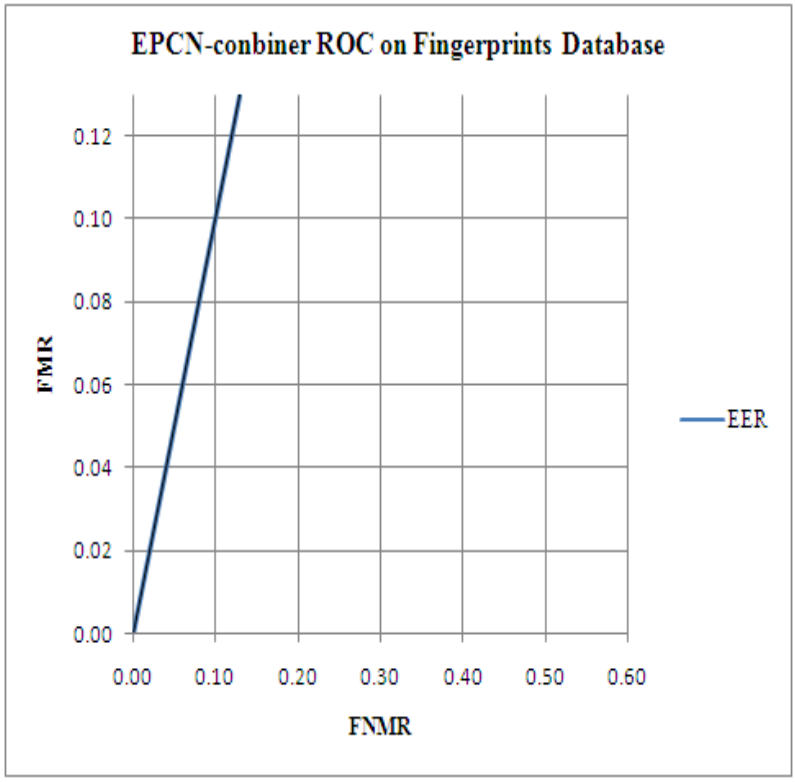


Figure 5.10: EPCN-combiner ROC on Fingerprint.

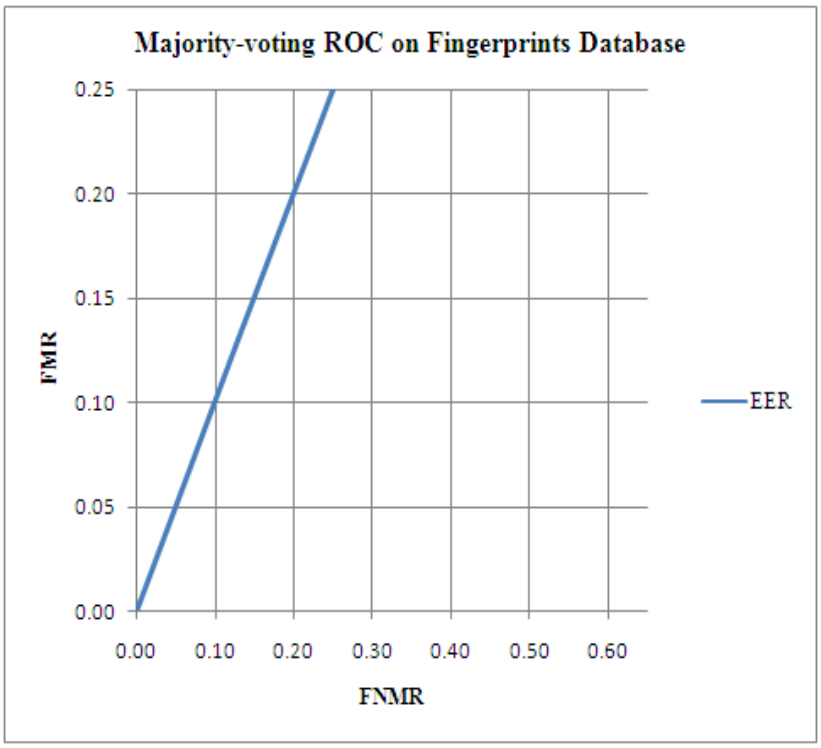


Figure 5.11: Majority Voting ROC on Fingerprint.

One of the main reason for larger variation in EER of FVC2004 and that of EPCN-combiner is that in FVC2004, almost all the available database are employed and the participating algorithms vary immensely. The multi-classifier utilized in this chapter does not employ any matching algorithms. It is a multi-classifier employed in different environment (from that of FVC2004 environment) whose input-output format is distinct and much different from that of FVC2004 participating algorithms. Numerical quantitative comparisons are to be treated with skepticism. The EER of FVC2004 may be compared to the error rate (i.e. (100 – x) % of performance, where x represent the performances quoted in table 5) of MCSPCN. In FVC2004, EER vary from 1.97% to 100%, while in MCSPCN, the error rate vary from 0% to 100%. Detailed results of FVC2004 are contained in [48] while that of MCSPCN on fingerprints are contained in Table 5.3

5.8 Comparison of the Multi-classifiers employed within the Thesis

- The multi-classifier of chapter four explores unconstrained handwritten numeral classification while that of chapter 6 employs different problem domain.
- Secondly, data partitioning methods for learning the multi-classifier of chapter 4 is different from data partitioning method for learning the multi-classifier of chapter 5. Within an input to a base classifier of chapter 4, the classes are ordered. This is not the case when fingerprint database are employed (i.e. in chapter 5).
- One of the main aims of employment of handwritten characters (chapter 4) on the multi-classifier is to explore its usefulness and examine its weaknesses. The *defects of the multi-classifier* are also examined with respect to large-scale multi-class fingerprint database.
- This lead to the development of combiner engine in chapter 5 to replace combiner unit.
- We may summarise that the combiner unit of chapter 4 supports large-scaled multi-class database poorly while the combiner engine of chapter 5 supports large-scaled multi-class database very well.
- Biases, overloading, and saturation effects are considerably minimized with the multi-classifier of chapter 5; this is not the case in chapter 4.

5.9 Summary

An advanced encoding scheme has been introduced for Multi-classifiers employed on problem domains possessing a large number of distinct classes with limited available training data. Results presented demonstrate an improved performance for the encoding scheme over that achievable via the majority voting method on a large-class database. It is noteworthy that the Multi-classifier arrives at a good level of performance despite the level of deformations, distortions, and noise present in the source fingerprints.

RAM-based Multi-classifiers does not template matching as do traditional fingerprint verification methods. Thus this MCS could be regarded as an intelligent, automatic, and template-free fingerprint recognition system.

The input arrangement of chapter 4 is systematic while statistical arrangement method is utilised in chapter 5. The bias problem was solved here via the implementation of the statistical arrangement method.

The overloading and saturation problems associated with large-class databases were solved by the sub-setting strategy of input, and appropriate selection of number of base classifier that participated in the Multi-classifier systems.

A combiner unit was employed in chapter 4 while a combiner engine is utilised in chapter 5. Results show that the combiner engine accommodates larger input classes.

Area of further research and development includes the application of this combination strategy to a wider range of large-class problem domains.

6. AN FPGA-BASED WEIGHTLESS NEURAL NETWORK HARDWARE

This chapter explores the significant practical difficulties inherent in mapping large artificial neural structures onto digital hardware. Specifically, a class of weightless neural architecture, the Enhanced Probabilistic Convergent Network, is examined due to the inherent simplicity of the control algorithms associated with the architecture. The advantages for such an approach follow from the observation that, for many situations for which an intelligent machine requires very fast, unmanned, and uninterrupted responses, a PC-based system is unsuitable, especially in electronically harsh and isolated conditions, The target architecture for the design is an FPGA, the Virtex-II pro which is statically and dynamically reconfigurable, enhancing its suitability for an adaptive weightless neural networks. This hardware is tested on a benchmark of unconstrained handwritten numbers from the National Institute of Standards and Technology (NIST), USA.

This chapter also examines the potential offered by adaptive hardware configurations of a class of the weightless neural architecture Enhanced Probabilistic Convergent Network targeted on a Virtex-II pro FPGA which is reconfigurable. The reconfiguration and adaptive capability of the Enhanced Probabilistic Convergent Network is a highly adaptive architecture offering a very fast, automated, uninterrupted response in potentially electronically harsh and isolated conditions. The reconfiguration and adaptive potential of EPCN is explored by the employment of a benchmark of unconstrained handwritten numerals from the Centre of Excellence for Document Analysis and Recognition (CEDAR).

6.0 Introduction

Early years of neural network hardware research involve multiple parallel processing elements (PE). Amsdahl was one of the main early researchers into neural network hardware. He showed that a task is worth parallelizing only when it is possible for about 50% of the task. “If about 50% of the task is parallelize, the total speed increase is only twofold; when more than 90% of the task can be parallelize could a speed increase of tenfold or greater occur.”: This is now known as Amsdahl’s law. Amsdahl law is a good guide as to when parallelizing leads to speed increase. Generally, only when about 90% of the processes constituting the task could be parallelized is parallelizing worth doing.

For a neural network to be implemented in hardware, adequate consideration must be given to floating-point, and recurrence decimal. Generally, precision is limited to certain number of significant figure. Since most neural network could not be implemented wholly

in hardware, different variety of hardware neural network has emerged and are categorised as:

1) Off-chip learning: - Off-chip learning occurs when learning the neural network is done on a computer using high precision. Weights/results of the learning process is downloaded onto a chip where the classification or recognition occurs.
2) Chip-in-the-loop learning: - Chip-in-the-loop learning is a situation whereby the forward propagation part of the learning algorithm occurs in the chip, but update and calculation of new weight value occurs on a computer.
3) On-chip learning: - On-chip learning and classification occurs entirely on chip. The EPCN is implemented on-chip, leading to limited precision of calculations. The EPCN implementation on-chip is also due to the fact that small amount of calculations are involved, thus discretization of values have very little effect on its performance because numerical errors are very small.

Previous chapters have implemented and applied neural networks, algorithms, and multi-classifiers for various purposes. In this chapter, the algorithms of EPCN will be considered in a hardware implementation. There are situations and environments that require the usage of neural network and do not demand urgency; these conditions are suitable for software-based EPCN. The conditions of emergencies (speed) and adverse surroundings motivate the consideration of EPCN algorithms in hardware.

Large-class databases and large artificial neural structure requires more time as compared to small ones. So that the hardware implementation of EPCN algorithms is motivated by the need to save time and resources while maintaining the same level of performance as compared to the software equivalent.

In the process of implementing the hardware equivalent of EPCN, use will be made of the learning and recognition algorithms of chapter 3. Learning and reasoning [25], in a digital hardware, may lead to adaptation and reconfiguration. Neural networks have shown to be well suited to learn from examples and adapt to non-linear environments, but many variants are rather resource intensive and therefore prohibitive in practical embedded applications [118]. However, one class of neural networks is more suited to implementation in hardware - the so-called weightless neural networks can be well matched to RAM (Random Access Memory) because their learning and recognition algorithms are mainly associated with reading from and writing to memory.

The aims and objectives of this chapter is to present the architecture, and the implementation of an adaptive RAM-based neural network, called Enhanced Probabilistic

Convergent Network (EPCN) [36], in a reconfigurable FPGA. Hardware implementations of EPCN are attractive for the following reasons:

1) Compact size and low power consumption compared to a PC based implementation (i.e. it becomes deployable in areas where a PC may not).
2) The binary weights of RAM-based neural networks are contained in RAM, and functions are converted to simple logic gates (AND, NOT, and OR). The logic combinations required to operate the EPCN are of lesser computational intensity than calculus operations.
3) Regular structure: Generally, RAM-based Artificial Neural Network (ANN) are easier to implement on hardware due to their regular structure.
4) The use of reconfigurable IC like FPGA to implement neural network allows fast prototyping and lends itself to modifications at low cost. This makes it a suitable testbed prior to large-volume production.
5) A well designed VHDL based hardware will allow a significant increase in processing thouroughput compared to a software based execution on a general processor.

Some authors envisaged that Machine Intelligent Quotient (MIQ) may also be a measure of its performance. Chalfant [22] introduced a MIQ and proposes the analysis of system architecture and configuration as the criteria for its measurement. Commuri [25] maintains that architecture of adaptation and learning at all levels of hierarchy simplifies the measurement of MIQ. Learning of a neural network by reconfiguration is demonstrated in [124] using a Virtex-II 6000 FPGA. In [122], Simoes employs ALTERA MAX + PLUS II in the implementation of Goal Seeking neuron (GSN) on Eraseable Programming Logic Device (EPLD). The EPLD is used in classification of British mail postal addresses. Spaanenburg [124] implements two neural networks, one is a feed-forward network to solve the problem of spatial and temporal computing. The second is implementation of Cellular Neural Networks (CNN) for image processing. The FPGA used is Virtex-II 6000 and the learning of these networks were made to depend on reconfiguration capability of this FPGA. Freeman [39], designed a co-processor based on a binary neural unit known as Correlation Matrix Memory (CMM) which is used for approximate high-speed search and match operations on large datasets. Botelho [14] implements Goal Seeking Neuron (GSN), a RAM-based neural network, on Khepera mobile robot for control and navigation. The RAM-based neural networks in [124],[14], designed on FPGA were deployed in autonomous systems. Most of these systems are application dedicated systems, often for

one purpose only. However, the principled EPCN system is generic and highly scaleable. The EPCN when implemented on FPGA could be employed both for prediction and recognition It would thus become suited to a multitude of applications. In this chapter however, the hardware is tested on a benchmark of unconstrained handwritten integers from National Institute of Standards and Technology (NIST), USA.

The possibility of reconfiguration and adaptability of FPGA-based EPCN will be introduced. The motives for the exploration of its reconfiguration and adaptability are significant and beneficial – these will be explained in subsequent paragraph.

Research into reconfiguration of artificial neural networks (ANN) is an increasingly significant area of investigation. This arises partly due to the improvement in performance possibilities offered in that it becomes possible for an ANN, when implemented in digital hardware, to be capable of adaptation and reconfiguration during learning [25]. Adaptation may also be in response to nonlinear environment. However, adaptation and reconfiguration may incur a high computation overhead, more so in practical applications [115]. This high computation overhead is however minimised in the class of neural network investigated in this chapter, the weightless neural network. This follows from the observation that the less the computation requirement, the faster an ANN is able respond to new input. This reduction in response time becomes very large when the ANN is implemented in hardware.

ANNs may also be grouped depending on the principle behind their implementation. Those whose behaviour closely mimics the intelligence of natural being e.g. the genetic algorithm, and those designed from mathematical concepts. Weightless neural networks, also called RAM based neural networks [7], are a subgroup of those designed from mathematical concept, in this case mathematical logic concept. Bledsoe and Browning in their pioneering work [10] (around 1959) made the first attempt to base their design of neural network on mathematical logic concept. More sophisticated networks have naturally been developed subsequently. These include the implementation of Enhanced probabilistic Convergent Networks (EPCN). The EPCN is an enhanced form of PCN [55]. The specific enhancements are as detailed in [85]. EPCN is a feed forward neural networks incorporating supervised learning with the addition that the mathematical logic is minimised even further when EPCN is implemented in a hardware.

A harware implementation of ANN offers significant advantages to a purely software implementation due to increased speed. For a weightless NN, the mathematical logic is of a reduced complexity than is the case with alternative NN when implemented in a digital

intergrated circuit (IC) - this allows an increase in speed. These advantages, amongst others, motivate the work of this chapter.

The aims and objectives of this chapter are two fold. One is to present the architecture and implementation of an adaptive RAM-based neural network, the Enhanced Probabilistic Convergent Network (EPCN) [85], in a reconfigurable FPGA. The second is to explore the reconfiguration and adaptive properties of the FPGA-based neural network.

The remainder of this chapter is organized as follows. Section 6.1 presents an overview of the EPCN, while section 6.2 introduces its hardware configurations. The experiments to test the configuration possibilities of FPGA based Hardware architecture of EPCN and its results are presented in section 6.2. The experiments and results obtained are presented in section 6.3. The experiments and results of re-configurability (or adaptive behaviour) of EPCN is presented in section 6.4. Analyses of results are presented in section 6.5. The chapter concludes with areas of further research and development in section 6.6.

6.1 The Enhanced Probabilistic Convergent Network

The architecture of EPCN consists primarily of four component layers as explained is section 5.2. It includes an optional feedback path (represented by dashed arrows) from the merge layer of the main group to the main-group. Each layer consists of component neurons which are themselves made up of storage locations known as RAM-locations as shown in Figure 5.2 of chapter 5. Details of the learning and recognition algorithms are contained in chapter 3.

6.1.1 Other similar weightless Neural networks

Almost all hardware implemented weightless NN are derived from either of these units: -

- WISARD discriminator [5]
- Correlation matrix memory [74]

To date, these units are used in various combinations to design weightless neural networks. A typical state-of-the-art design is employed by Bin Azar [47] who utilizes a WISARD discriminator to design a weightless NN for robot navigation. Azar [47] states that WISARD discriminator does not exhibit generalisation inherently. In his implementation, memorization and generalisation abilities were achieved by setting 120 neurons manually. The CMM relies on bitwise OR of input space during learning, and dot-product during recall phase. This find application in the design of C-NNAP [74].

Following is a comparison between EPCN designed in this chapter and a typical state-of-the-art design of FPGA based neural networks.

Table 6.1. Comparison of FPGA based, typical weightless neural network, and EPCN

Typical weightless NN	EPCN
Discriminator or correlation matrix memory utilized	No discriminator or correlated matrix memory utilized
May not be an n-tuple classifier	An n-tuple classifier
Does not favour generalisation	Inherent generalisation
May require manual training e.g. [3]	Does not permit manual training.
Binary ("1" or "0") output	Vector output – scaled probability values. Positive integer output only.

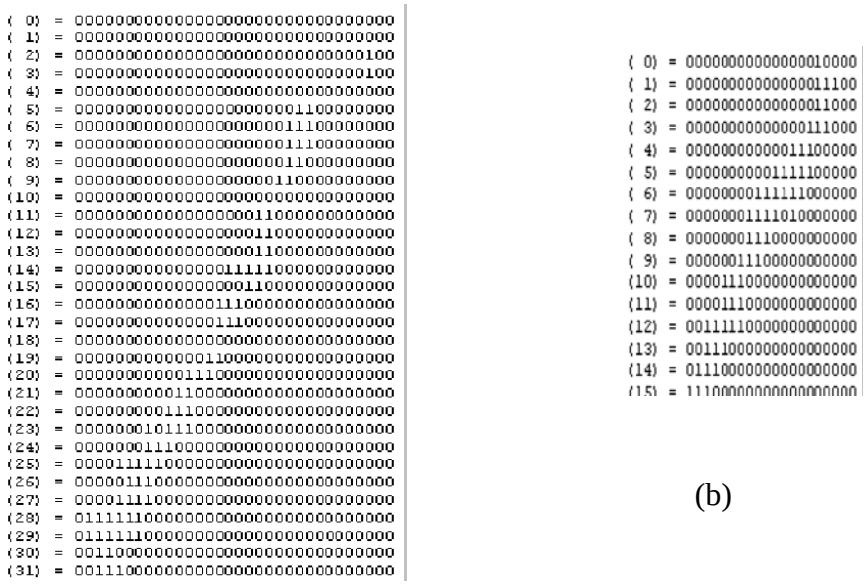
Having compared various FPGA based neural network, the architecture of EPCN will now be presented.

6.2 The FPGA-based hardware architecture of EPCN

In this section, the architecture of EPCN is proposed, by the thesis author that forms a complex hierarchical system. The design is sub-divided into pre-processing input data, core modules of EPCN, hashing function (unit), reconfiguration, and memory management.

6.2.1 Pre-processing

The EPCN's pre-processing steps include the reading in of the input data, or querying a terminal of input source. The EPCN expects the input values to be expressed in binary number. A compression algorithm, the Lempel-Zif algorithm [68] is included in the pre-processing steps. Most of the common identical data points in the classes will be removed by Lempel-Zif algorithm in order to permit real-time rescaling of input pattern as and when required. For example, the input, Figure 6.1(a), is pre-processed resulting in Figure 6.1(b) during input processing.



(b)

Figure 6.1: The pre-processing; (a) is pre-processed resulting in (b).

6.2.2 The EPCN Hardware

The EPCN is here described using the hierarchical system of design. The design is synthesised by Xilinx ISE during which EPCN is analysed and converted into digital circuit components. Figure 6.2 shows the main block modules constituting the EPCN architecture. In Figure 6.2, the train-block and the recognise-block are both connected to the input pre-processing block via the control unit and the hashing function that produces the addresses. The control unit initiates pre-processing when data are available at the input. After the completion of pre-processing of the input training data, it initiates the training processes (section 3.2.1). The train-block signals a finish flag when training completes. On reception of learn-flag-complete, the control unit checks the recognition input for data. When data is present, pre-pro cessing is done for the pattern meant for recognition. When the pre-processing-stop flag is detected by the control unit, the recognition block starts the recognition processes (section 3.2.2). The output block is monitored by the control unit through a feedback system. Iteration of the recognition processes stops when values in the output block are stable, by querying the output block, or after a pre-defined number of iteration steps.

The overriding majority of the EPCN block architecture consists of memory. Its functional behaviour is concentrated on data flow from and to these memory locations. The memory location in EPCN is described as single-port block RAM driven by a registered read address and a synchronous write operation.

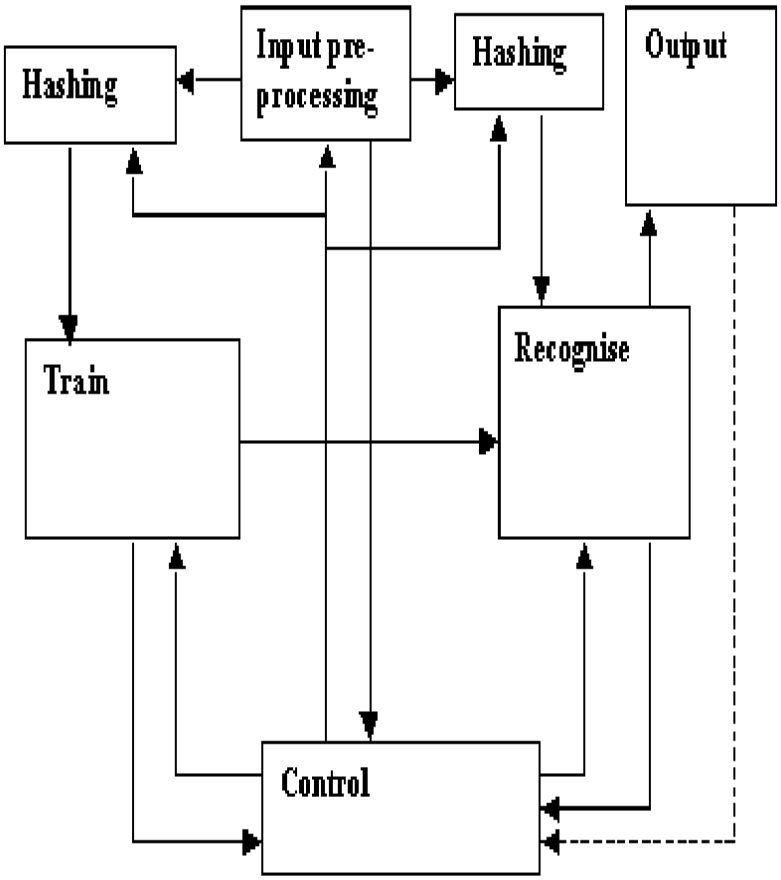


Figure 6.2: The block-diagram of EPCN FPGA architecture.

6.2.3 Hashing

A hashing function is often used to search and retrieve information from memory. Such a hashing function as employed by Freeman [39] is based on bit-folding, XOR, and pseudo-random number generator.

In this chapter however the hashing function implemented is based on XOR and Maximum-length Shift-register [67] code. A maximum-length shift-register code generates a systematic code with desired output length;

$$n = 2^m - 1 \tag{6.1}$$

Where m is the information bit derived from input pattern. The code words are normally generated by m-stage digital shift register with feedback. The generation of the code words depend on parity polynomials h(p) given by equation (6.2);

$$h(p) = p^k + h_{k-1}p^{k-1} + ... + h_0 p^0 \tag{6.2}$$

The maximum-length shift-register codes (MLSR) are dual of cyclic Hamming codes. Bits in the pattern become the information bits. The hashing is used for address (connectivity) formation. Data will be written to or retrieved from the LUT-RAM location whose address is so formed. Examples to illustrate this are given below.

Example I: when m = 3; equation (6.1) becomes

$$n = 2^m - 1 = 2^3 - 1 = 7;$$

This means that 7 addresses are required. In equation (6.2) the parity polynomial h(p) becomes;

$$h(p) = p^7 + h_6 p^6 + ... + h_0 p^0 \tag{6.3}$$

```
tcol =
{0000000000000000000000000000100 00000000000000000000000000000100
0000000000000000000000000000100 00000000000000000000000000000100
0000000000000000000000000000100 00000000000000000000000000000100
0000000000000000000000000000010 00000000000000000000000000000011
0000000000000000000000000000001 00000000000000000000000000000000}
tclas =
{0000000000000000000000000001001 00000000000000000000000000000110
0000000000000000000000000000101 00000000000000000000000000000111
0000000000000000000000000000011 00000000000000000000000000000001
0000000000000000000000000001000 00000000000000000000000000000100
0000000000000000000000000000010 00000000000000000000000000000000}
ttuple =
{0000000000000000000000000000111 00000000000000000000000000000111
0000000000000000000000000000111 00000000000000000000000000000100
0000000000000000000000000000010 00000000000000000000000000000110
0000000000000000000000000000101 00000000000000000000000000000011
0000000000000000000000000000001 00000000000000000000000000000000}
trow =
{0000000000000000000000000000110 00000000000000000000000000000110
0000000000000000000000000000110 00000000000000000000000000000110
0000000000000000000000000000010 00000000000000000000000000000101
0000000000000000000000000000011 00000000000000000000000000000001
0000000000000000000000000000100 00000000000000000000000000000000}
```

Figure 6.3. Formation of addresses by hashing from input patterns. This is prior to the learning process.

In equation (6.3) it is seen that the coefficient of p^7 is 1. h_i, (i = 0,1,2,…7) is such that $h_{k-1}p^{k-1}$ is an integer between 7 and 0. This is a constraint to be satisfied. Most of the values of $h_{k-1}p^{k-1}$ will be zero. Looking at Figure 6.3, it is seen that non of the values assigned to "ttuple" is greater than 7, the corresponding RAM location will be read from or written to

as the case may be. The RAM-location is found in a layer whose address is also generated by hashing, as values assigned to "tcol" variable, where "tcol" represents address of a layer. There it is seen that all values between 0 and 3 have been generated. To distinguish between wanted zeros and unwanted zeros, wherever h(p) has values greater than 3 or less than 0, this is set to 2^n and this make the location inaccessible.

Example II: Suppose ten connectivity are required none of which should be greater than 10?.

Answer: Recall that $2^4 > 10 > 2^3$. So that when m = 4; equation (6.1) becomes

$$n = 2^m - 1 = 2^4 - 1 = 15;$$

and in equation (6.2), the parity polynomial h(p) becomes;

$$h(p) = p^{15} + h_{14}p^{14} + ... + h_0 p^0 \qquad (6.4)$$

In equation (6.4) it is seen that the coefficient of p^{15} is 1. And h_i, (i = 0,1,…15) is such that $h_{k-1}p^{k-1}$ is an integer between 15 and 0. Since no connectivity should be greater than 10, h(p) is set to 2^n for those values that are not required. An example is shown in Figure 6.4, here the variable "rclas" shows all addresses derived lies between 0 and 10 inclusive. The "rclas" represents addresses of a neuron in the recognition phase.

To distinguish between wanted zeros and unwanted zeros, wherever h(p) has values greater than 10 or less than 0, this is set to 2^n and this make the location inaccessible.

Other addresses are derived similar to example I and II. Recall that information bits in a pattern characterise that pattern, and thus the connectivity is reproducible.

```
rcol =
{00000000000000000000000000000110 00000000000000000000000000000110
00000000000000000000000000000110 00000000000000000000000000000110
00000000000000000000000000000010 00000000000000000000000000000101
00000000000000000000000000000011 00000000000000000000000000000001
00000000000000000000000000000100 00000000000000000000000000000000}
rclas =
{00000000000000000000000000000010 00000000000000000000000000001001
00000000000000000000000000000110 00000000000000000000000000000101
00000000000000000000000000000111 00000000000000000000000000000011
00000000000000000000000000000001 00000000000000000000000000001000
00000000000000000000000000000100 00000000000000000000000000000000}
rtuple =
{00000000000000000000000000000111 00000000000000000000000000000111
00000000000000000000000000000111 00000000000000000000000000000010
00000000000000000000000000000110 00000000000000000000000000000101
00000000000000000000000000000011 00000000000000000000000000000001
00000000000000000000000000000100 00000000000000000000000000000000}
rrow =
{00000000000000000000000000000110 00000000000000000000000000000110
00000000000000000000000000000110 00000000000000000000000000000110
00000000000000000000000000000100 00000000000000000000000000000010
00000000000000000000000000000101 00000000000000000000000000000011
00000000000000000000000000000001 00000000000000000000000000000000}
```

Figure 6.4: Formation of addresses by hashing from input patterns. This is prior to the learning process.

6.2.4 Memory management

As indicated previously, the memory location in EPCN is described as single-port block RAM driven by a registered read address and a synchronous write operation in Xilinx's LUT-RAM. The memory management is monitored by the *control unit*, see Figure 6.2. During a read operation from a location, a write operation is disabled with respect to that location. During a write operation to a location, any read operation from that same location is disabled.

As concerning the buses, a read/write enable/disable operation depends on the addresses formed from the pattern. The FPGA consists of configurable logic blocks (CLBs). These CLBs are inter-connected by buses. Activating a bus, and which bus is being activated depends on if its address is formed. Activating a bus is necessary prior to a read/write operation; otherwise a read/write operation is not possible on that bus. A read and write addresses, with respect to one bus will not be formed at the same time. It is either a read address or a write address. It ensures that values are not written to and read from the same bus at the same time. This is commonly referred to as bus contention.

6.2.5 Reconfiguration

The structural architecture of EPCN and the size of its neuron are adaptive, changing with learning and classification. During learning and classification, an integer number called the *division* is required for *adjustment* purposes, [85]. The term *adjustment* refers to multiplying the integer value in a RAM location by the *division* and dividing by the number of training patterns per class. The *adjustment* is necessary for all classes to be treated equivalently when the number of pattern per class varies between classes. *Tuple-size* is the number of bits sampled from input data (at once) that characterize features of that data. For a *tuple-size* of n, $2^n - 1$ bit are sampled.

In practice the maximum size and structure of EPCN is naturally limited by the available hardware resources. The number of *pre-group layers*, the number of *main-group layers*, the *tuple-size*, and the *division* are often referred to as system parameters. The size of pre-group layer and the size of main-group layers are modifiable alongside the reconfiguration process.

The number *division* used during various *adjustment* phases could be chosen within a value from 1 to 2^{15}. This is the binary address range that fits in memory on FPGA. The possibility of the variability in system parameters is vital to static and dynamic reconfiguration. Modification to the value assigned to *division* is done by prefixing "constant divisn" with a "generic" statement. This is normally done before training and a recognition session pair.

The EPCN reconfiguration file is stored in Programmable Read Only Memory (PROM). Since the golden configuration is stored in revision 0 for FPGA's self-test, the EPCN reconfiguration file is stored in revision 1. The source-select switch is used to select any of the revision at any time required.

The FPGA is pre-programmed with various possible configuration options. The config-select, SW8, is a group of three switches, the combination of which gives the selection of one of eight possible configurations of EPCN. The source-select, SW9, is a group of two switches, the combination of which gives the selection of source of configuration for EPCN.

Using the config-select switches in conjunction with config-source switches it was found that it supports to a maximum of:

- Tuple-size = 4;
- Pre-group layers = 5;

- Main-group layers = 5;
- Class = 15;
- Number of neuron per layer 20-by-15;

The detection of pattern boundaries is automatically and dynamically done by the control unit (figure 6.2).

The functional activities of the pre-processing unit and the hashing function (unit) are monitored by the control unit to ensure that the size of the pattern used within the EPCN fall within the maximum neuron size possible. Secondly, it is always possible to adjust every pattern size appropriately before hashing. This solves the boundary problems. The solution to the boundary problems increases the range and type of input sources and reconfiguration flexibility of EPCN, which will be experimented on in section 6.3 and section 6.4.

6.3 Experimentations

The EPCN was designed and implemented using Xilinx ISE 9.21i. The NIST data base[1] has been used for testing the functionality of EPCN and has been found suitable for use. Testing was done by instantiating the EPCN in a test-bench and associating the NIST handwritten data set with its input.

The prototyping board is linked to the computer via a USB programming cable. Auto-recognition of the programming cable by Xilinx ISE enables the download of the bit file generated from EPCN, as shown in figure 6.5, and configuration of the board by impact (component of Xilinx ISE) using PROM file generated from EPCN.

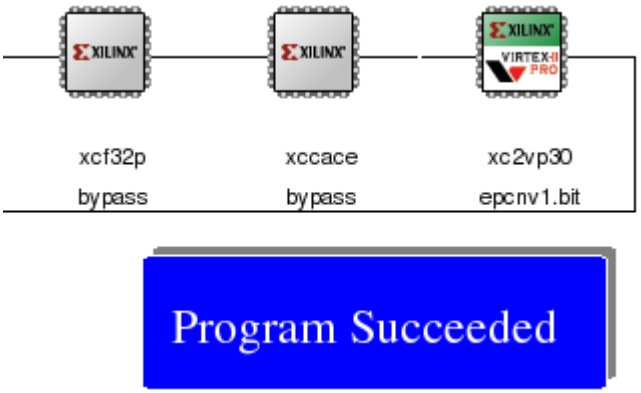


Figure 6.5: This shows that EPCN fits Virtex-II pro.

[1] National Institute of Standards and Technology (NIST) in Gaithersburg USA. NIST provide the handwritten simple form (HFS) of numerals, which were binarised and resize to 32-by-32.

To display the output of EPCN on board the FPGA, RS232 cable is connected via the com port and linked to the HyperTerminal at a baud rate of 2400 – 9600. Recognition results are displayed on the HyperTerminal in a PC. Internal to the Virtex-II pro FPGA is a 2MB SDRAM. External SDRAM at 2GB is also attached. This makes possible an increased number of LUTS generated and utilized during learning and recognition of EPCN in FPGA.

An example of the FPGA resource utilisation of EPCN is shown in Table 6.2. These are the resource requirements for the following EPCN architecture:

- Tuple-size = 3;
- Pre-group layers = 3;
- Main-group layers = 2;
- Class = 10;
- Number of neuron per layer 20-by-15;

Table 6.2: An extract of resource utilisation showing the conversion of EPCN to gate-level components.

Logic Utilisation	Used	Available	Used %
Total Number Slice Registers (SR)	1,612	27,392	5%
Total Number SR used as Flip-flops	1,611	1,612	99.9%
Number of 4 input LUTs	3,532	27,392	12%
Number of occupied Slices	2,492	13,696	18%
Number of Slices containing only related logic	2,492	2,492	100%
Number of bonded IOBs	24	416	5%
Number of GCLKs	2	16	12%

From Table 6.2, it is seen that the resource utilisation is relatively low for the given example network.

Using the NIST handwritten integers, the EPCN is trained on 0 to 9, and during recognition requested to recognise “1”. Data for training are selected from the training set

while patterns for recognition were selected from recognition set. Both training set and recognition set form disjoint sets.

```
grp.desout = {0000000000000000 0000011010001101 0000001010001101
0000001010010110 0000001001011001 0000000000000000 0000000000000000
0000000001111010 0000011010100010 0000000101101100}
        (0) = 0000000000000000
        (1) = 0000011010001101
        (2) = 0000001010001101
        (3) = 0000001010010110
        (4) = 0000001001011001
        (5) = 0000000000000000
        (6) = 0000000000000000
        (7) = 0000000001111010
        (8) = 0000001010100010
        (9) = 0000000101101100
```

Figure 6.6: Wrong recognition: A recognition result from EPCN when trained on “0” to “9”, and shown “1” in recognition phase.

Result of type figure 6.6 shown above is obtained when a pattern is shown to the network for recognition. Figure 6.6 shows the result when one input pattern is shown to the network for recognition. In this figure, the numbers 1,2,3,…,9, on the left-hand-side represents the classes while the binary numbers to the right-hand-side represents the probability (scaled by *division*) with which the

```
grp.desout = {0000000000000000 0000011010001101 0000011010001101
0000001010010110 0000001001011001 0000000000000000 0000000000000000
0000000001111010 0000011010100010 0000000101101100}
        (0) = 0000000000000000
        (1) = 0000011010001101
        (2) = 0000011010001101
        (3) = 0000001010010110
        (4) = 0000001001011001
        (5) = 0000000000000000
        (6) = 0000000000000000
        (7) = 0000000001111010
        (8) = 0000001010100010
        (9) = 0000000101101100
```

Figure 6.7: Ambiguous state: A recognition result from EPCN when trained on “0” to “9”, and shown “1” in recognition phase.

```
grp.desout = (0000000000000000 0000011010001101 0000011010001101
0000001010010110 0000001001011001 0000000000000000 0000000000000000
0000000001111010 0000011010100010 0000110101101100
0000000000000000)
          (0) = 0000000000000000
          (1) = 0000011010001101
          (2) = 0000011010001101
          (3) = 0000001010010110
          (4) = 0000001001011001
          (5) = 0000000000000000
          (6) = 0000000000000000
          (7) = 0000000001111010
          (8) = 0000011010100010
          (9) = 0000110101101100
```

Figure 6.8: Correct recognition: A recognition result from EPCN when trained on "0" to "9", and shown "1" in recognition phase.

pattern belong to that class. By varying the configuration of EPCN and showing it the same pattern for recognition, different other results are obtainable as shown in Figures 6.7 and 6.8. Three possibilities exist in recognition processes of EPCN. They are correct classification (Figure 6.8), ambiguous classification (Figure 6.7), and wrong classification (Figure 6.6). The binary numbers in Figure 6.7 and 6.8 has same meaning as in Figure 6.6.

6.4 Reconfiguration/Adaptive Experimentations

.The experimentation carried out here explores various configurations of EPCN. The EPCN was designed and implemented using Xilinx ISE. It was then tested in software by simulations prior to these experiments. The Source of database used in these experiments is:-

- The centre of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York, USA. Department of Computer Science. Unconstrained handwritten numbers from CEDAR were resized and binarised to 16-by-24 in dimension.

The config-select switch consists of three switches while the source-select switch is made up of two switches. In any session, learning or recognition, a combination of the three

switches on SW8 yields eight possible configurations which enables variation of configuration and system parameters of EPCN architecture. The config-source, consist two switches which are used to select sources of configuration. The various configurations of EPCN in these experiments reside in PROM (in Revision 1) and are fetched during reconfiguration automatically.

Preliminary investigations, that includes the available size of both the internal and external synchronous dynamic random access memory (SDRAM), has revealed that hardware resources supports maximum of 5 layers of pre-group and maximum of 5 layers of main-group. Guided by these hardware resource constraints, the experiment aims to explore various configuration possibility of EPCN and to determine the possible optimum configuration of EPCN. To this end, three experiments were performed on FPGA based EPCN using the database mentioned above. They are:-

- A case where division = 1000; main-group layers = 3; pre-group layer increase from 1 through to 5.
- A case where division = 1000; pre-group layer = 3; main-group layer increase from 1 through to 5.
- In the third experiment, the main-group layers = 3; pre-group layer = 3; division is increase from 100 through to 700.

Results of these experiments were recorded. They are graphically displayed in Figures 6.9, 6.10 and 6.11.

These same experiments have been performed on the software version of EPCN [85], by employing the same CEDAR database.

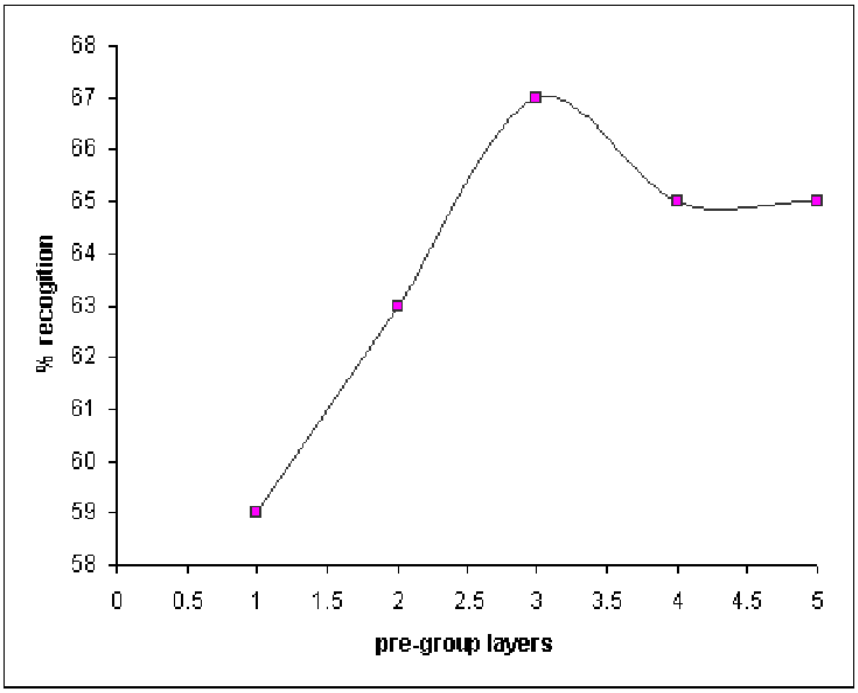


Figure 6.9: A plot of % recognition against number of pre-group layer; division = 1000; main-group layers = 3; pre-group layer increase from 1 through to 5.

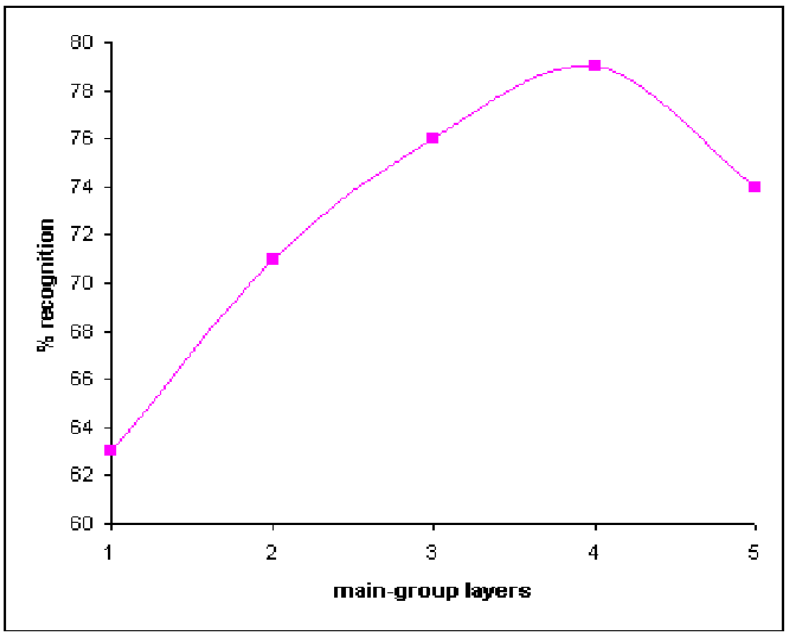


Figure 6.10: A plot of % recognition against number of main-group layer; division = 1000; pre-group layers = 3; main-group layer increase from 1 through to 5.

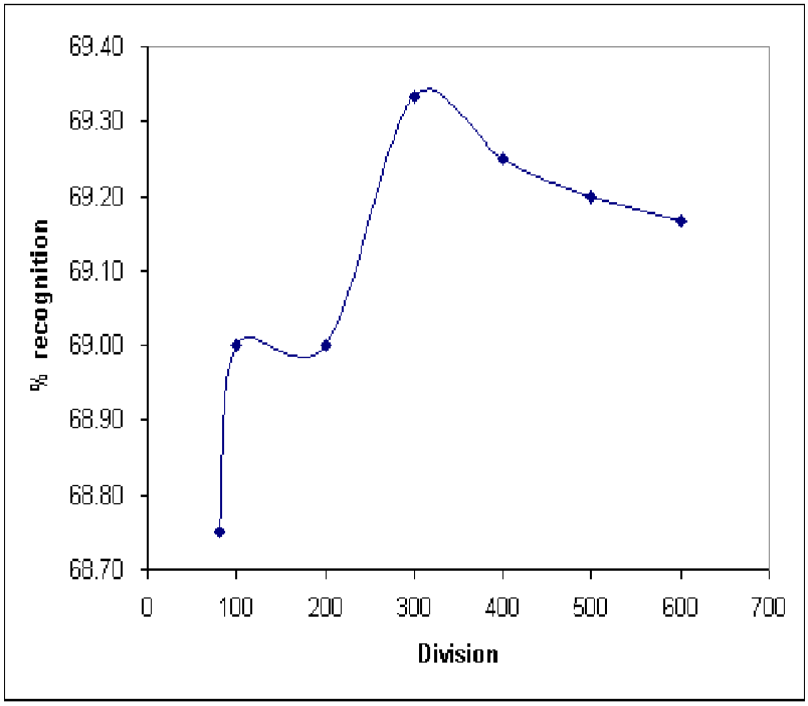


Figure 6.11: A plot of % recognition against *division*; the main-group layers = 3; pre-group layer = 3; division is increase from 100 through to 700.

6.5 Analysis

From the design and the example resource utilisation, Table 6.2, it could be inferred that static and dynamic variation both of precision (word length) and system parameters are possible. As these are fully supported by available resources, up to a certain maximum (see the sub-section 6.2.4).

Figures 6.6, 6.7 and 6.8 are result types obtainable from EPCN. Figure 6.6 is a case of wrong classification, while Figure 6.7 is an ambiguous state. Figure 6.8 shows correct recognition. These results are obtained when character "1"' is shown to EPCN. The different outputs, due to changes in system parameters, are indicative of the possibility of changes in decision due to changes in environment.

A pattern of 15-by-20 in size infers 300 neurons per layer. And there are many of such layers in any instance. This demonstrates the possibility of implementing an advanced and large weightless neural network, the EPCN, wholly on FPGA, the pre-processing steps inclusive.

6.6 Adaptive/Reconfiguration Analysis

The advantages of the FPGA implementation are that it is able to exploit the reconfiguration and adaptive capability of the EPCN which is advantageous for many situations for which an intelligent machine requires very fast, automated, uninterrupted responses, and in potentially electronically harsh and isolated conditions.

Figure 6.9 shows that the maximum percentage recognition occurs when the pre-group layer is 3. Figure 6.10 shows that the maximum percentage recognition occurs when the main-group layer is 4. Figure 6.11 shows that the maximum percentage recognition occurs when the division is 300.

Comparing Figures 6.9, 6.10 and 6.11, it may be observed that the performance is least dependent on *division* and that performance is most dependent on *main-group layers*. These results are identical to the result obtained from the PC-based EPCN when same input databases are used hence demonstrating the validity of the FPGA implementation. Further investigation and experimentation shows that the optimum system parameters are:-

Main-group layers = 4;

Pre-group layers = 3;

Division = 300;

Table 6.3 Comparison of Execution time

Platform	Execution time
Optically enhanced MLP	5.0e-10s
Virtex-II pro	7.5e-8s
PC: Intel® Core(TM)2 CPU 1.60GHz, 2039MB RAM	2.70e-2 s

These values are naturally dependent on the database employed and the number of classes. Also it is noteworthy that the hardware is of the order of 10^5 faster than an equivalent software implementation. A comparison between the speed of the FPGA-based EPCN, an optically enhanced Multilayer perceptron (MLP) [86][87], and a software based EPCN is shown in Table 6.4. There is clearly a substantial gain in speed by the FPGA-based EPCN over a software implementation. The EPCN is employed on human eye iris database and compared with other neural networks in table 6.5. Table 6.5 shows how different databases may give rise to different results (in Table 6.5). The results in Table 6.5 also depend on

configuration complexity and on source of database employed. The order of magnitude appears more general and thus more reliable.

Hardware constraints has been considered, and compared to equivalent software EPCN. These are tabulated in Table 6.4.

Table 6.4: Comparison of Hardware and Software implementation of EPCN. All numerical values referred are positive whole numbers.

HARDWARE	SOFTWARE
1) Pre-group layer is limited to 5 layers	Pre-group layers could be increased beyond 5
2) Main-group layer is limited to 5 layers	Main-group layers could be increased beyond 5
3) The word-length is limited to vary between 8 and 16.	The word-length could vary beyond these range.
4) Tuple-size is limited to integer values between 0 and 4	Tuple-size is not limited to integer values between 0 and 4.
5) The division could be given only 8 values per configuration set	The division could be given more than 8 values per configuration set

Experimental results further show that on comparing the software performances with the hardware performances:

1) Figure 3.5 is very similar in behaviour to figure 6.9.
2) Figure 3.6 is very similar in behaviour to figure 6.10.
3) Figure 3.7 is very similar in behaviour to figure 6.11

The FPGA-based EPCN may give the same result for exactly the same system parameters.

Table 6.5: Comparison of hardware EPCN with other neural networks implemented on other platforms. The database employed is *human eye Iris*

Methods/Database	Performance (%)	Platform/Circuit size	Time/Cycles
p-RAM (on 4 printed digits)	93%	IBM PC 66MHz	6200 sec.
EPCN (on handwritten numbers 0 - 9)	80%	Virtex-II Pro.; 2492 SLICEs	7.5e-8 sec
EPCN optimised (on Eye Iris)	99.17%	Virtex-II Pro.; 2402 SLICEs used	7.0e-8 sec
Information Theory (Mutual information, on Eye Iris)	99.05%	Mathlab 6.5	-
Back-propagation	-	Xilinx XC4005XL-PQ; 196 CLB used.	1879 ns
Space Vector Modulation (on Neural network)	max. Saving 83.22%	Xilinx XCV50hq240 FPGA, max. 1096 SLICEs	max. 149 cycles
Support Vector Machine (on SONAR, Diabetes, etc)	max. 94.4%	Spartan3 XC3s50pq208-5; max 1244SLICEs	-
Radial Basis Function (on pressure sensor)	97.20%	Virtex-II FPGA; 14334 SLICEs used	206 ms

Table 6.5 compares EPCN with other neural networks when employed in human eye Iris. From Table 6.5 and Figures 6.9, 6.10 and 6.11, it is deductible that the possibility of 16-bit word-length has a great effect on the identical result obtainable both from the hardware and the software EPCN.

6.7 Summary

The EPCN has been shown to be portably and wholly implement-able on FPGA. Pre-processing steps have been included in this design. The results demonstrate the possibility of implementation of a large, advanced, and adaptive weightless EPCN in a re-configurable FPGA. The FPGA based EPCN has been shown to be adaptive and reconfigurable. The results obtained here are comparable in performance terms to that of software-based EPCN. This is significant since hardware implementations of weightless classifiers are rare.

A shortcoming of these experiments is that interaction effects of these parameters were not investigated. This may be considered as an area of further experimentation and development. Other areas for further research include introduction of machine intelligent quotient (MIQ) as a means of self-assessment, and dynamic parameter tuning of the network.

7. CONCLUSION

7.1 Introduction

The advantages of RAM-based neural networks were enumerated in different places within the thesis. Introduction to weightless neural network afterwards focuses on EPCN whereby it was introduced. Modifications were made to configurations and connectivity, depending on areas of applications.

Chapter 3 reveals that other connectivity formation methods are possible. As this forms the basis of two types of PCN herein named fix-PCN and rand-PCN. We were able to explore various configuration possibilities for this neural network and, using handwritten characters, obtain a maximum performance of about 85% there. The levels of performance obtainable in this chapter could be improved upon.

This prompts investigations into possible multi-classifier systems. In chapter 4, a parallel multi-classifier was designed. Performances in excess of 85% were achievable. Further areas of study might be other forms of arrangement such as serial or hybrid arrangement of component classifiers. It is noteworthy that these depend on objectives of application. Low performance is not suitable for sensitive applications and also unsuitable in areas such as identification and security. In these areas, a very high % correct recognition is required. Secondly, a single PCN may be unable to cope with large-scaled multi-class databases due to problems among which are bias and saturation effects. A biometric database identifies itself with identification and security. Using PCN in a MCS in chapter 4, it is now possible to rapidly (and with high accuracy) classify large-scaled multi-class biometric databases. Though only fingerprint database was used in this chapter, there is no reason to suggest that it is not applicable to other large-class databases.

A novel advanced combination strategies were introduced in chapter 5. These combination methods were tested on fingerprint classification. After optimisation of the combination methods, performances of about 92% were obtained. Using PCN in a MCS with these combination methods, it is now possible to rapidly (and with high accuracy) classify large-scaled multi-class biometric databases. Its suitability for application to large-scaled multi-class database depends on the fact that external feature extraction procedures for input data were not required for correct classification. The MCS is capable of detecting

features of input database, autonomously, for their classification. This is an intrinsic property of this MCS, and applies to any input databases. Industrial application might be a good further step.

Accelerated character recognition and avoidance of collision may be required in electronically harsh and isolated conditions. Such conditions find itself in space exploration or in deep sea. Also, there are conditions for which an intelligent machine requires very fast, unmanned, and uninterrupted responses. These conditions make PC-based software very unsuitable, and form the content of chapter 6. Unsuitability of conditions conceived the idea of a hardware implementation. Attempts were then made in chapter 6 into a hardware design of EPCN. An FPGA based PCN was then applied to unconstrained handwritten character. This chapter demonstrates the possibility of implementation of an advanced RAM-based neural network wholly in FPGA. Following this is a question of its wider applicability, and also question of configuration issues.

Issues such as the advantages of a hardware implementation, when applied to sensitive and difficult area, were addressed in chapter 6. In chapter 6, system parameters of PCN and its various possible configurations were investigated. Results obtained were compared with other neural network systems. Hardware PCN was found suitable and applicable in areas mentioned in the previous paragraph. This is with a considerable speed and performance advantage as compared to many other systems.

7.2 Handwritten characters – Utilisation of a single Neural Network

The results of chapter 3 are obtained by the employment of a single weightless neural network, fix-EPCN and rand-EPCN independently, in turn on unconstrained handwritten characters. Filling of form by hand is still much in place in offices of industries and academics. The performance of EPCN in chapter 3 is at 87% maximum.

Application areas: - The advantage of EPCNs implemented in chapter 3 is that they could find application in offices where automated recognition of unconstrained handwritten characters, e.g. in bank cheques, in application forms etc., are required. An example application is given in section 3.4 which coincide with recognition of handwritten (hand writing of every day life) numerals. The EPCN in this chapter is unsuitable for applications that are security or health based.

Areas of further development: The coding of this EPCN is done in Matlab.

- The coding could be improved upon
- An optimisation algorithm could be introduced
- A multi-classifier could be constructed so as to improve the performances.

7.3 A Discussion on weightless Multi-Classifier Systems

Chapter 4 implements a multi-classifier system consisting only of weightless component classifiers. And performance rates of over 93% are obtainable. Though the multi-classifier does not utilize any of the classical improvement method such as Bagging or Boosting, the error rate of less than 10% has been obtained consistently.

Application areas: - The multi-classifier may be applied in low level security sector such as verification of absence or presence of materials in bulk. The disadvantage is its inapplicability in high-level security sectors. The multi-classifier is capable of accommodating more classes, and has a higher (about 20% more) percentage recognition rate as compared to a single EPCN. This Multi-classifier is suitable in recognition of hand-filled forms and handwritten characters. In addition to this, its suitability in biometric verification purposes is suggested.

Areas of further research: - This may include improving the coding to the combiner.

7.4 Classification of Large-Scale Multi-Class Databases

Also in chapter 4, the multi-classifier is tested on large-class databases (fingerprint databases). The component classifiers each were assigned ten classes. This arrangement does solve the problem of saturation, but does not solve the problem of bias. The performances of the multi-classifier on each class vary greatly. Such that for the multi-classifier to be very useful on large-scaled multi-class database, improvement on the MCS performances is required. Nevertheless, the performances of the multi-classifier on many classes were substantially above 62.5%. What has been achieved in the projects of this chapter are:

- The removal of saturation
- The utilisation of the multi-classifier on biometric databases serves as a pointer to the next-line-of-action, should it be required to utilize *normally* the multi-classifier on large-scaled multi-class databases.

Application areas: - It is suitable to medium-level security sectors such as biometric identification in industries. It is also suitable for character recognition such as found in (hand-filled) forms.

Areas of further research and development are: -

- The removal (or minimisation) of bias.
- Improvement in performance of the multi-classifier on large-class databases.
- Instead of employing a traditional method of boosting and/or bagging, it was decided to employ a novel method by replacing the gating function with a more advanced gating function so as to improve performance.

7.5 On Combination Strategy for Large-Scaled Multi-Class Database of Multi-Classifier

In chapter 5,

- A statistical arrangement method is introduced to solve the bias problem.
- A sub-setting strategy is introduced to solve the saturation problem
- A novel gating function is introduced to solve the problem of high memory demand, and increase speed.

These steps achieve a performance of 92% on average, on large-class databases, which in this case is a fingerprint database.

Potential Application areas: - An error rate of less than 10% implies that it may be used on medium to very large databases. The result obtained here is very good with respect to the database on which it is applied. It indicates that it may be used in industries that require low to medium level security, e.g. for human fingerprint recognition, handwritten recognition, level of alcohol in blood, etc. Generally, it is applicable in situations where the risk of false recognition is low.

The disadvantage is the low percentage recognition (average is 92%) when it comes to "high-level security" databases such as database of national security, databases related to health and hazards, etc. The reason is because the risk of false recognition is high (above 1%). It may be used on "high-level security" database only as an advisor since the output of the multi-classifier is a probability output.

Areas of further research: - Industrial deployment of the developed system.

7.6 Hardware-based EPCN

In chapter 6, the EPCN is ported to FPGA. The portability of EPCN to FPGA is significant and widens the areas where it may be applied. Though at an early stage of development, results shows that EPCN could be deployed in hardware for possible pattern recognition and prediction.

A significant achievement of the projects contained in chapter 6 is that it has become possible to implement an advanced and complex RAM-based neural network of this type, wholly in FPGA, which paves the way for other new areas of application of EPCN. The FPGA based EPCN is tested for adaptability and for reconfiguration. The results obtained (section 6.5) is good, and signifies:

- That FPGA based EPCN may be employ in offices of industries and academics where automated recognition of unconstrained handwritten characters is required.
- Wholly portable to hand-held equipment.
- May be employed in harsh surroundings.

Areas of application: - The hardware-based EPCN is highly adaptive and automatic with respect to its surrounding and to data. This implies its suitability in electronically isolated situations; e.g. in space exploration. It is also suitable for portable and/or embedded applications.

Areas of further research: An enhancement of the robustness of EPCN to vagaries of hardware is in order.

7.7 Summary

Chapter 7 has reviewed the achievement of previous chapters, the merits, and the demerits. Each weightless neural network and each multi-classifier implemented in each chapter is independently developed, and may be applied or used as such.

The results of projects detailed in this thesis have the following consequences for weightless neural systems. It means that (a) more connectivity methods are now possible for weightless neural networks; (b) a novel gating function is introduced for neural networks; (c) It is the first attempt at utilizing a weightless neural network as a trained combiner in a multi-classifier framework with a considerable success; (d) The EPCN is ported to FPGA. It signified that it may be possible for other advanced weightless neural networks to be ported also to FPGA.

Though a definition for intelligence may not be universally acceptable, and attempt has been made to define ANN as an intelligent system, it has nevertheless lead to achievements enumerated so far. As research extension to EPCN which will carry over to any MCS in which it is a component, it is suggested to include the following researches at project level:

- An attention mechanism;
- A consciousness mechanism.

These two modules are considered essential area of future research possibilities in order move EPCN toward an intelligent ANN system. Any other research extensions are possible and may be considered optional.

References

1. Abd Allah MM, Artificial Neural Network Based Fingerprint Authentication with Cluster Algorithm, Informatica 29, pp. 303 – 307, 2005.

2. Abhinar Saxena, Ashraf Saad, Evolving an artificial neural network classifier for condition monitoring of rotating mechanical systems, Applied Soft Computing ASOC – 214, pp.441-454, 25 October 2005.

3. Aleksander and Stohnam TJ, Guide to Pattern Recognition using random- access memories, Computera and Digital Techniques 2, pp.29-40, 1979.

4. Aleksander I, Morton H, Phenomenal weightless Machine, 17th Symp. On ANN, pp. 84 – 89, April 2009.

5. Aleksander I, Thomas WV, and Bowden PA, WISARD: a radical step forward in image recognition, Sensor Review 4(3), pp. 120-124, 1984.

6. Andrzej Bieszczad: Neurosolver: A step toward a neuromorphic general problem solver, Proceedings of IEEE World Congress on Computational Intelligence WCCI'94, Vol. III, pp. 1313 – 1318, 1994.

7. Austin (Ed.) J. RAM-based neural networks, World Scientific, 1998.

8. Bernard Michel et al., Support vectors machines regression for estimation of mars surface physical properties, 17th Symp. On ANN, pp. 195 – 200, April 2009.

9. Beyer J, Heesche W, Hauptmann W, Otte C. Heterogeneous mixture-of-expert for fusion of locally valid knowledge-based submodels, 17th Symp. On ANN, pp. 485 – 490, 2009.

10. Bledsoe WW, and Browning I. Pattern Recognition and reading by machine. In proc. Joint Comp. Conference, pp. 232-255, 1959.

11. Blue JL, Candela GT, Grother PJ, Chellappa R, Wilson CL, Evaluation of Pattern Classifiers for Fingerprint and OCR Applications. Pattern Recognition, pp. 485-501, 27 (1994).

12. Bogdan Gabrys, Dymitr Ruta, Genetic algorithm in classifier fusion, Applied soft computing 6, pp. 337 – 347, 2006

13. Boscsi B, Csato L. Dirichlet process-based component detection in state-space models, 17th Symp. On ANN, pp. 491 – 496, 2009.

14. Botelho SSC, Simões EV, Uebel LF, Barone DAC, High Speed Neural Control for Robot Navigation, IEEE International Conference on systems, man, and cybernetics, pp.421-429, Beijing, China, October 1996.

15. Bowmaker RG, and Coghill GG, Improved recognition Capabilities for goal seeking neurone, Electronic letters. 28, pp. 220-221, 1992.

16. Breiman L. Bagging predictors, Tech. Report 421, Dept of Statistics, University of California, Berkeley, 1994.

17. Breiman L. Combining predictors. In Sharkey, A, Editor, Combining Artificial Neural Nets, pp. 31-50, Springer-Verlag, 1999.

18. Bruderle D, et al: A software framework for tuning the dynamics of neuromorphic silicon towards biology: IWANN2007, LNCS 4507, pp. 479 -485, 2007.

19. Canuto A, Fairhurst MC, Howells WGJ, Improving Artmap learning through variable vigilance., International Journal of Neural Systems, voln.11 No.6, pp. 509-522., World Scientific Publishing Company. 2001.

20. Canuto A, Howells WGJ, and Fairhurst MC, The use of confidence measures to enhance combination strategies in multi-network neuro-fuzzy system. Connection Science, vol.12 no ¾, 2000, pp. 315-331.

21. Carpenter G, and Markuzon N, Artmap-IC and medical diagnosis: Instance Counting and inconsistent cases. Neural Networks. 11, pp. 323-336, 1998.

22. Chalfant EC, Lee S. Measuring the Intelligence of Robotic Systems: An Engineering Perspective, Proc. Int. Symposium on Intelligent Systems, Gaithersburg MD, pp. 370 – 375, October 1999.

23. Chikkerur S, Govindaraj V, Pankanti S, Bolle R, Minutiae Verification in Fingerprint Images Using Steerable Wedge Filters, ICVGIP, Calcutta, India, 2004.

24. Chikkerur S, Wu C, and Govindaraju V. A systematic approach for feature extraction in fingerprint images. In International Conference on Biometric Authentication, LNCS 3072, pp. 344 – 350, 2004.

25. Commuri S, Li Y, Hougen D, and Fierro R. Evaluating intelligence in unmanned ground vehicle teams, Performance Metrics for Intelligent Systems Workshop (PerMIS'04), NIST, Gaithersburg, MD, 2004.

26. Cristian Dima, Sensor and Classifier for outdoor obstacle detection, The Robotic Institute Carnegie Mellon University, 8 April 2003.

27. Dario Maio, Maltoni D, and Jain AK, FVC2004: Third Fingerprint Verification Competition, Int. Conf. On Biometric Authentication, Hong Kong, July 2003

28. Daouzli A, et. al.: Weights convergence and spike correlation in an adaptive neural network implementation on VLSI, Int. Conf. On Bio-inspired System and Signals, BIOSIGNAL, pp. 286 - 291, 2008.

29. David MacKay JC, Monte Carlo demonstration: Importance Sampling, Rejection sampling, Metropolis method, and Slice Sampling, Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, United Kingdom. February 2003.

30. De Canuto AM, Combining Neural Networks and Fuzzy Logic for Application in Character Recognition, PhD Thesis, University of Kent, 2001.

31. De Carvalho ACPLF, Combining two Neural Networks for Image classification In J. Austin (ed.) RAM-based neural networks. World Scientific, 1998.

32. De Cavalho A, Fairhurst MC, and Bisset DL, A lazzy learning approach to the training of GSN neural networks. In Proceeding of the ICANN 92, Elsevier, pp. 673-676, Brighton, UK, September 1992.

33. De Cavalho A, Fairhurst MC, and Bisset DL, Progressive learning algorithm for GSN feed-forward neural architectures. Electronic letters, 30(6), pp. 506-507, March 1994.

34. Dietterich G, Machine-learning research: Four current directions, The AI magazine 18(4), pp.97 – 136. 1998

35. Duin PW, The combining classifier: to train or not to train? ICPR2002, Quebec City, pp. 11 – 15, August 2002.

36. Dunja Mladenić, Janez Brank, Marko Grobelnik, Natasa Milic-Frayling, Feature Selection using Linear Classifier Weights: Interaction with classification Models SIGIR '04, Sheffield, South Yorkshire, UK, pp. 234 – 241, July 25 – 29 2004.

37. Fayyazi M, Navabi Z, Using VHDL Neural Network Models for Automatic Test Generation, 2nd Workshop on Libraries, Component Modeling and Quality Assurance, Toledo, Spain, April 1997.

38. Filho E, Fairhurst MC, and Bisset DL, Adaptive pattern recognition using goal-seeking neurones. Pattern recognition letters 12, pp. 131-138 March 1991.

39. Freeman M, Austin J, Designing a binary neural network co-processor, Digital System Design, 2005. Proceedings. 8th Conference on Euromicro, pp. 223 – 226, 30 Aug. to 3rd Sept. 2005.

40. Freeman M, Weeks M, Austin J. AICP: Aura Intelligence co-processor for Binary Neural Networks, IP-SOC, Grenoble, France, pp. 283 – 290, 2004.

41. Freeman W, and Aldeman E. The design and use of steerable filters. Transactions on PAMI, 13(9), pp. 891– 906, 1991.

42. Freund Y, Schapire R, A decision-theoretical generalisation of on-line learning and application to boosting, Journal of computer and systems sciences 55(1), pp. 119 – 139. 1997.

43. FVC2004, DB1_A.tar.gz. Available at http://biometrics.cse.msu.edu/fvc04db/index.html

44. German et al.: Neural networks and the bias/variance dilemma. Neural Computation 4(1), pp. 1-58, 1992.

45. Ghosh et al.: A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals. IEEE Journal of Ocean Engineering, 17(4), pp. 351-363, 1992.

46. Ghosh et al.: Integration of local and global neural classification for passive sonar signals. In proc. Of the Intl. Simulation Technology Conference, pp. 539-545, Houston, TX., 1992.

47. Hannan Bin Azhar MA, Dimond KR, Design of an FPGA based adaptive neural controller for intelligent robot navigation Digital System design, 2002, Proceedings. Euromicro Synposium 2002.

48. Hansen L.K., Salamon P.,: Neural network ensembles, IEEE Transaction on Pattern Análisis and Machine intelligence 12, pp. 993 – 1001, 1990.

49. Haykins S. Neural Networks, A Comprehensive Foundation, MacMillan Publishing, Englewood Cliffs, NJ, 1994.

50. Héla Zouari, Laurent Heutte, Yves Lecourtier, Using diversity measure in building classifier ensembles for combination method analysis, Advances in soft Computing,

Proc. of the 4th International Conference on Computer Recognition System, CORES, pp. 337 – 344, 2005.

51. Ho TK, The random subspace method for constructing decision forests, IEEE Trans. On Pattern Analysis and Machine Intelligence 20(8), pp. 832 – 844, 1998.

52. Hodgkin L, and Huxley AF, A quantitative description of membrane current and its application to conduction and excitation in nerve, Journal of physiology 117, pp. 500-544, 1952

53. Hollingum J, Automated Fingerprint Analysis Offers Fast Verif. Sensor Rev 3 (1992) 12-15. Institute Carnegie Mellon University, 8 April 2003.

54. Hopefield J, "Neurones with graded response have collective computational properties like those of two states neurons" in proceedings of the National academy of Science pp.2088-92 (pp.294 O. Karray), 1984.

55. Howells WGJ, Fairhurst MC, Bisset DL, PCN: The Probabilistic Convergent Network, Electronics Engineering Laboratories, University of Kent, Canterbury, Kent, CT2 7NT, U.K. November 1995.

56. Howells WGJ, Kola S, Statheros T, McDonald-Maier K, An Intelligent Fast-Learning Multi-Classifier System based on Weightless Neural Architectures, in: Proc. Of the Int. conf. On Recent Advances in Soft Computing (RASC 2006), K. Sirlantzis (Ed.), pp. 72 – 77, 2006.

57. Howells WGJ, Fairhurst MC, Faud Rahman: An exploration of a new paradigm for weightless RAM-based neural networks, Electronic, Connection Science, Voln. 12, No.1 pp.65-9, 2000.

58. http://bias.csr.unibo.it/fvc2004/databases.asp

59. Huetter, G (1988) “Solution of the travelling salesman problem with an adaptive ring”, in proceedings of the International conference on Neural Networks” pp.85-92

60. Hung D. C., Enhancement and feature purification of fingerprint images. Pattern Recognition, 26(11), pp. 1661–1671, 1993.

61. Hurricane Rita at http://www.chron.com/content/chronicle/special/05/rita/index.html

62. Igor Aleksander, from Wizard to Magnus: A family of weightless virtual neural machines, World Scientific, Singapore, pp. 18-30, 1998.

63. Impedovo S, Wang P, Bunke H. (Eds.): Automatic Bankcheque Processing, World Scientific Publ. Co., Singapore, 1997.

64. Internal technical Report, Automated Control and Guidance System – ACOS, University of Kent, Department of Electronics. Canterbury, Kent, CT2 7NT, U.K. July 2005.

65. Jain AK, Hong L, Bolle R, Online Fingerprint Verification. IEEE PAMI, p. 302-314, 19 (1997).

66. Jain AK, Prabakhar S, Hong L, Pankanti S, FingerCode: A Filterbank for Fingerprint Representation and Matching. In Proc. CV&PR Conf., Fort Collins, 1999.

67. Jang JR, “ANFIS: adaptive-network-based fuzzy inference system” IEEE Transaction on systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665-85, 1992.

68. John Proakis G. Digital Communication, 4th Edition, McGraw Hill Int. Edition, 2001.

69. John Skilling, David MacKay JC, Slice Sampling, Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, United Kingdom. February 2003.

70. Kaltenmerier A, Caesar T, Gloger J, Mandler E, Sophisticated topology of Hidden Markov Models for Cursive Script Recognition, in: Proc. Of 2nd Int. Conf. On Document Analysis and Recognition, Tsukuba Science City, Japan, pp. 139 – 142, 1993.

71. Kan Wing-kay, and Igor Aleksander; A probabilistic logic neuron network for associative learning in Neural computing architectures: the design of brain-like machines, pp. 156 - 171 ; ISBN:0-262-01110-7 , MIT Press Cambridge, MA, USA, 1989

72. Kanal L, Patterns in pattern recognition, IEEE Transactions on Information Theory. 20, pp. 697-722, 1974

73. Karray FO, De Silva C. Soft Computing and Intelligent System design, Addison Wesley, First Publication, 2004.

74. Kennedy JV, Austin J, Pack R, Cass B, C-NNAP: A dedicated processor for Binary Neural Networks. In the proceedings of the International Conference on Neural Networks '95, pp. 161 – 166, 1995.

75. Kim G, Govindaran V, Srihari S, Architecture for Handwritten Text Recognition Systems, in: S. –W. Lee (eds.), Advances in Handwritten Recognition, World Scientific Publishing. Co., pp. 163 – 172, 1999

76. Kuncheva LI, Combining pattern classifier: Methods and algorithm, John Wiley and sons Inc., 2004.

77. Kyoung Min Kim et al., Recognition of handwritten numerals using a combined classifier with hybrid features, A. fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, Springer – Verlag Heidelberg pp. 992 – 1000, 2004.

78. Lahoz D, and san Miguel C, A MLP neural network to predict the wind speed and direction at Zaragoza, Monografias del Seminario Matematico Garcia de Galdeano 33, pp. 293-300, 2006.

79. Lam L, Classifier combinations: Implementations and theoretical issues in MCS '00: proceedings of the first international workshop on multiple classifier systems, Springer-Verlag, London, UK, pp. 77 – 86, 2002.

80. Lopez J, Dorronsoro J, Rosen's projection method for SVM training, 17th Symp. On ANN, pp. 183 – 188, April 2009.

81. Lorrentz Pierre, The FPGA-based multi-classifier, Pattern Analysis and Applications, Volume 18, Issue 1, pp. 207-223, February 2015.

82. Lorrentz P, Howells WGH, McDonald-Maier KD, Design and analysis of a novel weightless neural based Multi-classifier, World Congress on Engineering, 2007.

83. Lorrentz P, Howells WGH, McDonald-Maier KD, An advanced combination strategy for multi-classifiers employed in large multi-class problem domains, Appl. Soft Comput. J. (March 2011), Volume 11, Issue 2, ISSN 1568 – 4946, 2151–2163, doi:10.1016/j.asoc.2010.07.014.

84. Lorrentz P, Howells WGH, McDonald-Maier KD, FPGA-based enhanced probabilistic convergent network for human iris recognition, 17th Symp. On ANN, pp. 319 – 324, 2009.

85. Lorrentz P, Howells WGH, McDonald-Maier KD, Enhanced Probabilistic Convergent Network, in: Proceedings of the 6th international conference on recent advances in Soft Computing (RASC 2006), K. Sirlantzis (Ed.), pp. 267 – 272, 2006.

86. Lumini, Nanni L, An advanced multi-modal method for human authentication featuring biometric data and tokenised random numbers, NeuroComputing, vol.69, no.13, pp.1706-1710, August 2006.

87. Maio D, and Maltoni D, Neural network based minutiae filtering in fingerprint images. In 14th International Conference on Pattern Recognition, pp. 1654–1658, 1998.

88. Maier KD, Beckstein C, Blickhan R, Fey D, and Erhard W, Standard cell-based implementation of a digital optoelectronic neural-network hardware, Applied Optics, Vol. 40, No. 8, pp. 1244 – 1252, 10 March 2001.

89. Maier KD, Beckstein C, Blickhan R, Fey D, and Erhard W, A multi-layer-perceptron neural network hardware based on 3D massively parallel optoelectronic circuits, in Proc. Of 6th Int. conf. On Parallel Interconnects, Anchorage, Alas. (IEEE Computer Society Press, Alamitos, Calif., 1999), pp. 73 – 80, 1999.

90. Maio D, and Maltoni D, Direct gray scale minutia detection in fingerprints. Transactions on PAMI, 19(1), 1997.

91. Maio D, Maltoni D, Jain AK, and Prabhakar S. Handbook of Fingerprint Recognition. Springer Verlag, 2003.

92. Massimo De Gregorio, Maurizio Giordano; Change Detection with Weightless Neural Networks. In CVPR Workshop, pp. 403-407, 2014.

93. Massimo De Gregorio, Giordano M, Rossi S, and Staffa M. Tracking deformable objects with WiSARD networks. In Workshop on Deformable Object Manipulation – INNOROBO2014, 2014.

94. Minsky M. Logical versus analog or symbolic versus connectionist or neat versus scruffy. AI Magazine, 12(2), pp. 34-51, 1991.

95. Mitchell T. Machine Learning, Morgan Kaufmann, San Mateo, CA, 1997.

96. Montgomery DC, et al., (2001), Engineering statistics, John Wiley & Sons Inc., Second Edition, 2001.

97. Morita M, Sabourin LS, Bortolozzi F, and Suen CY, Centre for Pattern Recognition and Machine Intelligence, Pontificia Universidade Catolica do Parana, IEEE, 2002.

98. Nahid Ardalani, Ahmadreza Khoogar, and Roohi H, A Comparison of Adaline and MLP Neural Network-based Predictors in SIR Estimation in Mobile DS/CDMA Systems, World Academy of Science, Engineering and Technology 9 2005.

99. Nanni L, Lumini A, Human authentication featuring signatures and tokenised random numbers NeuroComputing, vol.69, no.7-9, pp.858-861, March 2006.

100. Neil Yager, Adnan Amin, Fingerprint verification based on minutiae features; a review, Pattern Analysis and Application 7, pp. 94-113, 2004.

101. Perry Moerland and Emile Fiesler, Neural Network adaptation to hardware implementations, IDIAP-RR 97-17 in Handbook of Neural Computation, Institute of Physics Publishing and Oxford University Publishing, New York, Jan. 97.

102. Pétillot Y, Guibert L, and De Bougrenet de la Tocnaye JL, "Fingerprint recognition using a partially rotation invariant composite filter in a FLC joint transform correlator", Optics Communications **126**, pp. 213–219 (1996).

103. Pitts W, and McCulloch WS, "How we know universal: the perception of auditory and visual forms", Bulletin Math. Biophys. No. 9, pp. 27-47, 1947.

104. Prabhakar S, Jain A, and Pankanti S, Learning fingerprint minutiae location and type. Volume 36, pp. 1847–1857, 2003.

105. Prabhakar S, Jain A, Wang J, Pankanti S, and Bolle R, Minutiae verification and classification for fingerprint matching. In International Conference on Pattern Recognition, volume 1, pp. 25–29, 2000.

106. Prabhakar S, Pankanti S, and Jain AK, "Biometrics recognition: security and privacy concerns", IEEE Security & Privacy Magazine **1**, pp. 33–42 (2003).

107. Raffaele Cappelli, Maio D, and Maltoni D, A Multi-Classifier Approach to Fingerprint Classification**,** Pattern Analysis and Applications Special Issue on Fusion of Multiple Classifiers, vol. 5, no. 2, pp. 136 - 144, May 2002.

108. Ranawana R, Palade V, A neural network based multi-classifier system for gene identification in DNA sequences, Neural Comput. Appl. 14(2), pp. 122 – 131, 2005.

109. Ranawana R, Palade V, Multi-classifier systems – review and a roadmap for developers, University of Oxford Computing Laboratory, 24 April 2006.

110. Richard Mitchell, University of Reading, U.K., at http://www.personal.rdg.ac.uk/~shsmchlr/ nnetsmsc/nn09weightless.pdf, 09/07/2004.

111. Rodolfo J, Rajbenbach H, and Huignard JP, "Performance of a photorefractive joint transform correlator for fingerprint identification", Optical Engineering 34, pp.1166–1171, 1995.

112. Roli F, and Giacinto G, Hybrid Method in pattern recognition, chapter design of multiple classifier systems, Worldwide Scientific Publishing Co., pp. 199 – 226, 2002.

113. Ross A, Jain A, and Reisman J, "A hybrid fingerprint matcher", Pattern Recognition **36**, pp. 1661–1673 (2003).

114. Serkawt Farhan-Khola, Gareth Howells, Desing of a Genetic Feature Selection Algorithm for Neuron Input Mapping in N-tuple Classifiers, Department of Electronics, University of Kent, CT2 7NT, UK. 2006.

115. Shang L, Yi ., Ji L, Binary Image Thinning Using Autowaves Generated by PCNN, Neural Processing Letters, 25, pp. 49 – 62, 2007.

116. Shefa A, Dawwd, Ali Al-Saegh, RAM-Based Neural Network Parallel Implementation on a Reconfigurable Platform and Its Application for Handwritten

Digits Recognition; Al-Rafidain Engineering, Vol. 23, No. 2, pp. 76 -87, April 2015.

117. Shuang Yang, Antony Browne, Expert systems, vol. 2, No 5, November 2004.

118. Simões EV, Uebel LF, Barone DAC, Hardware Implementation of RAM Neural Networks, Pattern Recognition Letters, no. 17, pp. 421 – 429, 1996.

119. Simon Günter, Horst Bunke, Feature Selection Algorithm for the Generation of Multiple Classifier Systems and their Application to Handwritten Word Recognition, Department of Computer Science, University of Bern, Bern, Switzerland, March 4 2004.

120. Simon JC, Off-line cursive Word Recognition, Proc. Of the IEEE 80(7) , pp. 1150 – 1161, 1992

121. Simoncelli EP, and Farid H, Steerable wedge filters for local orientation analysis. Transactions on Image Processing, 5(9), pp. 1377 – 1382, 1996.

122. Sirlantis K, Howells WGJ, Gherman B, Novel Modular Weightless Neural Architecture for Biometric-based recognition, 17th Symp. On ANN, pp. 325 – 330, April 2009.

123. Siti Nurmaini, Ahmad Zarkasi, Simple Pyramid RAM-Based Neural Network Architecture for Localization of Swarm Robots; J. Inf Process Syst, Vol.11, No.3, pp.370-388, September 2015

124. Spaanenburg L, Alberts R, Slump CH, Van der Zwaag BJ, Natural learning of neural networks by reconfiguration, pp. 273-284 (2003), in: Rodriguez-Vazquez, A., Abbott, D. and Carmona, R. (eds.), SPIE Int. Symp. On Microtechnologies for the new Millennium, Vol. 5119 (Maspalomas, Gran Canaria, Spain), 2003

125. Stoianov A, Soutar C, and Graham A, “High-speed fingerprint verification using an optical correlator”, Optical Engineering **38**, pp. 99–107 (1999).

126. Suen C, Nadal L, Legault R, Mai T, Lam L, Computer Recognition of unconstrained Handwritten Numerals, Proc. Of the IEEE, 80(7), pp. 1162 – 1180, 1998.

127. Sukanesh R, Harikumar R, A Comparison of Genetic Algorithm & Neural Network (MLP) in Patient Specific Classification of Epilepsy Risk Level from EEG Signals, Engineering Letters, Feb. 2007.

128. Tagaki H, and Hayashi I, "A neural network-driven fuzzy reasoning", International journal of Approximate Reasoning, vol. 5, No. 3, pp. 119-212, 1991

129. Tico M, Immonen E, Ramo P, Kuosmanen P, Saarinen J, Fingerprint Recognition Using Wavelet features. In Proc. ISCAS, pp. 21-24. 2001

130. Tumer K, Ghosh J, Error correlation in ensemble classifiers connection science 8, pp.385 – 404, 1996

131. W. De Oliveira, Quantum RAM Based Neural Networks, 17th Symp. On ANN, pp. 331 – 336, 2009.

132. Wahab A, Chin SH, Tan EC, Novel Approach to Automated Fingerprint Recognition. IEE Proc. Vis. Image Signal Processing, 145 (3), pp. 160-166. 1998

133. Xiao, and Raafat H, Combining Statistical and Structural Information for Fingerprint Image Processing Classification and Identification, pp. 335–354. World Scientific, NJ, 1991.

134. Xilinx University Program Virtex-II Pro Development System: Hardware Reference Manual, UG069, March 2005.

135. Xu L, Krzyzak A, Suen CY, Associative Switch for Combining Multiple Classifiers and their Application in Handwritten Character Recognition, IEEE Trans. On System, Man, and Cybernetics SMC – 22(3), pp. 418 – 435, 1992.

……

Appendix

An EPCN Circuit