

Genetic Algorithm for Artificial Neural Network Training for the purpose of Automated Part Recognition

by

Stefan Buys, B.Tech Elec. Eng.

A dissertation submitted in compliance with the full requirements for the degree of

Magister Engineering: Mechatronic

in the

Faculty of Engineering, the Built Environment and Information Technology

Nelson Mandela Metropolitan University



Promoter: Prof. T. I van Niekerk

The copy of this dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the Nelson Mandela Metropolitan University and that no extracts from the thesis or information derived from it may be published without the author's prior consent.

I, Stefan Buys,

hereby declare that this work is my original work and all sources used or referred to have been documented and recognised.

Further this work has not been submitted in full or partial fulfilment of the requirements for any degree at another recognised educational institute.

Date

.....

Author's Signature

.....

Abstract

Object or part recognition is of major interest in industrial environments.

Current methods implement expensive camera based solutions. There is a need for a cost effective alternative to be developed. One of the proposed methods is to overcome the hardware, camera, problem by implementing a software solution. Artificial Neural Networks (ANN) are to be used as the underlying intelligent software as they have high tolerance for noise and have the ability to generalize. A colleague has implemented a basic ANN based system comprising of an ANN and three cost effective laser distance sensors. However, the system is only able to identify 3 different parts and needed hard coding changes made by trial and error. This is not practical for industrial use in a production environment where there are a large quantity of different parts to be identified that change relatively regularly. The ability to easily train more parts is required. Difficulties associated with traditional mathematically guided training methods are discussed, which leads to the development of a Genetic Algorithm (GA) based evolutionary training method that overcomes these difficulties and makes accurate part recognition possible.

An ANN hybridised with GA training is introduced and a general solution encoding scheme which is used to encode the required ANN connection weights.

Experimental tests were performed in order to determine the ideal GA performance and control parameters as studies have indicated that different GA control parameters can lead to large differences in training accuracy.

After performing these tests, the training accuracy was analyzed by investigation into GA performance as well as hardware based part recognition performance. This analysis identified the ideal GA control parameters when training an ANN for the purpose of part recognition and showed that the ANN generally trained well and could generalize well on data not presented to it during training.

Acknowledgements

I have been indebted in the preparation of this thesis to my supervisor, Prof. T.I van Niekerk, whose patience, insight and direction, as well as his academic experience, has been invaluable to me.

I am extremely grateful to Mr H. van Rooyen for allowing me access to his research and development into automated part recognition and artificial neural networks.

The informal support, encouragement and understanding of many friends have been indispensable.

Without the constant support and subtle pressure from my parents this thesis would certainly not have existed. I have to thank my father, Prof. F. Buys, who by example gave me the inspiration to further my academic career; it is to him that this research is dedicated.

To Anya, it is inconceivable to think that I would have been able to complete this research without you to help me get through the difficult times, and for all the emotional support, camaraderie, entertainment, caring and the endless supply of coffee you provided I thank you.

“If I have seen a little further it is by standing on the shoulders of Giants” – Sir Isaac Newton

Table of Contents

Abstract	i
Acknowledgements.....	ii
Table of Contents	iii
Abbreviations.....	vii
List of Figures	ix
List of Formulae.....	xi
List of Graphs	xii
List of Source Code	xiii
List of Tables	xiv
1. Chapter 1 – Introduction.....	1
1.1. Aim.....	2
1.2. Objectives of the study	2
1.3. Methodological Justification	2
1.4. Delimitations.....	4
1.5. Significance of Research	4
1.6. Organization of Thesis.....	7
1.7. Conclusion	7
2. Chapter 2 – Industrial Automated Part Recognition Neural Network Training and Genetic Algorithm based Optimization.....	8
2.1. Industrial Applications of Artificial Neural Networks and Genetic Algorithms.....	8
2.2. Comparing Back propagation and Genetic Algorithm based training.....	9
2.3. Genetic Algorithm based Artificial Neural Network training	12
2.4. Genetic Algorithm Control parameters	13
2.5. Automated Industrial Part Recognition	14
2.6. Conclusion	15
3. Chapter 3 – Experimental System Setup and Architecture.....	16
3.1. Experimental System Overview and Operation	16
3.1.1. System Training.....	16
3.1.2. Part Identification Operation.....	17
3.2. Hardware Architecture	19

3.2.1.	PLC	19
3.2.2.	IPC	21
3.2.3.	Data Collection Sensors.....	21
3.2.4.	Conveyor System	22
3.3.	Software Architecture	23
3.3.1.	Artificial Neural Network.....	23
3.3.2.	Genetic Algorithm	24
3.3.3.	Conveyor automation and data acquisition.....	25
3.3.4.	PLC Simulator	29
3.4.	Conclusion	34
4.	Chapter 4 – Neural Network Training Optimized for Part Recognition Applications.....	35
4.1.	Artificial Neural Network operation principle	35
4.1.1.	Advantages of Artificial Neural Networks	36
4.1.2.	Artificial Neural Network Applications	37
4.1.3.	Real-world Artificial Neural Network uses	37
4.2.	Feed-Forward Back-Propagation Neural Network	39
4.3.	Activation Functions	39
4.3.1.	Step function.....	39
4.3.2.	Linear function	40
4.3.3.	Sigmoid	41
4.3.4.	Ramp Function	41
4.3.5.	Ramp Function combined with Step Function.....	41
4.4.	Artificial Neural Network Architecture.....	42
4.5.	Artificial Neural Network Training.....	47
4.5.1.	Artificial Neural Network Training Methodologies.....	47
4.5.2.	Back-propagation Training	48
4.5.3.	Genetic Algorithm Training.....	49
4.6.	Conclusion	52
5.	Chapter 5 – Genetic Algorithms Applied to Neural Network Training.....	53
5.1.	Background on the way the GA works	53
5.1.1.	Problem Solution Encoding.....	56

5.1.2.	Population Creation and Initialization.....	56
5.1.3.	Fitness Evaluation / Fitness Function Definition.....	59
5.1.4.	Selection Operators	61
5.1.5.	Offspring Creation / Reproduction Operators.....	65
5.1.6.	Population Sorting Algorithm.....	68
5.1.7.	Algorithm Stopping Conditions	70
5.2.	Real-world Genetic Algorithm uses	71
5.2.1.	Automotive Design.....	71
5.2.2.	Engineering Design.....	71
5.2.3.	Evolvable Hardware	72
5.2.4.	Data Mining	72
5.2.5.	Optimized Telecommunications Routing.....	72
5.2.6.	Trip, Traffic and Shipping Routing	72
5.2.7.	Computer Gaming.....	73
5.2.8.	Encryption and Code Breaking.....	73
5.2.9.	Computer Aided Molecular Design.....	73
5.2.10.	Gene Expression Profiling.....	73
5.2.11.	Optimizing Chemical Kinetic Analysis.....	73
5.2.12.	Metamodeling of discreet-event simulation models.....	74
5.2.13.	Robotic Personality Generation	74
5.3.	Intelligent Part Recognition Training Algorithm Application.....	74
5.4.	Conclusion	81
6.	Chapter 6 – Part Recognition Performance.....	82
6.1.	Genetic Algorithm Control Parameters.....	82
6.1.1.	Training Set Data.....	82
6.1.2.	Population	83
6.1.3.	Offspring creation	84
6.1.4.	Genetic Cross-over and Mutation.....	85
6.2.	Genetic Algorithm Performance.....	88
6.2.1.	Training Data.....	88
6.2.2.	Population	90

6.2.3.	Offspring Creation.....	91
6.2.4.	Cross-over.....	92
6.2.5.	Mutation.....	94
6.2.6.	Overview of all test conditions.....	99
6.3.	Neural Network Performance with regard to Part Recognition	101
6.3.1.	PLC Simulator	102
6.3.2.	Hardware Testing	110
7.	Chapter 7 - Conclusion.....	118
	Bibliography.....	119
	Appendices.....	122
i.	Hardware Test Best Part ID Results	123
ii.	Hardware Test Part ID minus Misfire Results.....	128
iii.	Converged population for test Conditions 23.....	134
iv.	Converged population for test Conditions 26.....	136
v.	Training Data	138
vi.	PLC Simulator Source Code.....	144
vii.	Intelligent Neural Network Training utilizing Genetic Algorithm Source Code.....	145
viii.	Step 7 Part Recognition PLC Source Code.....	146

Abbreviations

AI	Artificial Intelligence
GA	Genetic Algorithm
GP	Genetic Programming
EP	Evolutionary Programming
ANN	Artificial Neural Network
AN	Artificial Neuron
BN	Biological Neuron
BP	Back-propagation
PLC	Programmable Logic Controller
EC	Evolutionary Computation
EA	Evolutionary Algorithm
DQM	Data Quality Management
EARGP	Evolutionary Algorithm for a Genetic Robots Personality
NMMU	Nelson Mandela Metropolitan University
SSE	Squared Sum Error
MSE	Mean Square Error
RSE	Root Square Error
RMSE	Root Mean Square Error
PCB	Printed Circuit Board
IPC	Industrial Personal Computer
OS	Operating System
OLE	Object Linking and Embedding
OPC	OLE for Process Control
CPU	Central Processing Unit
I/O	Input/Output

RAM Random Access Memory

DDR3 Data rate type three synchronous dynamic random access memory

GB GigaByte

GUI Graphical User Interface

List of Figures

Figure 1 – Adapted from Montana and Davis Experimental Results (Montana & Davis, 1989)	11
Figure 2 - Experimental System Setup Block Diagram.....	18
Figure 3 - PLC Test System.....	19
Figure 4 - Hardware Architecture	20
Figure 5 - Conveyor System.....	22
Figure 6 - Software Architecture.....	23
Figure 7 - PLC Simulator Weight Data Tab.....	30
Figure 8 - PLC Simulator Sensor Data Tab	31
Figure 9 - PLC Simulator Settings Tab	31
Figure 10 - PLC Simulator Main Neural Network Tab	32
Figure 11 - PLC Simulator Overview Excel Output	33
Figure 12 - PLC Simulator Individual Part Excel Output.....	33
Figure 13 - Basic Artificial Neuron	36
Figure 14 - Step Function.....	40
Figure 15 - Linear Function	40
Figure 16 - Ramp Function	41
Figure 17 - Ramp and Step Function.....	42
Figure 18 - Complete Neural Network Architecture.....	45
Figure 19 - Neural Network Architecture for each part	46
Figure 20 – GA based ANN training Flow Chart	51
Figure 21 - Landscape of solution space (Venables & Tan, 2007)	53
Figure 22 - Landscape of solution space with optimized solutions (Venables & Tan, 2007).....	54
Figure 23 - Basic Genetic Algorithm Flowchart.....	55
Figure 24 - Roulette Wheel Selection Operator	62
Figure 25- Rank Based Selection Operator	63
Figure 26 - Single Point Cross-over.....	66
Figure 27 – Two-point Cross-over	66
Figure 28 - Multi-point Cross-over	67
Figure 29 - Mutation operator.....	68
Figure 30 – Part Recognition Main Window.....	75
Figure 31 – Part Recognition General Settings Window	76
Figure 32 – Part Recognition Neural Network Settings Tab	76
Figure 33 – Part Recognition Neural Network Weights Tab	77
Figure 34 – Part Recognition Neural Network Training Data Tab.....	77
Figure 35 – Part Recognition Genetic Algorithm Settings Tab.....	79
Figure 36 - Part Recognition Genetic Algorithm Generation Creation Tab.....	79
Figure 37 - Part Recognition Genetic Algorithm GA Options Tab	80
Figure 38 - Part Recognition Genetic Algorithm Output Window	80
Figure 39 - Training Data Group Results	89

Figure 40 - Population Group Results (outliers removed).....	90
Figure 41 - Offspring Creation Group Results.....	92
Figure 42 - Cross-over Group Results.....	93
Figure 43 - Simatic Step 7 Part Targets Table.....	110
Figure 44 - Simatic Step 7 Part A Weight Table.....	111
Figure 45 - Simatic Step 7 ANN Output Table	111

List of Formulae

Equation 1 - Standard Deviation (DeCoster, 1998)	25
Equation 2- Artificial Neuron Output	36
Equation 3 - Step Function.....	39
Equation 4 - Step Function with binary output.....	39
Equation 5 - Linear Function	40
Equation 6 - Sigmoid Function	41
Equation 7 - Ramp Function.....	41
Equation 8 - Hybrid Ramp and Step Function.....	41
Equation 9 - Solution Space Description (Venables & Tan, 2007).....	53
Equation 10 - Basic Error Calculation.....	60
Equation 11 - RSE Calculation.....	61
Equation 12 - Mutated Gene	68

List of Graphs

Graph 1 - Part B X-axis Data Capture	28
Graph 2 - Part B Y-axis Data Capture	28
Graph 3 - Part E X-axis Data Capture	29
Graph 4 - Part E Y-axis Data Capture	29
Graph 5 - RSE vs. Generation for Cross-over group.....	93
Graph 6 - Mutation Group results	97
Graph 7 - Mutation Group results Mutation Factor	97
Graph 8 - Error vs Generation for Mutation Factor.....	98
Graph 9 - Genetic Algorithm Training Results - Average with outliers removed.....	100
Graph 10 - PLC Simulation Performance	106
Graph 11 - Part OK Identification minus Neural Network Misfires	109
Graph 12 - Hardware Test Results	116
Graph 13 - Hardware Test Results including Linear Trend.....	117

List of Source Code

Source Code 1 - Population Initialization	58
Source Code 2 - Population Initialization to f_{anin}	59
Source Code 3 - Population Sorting	70

List of Tables

Table 1 - Parts to be identified / recognised.....	6
Table 2 - Average RMSE comparisons (Gupta & Sexton, 1999)	10
Table 3 - RMSE Standard Deviations and Mean CPU time (Gupta & Sexton, 1999)	10
Table 4 - Part B X-point Training Data Standard Deviation	26
Table 5 - Part B Y-point Training Data Standard Deviation	26
Table 6 - Part E X-point Training Data Standard Deviation	27
Table 7 - Part E Y-point Training Data Standard Deviation	27
Table 8 - RSE Fitness Calculation.....	61
Table 9 - Error Fitness Calculation	61
Table 10 - Test Conditions 1 to 19	86
Table 11 - Test Conditions 20 to 35.....	87
Table 12 - Training Data Conditions	88
Table 13 - Population Conditions	90
Table 14 - Offspring Creation Conditions.....	91
Table 15 - Cross-over Conditions.....	92
Table 16 - Mutation Conditions.....	94
Table 17 - Genetic Algorithm Training Results - Average.....	99
Table 18 - PLC Simulator Results Conditions 1 to 11	102
Table 19 - PLC Simulator Results Conditions 12 to 23	103
Table 20 - PLC Simulator Results Conditions 24 to 35	104
Table 21 - PLC Simulation Performance	105
Table 22 - Part OK Identification minus Neural Network Misfires	108
Table 23- Conditions 4 Optimized Weights.....	112
Table 24 - Part Recognition Test 1 Result Summary	112
Table 25 - ANN Output Mapping	113
Table 26 - Part Recognition Test 1 Part A	114
Table 27 - Conditions 7 Optimized Weights.....	115
Table 28 - Part Recognition Test 2 Result Summary	115
Table 29 - Part Recognition Test 1 Part B	123
Table 30 - Part Recognition Test 1 Part C	124
Table 31 - Part Recognition Test 1 Part D	125
Table 32 - Part Recognition Test 1 Part E	126
Table 33 - Part Recognition Test 1 Part F.....	127
Table 34 - Part Recognition Test 2 Part A	128
Table 35 - Part Recognition Test 2 Part B	129
Table 36 - Part Recognition Test 2 Part C.....	130
Table 37 - Part Recognition Test 2 Part D	131
Table 38 - Part Recognition Test 2 Part E	132
Table 39 - Part Recognition Test 2 Part F.....	133

Table 40- Converged Population Conditions 24.....	134
Table 41 - Training Data Part A.....	138
Table 42 - Training Data Part B.....	139
Table 43 - Training Data Part C.....	140
Table 44 - Training Data Part D.....	141
Table 45 - Training Data Part E.....	142
Table 46 - Training Data Part F.....	143

1. Chapter 1 – Introduction

Humans recognize a multitude of objects in images with little effort. The images of the objects may vary somewhat in different viewpoints in many different sizes or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems in general.

Part recognition is an important problem in industrial vision. Efficient and accurate part identification is essential for flexible automation of almost all major manufacturing processes such as inspection, assembly sorting and binning.

At present camera based automated part recognition is the most commonly used method in industry.

The problems associated with this method are:

- Expensive hardware cost.
- Problems associated with lighting conditions.
- Problems associated with reflective surfaces.
- Complex training of new parts by specialist / trained personnel.
- Timely process to train new parts.

A major thrust in algorithmic development is the design of algorithmic models to solve increasingly complex problems. Enormous successes have been achieved through the modelling of biological and natural intelligence, resulting in so called “intelligent systems”. These intelligent systems include artificial neural networks, evolutionary computation, swarm intelligence and fuzzy systems. These systems form part of the field of Artificial Intelligence (AI).

In this research, two of the main paradigms of AI will be utilized in order to find an alternative solution to the part recognition problem. These are ANNs and Evolutionary Computation (EC) or GAs.

Multilayered feed-forward ANNs possess a number of properties which make them particularly suited to complex pattern classification problems. However, their application to some real world problems, such as part recognition, has been hampered by the lack of a training algorithm which reliably finds a nearly globally optimal set of weights in a relatively short time. GAs are good at exploring a large and complex solution space in an intelligent way to find values close to the global optimum. Hence, they are well suited to the problem of training feed-forward ANNs.

This research will investigate the various GA control parameters with the objective of finding the optimal conditions for these parameters when utilizing a GA for ANN weight optimization or training for the purpose of industrial part recognition.

1.1. Aim

To design, program and optimize a GA for the training of an ANN for the purpose of industrial part recognition.

1.2. Objectives of the study

The following objectives were accordingly specified for this project:

- Conduct a literature study in order to gain an understanding of current part recognition methods, ANNs and their training methods, GAs.
- Design and develop a GA to train an ANN for part recognition.
- Adapt an existing industrial hardware based ANN to this problem.
- Research the optimal GA control parameters and conduct experimentation to evaluate performance.
- Create a supporting experimental setup using appropriate measurement hardware and software in order to conduct supporting experimentation.
- Design and develop a system architecture, including all hardware, software and communication, to implement intelligent system software for industry application.

1.3. Methodological Justification

In order to accomplish the objectives, the fundamental research issues covered in this project include the following two paradigms of AI:

- ANNs

In his book, *Computational Intelligence: An Introduction*, Andries P. Engelbrecht describes ANNs as follows:

“The brain is a complex, nonlinear and parallel computer. It has the ability to perform tasks such as pattern recognition, perception and motor control much faster than any computer – even though events occur in the nanosecond range for silicon gates, and milliseconds for neural systems. In addition to these characteristics, others such as the ability to learn, memorize and still generalize, prompted research in algorithmic modelling of biological neural systems – referred to as artificial neural networks (NN)” (Andries, 2007).

An ANN will be trained to recognize a set of six defined parts/shapes.

- GAs

A GA is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomised, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, especially the *survival of the fittest* or *natural selection* laid down by Charles Darwin (Darwin, 1859). Since in nature, competition among individuals for scarce resources results in the fittest individuals dominating over the weaker ones

“Evolutionary computation (EC) has as its objective to mimic evolution, where the main concept is survival of the fittest: the weak must die. In natural evolution, survival is achieved through reproduction. Offspring, reproduced from two parents (sometimes more than two), contain genetic material of both (or all) parents - hopefully the best characteristics of each parent. Those individuals that inherit bad characteristics are weak and lose the battle to survive. This is nicely illustrated in some bird species where one hatchling manages to get more food, gets stronger, and at the end kicks out all its siblings from the nest to die.

Evolutionary algorithms use a population of individuals, where an individual is referred to as a chromosome. A chromosome defines the characteristics of individuals in the population. Each characteristic is referred to as a gene. The value of a gene is referred to as an allele. For each generation, individuals compete to reproduce offspring. Those individuals with the best survival capabilities have the best chance reproduce. Offspring are created by combining parts of the parents, a process referred to as crossover. Each individual in the population can also undergo mutation which alters some of the allele of the chromosome. The survival strength of an individual is measured using a fitness function which reflects the objectives and constraints of the problem to be solved. After each generation, individuals may undergo culling, or individuals may survive to the next generation (referred to as elitism). Additionally, behavioural characteristics (as encapsulated in phenotypes) can be used to influence the evolutionary process in two ways: phenotypes may include genetic changes, and/of behavioural characteristics evolve separately.” (Andries, 2007).

A GA will be developed for the purpose of training an ANN to be able to identify a set of parts. Although a set of six parts have been specified for this research, the system has been designed to be adaptable to other parts.

The GA has also been designed to be a reconfigurable system that can be adapted to other optimization problems and not ANN training only.

1.4. Delimitations

Although the system design allows the training of other parts / shapes, this research is limited to the identification of the six specified components as highlighted in *Table 1 (page 6)*.

The parts were selected based on the following criteria:

- Commonly used and available industrial automation parts.
- Physical size and shape allows them to fit onto the conveyor used as research platform.
- The geometry of the parts provides a wide range of surface shapes such as planes, cylinders, and spheres since such shapes are common in industrial parts.

Implementation is limited to a S7-300 Programmable Logic Controller (PLC), Festo conveyor and other hardware in use on the research platform.

The Object Linking and Embedding for Process Control (OPC) server in Section 3.1.2 *Part Identification Operation* that connects the *Neural Network* Visual Basic Application and the PLC is out of scope due to cost constraints and insignificance to the main research objectives.

1.5. Significance of Research

The automotive industry has an important impact on both the National South African- and especially the Regional Eastern Cape economy. The local companies primarily focus on production and manufacturing of motor vehicles and associated sub-components for export to a global market in the northern hemisphere. This scenario provides the regional economy and industry with a global link to industry and business.

As the Nelson Mandela Metropolitan University (NMMU) is the largest tertiary institution in the Eastern Cape with an engineering education program it is seen as one of the industry's largest sources of well trained automotive engineers. Due to highly innovative product development in this industry, driven towards better quality and lower cost, the educational contents of the engineering programs have to follow fast changing industry trends and needs.

In order to fully equip graduate engineers from the NMMU and enable them to support this demanding and fast changing industry, the University acquired a Siemens Automation and Motion control laboratory in the second quarter of 2006. This laboratory is used for accredited Siemens PLC training for industry as well as control system lecturing for Electrical Engineering students.

The intended applications for this laboratory are two-fold. Firstly it is to integrate PLC training into the curriculum of the Electrical-, Mechanical-, Industrial- and Mechatronic Engineering programs in order to prepare the graduates for real-world industrial applications as opposed to pure theoretical training. The second intended application is to promote and enable research into industrial application of new

technology, such as Artificial Intelligence that was previously seen as an Information Technology component only.

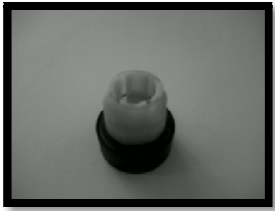





Part recognition plays a vital part in the modern manufacturing process. It is required for material handling, part sorting and quality control. Currently the most commonly used solutions are camera based systems. The main problems with these systems are their high cost, susceptibility to ambient light conditions (and the associated inaccuracy) and the expertise required by maintenance personnel to maintain them and train new parts.

A proposed alternative to camera based solutions are basic laser distance sensors providing dimensional data of the parts to a well trained Artificial Intelligence based system utilizing an ANN. While a previous student has managed to implement an ANN on industrial hardware for the purpose of part recognition, he had difficulty with training the ANN.

The proposed solution is to use a GA to train the ANN. While other researchers have managed to train ANNs by using a GA, none have researched the ideal GA control parameters for the application of part recognition.

This research will attempt to find the ideal control parameters for a GA when used to train an ANN for the purpose of part recognition.

Table 1 - Parts to be identified / recognised

Part ID	Part Description	Part Picture
Part A	Pushbutton	
Part B	Collar	
Part C	Contact	
Part D	Selector Switch	
Part E	Mushroom E-stop	
Part F	Cylinder	

1.6. Organization of Thesis

Overview of what each chapter contains.

Chapter 1 – Objectives, aim, delimitation and research significance are introduced.

Chapter 2 – Analyzes relevant theories, corresponding components, related technology and up-to-date development in the field of GA ANN training and part recognition in terms of literature study.

Chapter 3 – Describes overall system setup, hardware architecture, software components, implementation of subsystems and integration thereof to form a platform of intelligent part recognition ANN training using GAs.

Chapter 4 – ANN training background. Introduction to current back-propagation (BP) methods and the proposed GA based training.

Chapter 5 – Detailed explanation on the developed GA, overview on the related software components and their operation.

Chapter 6 – Research on test cases and the performance of each test case regarding GA and part recognition.

Chapter 7 – Conclusion

1.7. Conclusion

In this chapter the topics of automated object recognition, AI, ANN and GA were introduced. The research aim, objectives, delimitations and significance were also discussed.

In the next chapter relevant theory, corresponding components, related technology and up-to-date development in the field of GA based ANN training and automated part recognition is discussed in terms of a literature study.

2. Chapter 2 – Industrial Automated Part Recognition Neural Network Training and Genetic Algorithm based Optimization

In this chapter relevant theory, corresponding components, related technology and up-to-date development in the field of GA based ANN training and automated part recognition is discussed in terms of literature study.

Topics covered in the chapter are:

- Industrial applications of GAs and ANNs.
- Comparison of BP and GA based ANN training.
- GA based ANN training methodology.
- GA control parameters.
- Different methodologies of automated part recognition.

2.1. Industrial Applications of Artificial Neural Networks and Genetic Algorithms

ANNs combined with GAs for both weight optimization as well as architecture design has been successfully used in various industrial applications.

In their paper *Genetic Algorithm-Based Artificial Neural Network for Voltage Stability Assessment* Singh and Srivastava proposed a GA based BP ANN (Singh & Srivastava, 2011) for voltage stability margin estimation. As the voltage stability margin is an indication of the power system's proximity to voltage collapse, a fast and accurate estimation system is required. Their use of GAs in the training algorithm was included in the hope of avoiding the inherent local minimum problem of BP training. BP searches on the error surface by means of the gradient descent technique in order to minimize the error. It is therefore likely to get struck in a local minimum. They found that a GA based ANN learns faster while it at the same time provides more accurate voltage stability margin estimation when compared to that based on a BP algorithm.

Goa and Ovaska researched implementation of GA trained ANNs for fault detection in electrical motor drive systems in their paper *Genetic Algorithm Training of Elman Neural Network in Motor Fault Detection* (Gao & Ovaska, 2002). The GA aided training strategy for the ANN was introduced to improve the approximation accuracy and achieve better detection performance. Experiments with a practical automobile transmission gearbox with an artificial fault were carried out to verify the effectiveness of their method. They obtained encouraging fault detection results without any prior information on the gearbox model.

Demutgal et al. proposed another variation of a GA and ANN hybrid system in their paper *Fault Diagnosis on bottle filling plant using genetic-based neural network* (Demutgal, Unal, Tansel, & Yazicioglu, 2011) for the purpose of timely detection of the pneumatic system problems in industry. They proposed a system where back propagation is still used for ANN connecting weights optimization but a GA is implemented to

find the optimal ANN architecture. They found that the GA was able to establish a better ANN architecture for accuracy, estimation speed and compactness than the conventional trail-and-error method.

2.2. Comparing Back propagation and Genetic Algorithm based training

Before looking at the benefits of GA based ANN training, we first have to look at the, very popular, alternative that is BP training.

Hui et al. describe BP training of ANNs as follows:

“Back propagation (BP) is a powerful learning rule used in training feed forward multi-layer perceptrons that can be applied to a large range of classification problems. The training process aims at minimizing the difference between the target output and the actual output of the neural network. BP employs a simple optimization algorithm for this purpose. Because of its simplicity, BP is easy to understand and implement.” (Hui, Lam, & Chea, 1997).

They also identify two major limitations of BP training of ANNs. A slow convergence rate (attributed to its optimization component, the steepest descent algorithm) and its tendency to get stuck in a sub optimal local minima. They recommend the use of GA to escape this local minima (Hui, Lam, & Chea, 1997).

Gupta and Sexton compared the use of a GA and traditional BP training techniques of feed-forward ANNs in their paper: *Comparing backpropagation with a genetic algorithm for neural network training* (Gupta & Sexton, 1999).

The used a chaotic time series as an illustration to compare the GA and BP for effectiveness, ease-of-use and efficiency for training ANNs. Because ANNs generate complex error surfaces with multiple local optima, even for simple functions being estimated, BP tends to become trapped in local rather than global solutions. To overcome this local convergence problem for solving difficult non-linear optimization problems, such as pattern and part recognition, a number of techniques have been developed. The most common of these are the Evolutionary Programming (EP) approaches which include the GA.

Gupta and Sexton defined three measures to use for comparison of GA and BP training:

- Effectiveness: Refers to the accuracy of each algorithm in estimating the true functional form. The Root Mean Squared Error (RMSE) is used for a direct comparison between the GA and BP solutions.
- Ease-of-use: Deals with the effort needed for finding optimal algorithm settings for each problem. Finding these optimal control parameters are extremely important as they have a direct influence on the performance and accuracy of the NN training. Finding these ideal parameters form part of the main research of this project.
- Efficiency: The time (actual time, epochs or generations) taken by each algorithm to converge upon the optimal solution.

Table 2 - Average RMSE comparisons (Gupta & Sexton, 1999)

Hidden node count	6			4			2		
Data sets	SBP	EDBD	GA	SBP	EDBD	GA	SBP	EDBD	GA
In-sample	0.079	0.124	0.041	0.054	0.125	0.043	0.259	0.423	0.145
Test Set 1	0.135	0.183	0.050	0.960	0.174	0.055	0.348	0.478	0.164
Test Set 2	0.166	0.218	0.050	0.147	0.243	0.066	1.108	0.439	0.172

When looking at *Table 2 (page 10) comparisons* from Gupta and Sexton’s research, one can see that the GA far outperformed Standard BP (SBP) and Extended Delta-Bar-Delta (EDBD) BP methods in effectiveness when comparing the Root Mean Square Error (RMSE) that they produced for all hidden node count scenarios.

Table 3 - RMSE Standard Deviations and Mean CPU time (Gupta & Sexton, 1999)

Hidden node count	6			4			2		
Algorithm	SBP	EDBD	GA	SBP	EDBD	GA	SBP	EDBD	GA
Standard Deviation	0.210	0.012	0.007	0.024	0.023	0.010	0.077	0.089	0.013
Mean CPU time	611.200	623.800	329.300	570.300	583.200	245.100	497.900	532.100	134.400

When looking at *Table 3 (page 10) time* from Gupta and Sexton’s research, one can see that the GA far outperformed SBP and EDBD BP methods in efficiency when comparing the mean CPU time that they required for all hidden node count scenarios.

Montana and Davis describe a different genetic algorithm for training feed forward networks in their paper *Training Feed forward Neural Networks Using Genetic Algorithms* (Montana & Davis, 1989). In their research they found that it not only succeeds in its task but it outperforms BP, the standard training algorithm. They also identified drawbacks of BP:

“There are some drawbacks to back propagation. For one, there is the "scaling problem". Back propagation works well on simple training problems. However, as the problem complexity increases (due to increased dimensionality and/or greater complexity of the data), the performance of back propagation falls off rapidly. The performance degradation appears to stem from the fact that complex spaces have nearly global minima which are sparse among the local minima. Gradient search techniques tend to get trapped at local minima. With a high enough gain (or momentum), back propagation can escape these local minima. However, it leaves them without knowing whether the next one it finds will be better or worse. When the nearly global minima are well hidden among the local minima, back propagation can end up bouncing between local minima without much overall improvement, thus making for very slow training. A second shortcoming of back propagation is the following. To compute a gradient requires differentiability. Therefore, back propagation cannot handle discontinuous optimality criteria or discontinuous node transfer functions. This precludes its use on some common node types and simple optimality criteria.” (Montana & Davis, 1989).

Montana and Davis compared the performance of standard BP with GA for training of the feed-forward ANN. They used the standard BP algorithm with a learning rate of 0.5. The GA utilized two offspring creation operators namely Mutation and Cross-over.

When comparing them, they considered two generations of the GA to be equivalent to one epoch of BP. Their reasoning was that BP consists of looping through all training data performing firstly forward propagation and calculation of errors at the outputs and then secondly error BP and adjusting of weights. The second step required more computation in their networks of interest. The evaluation function of the GA performed the same calculations as the first step of BP by forward propagation and calculation of errors at the outputs and assigning this as fitness. The mutation and cross-over operators do very little computation.

Their experimental runs consisted of 10000 generations of the GA and 5000 epochs of BP. The results are shown in *Figure 1 (page 11)*. It can clearly be seen that the GA outperformed BP.

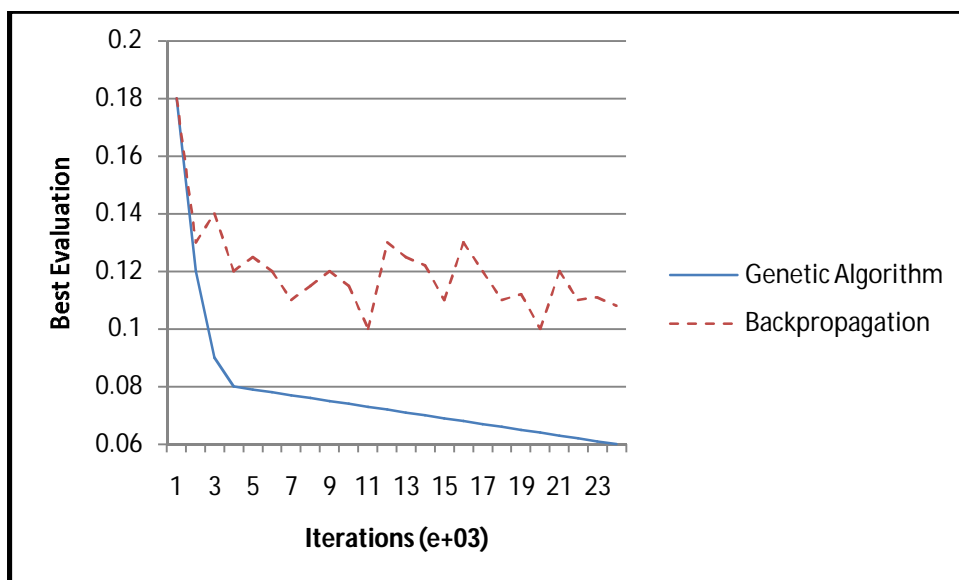


Figure 1 – Adapted from Montana and Davis Experimental Results (Montana & Davis, 1989)

Demetgul et al. also identified two major weaknesses of BP training of ANNs. Convergence to a local minimum and the need for several trial and error tests to find the optimal structure (Demetgul, Unal, Tansel, & Yazicioglu, 2011). They also point out the important advantage of GA based ANN training in avoiding the local minima in non-linear and multimodal optimization problems.

2.3. Genetic Algorithm based Artificial Neural Network training

Singh and Srivastava pointed out the advantage of GA ANN training that they operate in a population of possible solution candidates in parallel, instead of starting with a single candidate and iteratively operate on it using some sort of heuristics (Singh & Srivastava, 2011). This enables the search for a global minimum instead of a local as with BP training.

Dorsey et al. (Dorsey, Johnson, & Mayer, 1994) designed a Genetic Adaptive Neural Network Training algorithm and showed that the GA also works well for optimizing ANNs. This algorithm is different from other genetic search algorithms in that it uses real values instead of binary representations of the weights. This is also the case in the GA implemented in this research.

In their paper *Genetic Algorithm Training of Elman Neural Network in Motor Fault Detection* Goa and Ovaska note that GAs have the ability to evaluate the fitness of an individual and implement mutation and crossover operators in parallel. They go on by stating that a GA needs less prior information about the problem to be solved than conventional optimisation schemes, such as the steepest descent method, which require derivatives of objective functions. It is therefore attractive to employ a GA to optimise the parameters and structures of neural networks, instead of using the BP learning algorithm alone. As previously discussed, they also identify the well known problem of the steepest descent-based method of easily becoming trapped in local minima in the search space of ANN weights (Gao & Ovaska, 2002).

Goa and Ovaska also found that direct GA-based optimization of ANN weights are not always a better solution than BP. They found GA based optimization in Elman neural networks time consuming, because the size of the network grows drastically with the number of hidden nodes (Gao & Ovaska, 2002). Since, in the ANN architecture in this part recognition application, the amount of neurons in the hidden layers stays constant at a relatively low value, this is not a problem. The architecture of the ANN is discussed in Section 4.4.4 *Neural Network Architecture*.

Montana and Davis (Montana & Davis, 1989) implemented GA based training of ANN as follows:

1. Chromosome Encoding: The weights (and biases) in the neural network are encoded as a list of real numbers.
2. Evaluation Function: Assign the weights on the chromosome to the links in a network of a given architecture, run the network over the training set of examples, and return the sum of the squares of the errors.
3. Initialization Procedure: The weights of the initial members of the population are chosen at random with a probability distribution given by $t^{|r|}$. This is different from the initial probability distribution of the weights usually used in back propagation, which is a uniform distribution between -1.0 and 1.0. Our probability distribution reflects the empirical observation by researchers that optimal solutions tend to contain weights with small absolute values but can have weights with arbitrarily large absolute values. We therefore seed the initial population with genetic material which allows the genetic algorithm to explore the range of all possible solutions but which tends to favour those solutions which are a priori the most likely.

4. Operators: We created a large number of different types of genetic operators. The goal of most of the experiments we performed was to find out how different operators perform in different situations and thus to be able to select a good set of operators for the final algorithm.

2.4. Genetic Algorithm Control parameters

There are certain control parameters and limits that need to be specified before a GA can attempt to optimize the solution for a problem.

Singh & Srivastava defined these control parameters as: *“Before executing certain task, GA requires several parameters to be devised for its proper functioning. Some of them are gene encoding, population initialization, selection, reproduction, fitness evaluation, and so forth. The basic computing elements in GAs are genes, which are joined together to form a string of values referred to as a chromosome.*

Genes encoding is required to represent weights. To carry out efficient weight optimization in ANN, it is important to have a proper encoding scheme. There is no definite view in the literature about suitability of encoding schemes for chromosomes. In the present work, real value coding is adopted for gene encoding. Size of the population of individuals (chromosomes) is generated randomly to start the genetic search procedure. The size of population depends upon number of weights to be optimized multiplied by gene length. The number of weights to be optimized is determined from neural network configuration” (Singh & Srivastava, 2011).

Montana and Davis (Montana & Davis, 1989) identified five required components for artificial biological evolution:

- Encoding Solution: A way of encoding solutions to the problem in chromosomes of individuals.
- Fitness Function: A function that returns a rating or evaluation for each individual presented to it.
- Population Initialization: A way of initializing the population of chromosomes to random values that cover the entire search scope.
- Reproduction Operators: Operators that may be applied to parents when they reproduce to alter their genetic composition. The most common are mutation, crossover and elitism.
- GA Control Parameters: Parameter settings for the algorithm, the operators and so forth.

Montana and Davis (Montana & Davis, 1989) go on by stating that: *“The work described here only touches the surface of the potential for using genetic algorithms to train neural networks. In the realm of feed forward networks, there are a host of other operators with which one might experiment.”* One of the main objectives of this research, as defined in Section 1.2 *Objectives of the study*, is to experiment with these operators in order to find the ideal parameters for part recognition training.

2.5. Automated Industrial Part Recognition

Because of the importance of part recognition on industry, considerable effort has been dedicated to solving this problem. Until the 1970's optical techniques was the most common method applied to solving part recognition problems (Casasent & Psaltis, 1976).

Due to the technological advances in the microelectronics since the 1980's digital techniques have been applied (Hsu, Arsenault, & April, 1982).

Currently the most common approach to the problem in industry is implementing costly computerized camera based solutions.

Part recognition in computer vision is the task of recognizing a specified object in an image or video stream. The human brain and vision system has the ability to recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different viewpoints, in many different sizes or even when they are rotated. This task is still a challenge for computer vision systems in general.

In their book *Intelligent Vision Systems for Industry* B. Batchelor and P. Helan point out the following problem with current machine vision systems: "*the application of vision to automated assembly can be impressive to watch, but often deceptive. If one of the pieces to be assembled is rotated or moved slightly, then the system may not be able to cope with this change in its working environment*" (Batchelor & Whelan, 2002).

Perelmuter et al. presented work on industrial part recognition at the *5th International Conference on Education, Practice, and Promotion of Computational Methods in Engineering Using Small Computers* (Perelmuter, Carrera, Vellasco, & Pacheco, 1995).

They presented an application of an ANN for object recognition utilizing a camera based system. The application involved the pre-processing of a 2D-digital image, and its subsequent classification through a ANN trained with typical patterns extracted from the images of the objects.

One of the main advantages of an intelligent recognition or classifier system over traditional systems that they identified is that the intelligent classifier is a general purpose recognition system, being able to be applied to virtually any set of parts. This is also the case in this research project.

The main disadvantage that they identified is that the capture conditions are extremely critical in the resultant performance of the classifier. They found that adequate conditions, including lighting, is very important so that the borders extractor can detect the images' limits correctly.

Applications range from tasks such as industrial machine vision systems which, say, inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend and interpret the world around them.

Examples of applications of computer vision include systems for:

- Controlling processes in the field of robotics.
- Navigation in the field of robotics.
- Detecting events such as part counting.
- Organizing information.
- Modelling objects or environments.
- Automatic inspection and quality control in manufacturing applications.

Object recognition methods have the following applications:

- Image panoramas.
- Image watermarking.
- Global robot localization.
- Face Detection.
- Optical Character Recognition.
- Manufacturing Quality Control.
- Content-Based Image Indexing.
- Object Counting and Monitoring.
- Automated vehicle parking systems.
- Visual Positioning and tracking.
- Video Stabilization.

2.6. Conclusion

In this chapter relevant theory, corresponding components, related technology and up-to-date development in the field of GA based ANN training and automated part recognition has been discussed in terms of literature study.

ANNs and GAs, their use in industrial applications, GA control parameters and their importance as well as the various training methodologies were introduced.

While it is clear that a vast amount of research has been performed on different training techniques and methodologies of ANNs, it can be seen that very little has been investigated with regard to the GA control parameters for ANN training especially when applied specifically to automated part recognition.

In the next chapter the various hardware and software components, the system architecture and general operation of the experimental system setup is discussed.

3. Chapter 3 – Experimental System Setup and Architecture

In this chapter the research platform, system setup and architecture is described and discussed. In the first section of the chapter, a broad overview of the experimental test setup is described. As the main components of the research platform can be subdivided into hardware and software, the rest of the chapter is divided accordingly. The various hardware components, their operation and selection criteria are discussed in the second section of the chapter while the software components, their operation and the communication between them are discussed in the last.

3.1. Experimental System Overview and Operation

An overview of the experimental system setup can be seen in *Figure 2 (page 18)*.

There are two modes of operation for the experimental setup namely system training and part identification. Each of these modes is described below.

3.1.1. System Training

Before the system can successfully identify parts, the system has to be trained by presenting examples of the parts to be identified to it. The parts selected for this research was chosen as they are commonly used industrial components (push buttons, selector switches etc.) and also because they represent a wide variety of possible shapes. The selected parts can be seen in *Table 1 (page 6)*.

It is important to note that the system can be trained to identify any parts presented to it as long as it meets the following criteria:

- The part can physically fit onto the conveyor system. Refer to *Figure 5 (page 22)*.
- There is a significant geometric difference between the various parts as the ANN attempts to classify similar parts into trained groups. Refer to *Section 4.1 Artificial Neural Network operation principle* for more information on this generalization capability of ANNs.

The steps and processes in the training procedure are explained below (refer to *Figure 5 (page 22)*):

- a) The part to be trained is placed on the conveyor. The Start sensor detects the parts presence and starts the conveyor.
- b) As the part passes the Time-, X-axis and Y-axis sensors the dimensional data is recorded as stored in memory on the PLC.
- c) The Stop sensor detects the part as it reached the end of the conveyor and the conveyor is stopped.
- d) The captured dimensional data for that training run is transferred to an Excel File by means of an OPC server. The OPC component has been excluded from the experimental system setup because of the high cost involved. In the experimental system the captured data is manually copied from variable monitoring tables to an Excel document on the IPC.

- e) Depending on the amount of training data sets required for the test condition, steps a) to d) is repeated.
- f) The captured data in the Excel document is loaded into the developed Microsoft Visual Basic GA training application. This application and its operation are described in detail in *Chapter 5 – Genetic Algorithms Applied to Neural Network Training*.
- g) Once the GA application has evolved and optimized the required ANN connection weights for the part being trained, they are saved in an Excel document.
- h) The optimized connection weights are transferred from the Excel file to assigned PLC memory variables by means of an OPC server. The OPC component has been excluded from the experimental system setup because of the high cost involved. In the experimental system the evolved weights are manually copied from the Excel file on the IPC to the associated variable monitoring tables on the PLC.
- i) Steps a) to h) are repeated for each part, up to a maximum of six, to be trained.

3.1.2. Part Identification Operation

Once the system has been trained to identify the various parts, the part identification process is executed as follows:

- a) The part to be identified is placed on the conveyor. The part is detected by the Start sensor and the conveyor starts.
- b) The leading edge of the part is detected by the Time sensor and a timer is started (Siemens AG, 2011).
- c) The trailing edge of the part is detected by the Time sensor and the timer is stopped. The time taken for the part to pass is recorded in memory.
- d) The leading edge of the part is detected by the X- and Y-Axis sensor. As the part moves past the X- and Y-sensor, 10 dimensional points are captured and stored in memory.
- e) The leading edge of the part is detected by the Stop sensor and the conveyor stops.
- f) The captured Time, X- and Y-dimensional points are passed to the ANN as inputs.
- g) The ANN identifies the part and the appropriate output is switched HIGH.
- h) The process can then be repeated with the next part being loaded onto the conveyor.

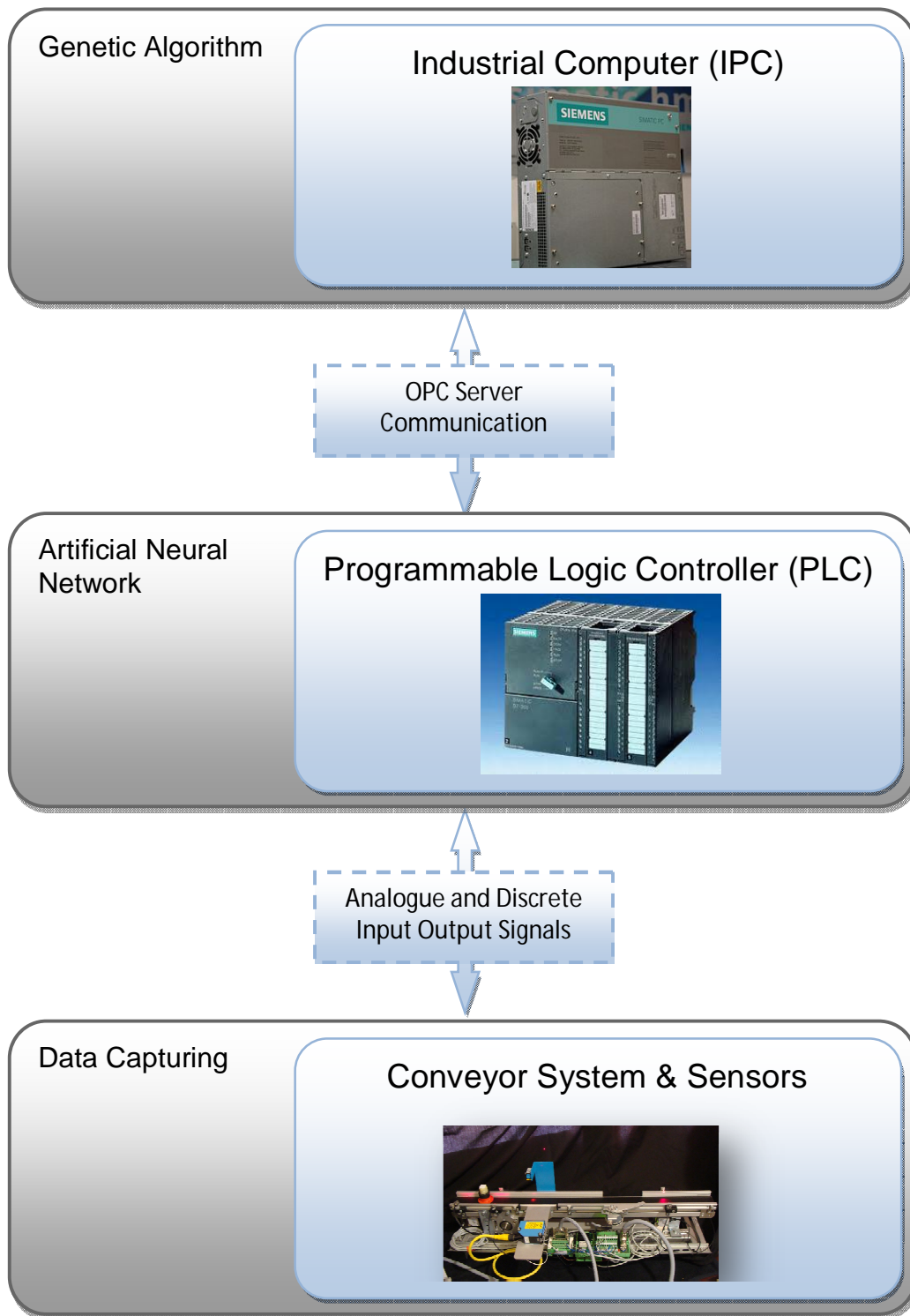


Figure 2 - Experimental System Setup Block Diagram

3.2. Hardware Architecture

In this section the hardware components of the experimental setup will be discussed. An overview of the hardware architecture can be seen in *Figure 4 (page 20)*.

The main hardware components of the experimental system setup are the PLC, Industrial PC (IPC), data collection sensors and the conveyor system.

3.2.1. PLC

The CPU hosts the ANN, data acquisition and conveyor control software while the Input/Output (I/O) Module captures the analogue and digital signals from the sensors and controls the conveyor with digital output signals.

The PLC consists of two main components that can be seen in *Figure 3 (page 19)*:

- CPU (Model Number: S7-300 ; Part Number: 313-6CF03-0AB0)
- I/O Module (Part Number: 334-0CE01-0AA0)

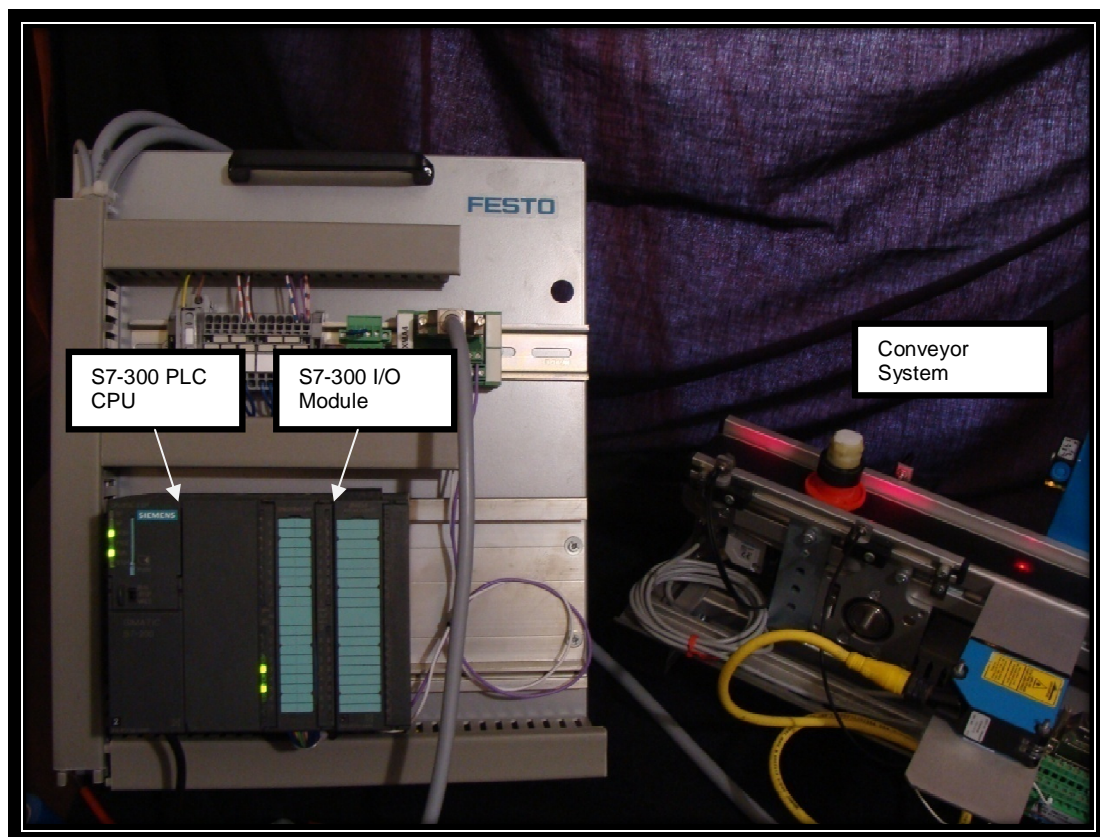


Figure 3 - PLC Test System

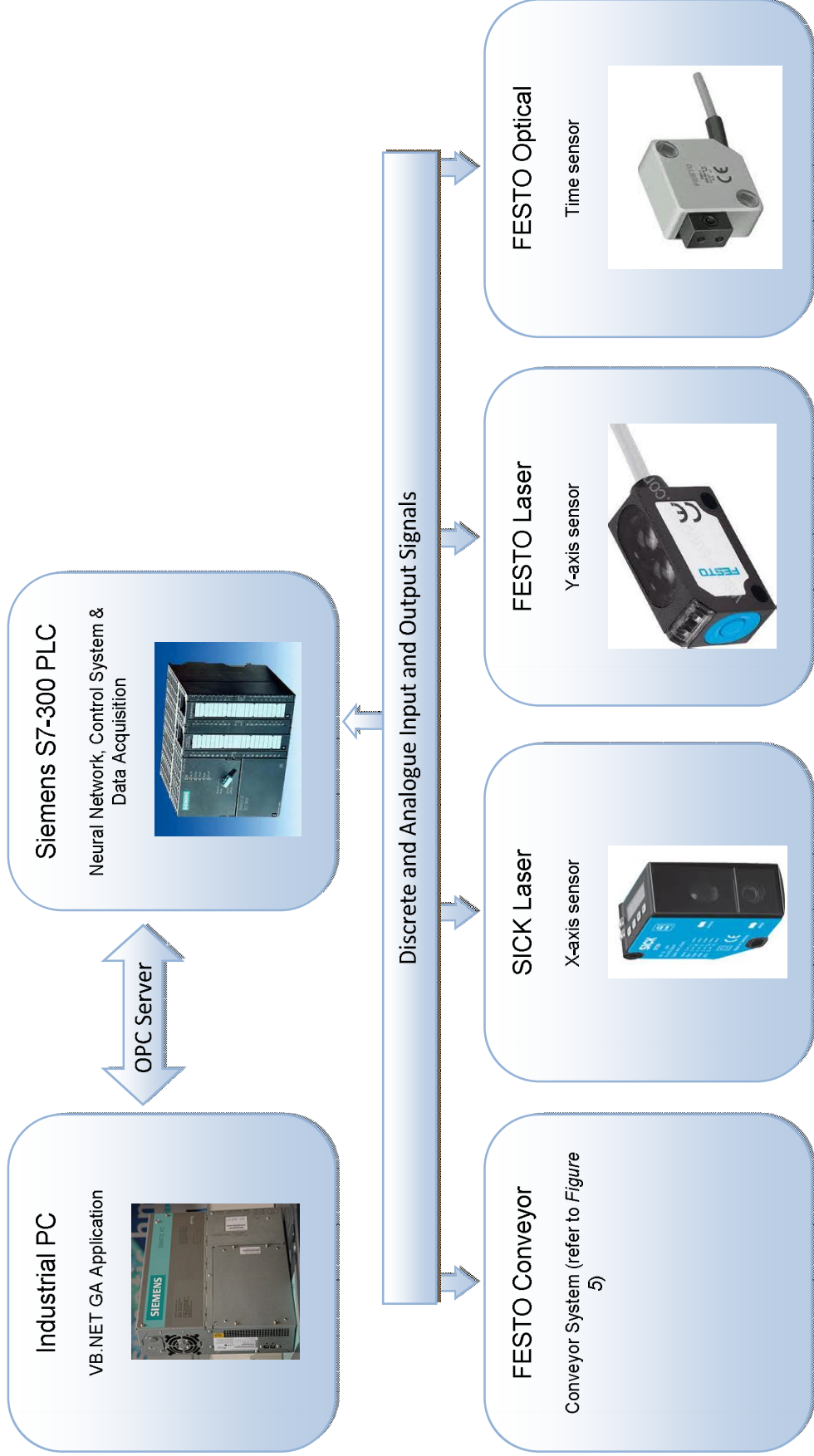


Figure 4 - Hardware Architecture

3.2.2. IPC

Due to the high cost of IPCs, the computer used in this research was not an industrial specification PC but a high end laptop computer.

The key specifications of the used computer are (Toshiba Corporation, 2011):

- Model Number: Toshiba Satellite Pro L650-15J.
- Processor: Intel Core i5-450M 2.40/2.66 GHz.
- Operating System: Microsoft Windows 7 Professional 64-bit.
- System memory: 6GB DDR3 1066MHz.

The key specifications of the computer recommended for industrial environments are (Siemens AG, 2011):

- Model Number: Siemens Simatic IPC547D.
- Processor: Configurable Intel Core up to i7.
- Operating System: Microsoft Windows 7 Ultimate 64-bit.
- System memory: Configurable DDR3 up to 32GB.

As the Operating System (OS) and key hardware specification of the used laptop is very similar to that of the recommended IPC for industrial environments, the GA performance results of the research will not be affected.

3.2.3. Data Collection Sensors

In this section the various data collection sensors will be discussed.

3.2.3.1. Time Sensor

The time taken for the part to pass the sensor is measured using a Reflex/Interrupt FESTO Fibre-optic device. The device operates in the red light frequency range. The IP65 protection rating of the device makes it a viable option to use in a production environment. Part number: SOEG-L-Q30-P-A-S-2L (FESTO AG & Co., 2002).

3.2.3.2. Sensor X

The X-axis dimensional data is captured using a red light 2nd class laser FESTO distance sensor. This sensor has a range of 40 – 150mm and a maximum sampling rate of 1kHz . The IP67 protection rating of the device makes it a viable option to use in a production environment. Part number: FESTO SOEG-RTD-Q20-PP-K-2L-T1 (FESTO AG & Co., 2007).

3.2.3.3. Sensor Y

The Y-axis dimensional data is captured using a red light 2nd class laser SICK distance sensor. This sensor has a range of 50 – 150mm, a response rate of less than 15ms and an accuracy of ± 0.5 mm. An analogue 4 - 20mA signal indicating the measured distance is output. This output is in turn measured by the S7-300 PLCs analogue current input (Siemens AG, 2011). The IP57 protection rating of the device makes it a viable option to use in a production environment. Part number: SICK AG DT-20-P254B. (SICK AG, 2008).

3.2.4. Conveyor System

The purpose of the conveyor system is to move the part to be identified past the various data capturing sensors. The conveyor system (Part Number: MP3-M-PF-700) and the various sensor placements can be seen in *Figure 5 (page 22)*.

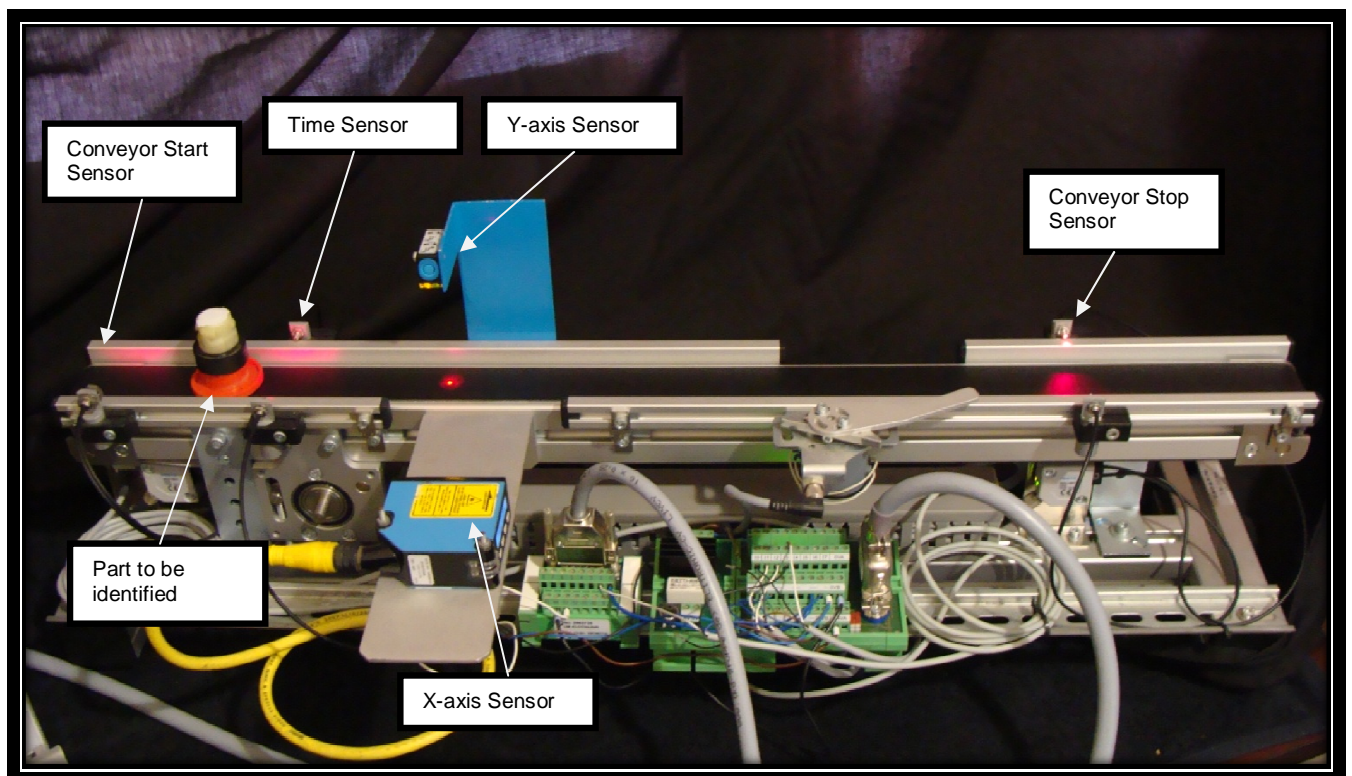


Figure 5 - Conveyor System

3.3. Software Architecture

An overview of the software architecture and the associated communication can be seen in *Figure 6 (page 23)*. The software components of the research platform consist of three main components namely:

- ANN.
- Conveyor automation and data acquisition.
- GA.

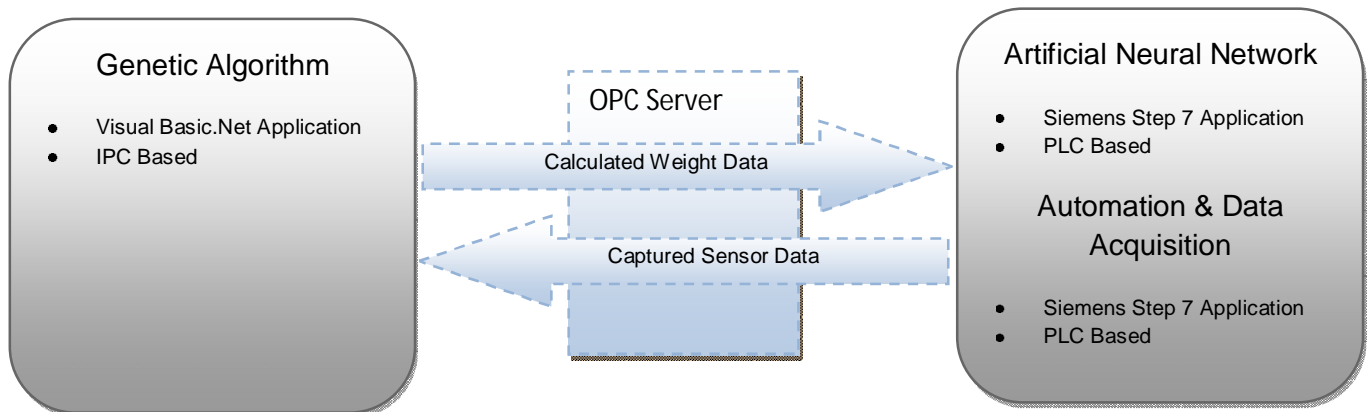


Figure 6 - Software Architecture

3.3.1. Artificial Neural Network

The main purpose of the ANN is to provide intelligence and decision making to the system. The ANN calculates, and decides, which of the trained parts is presented to it based on the inputs provided by the data acquisition component.

Credit for the initial software and concept development of the ANN in this application goes to Mr. Hennie van Rooyen as he developed this for use in his research into ANN based part recognition.

Various changes have been made to the originally developed neural network in order to perform this research into GA training:

- Removal of hard coded X- and Y-Neuron bias weight values. Both the X- and Y-neuron in the middle layer of the ANN had hard-coded weight values. These were determined by trial and error which is not practical for real world application as new parts would need to be added relatively frequently. These bias weights are now calculated by the GA during the training process.
- Removal of hard coded input weight values. Because of the limited and inaccurate ANN training the original system provided, the system was intolerant to outliers in input data or measurements. As a result, certain problematic data capture points

were nulled by manually setting the weight values for these inputs close to, or, zero. The problematic inputs were identified by manual inspection of input data. This is not viable in an industrial environment. All weight values are now calculated by the GA during the training process.

- Removal of hard coded fuzzy logic on the final output stage. Because of the limited and inaccurate training, the ANN would frequently incorrectly identify a part as another and thus misfire. As only three parts were used for the initial research, the misfire of one part for another was predictable. Basic fuzzy logic was hard coded to prevent this problem. This fuzzy logic can be simply explained as: IF identified as part A and part B, only fire part A. IF identified as part B only, identify as part B. This is not practical for real world application as new parts trained would result in a different fuzzy logic set. Accurate training resulted in avoiding the above misfire problem without the need for hard coded fuzzy logic.
- Adaption of the ANN architecture (refer to Section 4.4 *Neural Network Architecture* for more information). The original ANN was developed to recognize three parts. In order to increase the amount of parts between which the system can differentiate, more neurons and their associated connections had to be added to the architecture.

Software development was performed using Siemens Step 7 as this ANN component of the research platform is executed on a Siemens S7-300 PLC.

3.3.2. Genetic Algorithm

The main purpose of this software component is to train the ANN to correctly identify parts depending on their captured data vectors presented to it.

Software development was performed using Visual Basic 2008, part of the Microsoft Visual Studio 2008 suite.

Selection criteria for this programming language are:

- High level of computational power is required because of the complexity of the GA. The basic hardware specification capable of supplying this required processing power is a 64-bit quad-core Intel i5 processor with 6GB of DDR3 RAM. A software language developed to fully utilize this processing power is required.
- Complexity of calculations and programming loops.
- Object orientated programming to simplify operation.
- Complex data structures required (multi-dimensional variable arrays).
- Familiarity with software and development environment.

3.3.3. Conveyor automation and data acquisition

The main purpose of this software component is to control the conveyor that moves the part to be recognized past the Time, X- and Y-axis measurement sensors and to capture the various input data measurements.

Software development was performed using Siemens Step 7 as this component of the research platform is executed on a Siemens S7-300 PLC.

Once the leading edge of the part is detected the X- and Y-axis sensors capture ten measurements as the part passes. Ten part training data sets for the X- and Y-axis of part B and part E have been charted below (along with the average trend line in bold black). Part B and part E were randomly selected to be discussed.

Standard deviation is a widely used measure of variability or diversity and shows how much variation or scattering of data exists from the average. A low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data points are spread out over a large range of values.

In addition to expressing the variability of a population, standard deviation is commonly used to measure confidence in statistical conclusions. By looking at the standard deviation of the captured data sets we can determine whether there is good repeatability and confidence in the data capturing method.

The standard deviation is a measure of the average difference of each case from the mean, and can be calculated as

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n - 1}}$$

Equation 1 - Standard Deviation (DeCoster, 1998)

As can be seen in Table 4, 5, 6 and 7 which tables the average, standard deviation, standard deviation to average percentage and average standard deviation for the captured training data for X- and Y-points for part B and E, the standard deviation between the various training sets is very low. This indicates good confidence and repeatability of the data capturing method.

Table 4 - Part B X-point Training Data Standard Deviation

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Average	Standard Deviation	STDEV/AVG Percentage
Point X:1	4736	3840	5760	4736	6556	5376	8192	7552	7424	4736	5900.800	1 471.165	24.932%
Point X:2	3584	3584	3840	3712	4096	4096	6144	4480	4224	3712	4147.200	761.334	18.358%
Point X:3	3840	3584	3584	3584	3584	3712	4096	3712	3712	3712	3712.000	159.644	4.301%
Point X:4	3712	3584	3712	3712	3712	3584	3712	3712	3712	3840	3699.200	72.659	1.964%
Point X:5	3712	3456	3456	3584	3584	3584	3840	3712	3712	3712	3635.200	123.660	3.402%
Point X:6	3968	3584	3712	3712	3712	3968	3840	3840	3584	4224	3814.400	198.297	5.199%
Point X:7	7296	6144	5376	5888	4736	6144	4096	4608	3968	7552	5580.800	1 250.068	22.399%
Point X:8	7808	7936	8192	8192	8192	8192	6912	8192	7680	7680	7897.600	409.467	5.185%
Point X:9	4352	3584	4864	4352	5888	4992	7936	6912	7680	3840	5440.000	1 581.835	29.078%
Point X:10	3712	3456	3584	3712	3584	3712	3968	3712	3968	3712	3712.000	159.644	4.301%
	Average											618.777	11.912%

Table 5 - Part B Y-point Training Data Standard Deviation

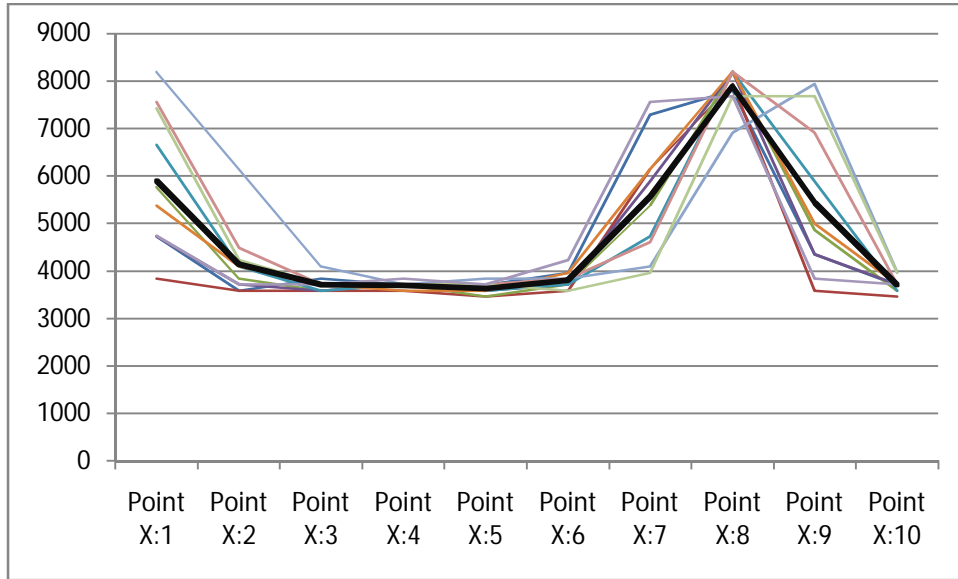
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Average	Standard Deviation	STDEV/AVG Percentage
Point Y:1	19584	19328	19712	19200	19712	19712	19328	19584	19200	19840	19520.000	235.634	1.207%
Point Y:2	19968	19328	19840	19200	19712	19840	20224	19712	19200	20352	19737.600	399.338	2.023%
Point Y:3	19584	19840	19840	19328	19840	19840	20096	20224	19456	19840	19788.800	271.194	1.370%
Point Y:4	19584	19328	19840	19200	19840	19840	20224	19840	19456	19840	19699.200	304.402	1.545%
Point Y:5	19584	19328	19840	19328	19840	19840	19968	19840	19584	20096	19724.800	259.180	1.314%
Point Y:6	19584	19328	19968	19456	20096	19968	19840	20352	19712	19840	19814.400	306.488	1.547%
Point Y:7	19584	19200	19840	19328	19840	19840	19712	19840	19584	19840	19660.800	235.248	1.197%
Point Y:8	19584	19200	19840	19328	19840	19840	19712	19968	19712	19840	19686.400	247.319	1.256%
Point Y:9	19584	19328	19968	19328	19968	19968	19584	20096	19968	16000	19379.200	1220.595	6.298%
Point Y:10	4864	4864	4864	4864	4864	4864	19456	4864	4864	4864	6323.200	4614.396	72.976%
	Average											809.379	9.073%

Table 6 - Part E X-point Training Data Standard Deviation

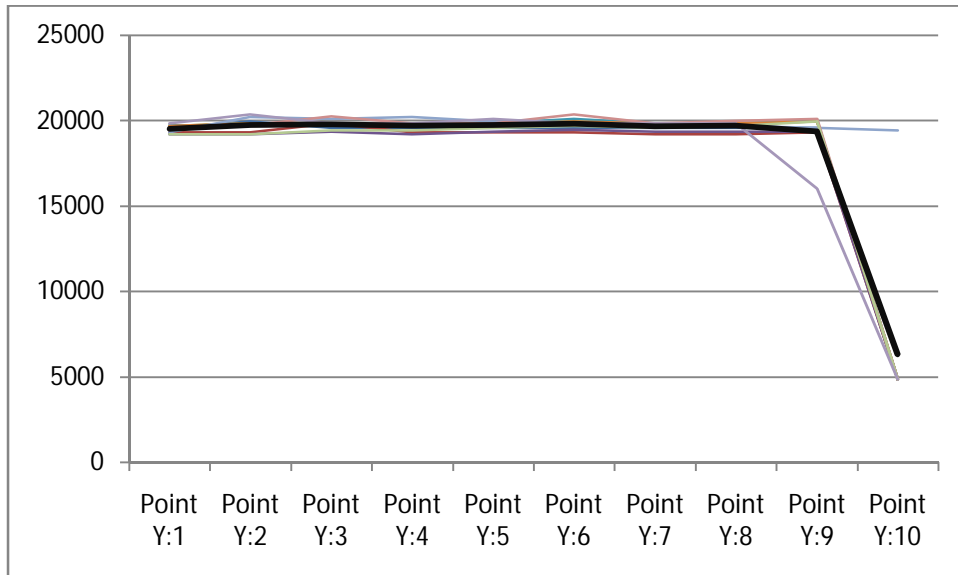
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Average	Standard Deviation	STDEV/AVG Percentage
Point X:1	24192	23168	23168	24704	24320	23296	24064	24448	23808	23296	23846.400	578.916	2.428%
Point X:2	23552	21376	22528	23296	24320	22400	23296	23424	21248	20480	22592.000	1 219.177	5.396%
Point X:3	23168	22528	21376	21888	24832	21248	20992	24320	21504	23040	22489.600	1 331.651	5.921%
Point X:4	24320	25600	23424	22784	25088	24192	22144	23680	25472	25856	24256.000	1 255.952	5.178%
Point X:5	23040	25472	23936	23680	24064	24064	22528	22656	24448	24448	23833.600	900.661	3.779%
Point X:6	20096	19200	20352	22656	17408	18944	21632	17664	19328	16512	19379.200	1 901.613	9.813%
Point X:7	13056	10240	13184	12672	11648	10752	13952	9856	13696	8960	11801.600	1 760.019	14.913%
Point X:8	5376	4992	4736	4992	5120	4864	5888	5120	4992	4736	5081.600	341.600	6.722%
Point X:9	4352	4608	4608	4736	4992	4608	4480	4864	4608	4096	4595.200	252.059	5.485%
Point X:10	3712	3712	4096	3968	3840	3840	3840	3584	3840	3584	3801.600	160.213	4.214%
	Average											970.186	6.385%

Table 7 - Part E Y-point Training Data Standard Deviation

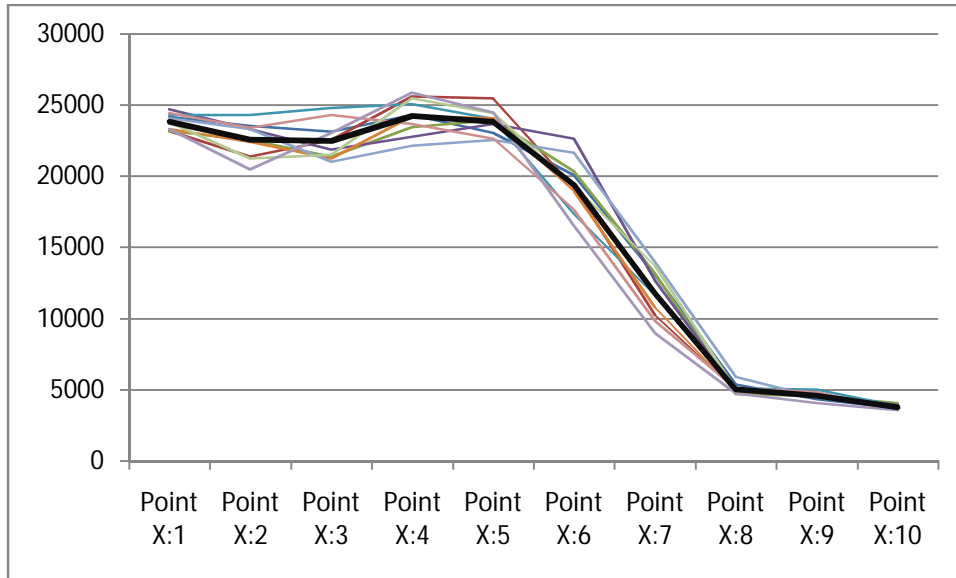
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Average	Standard Deviation	STDEV/AVG Percentage
Point Y:1	17408	17792	17792	17664	18048	17792	17792	18176	17920	18048	17843.200	219.225	1.229%
Point Y:2	18176	18176	18176	18048	18176	18176	18048	18176	18176	18304	18163.200	72.659	0.400%
Point Y:3	18304	18176	18304	18304	18176	18304	18304	18304	18304	18304	18278.400	53.970	0.295%
Point Y:4	18176	18048	18176	18304	18176	18176	18176	18048	18176	18176	18163.200	72.659	0.400%
Point Y:5	17792	17792	17920	17920	17664	17920	17920	17792	17792	17664	17817.600	100.968	0.567%
Point Y:6	17280	17024	17408	17408	16768	17280	17664	16768	17152	16768	17152.000	313.535	1.828%
Point Y:7	16384	16384	16512	16256	6912	16384	16384	5888	16384	6144	13363.200	4870.807	36.449%
Point Y:8	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864.000	0.000	0.000%
Point Y:9	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864.000	0.000	0.000%
Point Y:10	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864.000	0.000	0.000%
	Average											570.382	4.117%



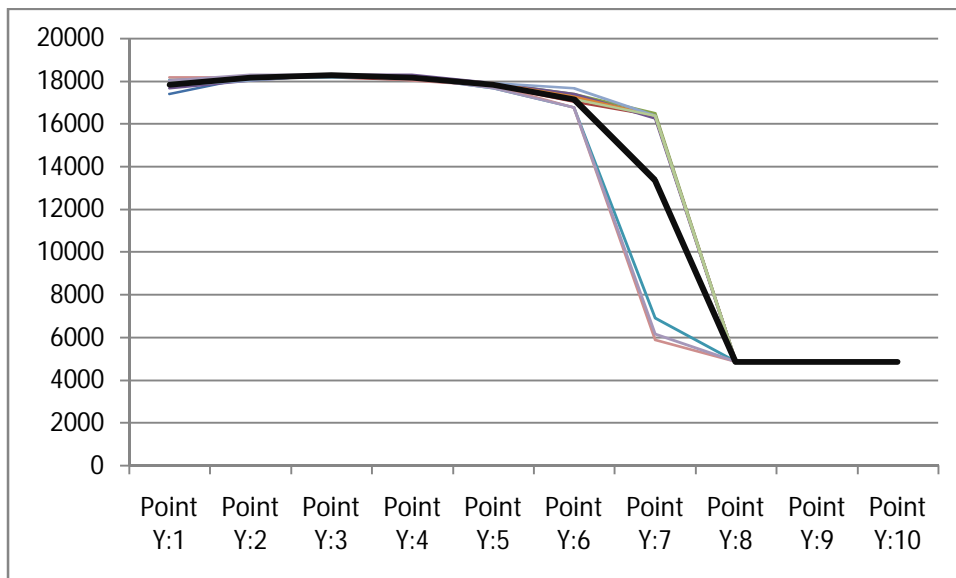
Graph 1 - Part B X-axis Data Capture



Graph 2 - Part B Y-axis Data Capture



Graph 3 - Part E X-axis Data Capture



Graph 4 - Part E Y-axis Data Capture

3.3.4. PLC Simulator

Evolved solutions to the ANN weight optimization problem for each of the 35 identified condition sets (as defined in Section 6.1 Genetic Algorithm Control Parameters) had to be evaluated using part data sets that were unfamiliar to the ANN in order to test the generalization ability of the ANN.

Ideally the best solution for each part for each training condition should be tested on the hardware research platform (as discussed in Section 3.2 *Hardware Architecture*) a minimum of 50 iteration sin order to test the quality of the optimized solution. Unfortunately that would result in 10 500 test iterations (6 parts x 35 conditions x 50 test iterations = 10 500 iterations).

This is impractical due to time required (estimated over 90 hours) as well as possible hardware damage.

A PLC simulator was developed as a solution to this problem; which fulfils the criteria of testing the evolved solutions with data sets previously unknown to the system (not used for training purposes) as well as being practical. Depending on these results, conditions for real-world hardware testing were selected.

Each part was passed through the conveyor system 50 times and the sensor data sets recorded for each iteration. These captured data sets were loaded into the PLC simulator along with the relevant evolved weight values for the relevant test condition set to be tested. The PLC simulator contains a duplicate of the ANN programmed into the hardware PLC. The output of the various ANNs for the current input data sets and weight sets are simulated and recorded to a readable Excel file.

Below are screenshots of the application with explanations of each step in the process.

- a) The evolved weight values to be evaluated is loaded into the PLC simulator application by either reading from an excel file or by manual entry in the *Weight Data* tab.

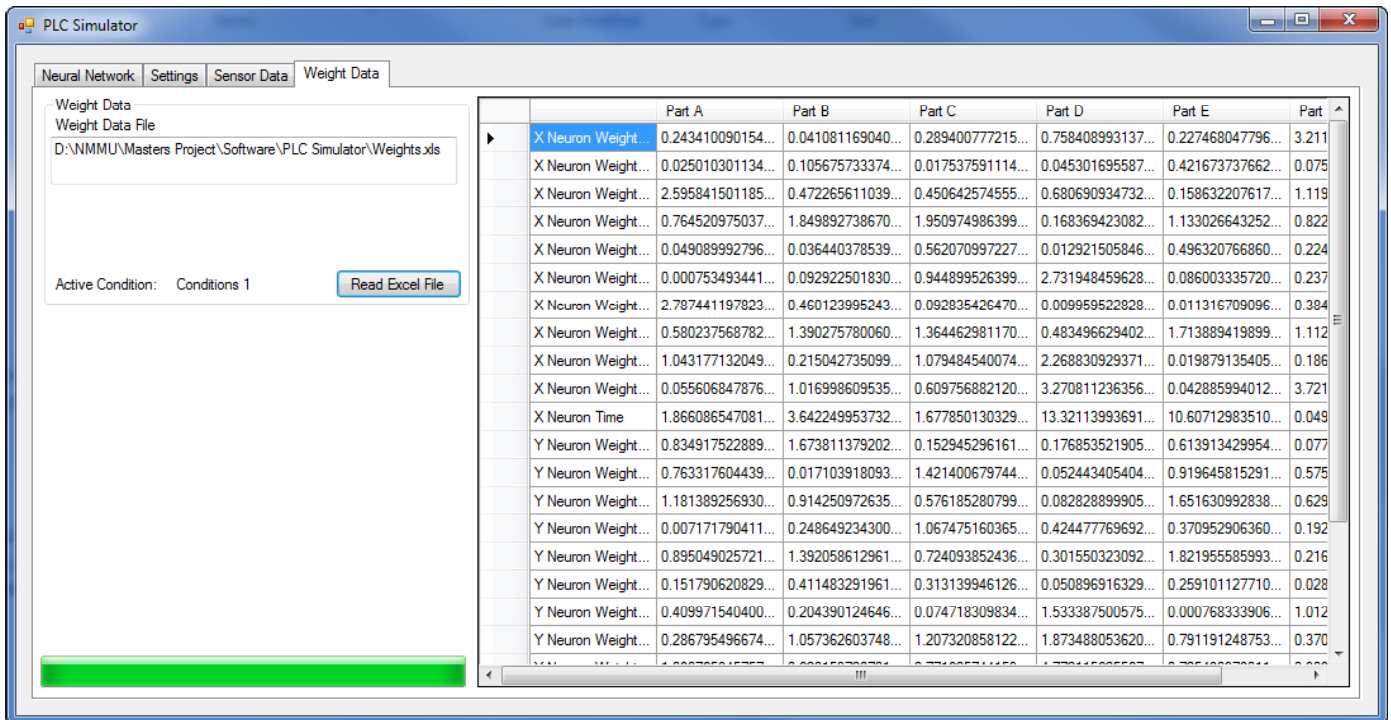


Figure 7 - PLC Simulator Weight Data Tab

- b) The test data sets previously captured for each of the sensor capturing points is loaded into the PLC simulator application from excel sheets or by manual entry in the *Sensor Data* tab.

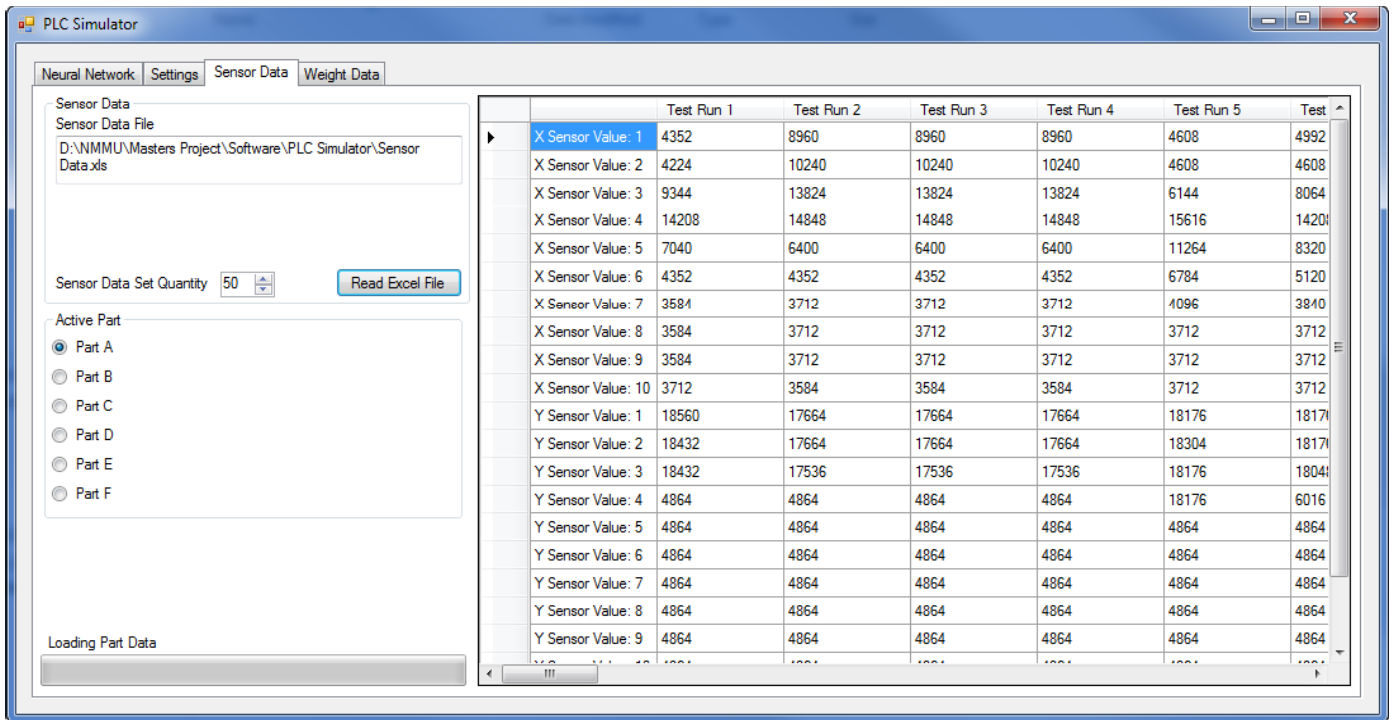


Figure 8 - PLC Simulator Sensor Data Tab

- c) Directory and file names for the application log files as well as result output files are specified in the *Settings* tab.

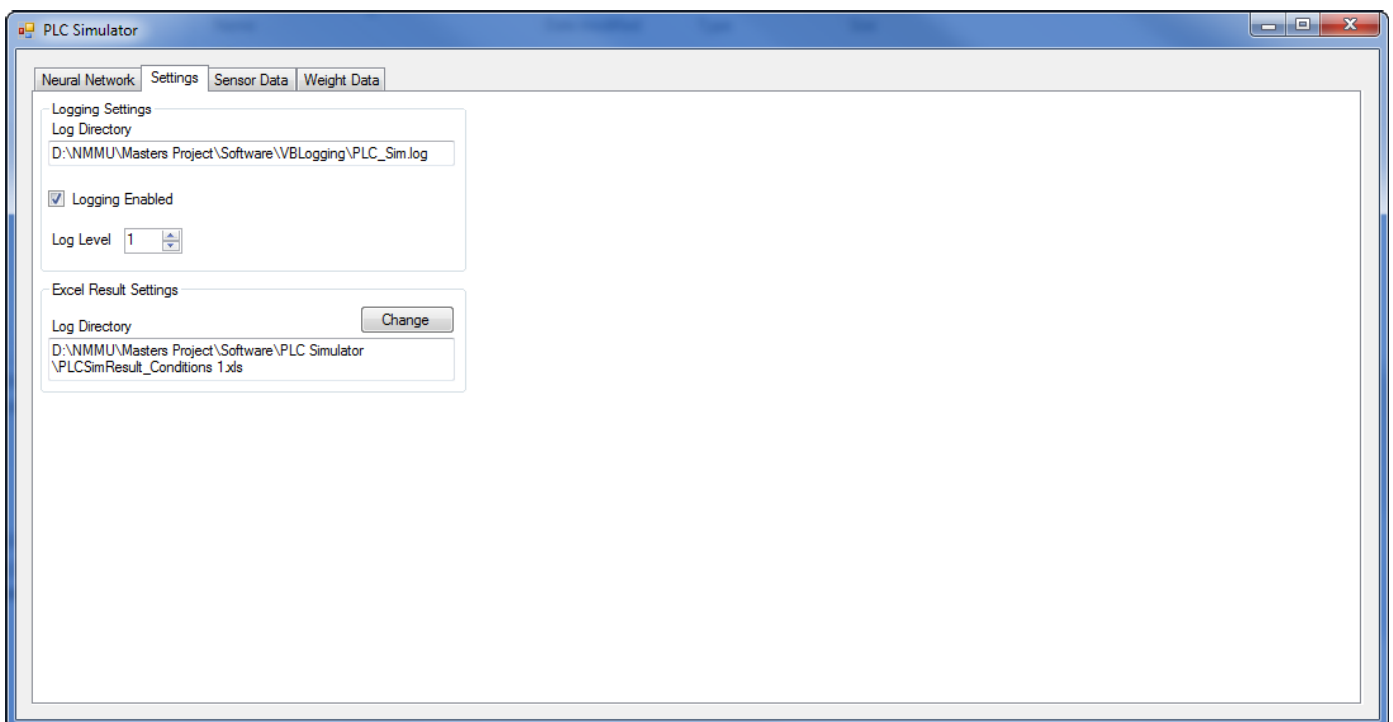


Figure 9 - PLC Simulator Settings Tab

- d) In the *Neural Network* tab the following group boxes are available:
- *Conditions to be simulated* group box.
 - i. Specify which of the 35 test conditions is to be evaluated.
 - ii. Specify which Parts are to be evaluated.
 - *Activation Function Limits* group box.
 - i. Activation function limits for the final output stage of each of the Neural Networks.
 - *Results* group box.
 - i. Results of the Neural Networks simulation.
 - *Breakdown* group box.
 - i. Summary of the Neural Networks Simulation Results.

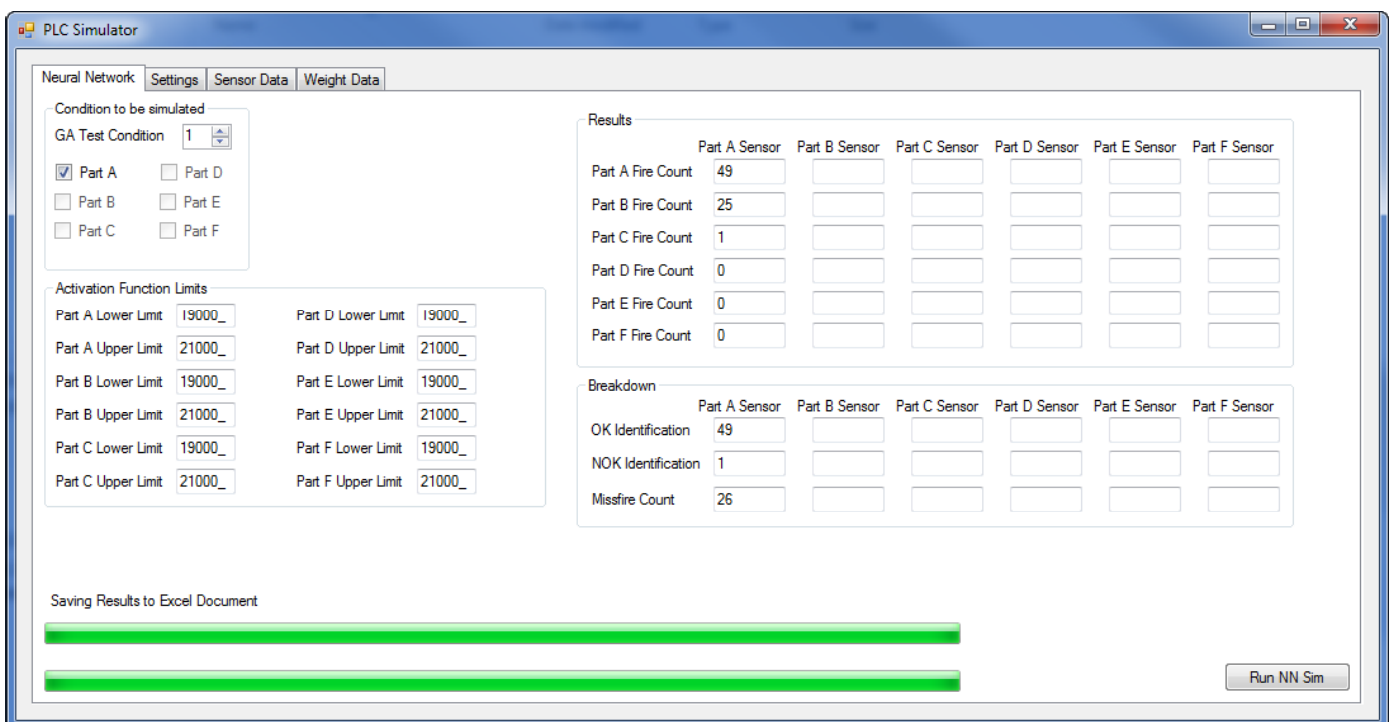


Figure 10 - PLC Simulator Main Neural Network Tab

e) A result summary is output into the specified Excel file.

	A	B	C	D	E	F	G	H	I
1	Conditions 1 on 09-29-2011 at 03:12:25 PM	Part A Sensor Data	Part A Sensor Data	Part A Sensor Data	Part A Sensor Data	Part A Sensor Data	Part A Sensor Data		
2	Part A Fire Count	49	0	0	0	0	0		
3	Part B Fire Count	25	38	1	0	0	0		
4	Part C Fire Count	1	0	48	0	0	0		
5	Part D Fire Count	0	0	0	47	6	0		
6	Part E Fire Count	0	0	0	2	48	0		
7	Part F Fire Count	0	0	0	0	0	49		
8	Identification OK	49	38	48	47	48	49	279	
9	Identification NOK	1	12	2	3	2	1	21	
10	Missfire	26	0	1	2	6	0	35	

Figure 11 - PLC Simulator Overview Excel Output

f) Detailed results for each ANN layer is output to the specified Excel file.

	A	B	C	D	E	F	G	H	I	J	K	L	
1		Sensor Data 1	Sensor Data 2	Sensor Data 3	Sensor Data 4	Sensor Data 5	Sensor Data 6	Sensor Data 7	Sensor Data 8	Sensor Data 9	Sensor Data 10	Sensor Data 11	Sensor
2	Part A X-Neuron	52661.54283	66578.35613	66578.35613	66578.35613	47347.3681	50489.04802	66753.81952	60869.61075	61660.87098	54481.29053	65977.12866	58
3	Part A Y-Neuron	66180.87839	63788.03959	63788.03959	63788.03959	65555.60063	65219.46918	63940.71524	63880.37834	63771.13309	64128.76099	63636.82177	63
4	Part A Output	20669.63748	19981.38327	19981.38327	19981.38327	20456.36427	20364.11398	20029.26854	19988.6306	19957.79041	20041.6123	19932.35908	19
5	Part A Result	1	1	1	1	1	1	1	1	1	1	1	1
6	Part B X-Neuron	43189.00723	47424.61474	47424.61474	47424.61474	45154.47845	43092.65936	44408.0262	47783.06376	51792.90376	41698.72696	48899.07162	43
7	Part B Y-Neuron	68489.56846	66157.52879	66157.52879	66157.52879	70920.60595	67777.81783	66274.72817	69308.58714	69062.33692	69459.45241	66040.50466	66
8	Part B Output	19706.14381	20890.94918	20890.94918	20890.94918	20546.2659	19615.98981	19919.56436	21267.93008	22551.68039	19301.61877	21360.80862	1
9	Part B Result	1	1	1	1	1	1	1	0	0	1	0	0
10	Part C X-Neuron	52701.66078	57295.16234	57295.16234	57295.16234	59119.93892	54098.51408	53557.61783	61936.71053	67021.75651	54559.57561	59383.50371	53
11	Part C Y-Neuron	63096.203	61351.26628	61351.26628	61351.26628	76918.25862	63682.06967	61426.46546	74879.75898	74722.09769	75270.63933	61277.51456	61
12	Part C Output	22661.83591	24619.62644	24619.62644	24619.62644	25436.47439	23259.87261	23023.41802	26634.67747	28806.23571	23484.6775	25511.42659	22
13	Part C Result	0	0	0	0	0	0	0	0	0	0	0	0
14	Part D X-Neuron	46372.78358	53223.95819	53223.95819	53223.95819	51699.26159	48473.8003	52774.02253	58224.2764	56453.20435	50925.9406	52723.76576	51
15	Part D Y-Neuron	56982.73231	56709.78032	56709.78032	56709.78032	62537.55167	57358.58714	56734.29339	62102.67359	62011.79222	62160.41063	56699.17823	56
16	Part D Output	13687.87004	14122.21691	14122.21691	14122.21691	15079.1587	13905.02809	14094.89565	15460.8709	15319.09508	14955.6019	14084.92836	13
17	Part D Result	0	0	0	0	0	0	0	0	0	0	0	0
18	Part E X-Neuron	30718.21286	35639.22039	35639.22039	35639.22039	34559.38141	31748.8265	33124.06771	38041.19884	41871.69088	34460.16971	36906.97408	32
19	Part E Y-Neuron	82073.10369	79336.8879	79336.8879	79336.8879	86234.95382	81395.04305	79556.80555	84046.11201	83911.54024	84414.20853	79125.47913	79
20	Part E Output	13974.46291	13687.38276	13687.38276	13687.38276	14751.77493	13897.64136	13646.70393	14508.44693	14602.25818	14459.27185	13691.8966	13
21	Part E Result	0	0	0	0	0	0	0	0	0	0	0	0
22	Part F X-Neuron	58908.54361	79301.38307	79301.38307	79301.38307	59225.19359	60294.76089	78532.41103	83096.1207	84458.57174	74514.0943	79702.81504	74
23	Part F Y-Neuron	40899.9801	39824.42564	39824.42564	39824.42564	43196.83469	40702.79569	39905.42933	42263.14841	42228.17572	42441.61092	39743.84834	39
24	Part F Output	7980.771944	8138.257392	8138.257392	8138.257392	8378.870063	7970.2588	8139.235941	8618.848021	8635.677622	8505.685604	8131.198325	80
25	Part F Result	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12 - PLC Simulator Individual Part Excel Output

3.4. Conclusion

In this chapter the research platform, system setup and architecture has been described and discussed. A broad overview of the experimental test setup is described, the various hardware and software components, their operation and selection criteria are discussed.

A detailed analysis of the data capturing method and actual captured data is performed as the accuracy of the input data to the ANN is of extreme importance. The analysis revealed that the data capturing is accurate and repeatable enough to avoid excessive statistical outliers but still with enough variation to allow the ANN to generalize and not over fit. The results also show that the selected hardware (sensors, conveyor and control) meets the required criteria.

In the next chapter the mathematical fundamentals, operation, training methodologies, architectures and parameters of ANNs are discussed.

4. Chapter 4 – Neural Network Training Optimized for Part Recognition Applications

In this chapter the history, operational principles and components of ANNs are discussed. In the first section of this chapter the mathematical fundamentals, advantages and applications of ANNs are discussed. Various network architectures, activation functions and finally training methodologies are discussed.

The human brain has the ability to perform amazingly complex tasks namely: we can walk, talk, read, write, and recognize hundreds of faces and objects, irrespective of colour, distance, orientation and lighting conditions. In actual fact it is not the brain on its own that perform these tasks but the entire human body. The brain plays an essential role in this process, but it should be noted that the body itself (the shape or the anatomy, the sensors, their position on the body) and the materials from which it is constructed also do a lot of useful work in intelligent behaviour.

There are a variety of ANNs, each with slight advantages to the given task. Although each network may vary slightly, there is still common ground on which every ANN is based.

ANNs are mathematical models of real world systems which are built by tuning a set of parameters. These parameters are known as weights which describe the model by mapping a set of values, called inputs, to an associated set of values, called outputs. Like the human brain, the ANN consists of a number of Artificial Neurons (AN), which processes information provided to its inputs.

In the past knowledge based expert systems were used to model decision making tasks. These systems were very accurate but offered little scope and tolerance outside their programmable field. These systems were reliant on rules and sets of equations which gave specific answers from specific inputs. This was extremely fast at computing the answer as long as the rules related to the problem were known. For such cases where the rules relating to the problem were not known they could not be implemented. For this reason ANNs became more of an interest

4.1. Artificial Neural Network operation principle

An AN is a mathematical function conceived as a crude model, or abstraction of biological neurons (BN). ANNs consist of multiple ANs networked together with interlinking connections. ANs in one layer are connected, fully or partially, to the ANs in the next layer. Feedback connections to previous layers are also possible.

The AN receives one or more inputs (representing the one or more dendrites) and sums them to produce an output (representing a BNs axon). Usually the sums of each node are weighted with input signals inhibited or excited through negative and positive numerical weights associated with each connection to the AN, and the sum is passed through function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions.

The AN collects all incoming signals, or input vector, and computes a net input signal as a function of the respective weights. The net input signal serves as input to the activation function which calculates the output signal of the AN.

The output of an AN is calculated as the weighted sum of all the input signals.

$$net = \sum_{i=1}^I z_i \cdot v_i$$

Equation 2- Artificial Neuron Output

Where v_i is the weight of each input z_i .

A typical AN structure is depicted in *Figure 9 (page 31)*.

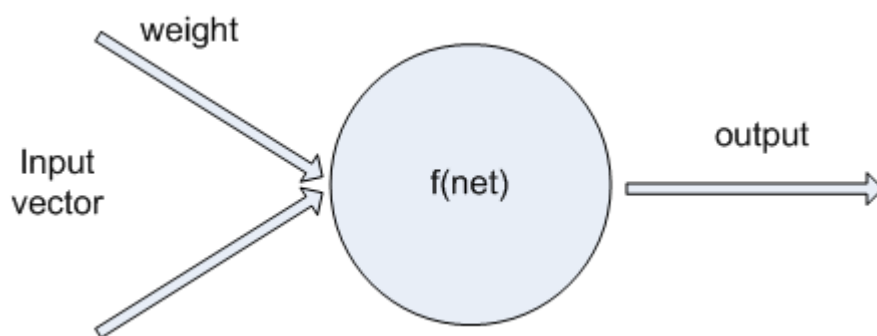


Figure 13 - Basic Artificial Neuron

4.1.1. Advantages of Artificial Neural Networks

ANNs have the ability to derive meaning from complicated, incomplete or imprecise data sets. This enables them to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained ANN can be thought of as an expert in the category of information it has been given to analyse. This expert can then be used to provide projected results given new sets of input data.

Other advantages include:

- Adaptive learning.

The ability to learn how to do tasks based on the initial training data available.

- Self-Organisation.

The ability to create its own organisation or representation of the information it receives during the learning process.

- Real Time Operation.

ANN computations may be carried out in parallel. Multi-core processors combined with operating systems supporting task multi-threading are allowing ANNs to take advantage of this capability.

4.1.2. Artificial Neural Network Applications

ANNs have been successfully implemented in a number of fields and have proven their ability to solve complex problems. Some of these applications include:

- Classification or pattern recognition.

A set of inputs are grouped together with similar input data sets.

- Pattern completion.

The missing data from an incomplete input data set is predicted.

- Optimization.

The optimal values for parameters in an optimization problem are found.

- Control.

An action is suggested depending on the input data set presented.

- Function approximation.

Functional relationships between input and output vectors are learnt and predicted.

- Data mining / Knowledge discovery.

Discovery of hidden patterns, and thus information, from raw data.

4.1.3. Real-world Artificial Neural Network uses

ANNs have been successfully implemented in a number of real-world applications and have proven their ability to solve complex problems. Some of these applications include:

- Anomaly detection applications.

ANNs are able to identify certain data records which do not fit the pattern of their peers.

- Data mining, cleaning & validation.

Data mining can be achieved by determining which records suspiciously diverge from the pattern of their peers.

- Handwriting and typewriting recognition.

Hand- and Typewriting recognition can be achieved by imposing a grid over the writing; each square of the grid then becomes an input to the neural network. This is called *Optical Character Recognition*.

- Voice recognition.

Voice recognition can be achieved by analyzing the audio oscilloscope pattern, much like a stock market graph.

- Medical Applications.

ANNs are an active research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modelling parts of the human body and recognising diseases from various scans.

ANNs are ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. ANNs learn by example so the details of how to recognise the disease are not needed. What is needed is a set of examples, or training data sets, that are representative of all the variations of the disease.

- Business Applications.

Business is a diverted field with several general areas of specialisation such as accounting or financial analysis. Almost any ANN application would fit into one business area or financial analysis.

There is potential for using ANNs for business purposes, including resource allocation and scheduling. There is also a strong potential for using ANNs for database mining, searching for patterns implicit within the explicitly stored information in databases.

ANNs have been successfully applied to exchange rate prediction applications (Knowles, Hussain, Deredy, & Lisboa, 2009), stock return predictions (Shi, Tan, & Ge, 2009), time series data analysis (Zhang, Ultra High Frequency Trigonometric Higher Order Neural Networks for Time Series Data Analysis, 2009), financial time series predictions (Liatsis, Hussain, & Milonidis, 2009) and stock index modelling (Chen, Wu, & Wu, 2009).

4.2. Feed-Forward Back-Propagation Neural Network

A popular type of ANN, and the type of network implemented in this research, which is often used for engineering problems, is called a Multi-Layer Feed-Forward Back Propagation Neural Network. It consists of a layered ANN with feed-forward connections through two hidden layers to the output layer which process the results. The term feed-forward is used when the network operates in one direction only and does not loop back into itself. The input layer receives information as discussed in Section 3.2.3 *Data Collection Sensors*. The output layer processes the information generated and applies the final binary activation function. The intelligence of the network lies in the hidden layers. The hidden layers link the input layer with the output layer. It extracts information from the input layer and remembers useful features and sub features to predict the outcome of the network. The goal is to map the input vector onto the output vector with minimum error to the original output data specified during the training process. Most previous researchers used the supervised learning technique. For each input vector presented to the network there is an associated output vector. The most widespread implementation of supervised training employs error BP techniques which will be discussed in Section 4.5.2 *Back-propagation Training*.

4.3. Activation Functions

The Activation Function (f_{AN}) calculates the output of the neuron by analysing the net input signal and bias. Commonly used activation functions are discussed below:

4.3.1. Step function

The output, $f_{AN}(net)$ of this transfer function is binary, depending on whether the weighted sum input meets a specified threshold, θ . The output is set to 1 if the threshold is met; otherwise it is set to 0.

$$f_{AN}(net - \theta) = \begin{cases} y_1 & \text{if } net \geq \theta \\ y_2 & \text{if } net \leq \theta \end{cases}$$

Equation 3 - Step Function

$$f_{AN}(net - \theta) = \begin{cases} 1 & \text{if } net \geq \theta \\ 0 & \text{if } net \leq \theta \end{cases}$$

Equation 4 - Step Function with binary output

It is especially useful in the last layer of a network intended to perform binary classification of the inputs.

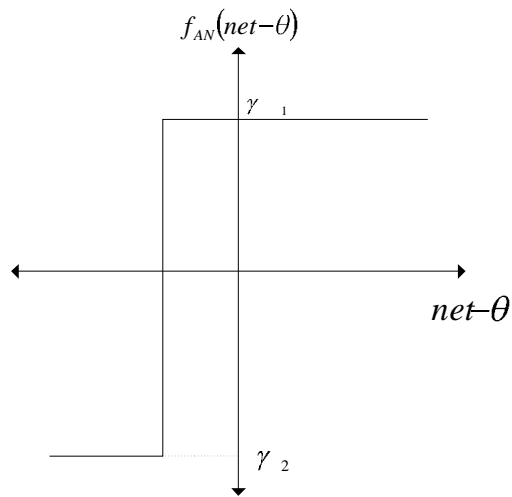


Figure 14 - Step Function

4.3.2. Linear function

The output unit is simply the weighted sum of its inputs plus a *bias* term. A number of such linear neurons perform a linear transformation of the input vector. This is usually more useful in the first layers of a network. A number of analysis tools exist based on linear models, such as harmonic analysis, and they can all be used in neural networks with this linear neuron. The bias term allows us to make affine transformations to the data. A slope, λ , can also be applied to the output.

$$f_{AN}(net - \theta) = \lambda(net - \theta)$$

Equation 5 - Linear Function

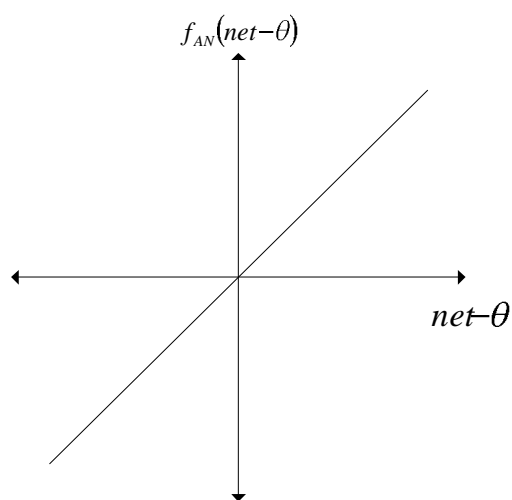


Figure 15 - Linear Function

4.3.3. Sigmoid

A Sigmoid function such as the logistic function also has an easily calculated derivative, which is important when calculating the weight updates in the network due to low processing overheads. It is commonly seen in multilayer ANNs using BP training.

$$f_{AN}(net - \theta) = \frac{1}{1 + e^{-\lambda(net - \theta)}}$$

Equation 6 - Sigmoid Function

4.3.4. Ramp Function

$$f_{AN}(net - \theta) = \begin{cases} \gamma & \text{if } net > \theta \\ net - \theta & \text{if } -\epsilon \leq \theta \text{ and } \geq \omega \\ -\gamma & \text{if } net < \omega \end{cases}$$

Equation 7 - Ramp Function

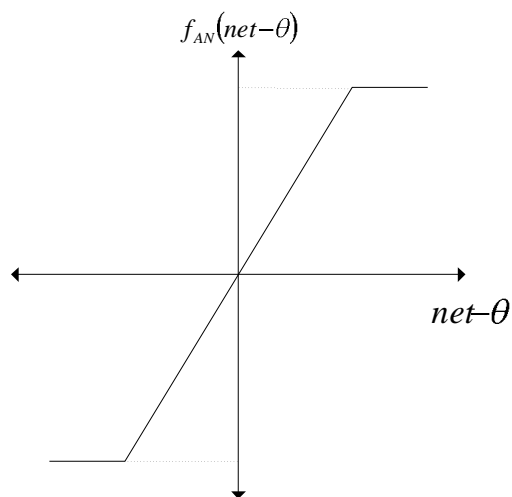


Figure 16 - Ramp Function

4.3.5. Ramp Function combined with Step Function

This function is a hybrid of the ramp- and step function where the output is binary dependant on predefined limits. This function is applied to the final output stage of the ANN used in this research.

$$f_{AN}(net) = \begin{cases} 0 & \text{if } net > \theta \\ 1 & \text{if } net \leq \theta \text{ and } \geq \omega \\ 0 & \text{if } net < \omega \end{cases}$$

Equation 8 - Hybrid Ramp and Step Function

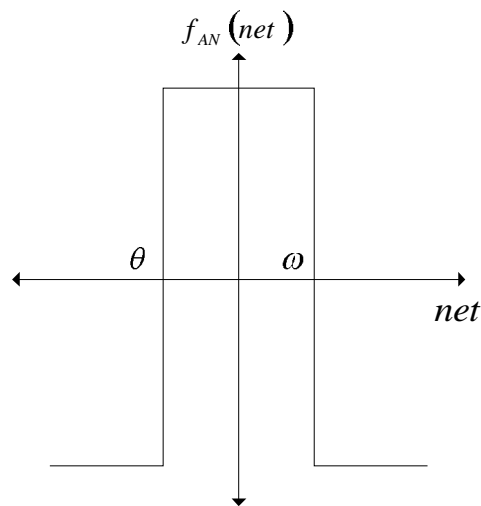


Figure 17 - Ramp and Step Function

4.4. Artificial Neural Network Architecture

Any ANN architecture is comprised of one or more layers of neurons, each with their individual activation function and interconnecting weighted connections. As such, the basic ANN architecture can be classified as either single-or multi-layer.

- Single-layer feed-forward networks

Single-layer feed-forward ANNs are composed of one layer, excluding the input layer, only. Haykin describes single-layer networks as follows:

“In a layered neural network the neurons are organized in the form of layers. In the simplest form of a layered network, we have an input layer of source nodes that projects onto an output layer of neurons (computational nodes), but not vice versa. In other words, this network is strictly a feedforward or acyclic type.” (Haykin, Neural Networks: A Comprehensive Foundation, 1999)

“Such a network is called a single layer network, with the designation “single –layer” referring to the output layer of computational nodes (neurons). We do not count the input layer of source nodes because no computation is performed there.” (Haykin, Neural Networks: A Comprehensive Foundation, 1999)

- Multilayer feed-forward ANNs

Multilayer feed-forward ANNs are composed of more than one layer, excluding the input layer. Haykin describes Single-layer Networks as follows:

“The second class of feed forward neural networks distinguishes itself by the presence of one or more hidden layers, whose computational nodes are correspondingly called hidden neurons or hidden units. The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is

enabled to extract higher order statistics. In a rather loose sense the network acquires a global perspective despite its local connectivity due to the extra set of synaptic connections and the extra dimension of neural interactions” (Haykin, Neural Networks: A Comprehensive Foundation, 1999)

“The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computational nodes) in the second layer (i.e., the first hidden layer). The output signals of the second layer are used as inputs to the third layer, and so on for the rest of the network. Typically the neurons in each layer of the network have as their inputs the output signals of the preceding network only. The set of output signals of the neurons in the output (final) layer of the network constitutes the overall response of the network to the activation pattern supplied by the source nodes in the input (first) layer.” (Haykin, Neural Networks: A Comprehensive Foundation, 1999)

The architecture implemented in this research is a multilayer, partially connected feed-forward ANN.

In his book, *“Neural Networks: A Comprehensive Foundation”*, Simon Haykin describes fully and partially connected neural networks as:

“A Neural Network is said to be fully connected in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer. If, however, some communication links (synaptic connections) are missing from the network, we say that the network is partially connected” (Haykin, Neural Networks: A Comprehensive Foundation, 1999).

It is a multilayer ANN in the sense that it consists of four layers namely the input layer, two hidden or computational layers and an output layer. It is partially connected in the sense that although all neurons in the input layer connect to all neurons in the 1st hidden layer, all the neurons in the 1st hidden layer do not connect to all the neurons in the 2nd hidden layer. Please refer to *Figure 18 (page 45)*.

The various layers can be described as follows:

- Input Layer.

The activation function is linear.

Input vectors are based on the measured dimensions from the X-, Y-axis and time laser sensors.

$Z_{01} - Z_{10}$ are linear inputs from the X-axis laser sensor.

$Z_{11} - Z_{20}$ are linear inputs from the Y-axis laser sensor.

Z_{21} is an input from the time sensor.

- Middle / Hidden Layers.

The activation function is linear.

Neurons Y_1, Y_3, Y_5, Y_7, Y_9 and Y_{11} calculate a weighted sum for the X-axis inputs for their individual part specific optimized weights.

Neurons $Y_2, Y_4, Y_6, Y_8, Y_{10}$ and Y_{12} calculate a weighted sum for the Y-axis inputs for their individual part specific optimized weights.

The time input layer neuron, Z_{21} , is connected to all the 1st hidden layer neurons.

X_1, X_2, X_3, X_4, X_5 and X_6 calculate a weighted sum of their relevant X- and Y-axis neurons.

The output from this layer, the 2nd hidden layer, is linear.

This linear output is used to calculate the fitness of an individual during the training process.

One of the requirements for GA based ANN training is that the calculated fitness value contains more information than a Boolean True or False answer. This will be discussed in Section 4.5.3 *Genetic Algorithm Training*.

- Output Layer.

The sole purpose of this layer is to provide a Boolean output depending on whether the output from the previous layer falls within a certain target range and thus the associated weight ($U_{1,1} - U_{6,6}$) is a constant value of 1. The activation function is a stepped ramp hybrid function as described in Section 4.3.5 *Ramp Function combined with Step Function*. A Boolean output from O_1, O_2, O_3, O_4, O_5 and O_6 indicates whether part A, part B, part C, part D, part E or part F has been identified.

Because of the partial interconnectivity of the ANN, it is possible to subdivide the network into six logical and functional separate smaller ANNs, one for each part to be identified as can be seen in *Figure 19 (page 46)*. This is an important feature as it enables the separate training of weights for each part's ANN.

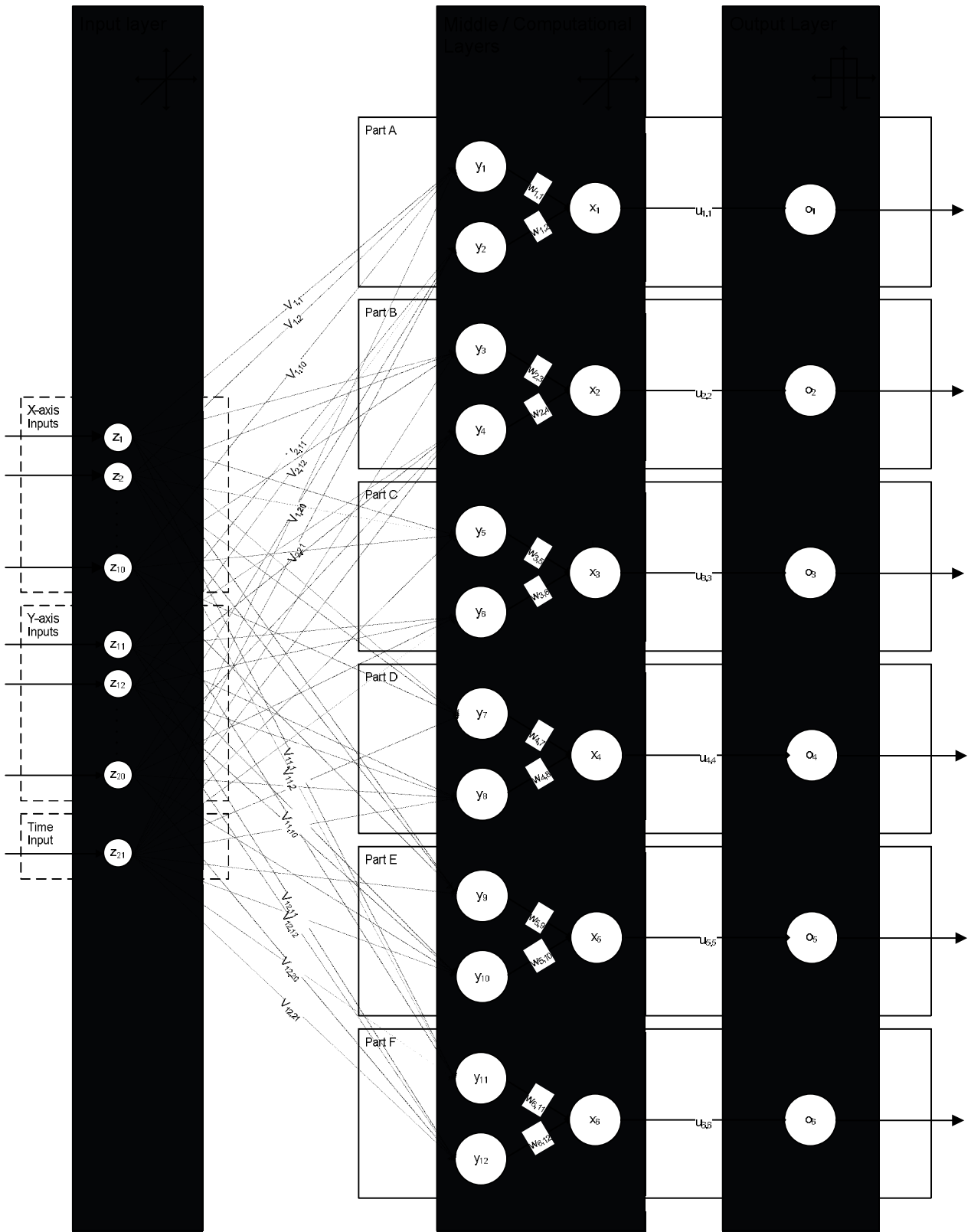


Figure 18 - Complete Neural Network Architecture

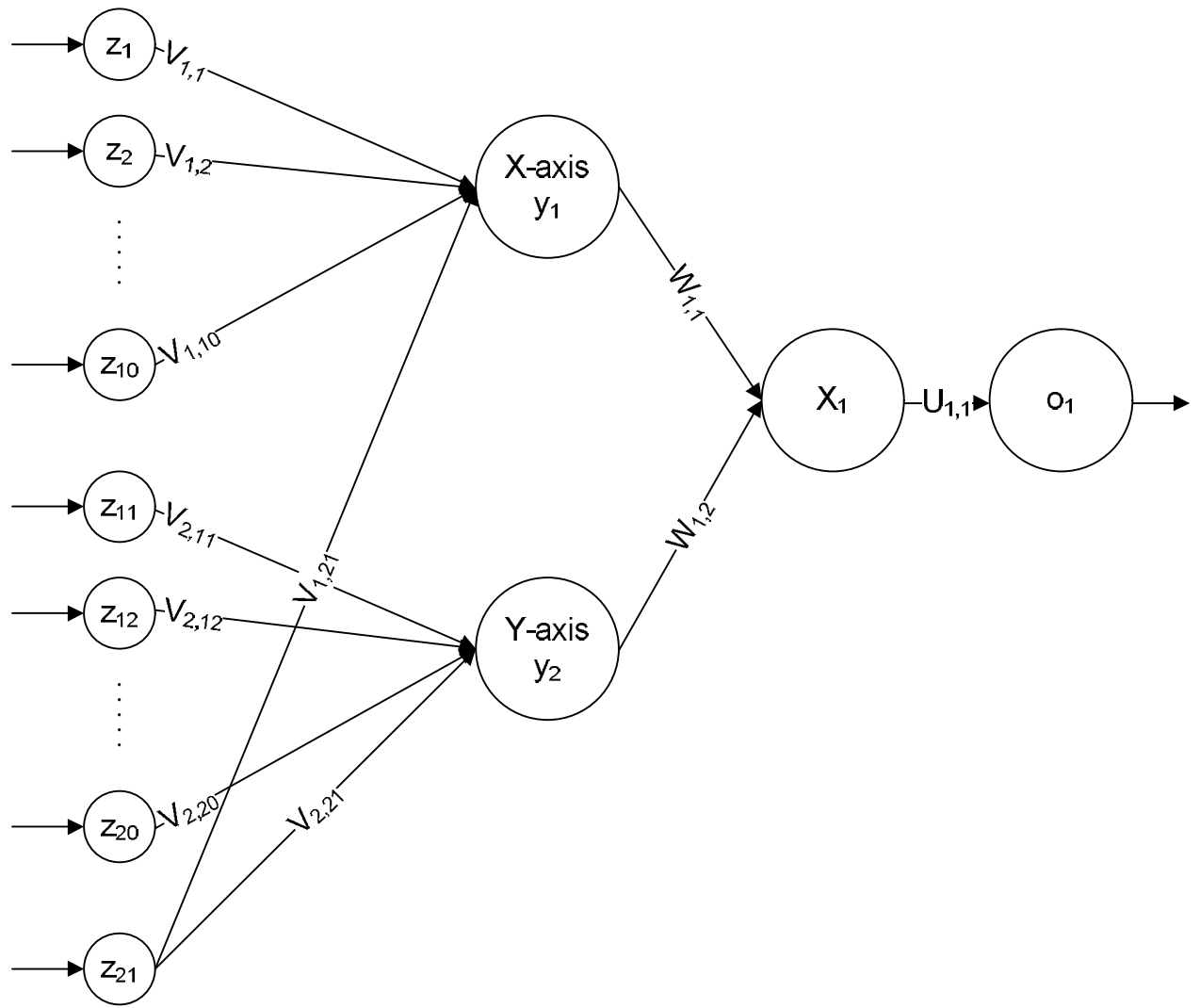


Figure 19 - Neural Network Architecture for each part

4.5. Artificial Neural Network Training

The intelligence of the ANN is embedded in the various weight values assigned to each connection. These weights are modified by learning rules. The three commonly used types of learning rules are supervised, reinforcement, and unsupervised.

4.5.1. Artificial Neural Network Training Methodologies

4.5.1.1. Supervised Learning

The term supervised is used both in a very general and narrow technical sense. In the narrow technical sense supervised means that for a certain input the corresponding output is known, the ANN is to learn the mapping from inputs to outputs. In supervised learning applications, the correct output must be known and provided to the learning algorithm. The task of the network learning algorithm is to find the mapping. The weights are changed depending on the magnitude of the error that the network produces at the output layer: the larger the error, i.e. the discrepancy between the output that the network produces (the actual output) and the correct output value (the desired output), the more the weights change. This is why the term *error-correction learning* is also used. The most used example of supervised learning is BP. BP is very powerful and there are many variations of it. Such learning algorithms are used in the context of feed-forward networks and requires a multi-layer network. BP will be investigated in this research. Although not typically used, the GA training used in this research is also a form of supervised learning.

4.5.1.2. Reinforcement Learning

In reinforcement learning, the only known detail of the output is a discreet answer of whether the output is correct or not. No detail of the amount of error is available.

The problem of attributing the error to the right cause is called the credit assignment or blame assignment problem. It is fundamental to many learning theories. It is used to designate learning where a particular behaviour is to be reinforced. An example would be robot training. The robot receives a positive reinforcement signal if the result was good, no reinforcement or a negative reinforcement signal if it was bad. If the robot has managed to pick up an object, has found its way through a maze, or if it has managed to shoot the ball into the goal, it will get a positive reinforcement. Reinforcement learning is not tied to ANNs: there are many reinforcement learning algorithms in the field of machine learning in general.

4.5.1.3. Unsupervised Learning

Hebbian learning and competitive learning typically fall under this type of learning. Hebbian learning is when two nodes are active simultaneously the connection between them is strengthened.

4.5.2. Back-propagation Training

Back-propagation is the most common type of feed-forward ANN training used. The method processes records one at a time and train or learn by comparing their prediction of the data set output with the known actual target. The errors from the initial prediction of the first record is fed back into the network and used to modify the networks algorithm, hence the name back propagation. The term feed-forward describes how this neural network processes the input data set vector. The BP learning algorithm calculates the error between the generated output and target output and uses the estimated error to modify the weight in response to the errors. This method sends the error backwards through the network adjusting the weights accordingly. The error function is based on the Mean Square Error (MSE) value used to evaluate the goodness of fit in statistical problems. For problems where the network prediction closely matches the target solution, the error function will be very small. In contrast, poor network representations of problems will produce high error values.

The algorithm changes its weight to follow the steepest path towards the bottom of the valley shaped error surface (an example of the solution / error space or surface can be seen in *Figure 21 (page 53)*). This continues until a set of weights, which processes data accurately for the application, is found or optimized. These weights are normally adjusted based on a gradient-descent technique.

In his book, "*An Introduction to the Modeling of Neural Networks*", P. Paretto identifies two of the most troublesome defects of gradient descent BP training namely the *Overshooting Effect* and the *Gully Effect*. Both of these problems are related to the momentum, or rate of descent, of gradient descent training methods. If the rate of change is too small, the training algorithm might get stuck in a local minimum. If the rate of change is too large, the training algorithm might jump over small local minimum valleys (Paretto, 1994).

Although there is no general rule as to how many training data sets is needed to generate the ideal weight for the layers, it is known that the highest accuracy and efficiency obtained is by the network that requires the fewest weights needed to process the training data to a specified accuracy. To determine the optimal architecture of the ANN there has to be a trade off between the networks being large enough to learn the relationships but small enough so it can still be generalised and over fitting is avoided. What is known is that data sets chosen for the training must cover the entire range of expected inputs to the ANN. Some statistical analysis and manipulation of training data, such as the removal of outliers generally result in higher accuracy and smoother generalisation (please refer to Section 3.3.3 *Conveyor automation and data acquisition* for more information on how outliers were handled in this research).

After adjusting the connecting weights new inputs are presented to the network and are interpolated. Each successive round of inputs is called an epoch. The network compares the new input data set and must recognise whether it is similar to previous patterns. The difference between the output of the final layer and the desired output is BP to the previous layer and the connection weights are adjusted accordingly.

4.5.3. Genetic Algorithm Training

As previously discussed, the most popular training method for multilayer ANNs is the BP algorithm. Weights connected to the output layer are adjusted directly from the output error. The errors are back-propagated and the weights back-adjusted layer by layer from the output layer to the input layer. This process is repeated for every epoch, or training iteration, until all the available training data have been used. To accurately train an ANN typically requires many epochs in order to obtain the optimal connection-strength or weights. The guidance of the connection parameter optimisation process is based on the gradient of the error.

In their paper, *Artificial evolution of neural networks and its application to feedback control*, Yun Li and Alexander Häußler identified the following drawbacks of BP training of ANNs:

1. *“Gradient guidance passively adjusts parameters from the performance index, which must be differentiable or “well-behaved” and thus may not allow modified error terms that suit real engineering problems;*
2. *It is difficult to train a direct feedback neurocontroller that meets constraint conditions in practical applications;*
3. *The trained parameters may be local optima, although parallel training may overcome this problem to a certain extent;*
4. *Different control parameters in ANN training (e.g., the learning rate and momentum rate) may result in different minimum rms errors for the same (large) number of epochs;*
5. *The architecture of the network usually needs to be fixed prior to parameter training and thus optimal architectures or topologies cannot be revealed for different types of data or applications;*
6. *Using BP is difficult to obtain an ANN that matches multiple data sets that are extracted from different operating conditions of the same plant;*
7. *It is usually difficult to incorporate knowledge and expertise that the designer may already have on the network design; and*
8. *It is almost impossible to minimise the input to an ANN.” (Li & Häußler, 1996)*

GAs are very effective in searching unknown, irregular and complex spaces for optimisation. GAs have the ability to simultaneously evaluate performance at multiple points in parallel while intelligently searching through the solution space and can thus approach the global optima for almost any type of problem presented to it without the need for differentiation. Recent development in this subset of EC has shown the strength of GAs in overcoming the drawbacks listed above. As previously discussed in Chapter 2, training by a GA can be more efficient, in terms of the convergence speed, than BP and that GA trained networks can effectively lead to a lower error as it searches for a global minimum.

In order to perform the training of an ANN the amount of weights required are encoded as a string of real values. This string, or array, is then a representation of an individual and its genetic information in

the GA. Many such individuals are initially randomly generated to form a population of candidate solutions.

As can be seen in the ANN architecture depicted in *Figure 18 (page 45)*, 24 weights are required for each part ANN. These 24 weights are comprised of:

- 10 weights for the 10 X-axis dimensional inputs.
- 1 weight for the X-axis time input.
- 10 weights for the 10 Y-axis dimensional inputs.
- 1 weight for the Y-axis time input.
- 1 weight for the X-axis neuron to output neuron connection.
- 1 weight for the Y-axis neuron to output neuron connection.

As the final output stage neuron, O_1 , purely acts as a stepped range activation function, a constant weight value of 1 is used and need not be optimized by the GA.

The learning process is described below (refer to *Figure 20 (page 51)*):

- a) Training data sets are captured and loaded into the GA training/optimization application.
- b) An initial population is created and genes initialized to random values within a specified range.
- c) Training set input data is fed into the ANN along with the individual currently being evaluated gene values as weights.
- d) The linear output of the ANN (neuron X_1 in *Figure 18 (page 45)*) is compared to the target value. This is used to evaluate the fitness of the current individual.
- e) Step c) and d) is repeated for each individual in the population.
- f) If a defined GA stopping condition (as discussed in Section 5.1.7 *Algorithm Stopping Conditions*) is met, the training process terminates. If not, the next generation is created by selecting parent individuals and creating offspring. The process is then repeated until a stopping condition is met.

The size of the population, the probability rates for cross-over and mutation, type of cross-over, mutation rate and parent selection algorithms are all control parameters of the GA.

“The task of selecting crossover and mutation rates in a GA is much easier than determining the learning and momentum rates in a BP based algorithm. In the optimisation process of ANN parameters, the GA search is guided by analysis and exploration of the fitness of every individual in the evolving population. The relative fitness of an individual in the population is the criterion that determines the probability of its reproduction and survival.” (Li & Häußler, 1996)

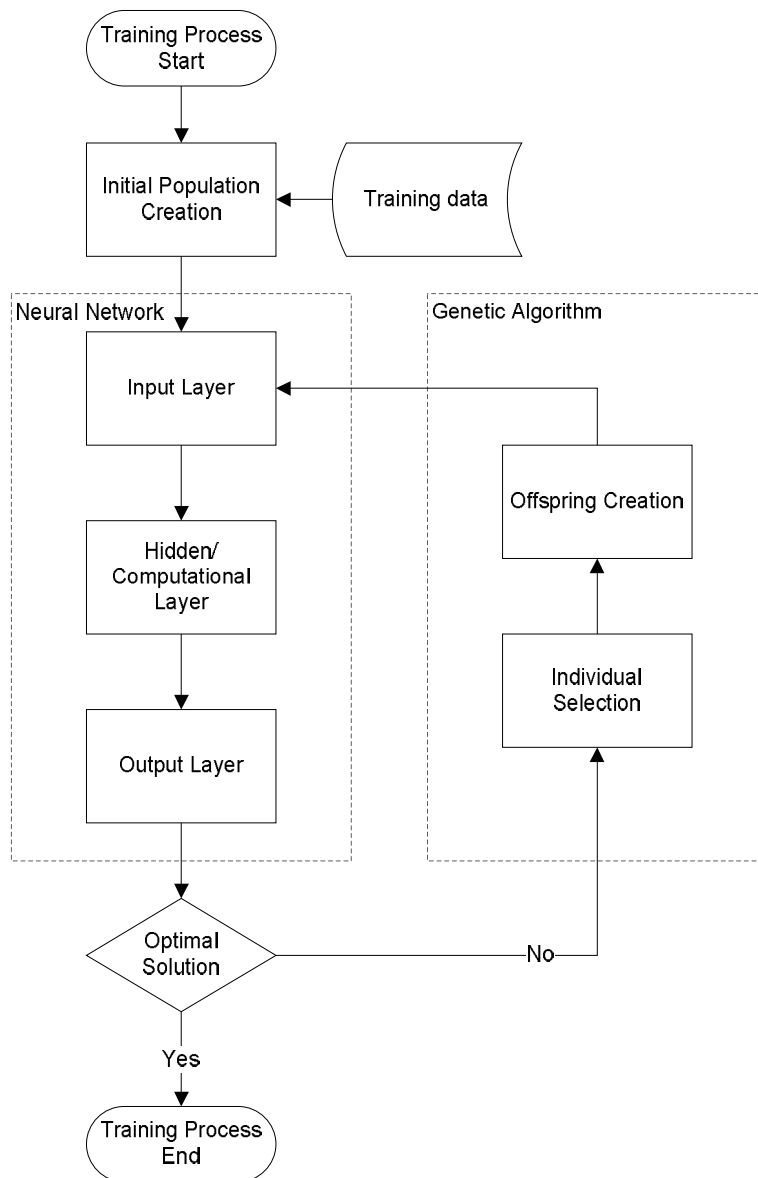


Figure 20 – GA based ANN training Flow Chart

“Since the GA is a fitness evaluation based search, as opposed to calculus based optimisation, method, the analytical requirement of the fitness function is thus much more relaxed than an objective or cost function used in traditional optimisation techniques.” (Li & Häußler, 1996)

If the below criteria are met, the ANN can be trained by a GA:

- The solution space is known and can be encoded as gene values.
- The fitness of each individual is analysable and can be evaluated.
- The calculated fitness value contains more information than a Boolean True or False answer, as otherwise the reproduction will be completely polarised and the GA will become a multi-point random search.

4.6. Conclusion

In the chapter the general operational principle of ANNs, their advantages, applications, various architectures and training methods were discussed.

The mathematical and operational foundations of the activation functions as well as the hybrid utilized in the project are explained in detail.

Various types of commonly used ANNs, their mathematical and operational foundations are introduced. The type of ANN and the associated selection criteria as used in this research is explained.

The ANN architecture, its various layers and their activation functions, the associated inputs and outputs as well as the various AN connection weights as used in this project is introduced and discussed in detail.

The most commonly used method of ANN training, BP, is discussed. The methodology of GA based ANN training as used in this project is explained.

In the next chapter a similar topics will be discussed with regard to GAs. The history, methodology and real-world uses are discussed. The application specific GA software, its operation, use and development is also discussed.

5. Chapter 5 – Genetic Algorithms Applied to Neural Network Training

In this chapter background on the operating principle and history of GAs as well as the ANN training application used in this research is discussed. A detailed description and operation guide of the developed GA training application is also given.

5.1. Background on the way the GA works

The GA in this application can be thought of as the search through the solution space of possible chromosome values for the set of chromosome values that will result in the lowest possible ANN output error. In that sense of an Evolutionary Algorithm (EA) is a stochastic search for an optimal solution to a given problem.

Figure 21 (page 53) and Figure 22 (page 54) describes the solution space for Equation 9 (page 53). In Figure 21 (page 53) it can be seen that each of the six candidates are represented by its coordinates. In optimization problems one looks for the lowest valley or highest hill depending on the problem being optimized.

Equation 9 - Solution Space Description (Venables & Tan, 2007)

$$f(x, y) = (1 - x)^2 e^{-x^2 - (y+1)^2} - (x - x^3 - y^3) e^{-x^2 - y^2}$$

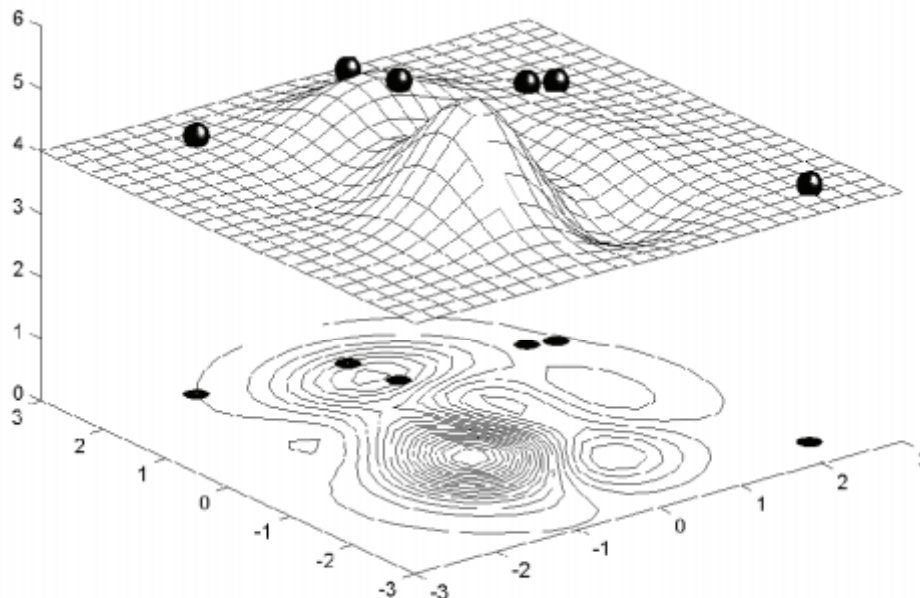


Figure 21 - Landscape of solution space (Venables & Tan, 2007)

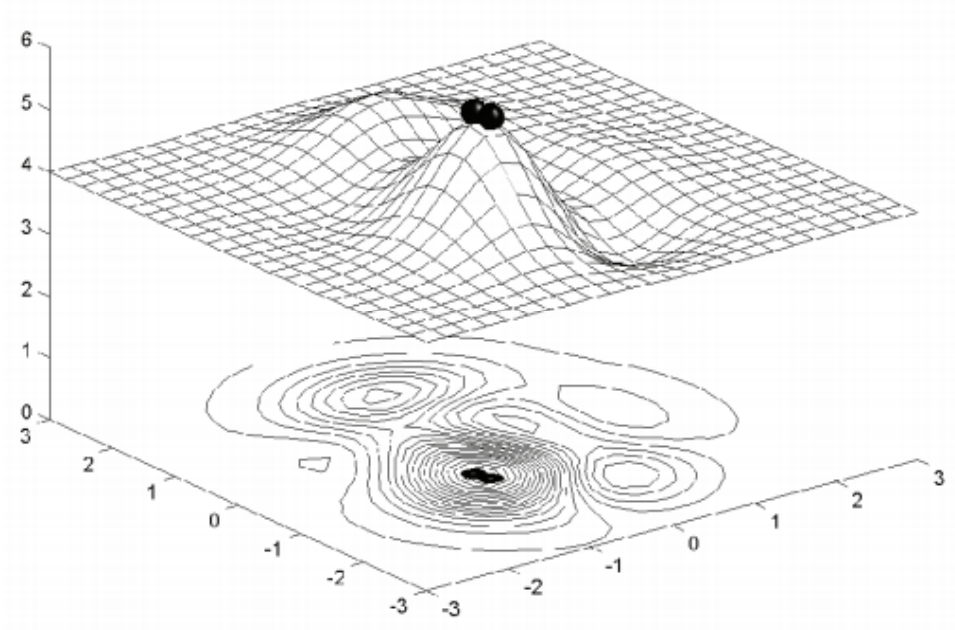


Figure 22 - Landscape of solution space with optimized solutions (Venables & Tan, 2007)

In *Figure 22 (page 54)* it can be seen that the six individuals have, over successive generations, converged on the optimal solution.

The evolutionary search process is influenced by the following main components:

- Problem solution encoding.
- Population creation and initialization.
- Fitness evaluation / fitness function definition.
- Selection operators.
- Offspring creation / reproduction operators.
- Population sorting algorithms.
- Algorithm stopping conditions.

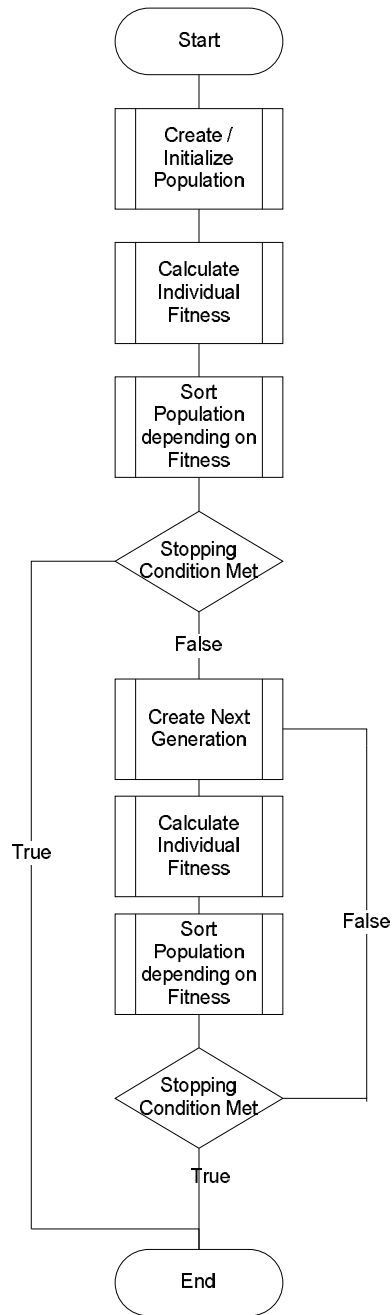


Figure 23 - Basic Genetic Algorithm Flowchart

The various components and the specific application of each in this research are described in the following subsections:

5.1.1. Problem Solution Encoding

“In nature, organisms have certain characteristics that influence their ability to survive and reproduce. These characteristics are represented by long strings of information contained in the chromosomes of the organism. Chromosomes are structures of compact intertwined modules of DNA, found in the nucleus of organic cells. Each chromosome contains a large number of genes, where a gene is the unit of heredity. Genes determine many aspects of anatomy and physiology through control of protein production. Each individual has a unique set of genes. An alternative form of a gene is referred to as an allele.” (Andries, 2007)

In this research the terms chromosome and individual are used interchangeably due to the fact that each individual is represented by a single chromosome string.

“In the context of EC, each individual represents a candidate solution to an optimization problem. The characteristics of an individual are represented by a chromosome, also referred to as a genome. These characteristics refer to the variables of the optimization problem, for which an optimal assignment is sought. Each variable that needs to be optimized is referred to as a gene, the smallest unit of information. An assignment of a value from the allowed domain of the corresponding variable is referred to as an allele.” (Andries, 2007)

“An important step in the design of an EA is to find an appropriate representation of candidate solutions (i.e. chromosomes). The efficiency and complexity of the search algorithm greatly depends on the representation scheme. Different EAs from the different paradigms use different representation schemes. Most EAs represent solutions as vectors of a specific data type.” (Andries, 2007)

In this research each possible solution, or individual, is represented as vectors of float, or double, data types. As the total number of weights that are required for the part recognition ANN is 24 for each part (as discussed in Section 4.5.3 *Genetic Algorithm Training*), each individual is composed of 24 floating point values.

5.1.2. Population Creation and Initialization

GAs are evolutionary population based algorithms. The first step in the algorithm is to create and initialize a population of individuals with random genetic material. There are various factors that need to be considered during this process of which the two most important are initial population size and initial genetic material range.

5.1.2.1. Population Size

The population size influences the computational complexity and possible search scope.

A larger population will result in higher computational complexity for each generation as well as increase time required to evaluate, sort and evolve each generation. The advantage of a larger population is that it will increase the search scope in the possible solution space of the problem being optimized.

5.1.2.2. Initialization values

The initial genetic values assigned to the various individuals in the population directly affect the range of possible solution space that will be searched by the evolutionary process.

The initial values assigned should be uniformly spread throughout the entire range of allowed values in order to ensure that the entire solution space is searched. If a certain range of values are left out of the initialization there is a good chance that that area of the solution space will not be searched unless values are evolved to that region by mutation.

“Evolutionary algorithms are stochastic, population-based algorithms. Each EA therefore maintains a population of candidate solutions. The first step in applying an EA to solve an optimization problem is to generate an initial population. The standard way of generating an initial population is to assign a random value from the allowed domain to each of the genes of each chromosome. The goal of random selection is to ensure that the initial population is a uniform representation of the entire search space. If regions of the search space are not covered by the initial population, chances are that those parts will be neglected by the search process.

The size of the initial population has consequences in terms of computational complexity and exploration abilities. Large numbers of individuals increase diversity, thereby improving the exploration abilities of the population. However, the more the individuals, the higher the computational complexity per generation. While execution time per generation increases, it may be the case that fewer generations are needed to locate an acceptable solution. A small population, on the other hand will represent a small part of the search space. While the time complexity per generation is low, the EA may need more generations to converge than for a large population.

In the case of a small population, the EA can be forced to explore more of the search space by increasing the rate of mutation” (Andries, 2007)

While increasing the mutation rate could force the EA to search a larger area of the solution space, it could also result in the optimal solution being missed as the smaller areas of the solution space are not being explored.

```

ReDim Population(PopulationSize, PopulationWidth)
For Individual As Integer = 0 To PopulationSize
    For Gene As Integer = 1 To ComponentCount
        Dim number = rand.NextDouble()
        Population(Individual, Gene) = number * PopInitWeight
    Next
Next

```

Source Code 1 - Population Initialization

The Section of source code in *Source Code 1 (page 58)* is used to initialize the population by making use of the *Random Visual Basic.NET Class*. All gene values are initialized to floating real values in the range [0,1].

The variable *PopIniWeight* is a user defined variable to allow scaling of the random initialization value.

In his book, *Computational Intelligence: An Introduction*, Andries P. Engelbrecht discusses weight initialization methods when using gradient based optimization methods of training of ANNs. Although gradient based training was not used in this research the theory is still applicable.

“A sensible weight initialization strategy is to choose small random weights centered around 0. This will cause net input signals to be close to zero. Activation functions then output midrange values regardless of the values of input units. Hence there is no bias toward any solution. Wessels and Barnard [898] showed that random weights in the range $[\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$ is a good choice, where fanin is the number of connections leading to a unit.” (Andries, 2007)

As the amount of connections leading to each unit in this application is 24, the range can be calculated as follows.

$$\begin{aligned}
 \text{Range} &= \left[\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}} \right] \\
 &= \left[\frac{-1}{\sqrt{24}}, \frac{1}{\sqrt{24}} \right] \\
 &= [-0.204124, 0.204124]
 \end{aligned}$$

```

ReDim Population(PopulationSize, PopulationWidth)

fanin = 1 / (Math.Sqrt(ConnectionAmount))

DevidingFactor = 0.5 / fanin

For Individual As Integer = 0 To PopulationSize
    For Gene As Integer = 1 To ComponentCount
        Dim number = rand.NextDouble()

        number = number - 0.5

        number = number / DevidingFactor

        Population(Individual, Gene) = number
    Next
Next

```

Source Code 2 - Population Initialization to f_{anin}

The Section of code in *Source Code 2 (page 59)* is used to initialize the population by making use of the *Random Visual Basic.NET Class*. The random values are then scaled to fall within the calculated range of [-0.204124, 0.204124].

5.1.3. Fitness Evaluation / Fitness Function Definition

The fitness of an individual is a value that indicates how far or close an individual is from the optimal solution. The fitness function is the problem specific function which calculates and assigns a fitness value to the individual.

The most difficult and important concept of genetic programming is the fitness function as it is the sole means of judging the quality of an evolved solution. Hui et al. describe the fitness function as follows: *"In the context of neural network training, the fitness function may be the total error function"* (Hui, Lam, & Chea, 1997)

The fitness evaluation is also important in the parent selection stage as fitter, closer to optimal; individuals stand a good chance of either being selected as parents or to survive through to the next generation. This enables them to pass along their good genetic information to the next generation.

"In the Darwinian model of evolution, individuals with the best characteristics have the best chance to survive and to reproduce. In order to determine the ability of an individual of an EA to survive, a mathematical function is used to quantify how good the solution represented by a chromosome is.

Usually, the fitness function provides an absolute measure of fitness. That is, the solution represented by a chromosome is directly evaluated using the objective function. As a final comment on the fitness function, it is important to emphasize its role in an EA.

The evolutionary operators, e.g. selection, crossover, mutation and elitism, usually make use of the fitness evaluation of chromosomes. For example, selection operators are inclined towards the most-fit individuals when selecting parents for crossover, while mutation leans towards the least-fit individuals.” (Andries, 2007)

Utilizing GAs for design work has been criticized as a lazy method of design. The counter argument is the amount of effort and planning involved in designing a workable and accurate fitness function. Even though it is no longer the human designer, but the computerized GA, that comes up with the final design, it is the human designer who has to design the fitness function. If this is designed wrongly, the algorithm will either converge on an inappropriate solution, or will have difficulty converging at all.

Another important design aspect of the fitness function is the efficiency and speed of execution, as a typical GA must be iterated many times in order to produce a usable result. This is especially the case for the discussed part recognition application as new parts need to be trained quickly and accurately.

Please refer to Section 4.5.3 *Genetic Algorithm Training* for more general information on the training of ANNs by utilizing GAs.

The fitness function implemented in this application can be explained as follows:

- ANN target output for the part being trained is specified by the user.
- Input training sets for the part in training are captured.
- Each of the provided training sets are individually input into a simulated ANN identical to the ANN programmed into the PLC.
- The output, before applying the step function, of the simulated ANN for each training set is recorded and the RSE to the ANN target is calculated.
- The sum total of all the training sets RSEs are assigned as the fitness of the individual.
- The lower the fitness value, the better the solution as it indicates a lower error value.

RSE is used as a negative error value would falsely improve the fitness of an individual. *Table 8 (page 61)* and *Table 9 (page 61)* illustrate this possibility. As an example, when the fitness of an individual is evaluated using RSE, its true fitness and error value of 1472.52 is calculated. When using the basic error calculation results in an incorrect *better* fitness of 210.54.

$$\text{Error} = \text{ANN Output} - \text{ANN Target}$$

Equation 10 - Basic Error Calculation

$$RSE = \sqrt{(ANN\ Output - ANN\ Target)^2}$$

Equation 11 - RSE Calculation

Table 8 - RSE Fitness Calculation

Training Data Set	ANN Output	ANN Target	RSE
Data Set 1	19926.49	20 000.00	73.51
Data Set 2	20259.46	20 000.00	259.46
Data Set 3	19725.71	20 000.00	274.29
Data Set 4	20371.53	20 000.00	371.53
Data Set 5	19506.26	20 000.00	493.74

1 472.52

Table 9 - Error Fitness Calculation

Training Data Set	ANN Output	ANN Target	Error
Data Set 1	19926.49	20 000.00	73.51
Data Set 2	20259.46	20 000.00	-259.46
Data Set 3	19725.71	20 000.00	274.29
Data Set 4	20371.53	20 000.00	-371.53
Data Set 5	19506.26	20 000.00	493.74

210.54

5.1.4. Selection Operators

In order to ensure the preservation of good genetic material, parent selection should be largely based on the fitness of individuals. This is the main objective of selection operators. At the end of each generation the population for the next generation is selected. The new population can consist of both newly created offspring and parents that survive through to the next generation. Although most of the unfit, genetically bad, individuals are killed off at the end of the generation a few are allowed to survive as they improve the genetic diversity of the population.

Selection operators are characterized by their selective pressure which relates to the time required for the population to converge. It is defined as the speed at which the best solution will occupy the entire population by repeated application of the selection operator alone. An operator with a high selective pressure decreases diversity of genetic material in the population more rapidly than operators with a low selective pressure, which may lead to premature convergence to suboptimal, local minimum, solutions. A high selective pressure limits the exploration abilities of the population.

Generally the more fit individuals, with the lowest error, are selected as parents for cross-over and mutation operators as they most likely to contain good genetic material.

The mutation operator is also applied to less fit individuals as they allow exploration of wider areas of the solution space for a global optimum whereas the elite individuals generally converge in one area of the solution space which might only be a local optimum.

The most commonly used selection operators are discussed in this section.

5.1.4.1. Roulette Based Selection

Parents are selected according to their fitness. This procedure, also known as fitness proportionate selection, can be thought of as a roulette wheel being spun once for each available slot in the next population. Each solution has a portion of the roulette allocated in proportion to their fitness. In this scheme it is possible to choose the best individual more than once, and chances are that the worse individual has a very slim chance of being selected and surviving to the next generation. See *Figure 24 (page 62)* for an example.

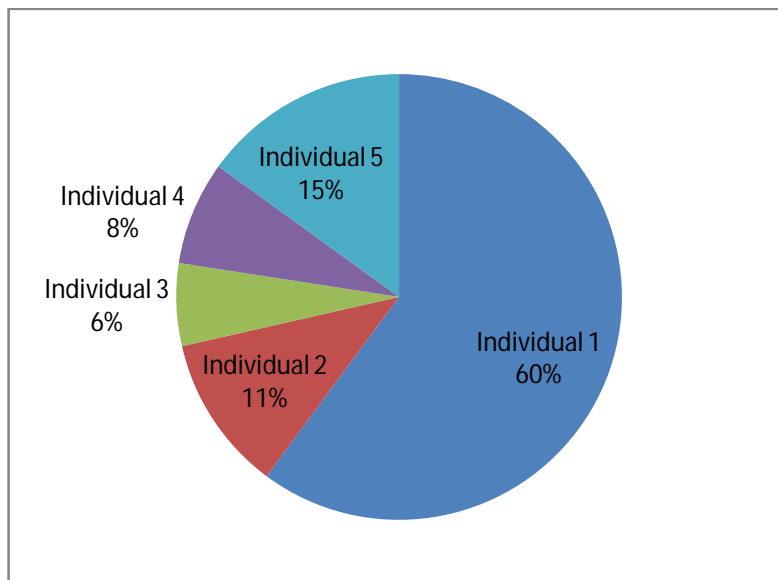


Figure 24 - Roulette Wheel Selection Operator

5.1.4.2. Rank Based Selection

“Rank-based selection uses the rank ordering of fitness values to determine the probability of selection, and not the absolute fitness values. Selection is therefore independent of actual fitness values, with the advantage that the best individual will not dominate in the selection process.”
(Andries, 2007)

One of the problems associated with roulette based selection is that, in an situation where the fitness's of individuals differs very much and one individual is much better than the rest, the good individual will be selected numerous times and cause the population to prematurely converge on

a local minimum. If, for example, the best chromosome fitness is 90% of the roulette wheel then the other chromosomes will have very few chances to be selected.

Rank based selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have a fitness of 1, second worst 2 etc. and the best will have fitness N (where N is the number of individuals in the population).

After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

“Rank selection is an alternative method whose purpose is also to prevent too-quick convergence. In the version proposed by Baker (1985), the individuals in the population are ranked according to fitness, and the expected value of each individual depends on its rank rather than on its absolute fitness. There is no need to scale fitnesses in this case, since absolute differences in fitness are obscured. This discarding of absolute fitness information can have advantages (using absolute fitness can lead to convergence problems) and disadvantages (in some cases it might be important to know that one individual is far fitter than its nearest competitor). Ranking avoids giving the far largest share of offspring to a small group of highly fit individuals, and thus reduces the selection pressure when the fitness variance is high. It also keeps up selection pressure when the fitness variance is low: the ratio of expected values of individuals ranked i and $i+1$ will be the same whether their absolute fitness differences are high or low” (Melanie, 1999)

Figure 25 (page 63) shows how the situation changes after changing to a rank based selection operator.

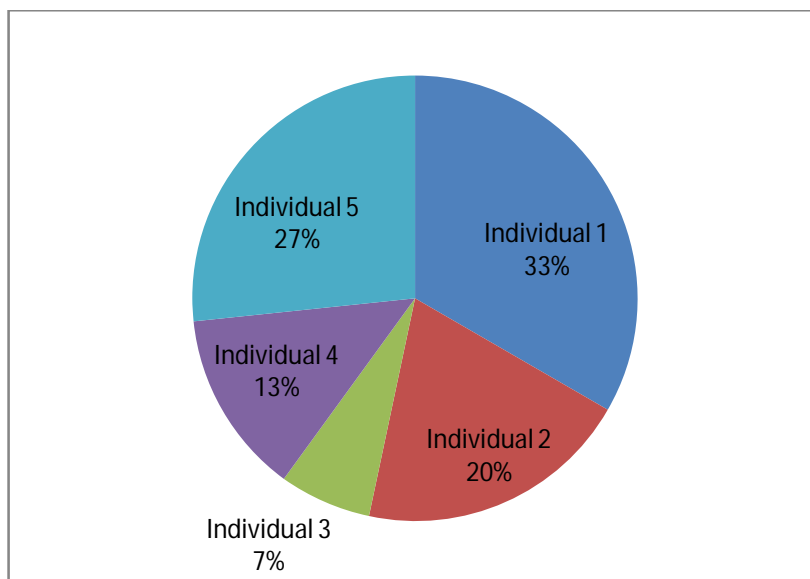


Figure 25- Rank Based Selection Operator

5.1.4.3. Random Selection

Random selection is the simplest selection operator, where each individual has the same probability to be selected. No fitness information is used, which means that the best and the worst individuals have exactly the same probability of being selected as parent thus having their genetic material surviving to the next generation. Random selection has a very low selective pressure. While it is good to have some randomly selected individuals as it increases the range and variety of the population's genetic pool, it also counter acts the base principle of GAs, natural selection.

5.1.4.4. Tournament Selection

Rank scaling requires sorting the entire population by rank which adds computational complexity and also increases execution time dramatically as it needs to be performed for each generation.

Tournament selection is similar to rank selection in terms of selection pressure, but it is computationally more efficient and more amenable to parallel implementation.

Two individuals are chosen at random from the population. A random number is then chosen between 0 and 1. If *the random number is smaller than a preset parameter* value (for example 0.80), the fitter of the two individuals is selected to be a parent; otherwise the less fit individual is selected. The two are then returned to the original population and can be selected again.

Another implementation of tournament selection is described by Andries P. Engelbrecht.

“Tournament selection selects a group of n_{ts} individuals randomly from the population, where $n_{ts} < n_s$ (n_s is the total number of individuals in the population). The performance of the selected n_{ts} individuals is compared and the best individual from this group is selected and returned by the operator. For crossover with two parents, tournament selection is done twice, once for the selection of each parent.

Provided that the tournament size, n_{ts} , is not too large, tournament selection prevents the best individual from dominating, thus having a lower selection pressure. On the other hand, if n_{ts} is too small, the chances that bad individuals are selected increase.

Even though tournament selection uses fitness information to select the best individual of a tournament, random selection of the individuals that make up the tournament reduces selective pressure compared to proportional selection. However, note that the selective pressure is directly related to n_{ts} . If $n_{ts} = n_s$, the best individual will always be selected, resulting in a very high selective pressure. On the other hand, if $n_{ts} = 1$, random selection is obtained.” (Andries, 2007)

5.1.4.5. Elitism

Elitism is the selection operator where the best, fittest individuals of the current generation are allowed to survive through to the next generation. Such individuals can be lost if they are not selected to reproduce or if they are destroyed by crossover or mutation. The rest of the offspring is created by classical reproduction operators. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution. If a large percentage of offspring creation is allocated to elitism it can cause the population to prematurely converge on a local, sub-optimal minimum solution as it drastically decreases the genetic diversity of the population.

5.1.4.6. Hall of Fame

The hall of fame is a selection scheme where the best individual of each generation is selected to be inserted into a separate population or “hall of fame”. The hall of fame will therefore contain an archive of the best individuals found from the first generation. This elite genetic material in the hall of fame can be used as a parent pool for the crossover operator. This method ensures that, as with elitism, good genetic material isn’t lost by mutation or cross-over but has the advantage that it can’t cause the population to prematurely converge on a sub-optimal solution.

In this research a combination of elitism and random selection is used as selection operators with the selective pressure of each defined by the options set in software.

5.1.5. Offspring Creation / Reproduction Operators

Reproduction is the process of producing offspring from selected parents by applying crossover and/or mutation operators.

5.1.5.1. Elitism

The concept of elitism has already been introduced and discussed in Section 5.1.4.5. *Elitism*.

5.1.5.2. Crossover

Crossover is the process of creating one or more new individuals through the combination of genetic material randomly selected from one or more parents.

If parent selection focuses on the most-fit individuals, the selection pressure may cause premature convergence due to the lack of genetic diversity in the new populations.

Crossover operators can be divided into three main categories based on the number of parents used. The three main categories are:

- Asexual.

Offspring is created by rearranging the gene values of one individual.

- Sexual.

Offspring is created by combining the genetic material of two parent individuals. This is the most common form of the crossover operator.

- Multi-recombination.

Offspring is created by combining the genetic material of more than two parent individuals.

The sexual crossover category can be further classified as:

- Single Point Crossover.

Both parent chromosomes are split at a randomly determined crossover point. A new individual, or offspring, is created by appending the second part of the second parent to the first part of the first parent as illustrated in *Figure 26 (page 66)*.

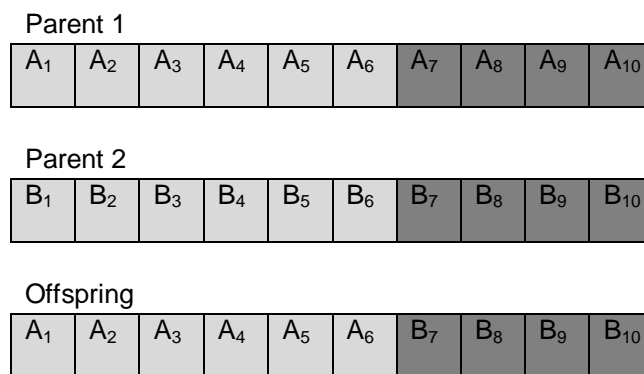


Figure 26 - Single Point Cross-over

- Two-point Crossover

In two-point crossover, both parents are split at two points and a new offspring is created by using parts number one and three from the first, and the middle part from the second parent chromosome. This is illustrated in *Figure 27 (page 66)*.

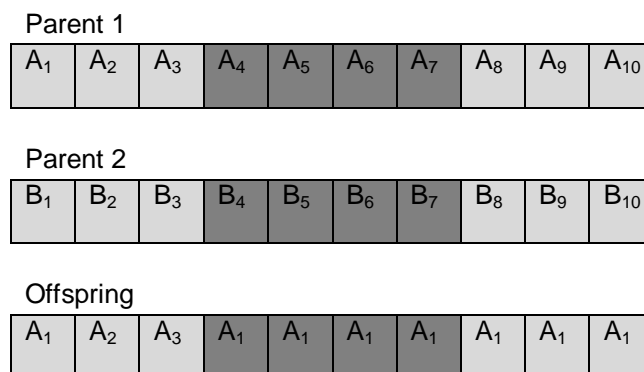


Figure 27 – Two-point Cross-over

- Multi-point Crossover

In Multi-point crossover, random chromosomes are selected from each parent individual and combined to form offspring. This is illustrated in *Figure 28 (page 67)*.

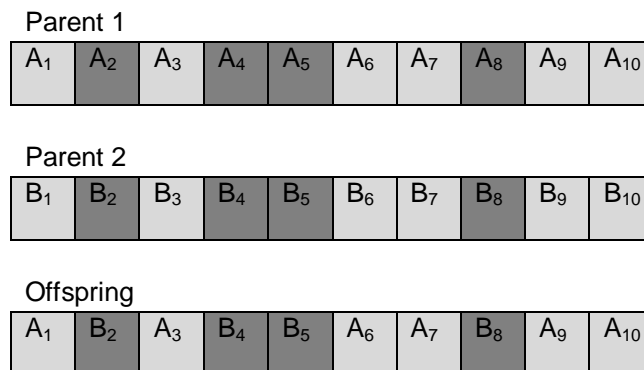


Figure 28 - Multi-point Cross-over

5.1.5.3. Mutation

In individuals composed of real string values, this can be achieved by randomly modifying the value of one or more genes in the parent individual of a gene, as illustrated in *Figure 29 (page 68)*.

The problem with the above discussed crossover methods is that no new information is introduced: each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations.

The main objective of mutation is to introduce new genetic material into the population, thereby increasing genetic diversity. With the new gene values, the GA may be able to arrive at a better solution than was previously possible. Mutation is an important part of genetic search and optimization as it helps prevent the population from stagnating at any local minimum in the solution space.

Applying a large degree of mutation to highly fit individuals could damage good genetic material from the gene pool. For this reason, very small mutation changes are normally made to genetic material. The alternative is to increase the mutation amount for less fit individuals. Another approach is to increase the mutation amount during the first few generations in order to promote search space exploration in the first generations and then reduce it over time to allow for exploitation during the final generations.

*“A common view in the GA community, dating back to Holland's book *Adaptation in Natural and Artificial Systems*, is that crossover is the major instrument of variation and innovation in GAs, with mutation insuring the population against permanent fixation at any particular locus and thus playing more of a background role.”* (Melanie, 1999)

$$B_1 = A_3 + \theta$$

where θ is a randomly generated real number

Equation 12 - Mutated Gene

Parent

A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	A ₁₀
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------

Offspring

A ₁	A ₂	B ₁	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	A ₁₀
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------

Figure 29 - Mutation operator

5.1.6. Population Sorting Algorithm

After each new generation has been created and the individual's fitness' have been calculated, the population needs to be sorted accordingly. The lower the amount of error, the higher the individual's fitness.

The efficiency of this sorting algorithm is very important as it has a major influence on the computational complexity and the time taken by each generation.

The sorting algorithm used in this research can be seen in *Source Code 3 (page 70)*. Each repetition involves reaching through the entire population array from the top down for the individual with the highest fitness. Once this individual is found, it is moved into a new temporary sorted population array and subsequently removed from the original array. The process is then repeated to find the next best individual. Once all the positions in the new temporary array have been filled, the new temporary population array becomes the active population array.

The algorithm used in this research is similar to a commonly used sorting algorithm called Insertion Sort. Although it is less efficient on large lists r arrays of data, such as large populations, it does provide several advantages:

- Simple implementation.
- Efficient for small data sets (as is normally the case with GAs).
- Adaptive for datasets that are already substantially sorted.
- More efficient than most other simple quadratic algorithms such as bubble sort.
- Stable as it does not change the relative order of elements with equal fitness.
- In-place as it only requires a constant amount of memory space.

```

Private Sub SortPopulation()

    ' Logging
    SaveToLogFile("Sorting of Population Started", 2)

    ' Temp Population variable
    Dim SortedPopulation(PopulationSize, PopulationWidth) As Double
    ' Temp to store best individual of that cycle
    Dim BestIndividual(PopulationWidth) As Integer
    ' Best individual number of that cycle
    Dim BestIndividualOfCycle As Integer
    ' Best Fitness of that loop
    Dim BestLoopFitness As Double

    ' Sort Fitness from High to Low
    If radSortHigh.Checked = True Then

        ' Loop through to build sorted population
        For SortedPopCounter As Integer = 0 To PopulationSize

            BestLoopFitness = 0

            For LoopCurPopCounter As Integer = 0 To PopulationSize

                ' If current Individual Fitness is better than previous best
                fitness
                If Population(LoopCurPopCounter, FitnessPosition) >
                BestLoopFitness Then
                    ' Save this as new highest fitness
                    BestLoopFitness = Population(LoopCurPopCounter,
                    FitnessPosition)

                    BestIndividualOfCycle = LoopCurPopCounter

                End If

            Next

            ' Store best individual of that cycle in new sorted population
            For LoopChromo As Integer = 0 To PopulationWidth
                SortedPopulation(SortedPopCounter, LoopChromo) =
                Population(BestIndividualOfCycle, LoopChromo)
            Next

            ' Decrease Best
            Population(BestIndividualOfCycle, FitnessPosition) = 0

        Next

    ' Sort Fitness from Low to High
    ElseIf (radSortLow.Checked = True) Then

        ' Loop through to build sorted population
        For SortedPopCounter As Integer = 0 To PopulationSize

```

```

BestLoopFitness = 99999

For LoopCurPopCounter As Integer = 0 To PopulationSize
    ' If current Individual Fitness is better than previous best
    fitness
    If Population(LoopCurPopCounter, FitnessPosition) <
BestLoopFitness Then
        ' Save this as new highest fitness
        BestLoopFitness = Population(LoopCurPopCounter,
FitnessPosition)

        BestIndividualOfCycle = LoopCurPopCounter

    End If

Next

    ' Store best individual of that cycle in new sorted population
    For LoopChromo As Integer = 0 To PopulationWidth
        SortedPopulation(SortedPopCounter, LoopChromo) =
Population(BestIndividualOfCycle, LoopChromo)
    Next

    ' Increase Best
    Population(BestIndividualOfCycle, FitnessPosition) = 99999

Next

End If

    ' Sorted Population overrides original unsorted population
    Population = SortedPopulation

    ' Logging
    SaveToLogFile("Sorting of Population Completed", 2)

End Sub

```

Source Code 3 - Population Sorting

5.1.7. Algorithm Stopping Conditions

“The number of generations that evolve depends on whether an acceptable solution is reached or a set number of iterations is exceeded. After a while all the chromosomes and associated costs would become the same if it were not for mutations. At this point the algorithm should be stopped.” (Haupt, 2004, p. 47)

As can be seen in *Figure 23 (page 55)* the evolutionary operators are repeatedly applied in the GA until a stopping condition is satisfied. The simplest stopping condition is to set a limit on the number of generations that is allowed to be created. This limit should not be too small, otherwise the EA will not

have sufficient time to search the solution space. This is especially the case if the population size is very small.

In addition to the limit on the amount of generations there is normally a criterion to evaluate if the population has converged, whether this convergence is on a sub-optimal solution or not. Convergence is loosely defined as the event when the population becomes stagnant. In other words, when there is no more genetic change in the majority of the population and most of the individuals have the same genetic makeup.

The last commonly used algorithm stopping condition is to stop the evolutionary process once an adequate solution has been found.

To summarize the various stopping conditions commonly used:

- A solution is found that results in the minimum error specified.
- A specified number of generations have been reached.
- An allocated time limit has been reached.
- The highest ranking / best fitness individual has reached a plateau and has been stuck in one area of the solution space for a set number of generations without change. In this research this is also referred to as *Idle Fitness*.
- Manual inspection.
- Combinations of the above.

5.2. Real-world Genetic Algorithm uses

GAs have been successfully utilized in the areas described in the following subsections:

5.2.1. Automotive Design

GAs have been used to both design composite materials and aerodynamic shapes for race cars and regular means of transportation (including aviation) can return combinations of best materials and best engineering to provide faster, lighter, more fuel efficient and safer vehicles.

5.2.2. Engineering Design

Optimizing a range of materials to optimize the structural and operational design of buildings, factories, machines, etc. is a rapidly expanding application of GAs. These are being created for such uses as optimizing the design of heat exchangers, robot gripping arms, satellite booms, building trusses, flywheels, turbines, and various other computer-assisted engineering design applications. There is work to combine GAs optimizing particular aspects of engineering problems to work together, and some of these can not only solve design problems, but also project them forward to analyze weaknesses and possible point failures in the future so these can be avoided.

5.2.3. Evolvable Hardware

Evolvable hardware applications are electronic circuits created by GA computer models that use stochastic (statistically random) operators to evolve Printed Circuit Board (PCB) routing and component layout. GAs could evolve combinations of component placing and track routes than minimize PCB size and weight.

5.2.4. Data Mining

S. Das and B. Saha researched the application of a GA to data mining. Data quality is important to organizations as it is used as a tool for assessing data quality. The goal of Data Quality Mining (DQM) is to employ data mining methods in order to detect, quantify, explain and correct data quality deficiencies in very large databases. They tried to develop Multi-objective GA based approach utilizing linkage between feature selection and association rule. Their main motivation for using GA in the discovery of high-level prediction rules is that they perform a global search and cope better with attribute interaction than the greedy rule induction algorithms often used in data mining. They found that the use of a multi-objective evolutionary framework for association rule mining offers a tremendous flexibility to exploit in further work (Das & Saha, 2009).

5.2.5. Optimized Telecommunications Routing

GAs are being developed that will allow for dynamic and anticipatory routing of circuits for telecommunications networks. These could take notice of a system's instability and anticipate re-routing needs. Other GAs are being developed to optimize placement and routing of cell towers for best coverage and ease of switching.

5.2.6. Trip, Traffic and Shipping Routing

GA applications originally designed to solve a well known game theory problem known as the *Travelling Salesman Problem* can be used to plan the most efficient routes and scheduling for travel planners, traffic routers and even shipping companies. The shortest routes for travelling, the timing to avoid traffic tie-ups and rush hours, most efficient use of transport for shipping, even to including pickup loads and deliveries along the way can be optimized by GAs.

Khosravi and Co. researched the use of a genetic algorithm-based method that automates the neural network model selection of a Neural Network used for travel time prediction. Model selection and parameter adjustment is carried out through minimization of a prediction interval-based cost function, which depends on the width and coverage probability of constructed prediction intervals. The experiments that they conducted using the bus and freeway travel time datasets demonstrated the suitability of their proposed method for improving the quality of constructed prediction intervals in terms of their length and coverage probability (Khosravi, Mazloumi, Nahavandi, Creighton, & Van Lint, 2011).

5.2.7. Computer Gaming

GAs are increasingly being applied to artificial intelligence for computer simulated players in computer games. These GAs have been programmed to incorporate the most successful strategies from previous games, the programs learn, and usually incorporate data derived from game theory in their design. Game theory is useful in most all GA applications for seeking solutions to whatever problems they are applied to, even if the application really is a game.

5.2.8. Encryption and Code Breaking

On the security front, GAs can be used both to create encryption for sensitive data as well as to break those codes. Encrypting, decrypting and data have been important in the computer world ever since there have been computers.

5.2.9. Computer Aided Molecular Design

The design of new chemical molecules is a growing field of applied chemistry in both industry and medicine. GAs are used to aid in the understanding of protein folding, analyzing the effects of substitutions on those protein functions, and to predict the binding affinities of various designed proteins developed by the pharmaceutical industry for treatment of particular diseases. The same sort of GA optimization and analysis is used for designing industrial chemicals for particular uses, and in both cases GAs can also be useful for predicting possible adverse consequences. This application has and will continue to have great impact on the costs associated with development of new chemicals and drugs.

5.2.10. Gene Expression Profiling

The development of microarray technology for taking snapshots of the genes being expressed in a cell or group of cells has been a boon to medical research. GAs are being developed to make analysis of gene expression profiles much quicker and easier. This helps to classify what genes play a part in various diseases, and further can help to identify genetic causes for the development of diseases. The ability to do this quickly and efficiently will allow researchers to focus on individual patients unique genetic and gene expression profiles, enabling the hoped-for *personalized medicine*.

5.2.11. Optimizing Chemical Kinetic Analysis

GAs are proving very useful toward optimizing designs in transportation, aerospace propulsion and electrical generation. By being able to predict the chemical kinetics of fuels and the efficiency of engines, more optimal mixtures and designs can be made available quicker to industry and the public. Computer modelling applications in this area also simulating the effectiveness of lubricants and can pinpoint optimized operational vectors, and may lead to greatly increased efficiency all around well before traditional fuels run out.

5.2.12. Metamodeling of discrete-event simulation models

B. Can and C. Harvey researched the use of Genetic Programming (GP) and ANNs in the development of surrogate models of complex systems. The purpose of their research was to provide a comparative analysis of GP and ANNs for metamodeling of discrete-event simulation models. Three stochastic industrial systems were studied: an automated material handling system in semiconductor manufacturing, an inventory model and a serial production line. The results of the study showed that GP provides greater accuracy in validation tests, demonstrating a better generalization capability than ANN. GP however requires more computation in metamodel development than when compared to ANN (Can & Harvey, 2012).

5.2.13. Robotic Personality Generation

K. Lee investigated a way to generate a robot genome that contributes to defining the personality of a software robot or an artificial life in a mobile phone that is both complex and feature-rich, but still plausible by human standards for an emotional life form. He proposed a neural network algorithm for a genetic robot's personality and an upgraded version of a previously introduced evolutionary algorithm for a genetic robot's personality (EAGRP). Both tools demonstrated good performance in generating the robot genome with a personality that was both complex and feature-rich and still plausible by human standards for an emotional life form. He found that EAGRP showed a comparative advantage especially in terms of variability and intensity (Lee, 2011).

5.3. Intelligent Part Recognition Training Algorithm Application

The main component of the research platform is the GA software and its related functionality. Previously captured training data sets for the various parts are loaded into the application and GA control parameters are specified.

The purpose of this software component is to optimize the problem of ANN weight calculation. Results are graphically output to the user or saved to Excel files.

Microsoft Visual Basic 2008 was selected as the development tool because of previous knowledge and familiarity of the development package as well as meeting the required functionality of complex array handling etc.

The complete software project, including the source code, included components and published installation file is available on the accompanying compact disk.

The various Graphical User Interface (GUI) windows of the application are described below:

- When the application starts, the main window in *Figure 30 (page 75)* opens up. The four main sections of the application are:
 - Genetic Algorithm.
This form contains all options and parameter settings for the GA.

- Neural Network.
This form contains all options, training set input options and evolved weight outputs.
- Settings.
This form contains all application setting options such as log directories and log levels as can be seen in *Figure 31 (page 76)*. Four log levels are provided as detailed logging is extremely processor intensive which negatively influences the performance of the GA.
- Output.
This form contains the application output window. Data that is output is dependent on the specified log level and contains info such as parameter options, generation progression etc. See *Figure 38 (page 80)*.

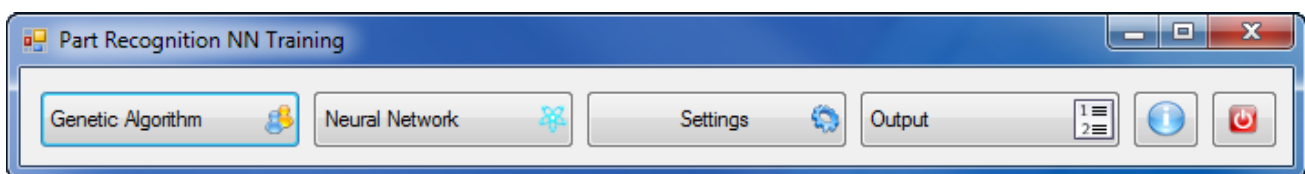


Figure 30 – Part Recognition Main Window

- The *Neural Network* form contains the following tabs:
 - Training Data.
Training Data sets are loaded for each part from excel files or by manual entry as can be seen in *Figure 34 (page 77)*.
 - Weights.
Evolved weights are displayed in the window. Results can be saved to an Excel file. See *Figure 33 (page 77)*.
 - Settings.
Settings such as source and output excel file directories, amount of X- and Y-axis data points in training sets, weights quantity to be calculated and training sets resented are specified here. See *Figure 32 (page 76)*.

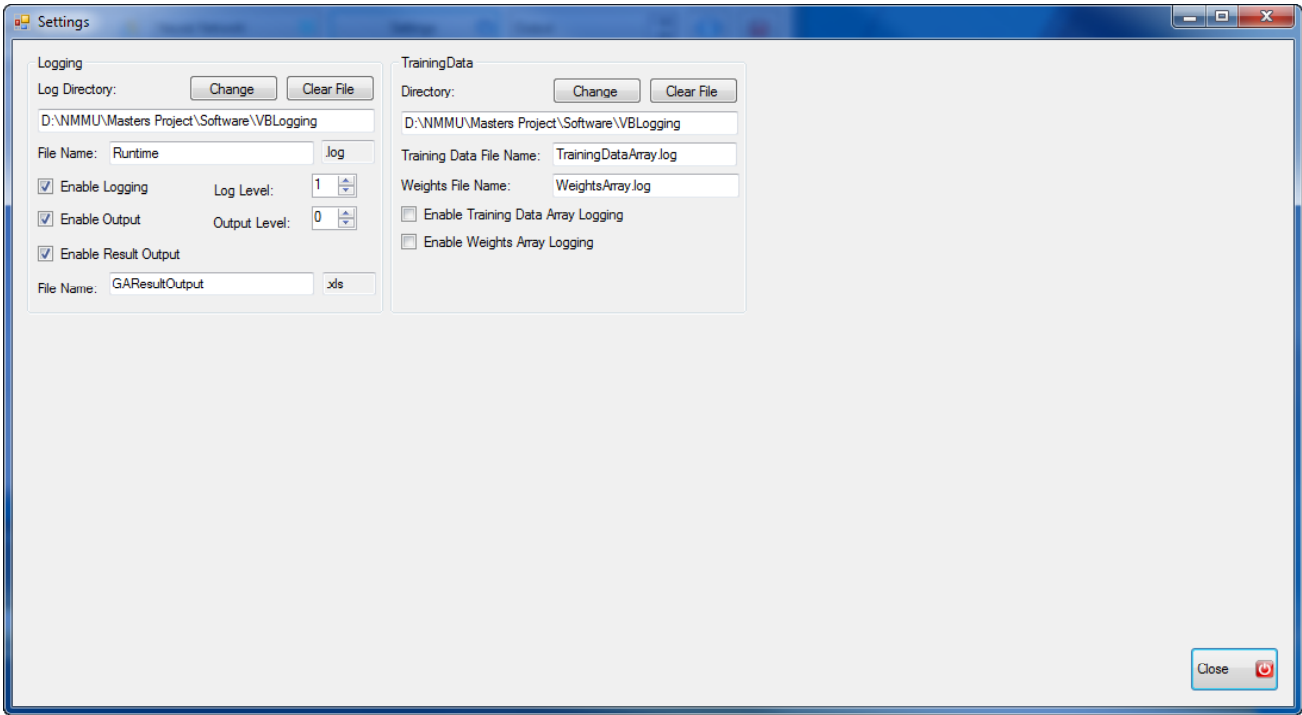


Figure 31 – Part Recognition General Settings Window

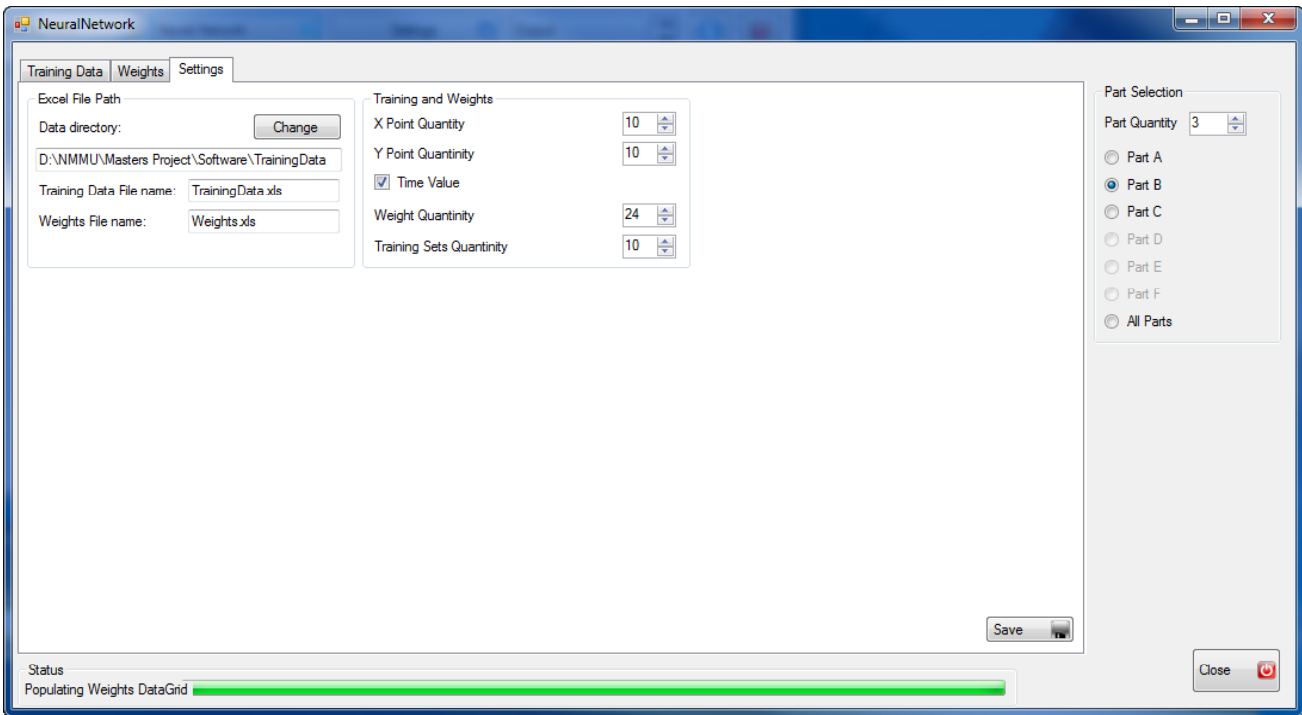


Figure 32 – Part Recognition Neural Network Settings Tab

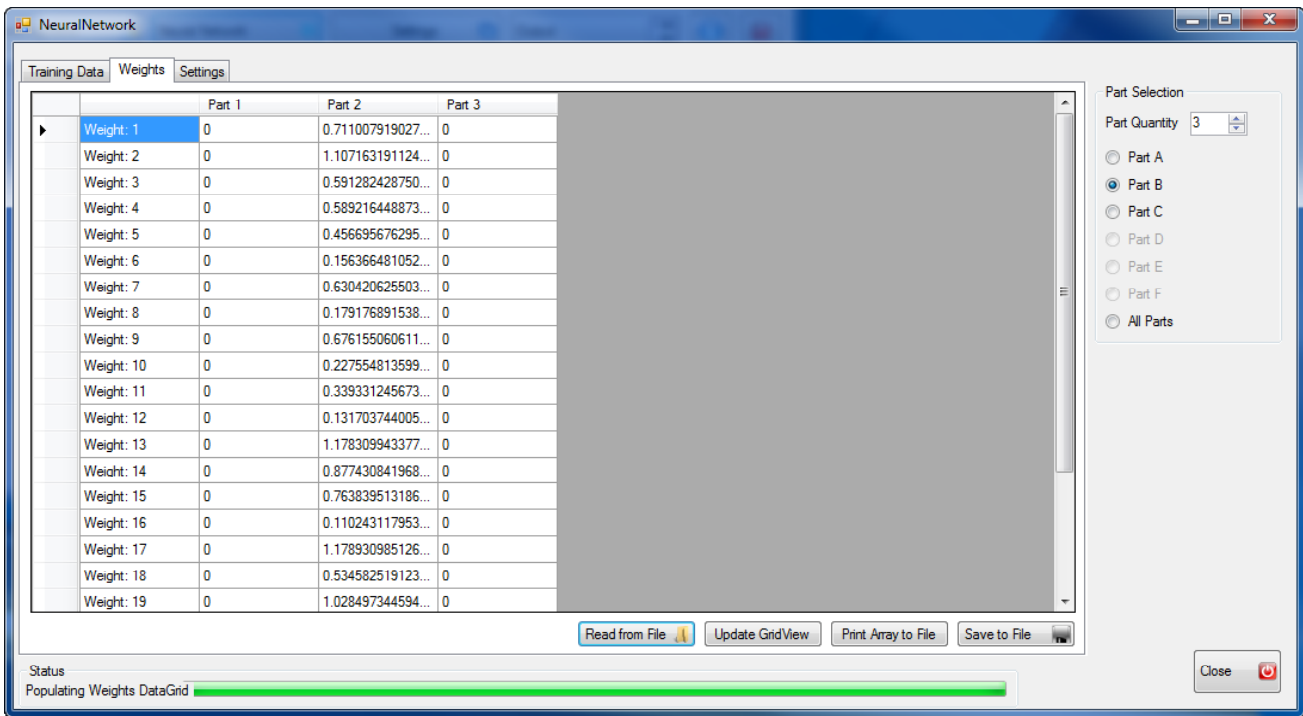


Figure 33 – Part Recognition Neural Network Weights Tab

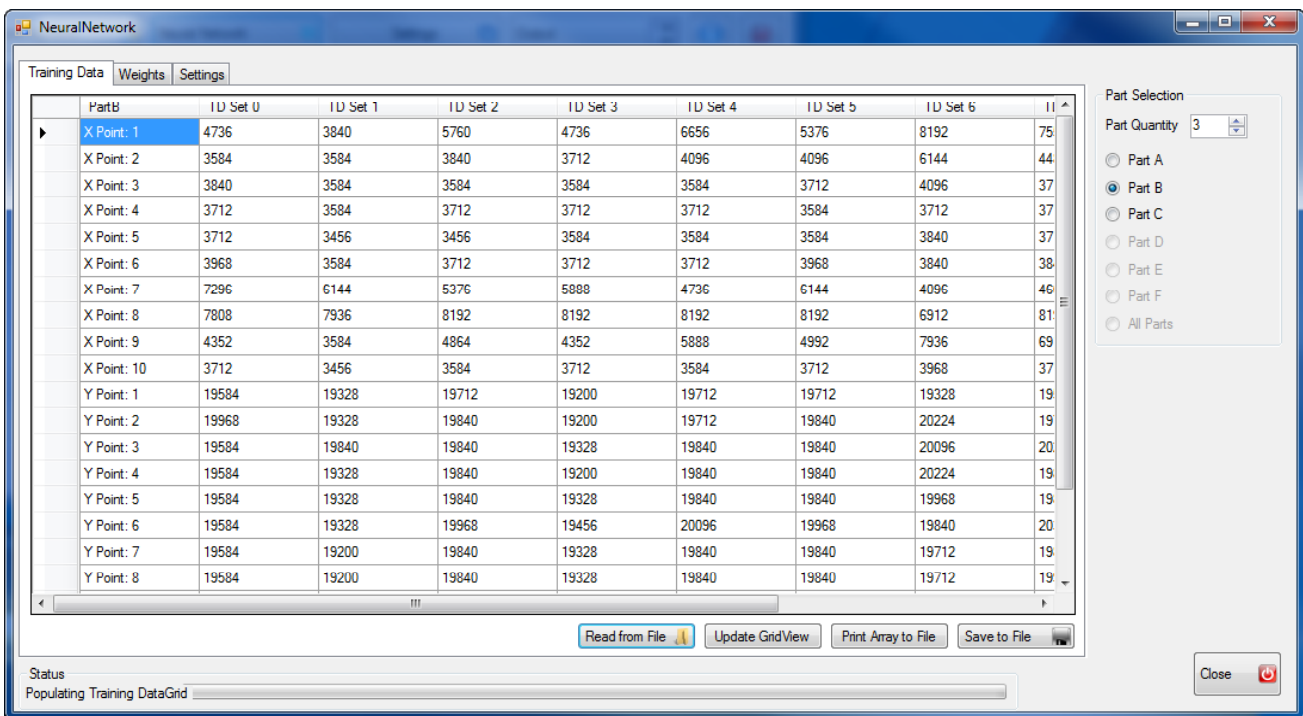


Figure 34 – Part Recognition Neural Network Training Data Tab

- The *Genetic Algorithm* form contains the following tabs:
 - GA Options.

General parameter settings for the Genetic Algorithm are specified in this tab which can be seen in *Figure 37 (page 80)*. The various group boxes are:

- i. Population Options.
Population size and the amount of genes in each individual is specified here.
 - ii. Stopping Condition.
Stopping conditions for the GA are specified here. A combination of three options namely *minimum error*, *maximum generation count* and *idle error* can be specified. For a detailed discussion on these options please refer to Section 5.1.7 *Algorithm Stopping Conditions*.
 - iii. GA Running Options.
Functionality is provided to execute the training process for multiple part ANNs successively without user interaction. The training process can also be executed for multiple iterations should it be required.
 - iv. Neural Network.
Target output for each part ANN is to be specified here. Two fitness functions are also provided: *Fitness Exponent* (used purely for initial GA optimization functionality testing) and *Fitness NN* which is used for ANN weight optimization.
- Generation Creation.
Detailed parameter settings for offspring creation are specified in this tab which can be seen in *Figure 36 (page 79)*. The various group boxes are:
 - i. Population Init.
A scaling factor for initial population gene values can be specified. Functionality to initialize gene values to within limits depending on neuron connection amount can also be specified. See Section 5.1.2 *Population Creation and Initialization* for a detailed description of the latter.
 - ii. Composition.
Each generation is created by a combination of *elitism*, *cross-over* and *mutation*. Functionality is provided to specify the percentage of each.
 - iii. Mutation Options.
The random number generation class of Visual Basic 2008 creates random values between 0 and 1. These values are used to mutate individual genes in individuals. A scaling factor for these random values can be specified. Functionality is provided to select whether a random or user specified amount of genes are to be mutated and also whether random or elite individuals are to be mutated.
 - iv. Cross-over Options.
Functionality is provided for the user to specify how many genes are to be crossed over as well as parent selection criteria.

- Settings.

The population for each generation can be output to a log file. The directory and filename for this log file is specified. There is an option to disable this logging as this is extremely processor intensive, negatively influencing the performance of the GA. Refer to *Figure 32 (page 76)*.

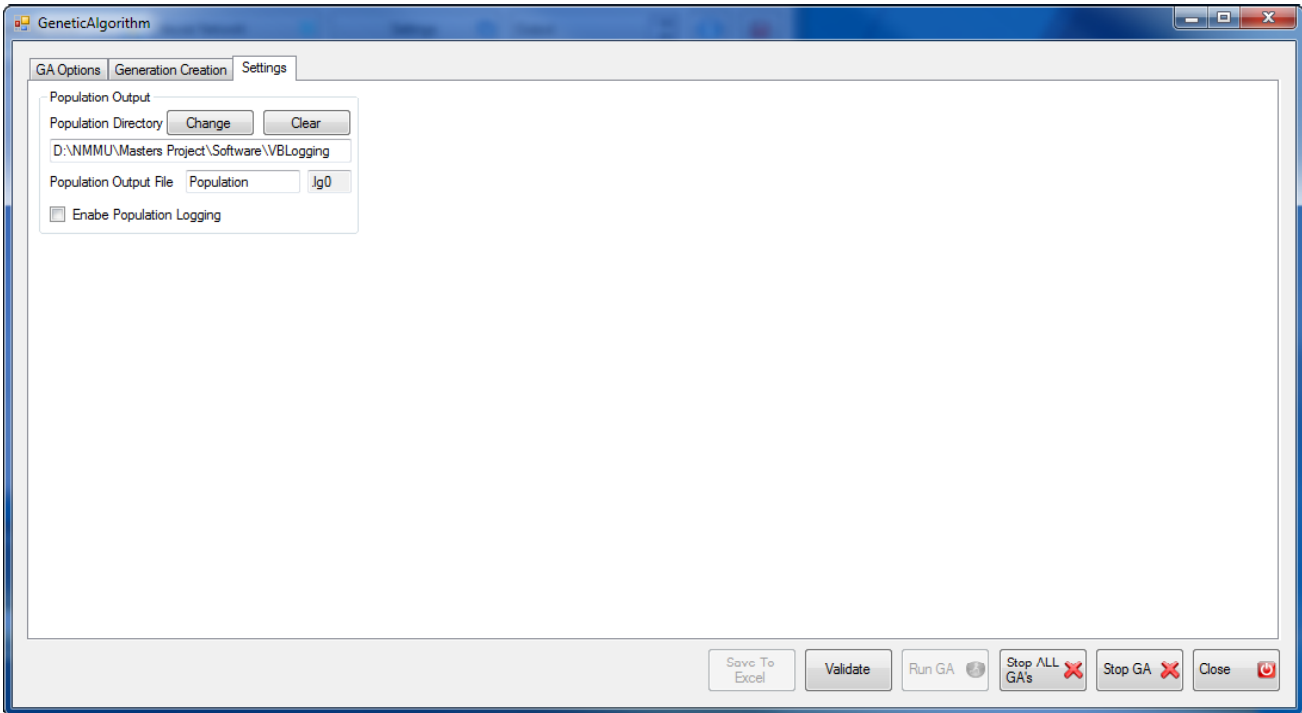


Figure 35 – Part Recognition Genetic Algorithm Settings Tab

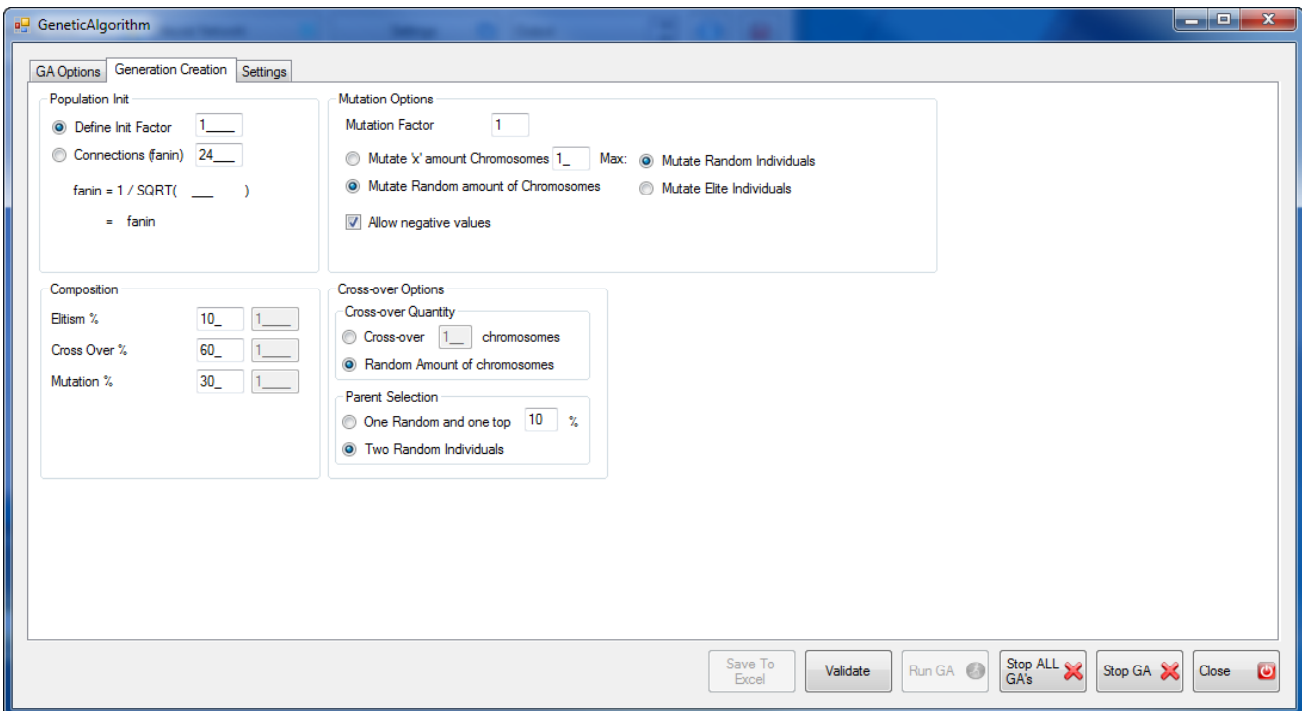


Figure 36 - Part Recognition Genetic Algorithm Generation Creation Tab

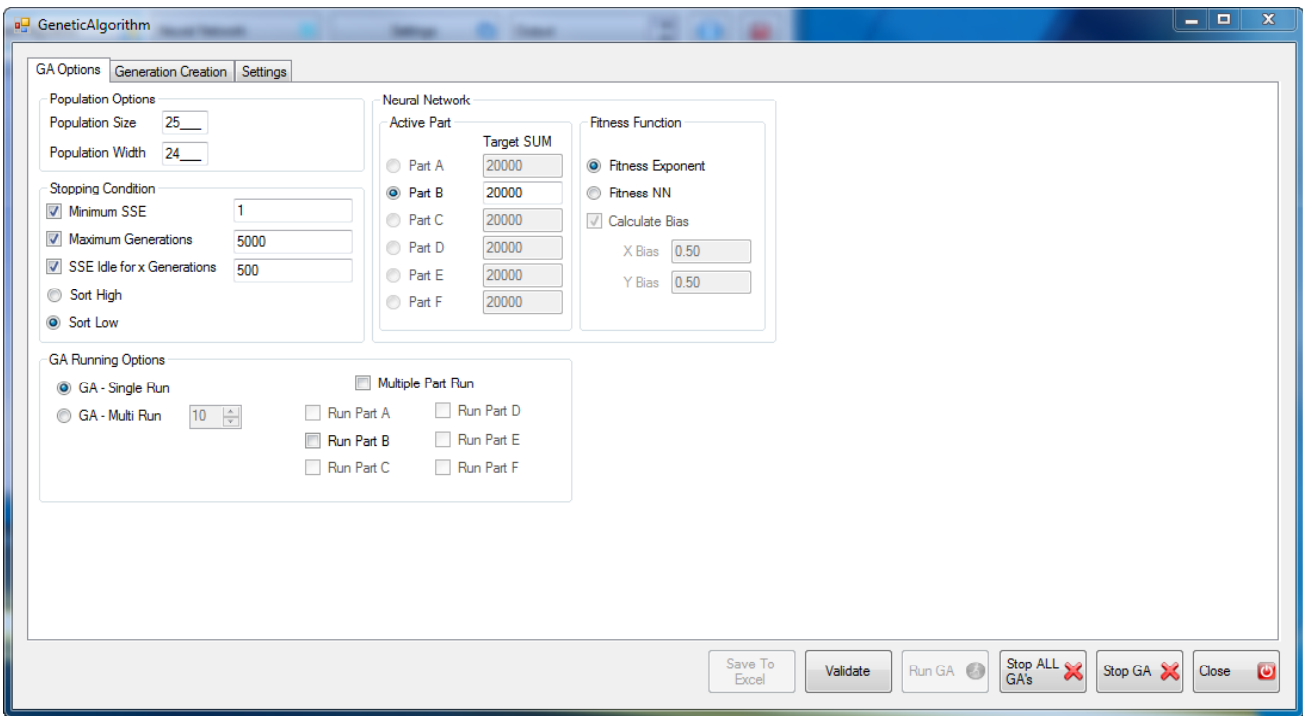


Figure 37 - Part Recognition Genetic Algorithm GA Options Tab

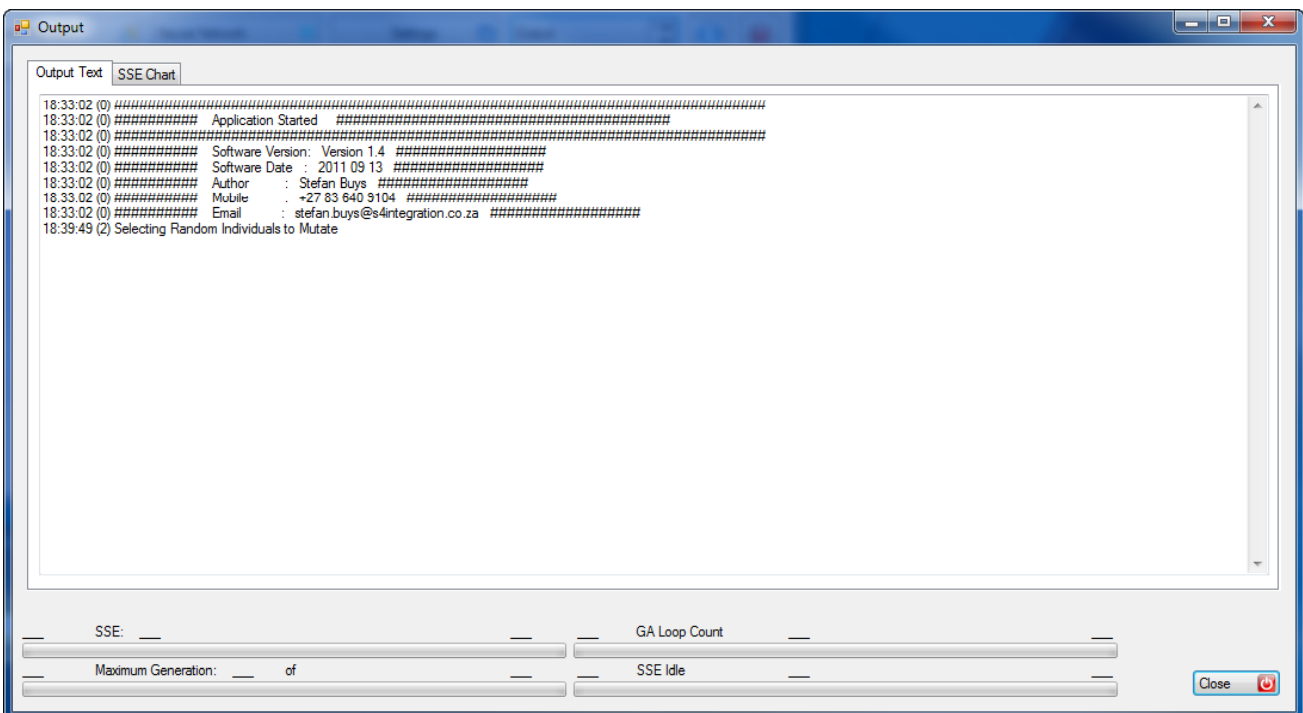


Figure 38 - Part Recognition Genetic Algorithm Output Window

5.4. Conclusion

In this chapter background on the operating principle and history of GAs as well as the ANN training application used in this research was discussed.

The various components of a GA are discussed in detail. Commonly used problem solution encoding schemes, their importance and the specific type developed for this project is discussed. The initial population creation and initialization along with the developed associated software and methodology is discussed. The importance of an accurate fitness function as well as the methodology and mathematical operational principle developed to accurately calculate the fitness of each individual for this project was discussed. Various selection operators, offspring creation methods, population sorting algorithms and GA stopping conditions as well as those implemented in the developed software and included in this research was discussed in detail.

The operation of the developed GA software is discussed in detail along with a description of the various options and GA control parameter settings available to the user in the GUI.

The next chapter described the supporting experimentation that was performed, their methodology as well as their results.

6. Chapter 6 – Part Recognition Performance

In this chapter the performance of the GA training as well as the ANN part recognition is discussed. The first section describes the various identified GA training control parameters. The rest of the chapter deals with the test results. First the effects of the various test cases with regard to the GA performance is analysed and discussed. Secondly the effects of the various test cases with regard to the PLC Simulator performance is analysed and discussed. Lastly the results of the hardware part recognition tests are discussed.

6.1. Genetic Algorithm Control Parameters

Various GA control parameters have been identified and classified into groups. A range of practical values for these control parameters were defined and 35 combinations, or test conditions, of these parameters were defined. In order to isolate the effects of each control parameter, only one set of parameters is adjusted at a time while the others maintain a preset default condition.

The identified GA control parameters have been classified into four main groups and their effect on the performance of the GA and ANN part recognition investigated.

6.1.1. Training Set Data

The *training set data* group consist of the amount of data sets used during the training process as well as the target value for each ANN.

- Training set quantity.

The effect on the performance of the GA and ANN part recognition as a result of the amount of training data sets provided is investigated in this section. More training sets could cause the ANN to over fit, subsequently hampering its ability to generalize. Not enough could result in inaccurate generalization of the ANN. The amount of training data also has a direct effect on the amount of time, or generations, taken by the GA to converge on a solution as it adds complexity to the solution space and drastically increases computational overheads.

The range of values for this control parameter was selected as:

- i. 10 Training Data Sets.
- ii. 20 Training Data Sets.

- Training Target.

The ANN is trained by the GA to output certain target value for each part. The effect on the performance of the GA and ANN part recognition as a result of the target values specified is investigated in this section. If the same targets are specified for all ANN output neurons the ANN might misfire, too much of a range and ANN loses accuracy as weights cannot accommodate such a wide range of targets.

The range of values for this control parameter was selected as:

- i. Target of 20 000 for all part output neurons.
- ii. Different targets for each part output neuron.
 - Part A – 15 000.
 - Part B – 30 000.
 - Part C – 45 000.
 - Part D – 60 000.
 - Part E – 75 000.
 - Part F – 90 000.
- iii. Different targets for each part output neuron.
 - Part A – 16 000.
 - Part B – 18 000.
 - Part C – 20 000.
 - Part D – 22 000.
 - Part E – 24 000.
 - Part F – 26 000.

6.1.2. Population

The *population* group consists of the GA population size as well as the initialization of the first generation population.

- Population Size.

The effect on the performance of the GA and ANN part recognition as a result of the size of the GA population is investigated in this section. A very large population will result in a big area of the possible search, or solution, space being searched with each generation but it will negatively affect the amount of computational processing required during each generation, increasing the computational time required for each generation. A very small population will have the inverse affect with a small area of the solution space searched during each generation, but will decrease the amount of computational time required to process each generation.

The range of values for this control parameter was selected as:

- i. 25 Individuals.
- ii. 100 Individuals.
- iii. 200 Individuals.

- Population Initialization.

The effect on the performance of the GA and ANN part recognition as a result of the initial values assigned to the genetic information of each individual is investigated in this section. If the population is initialized to values within a very large range, it will result in

a very large search, or solution, space being covered. This has both advantages and disadvantages as the optimal solution will most probably fall within the area being searched, but it will also cause an uneven distribution of individuals throughout the solution space with large areas between individuals not being searched. If the initialization range is too small it could also result in the entire solution space not being searched and the GA converging on a sub-optimal, local minimum.

The range of values for this control parameter was selected as:

- i. Scaling factor of $1 \times rand[0; 1]$.
- ii. Scaling factor of $10 \times rand[0; 1]$.
- iii. Random value within range $\pm \frac{1}{\sqrt{f_{min}}}$

6.1.3. Offspring creation

The individuals of each new generation can be created by one or more of the following three reproduction operators: elitism, cross-over and mutation. The effect that the implementation ratio of these operators has on the performance of the GA and ANN part recognition is investigated in this section.

While a high ratio of elitism ensures that good genetic information does not get lost during reproduction, it could also lead to premature convergence of the population on a local, sub-optimal minimum.

Mutation ensures that new genetic material is introduced into the population, thereby searching all areas of the solution space. If the ratio of mutation is too high it could lead to the loss of good genetic information as good- could be mutated into bad genetic information.

The rate of mutation is also an important factor as it controls the size of the mutation change and in turn, the size of the jump in the solution space. If the mutation rate is too small, it will result in only a section of the solutions space being searched thus causing the population to converge on a sub-optimal solution. If the mutation rate is too large, it will result in the GA jumping over a low error point in the solution space.

Cross-over also assists in keeping good genetic information within the next generation of the population. The disadvantage of cross-over is that it does not introduce any new genetic information into the population; it only recombines already existing information. If the ratio of cross-over is too high it could lead to convergence on a sub-optimal, local minimum.

The range of values for this control parameter was selected as:

- i. Majority mutation with offspring creation ratio of 10% mutation, 10% cross-over and 80% elitism.
- ii. Majority cross-over with offspring creation ratio of 10% mutation, 80% cross-over and 10% elitism.
- iii. Majority mutation with offspring creation ratio of 80% mutation, 10% cross-over and 30% elitism.
- iv. Evenly distributed offspring creation ratio of 33% mutation, 33% cross-over and 33% elitism.

6.1.4. Genetic Cross-over and Mutation

Both the cross-over and mutation reproductive operators have various control parameters. The effect that these parameters have on the performance of the GA and ANN part recognition is investigated in this section.

Cross-over Control Parameters.

- Parent Selection.
Both parents are either randomly selected or one parent is selected from the top 25% of the current population and the other randomly.
- Gene Cross-over Quantity.
Either one gene is randomly selected to be crossed-over or a random number of genes are selected to be crossed-over.

Mutation Control Parameters

- Mutation Scaling Factor.
A random value between -0.5 and 0.5 is generated for mutation. This random value can be scaled by multiplying with 0.5, 1 or 5.
- Negative Gene Values.
During mutation, the possibility exist that gene values could be mutated to a negative value. This is either allowed or not.
- Gene Mutation Quantity.
Either one gene is randomly selected to be mutated or a random number of genes are selected to be mutated.

During the investigation into the effects of the cross-over control parameters, 80% of offspring is created by cross-over in order to accentuate the effects. The same applies to the mutation control parameter investigation.

Table 10 – Test Conditions 1 to 19

	Training Data		Population		Offspring Creation			Cross-over		Mutation		
	Training Set Quantity	Training Target	Population Size	Population Initialization	Elitism	Cross Over	Mutation	Cross-over Parent Selection	Cross-over Quantity	Mutation Factor	Allow Negative Values	Mutate Quantity
Conditions 1 - Default	10	20000	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 2	10	15k - 90k	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 3	10	16k - 26k	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 4	20	20000	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 5	20	15k - 90k	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 6	20	16k - 26k	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 7	10	20000	25	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 8 - Default	10	20000	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 9	10	20000	200	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 10	10	20000	25	10	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 11	10	20000	100	10	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 12	10	20000	200	10	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 13	10	20000	25	$\pm \frac{1}{\sqrt{f_{amin}}}$	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 14	10	20000	100	$\pm \frac{1}{\sqrt{f_{amin}}}$	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 15	10	20000	200	$\pm \frac{1}{\sqrt{f_{amin}}}$	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 16	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 17	10	20000	100	1	10%	80%	10%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 18	10	20000	100	1	80%	10%	10%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 19 - Default	10	20000	100	1	33%	33%	33%	RANDOM	RANDOM	1.0	TRUE	RANDOM

Table 11 - Test Conditions 20 to 35

	Training Data		Population		Offspring Creation			Cross-over		Mutation		
	Training Set Quantity	Training Target	Population Size	Population Initialization	Elitism	Cross Over	Mutation	Cross-over Parent Selection	Cross-over Quantity	Mutation Factor	Allow Negative Values	Mutate Quantity
Conditions 20	10	20000	100	1	10%	80%	10%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 21	10	20000	100	1	10%	80%	10%	RANDOM	1 CHROMO	1.0	TRUE	RANDOM
Conditions 22	10	20000	100	1	10%	80%	10%	TOP 25%	RANDOM	1.0	TRUE	RANDOM
Conditions 23	10	20000	100	1	10%	80%	10%	TOP 25%	1 CHROMO	1.0	TRUE	RANDOM
Conditions 24	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	0.5	TRUE	RANDOM
Conditions 25	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	1.0	TRUE	RANDOM
Conditions 26	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	5.0	TRUE	RANDOM
Conditions 27	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	0.5	FALSE	RANDOM
Conditions 28	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	1.0	FALSE	RANDOM
Conditions 29	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	5.0	FALSE	RANDOM
Conditions 30	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	0.5	TRUE	1 CHROMO
Conditions 31	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	1.0	TRUE	1 CHROMO
Conditions 32	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	5.0	TRUE	1 CHROMO
Conditions 33	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	0.5	FALSE	1 CHROMO
Conditions 34	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	1.0	FALSE	1 CHROMO
Conditions 35	10	20000	100	1	10%	10%	80%	RANDOM	RANDOM	5.0	FALSE	1 CHROMO

6.2. Genetic Algorithm Performance

In this section the performance of the GA training under the various testing conditions is discussed.

The three most important performance factors are:

- Absolute Sum Error.
- Generation Count.
- Relative Computational Time.

In order to provide a more accurate representation of the GA performance, the training for each part of the 6 parts was repeated 10 times under each of the 35 defined test conditions. An average of the test results for each of the 6 parts for each of the 10 test runs was then calculated.

The results of each of the control parameter groups discussed in the previous section are discussed below.

6.2.1. Training Data

The table below summarizes the test conditions in the *training data* group.

Table 12 - Training Data Conditions

	Training Data	
	Training Set Quantity	Training Target
Conditions 1 - Default	10	20000
Conditions 2	10	15k - 90k
Conditions 3	10	16k - 26k
Conditions 4	20	20000
Conditions 5	20	15k - 90k
Conditions 6	20	16k - 26k

The chart below shows the results of the Training Data group.

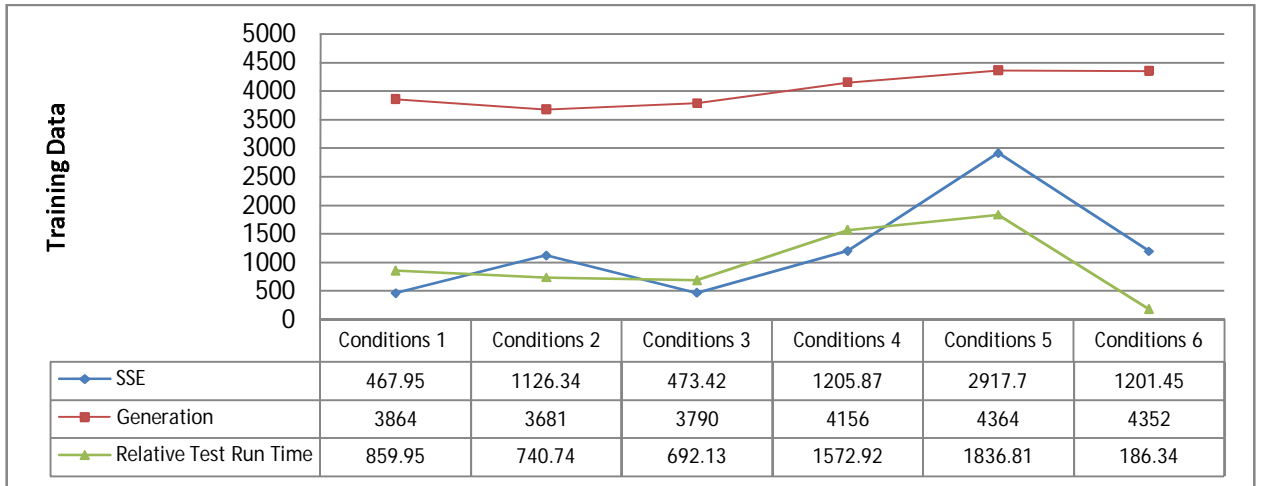


Figure 39 - Training Data Group Results

It can be seen that test Conditions 1 resulted in the lowest average error while test Conditions 5 resulted in the highest. It should be noted that test Conditions 1 had a target output of 20 000 for all parts while Conditions 5 had a different target output for each part ranging from 15 000 to 90 000. The higher error in Conditions 5 could be the result of the complex error space resulting from such a wide target range as well as the ANN that struggles to accommodate such a wide range of target outputs with its current architecture and amount of connecting weights. A more complex ANN architecture, with more connecting weight could possibly allow this wide range of target outputs to be accommodated. The increased amount of generations required for the population to converge as well as the increased amount of relative computation time can also be attributed to the same reasons as the increase in error.

6.2.2. Population

The table below summarizes the test conditions in the *population* group.

Table 13 - Population Conditions

	Population	
	Population Size	Population Initialization
Conditions 7	25	1
Conditions 8 - Default	100	1
Conditions 9	200	1
Conditions 10	25	10
Conditions 11	100	10
Conditions 12	200	10
Conditions 13	25	$\pm \frac{1}{\sqrt{fanin}}$
Conditions 14	100	$\pm \frac{1}{\sqrt{fanin}}$
Conditions 15	200	$\pm \frac{1}{\sqrt{fanin}}$

The figure below, *Figure 40 (page 90)*, shows the results of the *population* group.

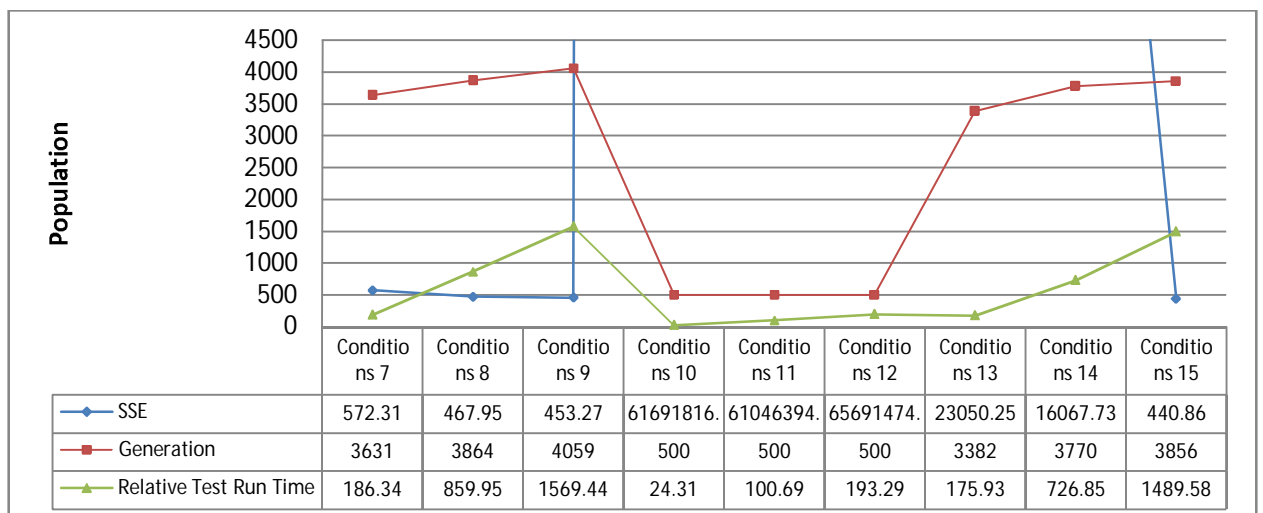


Figure 40 - Population Group Results (outliers removed)

It can clearly be seen that the high population initialization scaling factor of test Conditions 10, 11 and 12 resulted in very large error values. This can be attributed to the initial genetic information placing the individuals outside of the solution space for this specific optimization problem. In *Table 17 (page 42)* it can be seen that the GA stopping condition for all 60 test runs (6 parts with 10 test runs each) for all three these test conditions were *Squared Sum Error (SSE) Idle*. This further gives credibility to the theory of the individuals being placed outside the solution space by the large initialization scaling factor as neither cross-over nor mutation could get them inside the solution space before the set limit of *SSE Idle* generations was reached.

Although the generations required for the populations to converge is relatively constant for Conditions 7, 8 and 9 and then again for Conditions 13, 14 and 15, there is a clear increase in the relative computational time when looking at Conditions 7, 8 and 9 and then again with Conditions 13, 14 and 15. This increase is perfectly in line with the increase in population size for these conditions as predicted in Section 6.1.2 *Population*.

When looking at Conditions 7, 8 and 9 and again at Conditions 13, 14 and 15 it can be seen that the larger population size does result in a lower error but this comes at a very large computational complexity, and thus computational time, cost.

The lowest error was produced by test Condition 15. This can be attributed to the large population size as well as initialization to the range of $\pm \frac{1}{\sqrt{f_{anin}}}$ as discussed in section 5.1.2.2 *Initialization values*.

6.2.3. Offspring Creation

The table below summarizes the test conditions in the *offspring creation* group.

Table 14 - Offspring Creation Conditions

	Offspring Creation		
	Elitism	Cross Over	Mutation
Conditions 16	10%	10%	80%
Conditions 17	10%	80%	10%
Conditions 18	80%	10%	10%
Conditions 19 - Default	33%	33%	33%

Figure 41 (page 92) below shows the results of the *offspring creation* group.

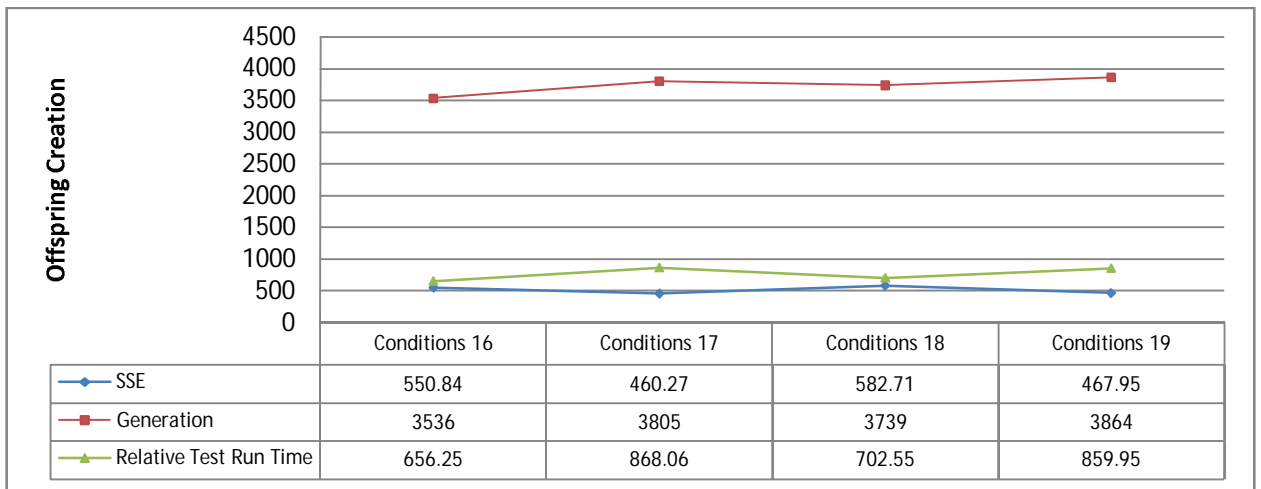


Figure 41 - Offspring Creation Group Results

Although Conditions 17, with the highest rate of cross-over, resulted in the lowest error it also resulted in the highest relative test run time and the second highest generation count required for the population to converge. The higher computational time can be explained by the slightly more complex computation required with the cross-over operator as two parents need to be selected and operated whereas the other two reproduction operators, elitism and mutation, only require one parent to be selected and operated.

6.2.4. Cross-over

The table below summarizes the test conditions in the *cross-over* group.

Table 15 - Cross-over Conditions

	Cross-over	
	Cross-over Parent Selection	Cross-over Quantity
Conditions 20	RANDOM	RANDOM
Conditions 21	RANDOM	1 CHROMO
Conditions 22	TOP 25%	RANDOM
Conditions 23	TOP 25%	1 CHROMO

Figure 42 (page 93) below summarizes the test results for the *cross-over* group.

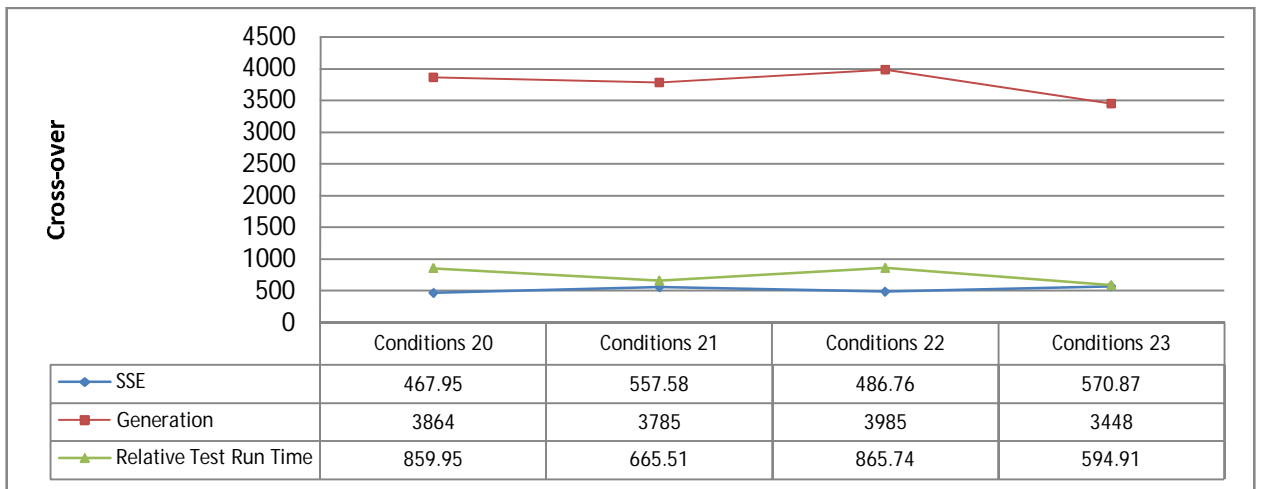
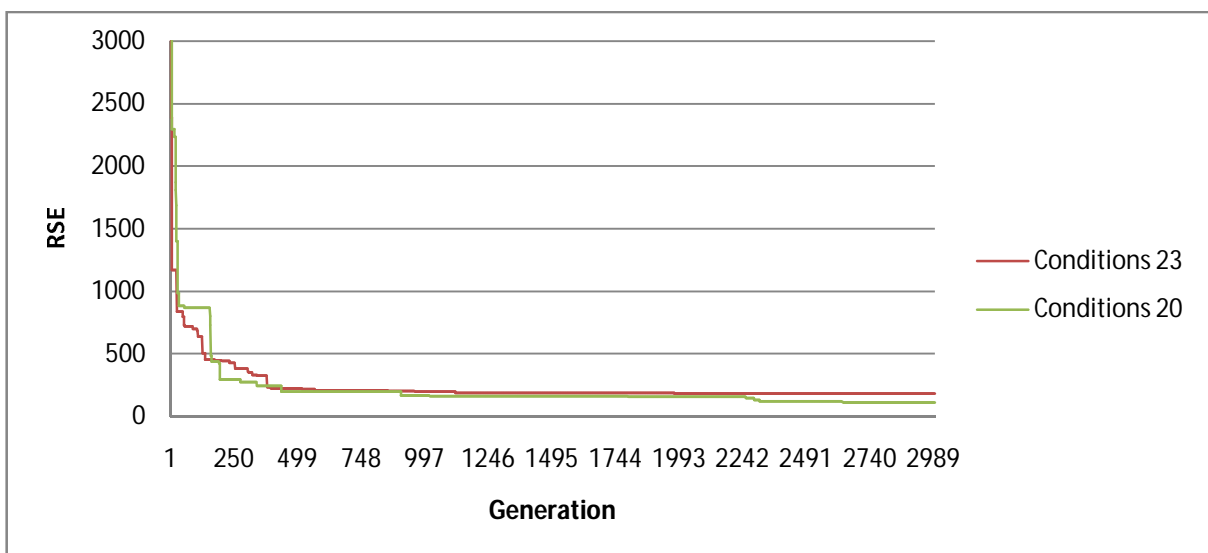


Figure 42 - Cross-over Group Results

Whilst the lowest error value was obtained by test Conditions 20, test Conditions 23 converged in 10% fewer generations as well as 30% less relative computational time. Conditions 23 did however prematurely converge, resulting in an error value 18% higher than Conditions 20.

It can be seen in the Root Square Error (RSE) vs. Generation graph, *Graph 5 (page 93)*, below that selecting at least one parent from the elite, top 25% of the population (Conditions 23) ensures that the good genetic information of that generation gets used for the next and in doing so, results in quick convergence of the population. It does however cause the population to prematurely converge on a local minimum as the offspring creation does not easily allow for new genetic information to enter the elite individuals of the population. While test Conditions 20 took longer to find a good solution, especially in the first few generations, it ultimately found the better global solution.



Graph 5 - RSE vs. Generation for Cross-over group

6.2.5. Mutation

The table below summarizes the test conditions in the *mutation* group.

Table 16 - Mutation Conditions

	Mutation		
	Mutation Factor	Allow Negative Values	Mutate Quantity
Conditions 24	0.5	TRUE	RANDOM
Conditions 25	1	TRUE	RANDOM
Conditions 26	5	TRUE	RANDOM
Conditions 27	0.5	FALSE	RANDOM
Conditions 28	1	FALSE	RANDOM
Conditions 29	5	FALSE	RANDOM
Conditions 30	0.5	TRUE	1 CHROMO
Conditions 31	1	TRUE	1 CHROMO
Conditions 32	5	TRUE	1 CHROMO
Conditions 33	0.5	FALSE	1 CHROMO
Conditions 34	1	FALSE	1 CHROMO
Conditions 35	5	FALSE	1 CHROMO

The possible solution space has been described as valleys and peak. *“The evaluation function defines a response surface that is much like a topography of hills and valleys, and the problem of finding the best solution is similar to searching for a peak on a mountain range while walking in a dense fog. You can only sample new points in your immediate vicinity and you can only make local decisions about where to walk next. If you always walk uphill, you’ll eventually reach a peak, but this might not be the highest peak in the mountain range. It might be just a “local optimum.” You might have to walk downhill for some period of time in order to find a position such that a series of local decisions within successive neighbourhoods leads to the global peak.”* (Michalewicz & Fogel, 2000).

As we are trying to find a low error value in this GA application, we are looking for the lowest valley point in the solution space.

As previously discussed in Section 6.1.4 *Genetic Cross-over and Mutation*, the rate of mutation is an important factor in offspring creation. A large mutation rate could cause the GA jumping over a local minimum and a small mutation rate could cause the GA to only search a small area of the solution space.

This can clearly be seen in *Graph 7 (page 97)* when looking at the error and generation count of test Condition group 24, 25 and 26 as well as Condition group 31, 31 and 32. As the rate of mutation increases, the required generation count for the population to converge decreases, but the convergence is premature as a lower point in the solution space is missed.

This statement is also supported when looking at the results of test Conditions 26. This Condition, with the highest mutation rate of 5, required the least amount of generations to converge. It is clear that the convergence was premature at a sub-optimal local minimum when comparing the error value of 833 that it yielded to the lowest yielded error of 381, 54% less, by Conditions 30.

This can also be seen when looking at the Error vs. Generation Graph, *Graph 8 (page 98)*, where the error is plotted from a random test run from Conditions 30, 31 and 32. It can be seen that, although the error initially decreases at a quicker rate with the conditions with higher mutation rates, they do tend to jump over the smaller minimum valleys in the solution space than the conditions with lower mutation rates.

The amount of genes that are mutated is also an important factor. When more than one gene is mutated, the individual is moved in more than one direction/dimension in the solution space. This can cause a good individual to be moved in the wrong direction in one of the dimensions in the solution space. Mutating only one gene at a time ensures a more controlled movement through the solution space as the individual is only moved in one dimension before being re-evaluated by the fitness function.

This can be seen when looking at the test results. Test Conditions 24 – 29 mutated a random amount of genes per generation while test Conditions 30 – 35 mutated only 1 gene per generation. The average error of Conditions 24 – 29 is 611, 24.4% higher than the average error of 461 yielded by test Conditions 30 – 35.

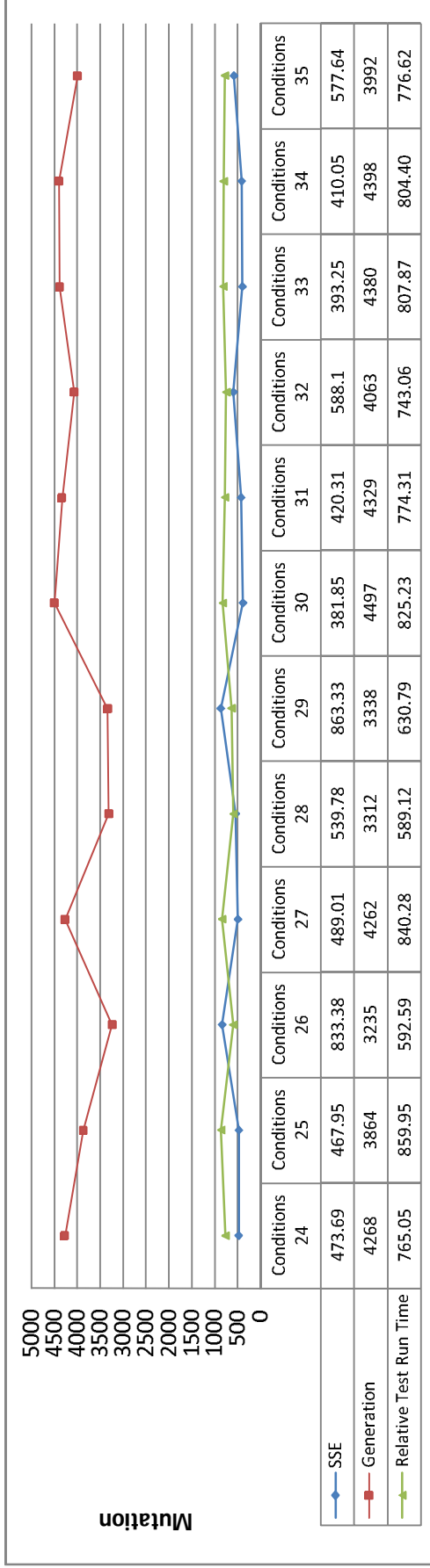
The above two statements are supported by the test results as test Conditions 30 yielded the lowest error. This test condition set contains both the smallest mutation factor and only 1 gene mutation per generation.

If the initial population is initialized to positive values only, as was the case with most test conditions, mutation is the only way that negative valued genetic information will be introduced into the gene pool. Allowing mutation of gene values to both positive and negative values doubles the size of the possible solution space.

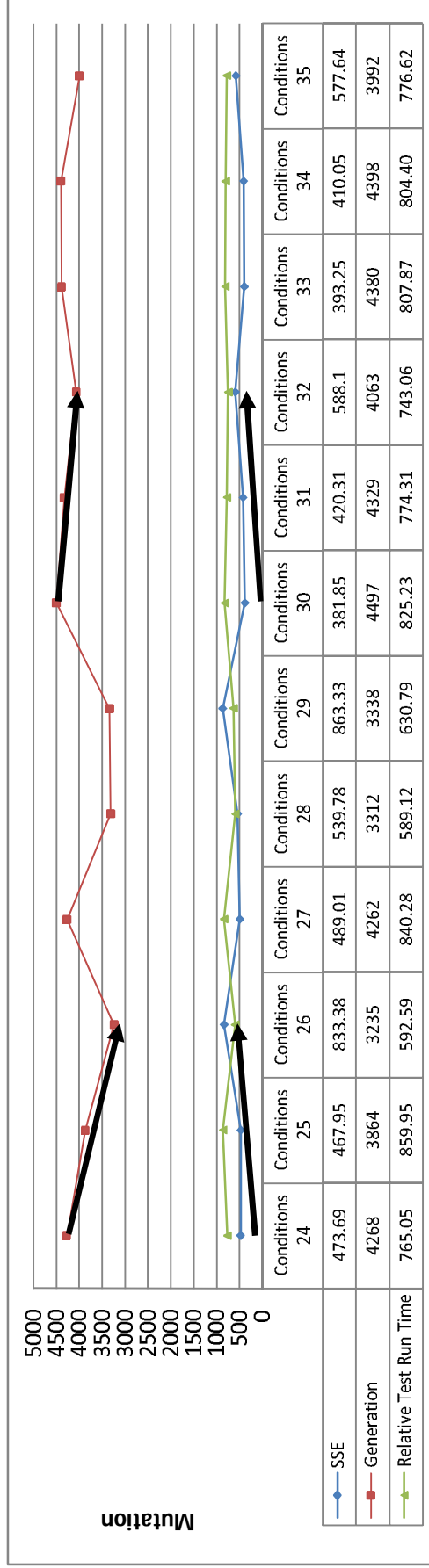
When looking at the end application of the gene as a neuron connection weight, a negative weight value in the neural network will attenuate that specific input signal.

When we investigate the average error of the test conditions where negative values were allowed we find an error of 527.54 as opposed to the average error value of a non-negative population of 545.51. The difference in these error values equate to a very small difference of only 3.2%. This small effect that this large increase in search space has can be explained because negative values would cause

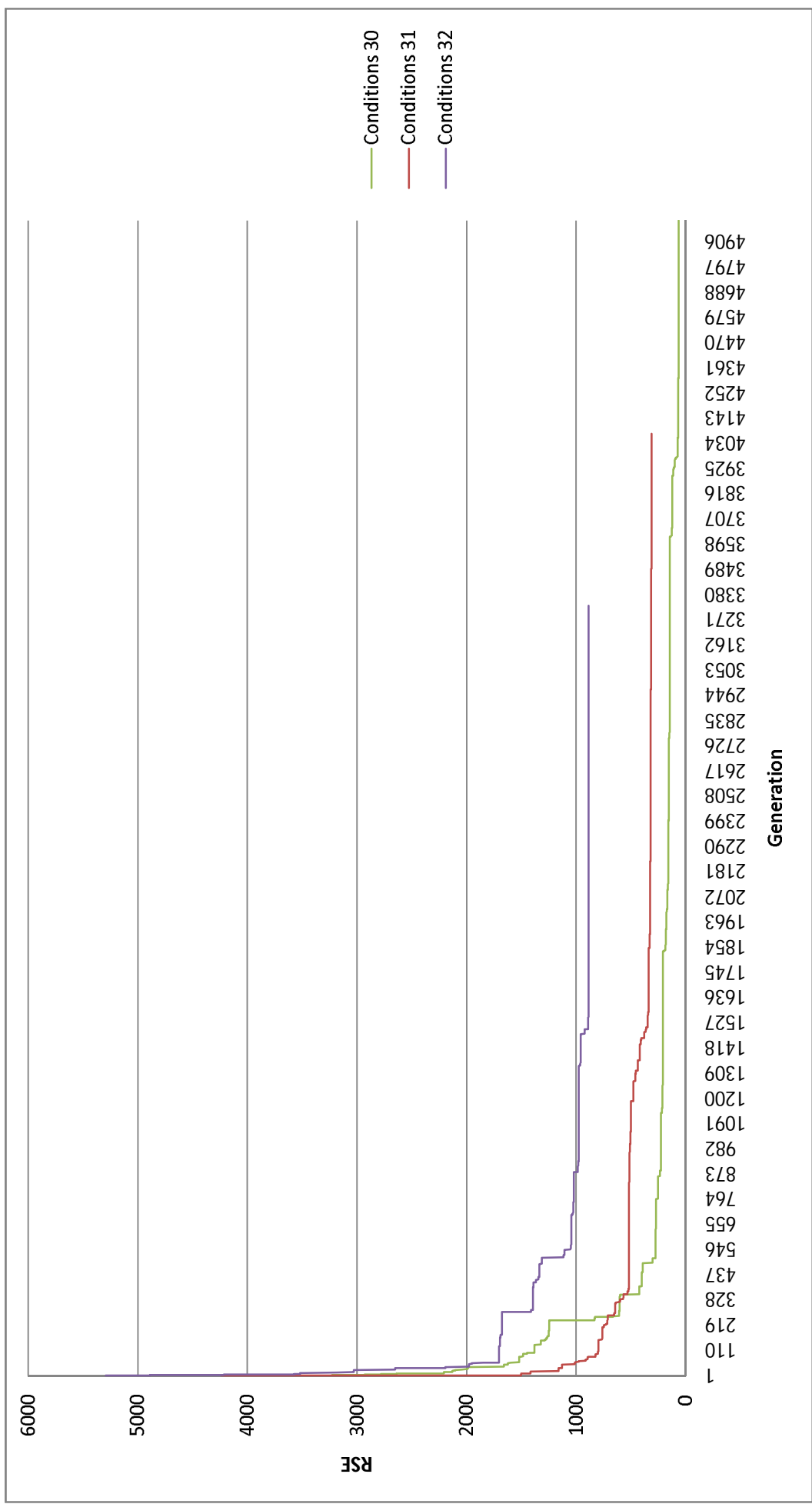
non-required attenuation. When looking at the converged example population in appendix *iv*. *Converged population for test Conditions 26* it can be seen that even though negative values are allowed, no negative gene values were found as optimal.



Graph 6 - Mutation Group results



Graph 7 - Mutation Group results Mutation Factor



Graph 8 - Error vs Generation for Mutation Factor

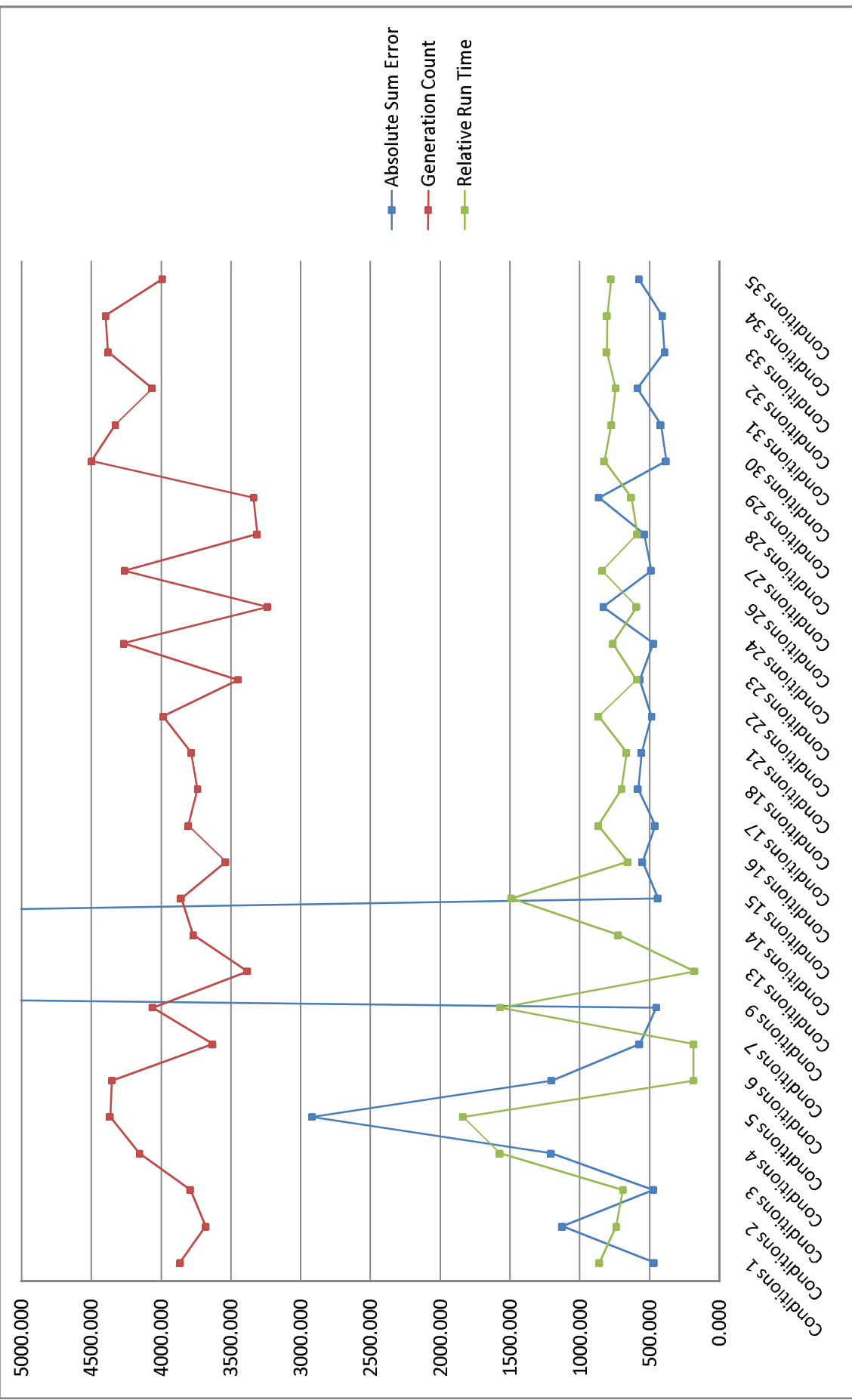
6.2.6. Overview of all test conditions

Each test Condition had a total of 60 test runs, 10 runs for each of the 6 parts. The average error, generation count to reach a stopping condition, average test run time, total test run time for all 60 test runs as well as the stopping conditions are tabled below.

Test Condition 30 resulted in the lowest error but also required the highest number of generations to converge on a solution.

Table 17 - Genetic Algorithm Training Results - Average

	SSE	Generation	Test Run Time	Total Time	Stopping Condition 1 - Sum Error Target Reached	Stopping Condition 2 - Max Generations	Stopping Condition 3 - Sum Error Idle
Conditions 1	467.95	3864	00:12:23	00 12:23:10	0	25	35
Conditions 2	1126.34	3681	00:10:40	00 10:39:58	0	23	37
Conditions 3	473.42	3790	00:09:58	00 09:57:34	0	27	33
Conditions 4	1205.87	4156	00:22:39	00 22:38:55	0	22	38
Conditions 5	2917.70	4364	00:26:27	01 02:26:44	0	35	25
Conditions 6	1201.45	4352	00:02:41	00 20:32:20	0	35	25
Conditions 7	572.31	3631	00:02:41	00 02:40:31	0	16	44
Conditions 9	453.27	4059	00:22:36	00 22:35:48	0	29	31
Conditions 10	61691816.20	500	00:00:21	00 00:20:56	0	0	60
Conditions 11	61046394.19	500	00:01:27	00 01:27:20	0	0	60
Conditions 12	65691474.83	500	00:02:47	00 02:47:27	0	0	60
Conditions 13	23050.25	3382	00:02:32	00 02:32:19	0	21	39
Conditions 14	16067.73	3770	00:10:28	00 10:27:44	0	27	33
Conditions 15	440.86	3856	00:21:27	00 21:27:29	0	26	34
Conditions 16	550.84	3536	00:09:27	00 09:27:02	0	22	38
Conditions 17	460.27	3805	00:12:30	00 12:30:02	0	23	37
Conditions 18	582.71	3739	00:10:07	00 10:06:54	0	24	36
Conditions 21	557.58	3785	00:09:35	00 09:35:28	0	27	33
Conditions 22	486.76	3985	00:12:28	00 12:27:58	0	29	31
Conditions 23	570.87	3448	00:08:34	00 08:33:39	0	21	39
Conditions 24	473.69	4268	00:11:01	00 11:00:54	0	35	25
Conditions 26	833.38	3235	00:08:32	00 08:32:05	0	15	45
Conditions 27	489.01	4262	00:12:06	00 12:06:03	0	33	27
Conditions 28	539.78	3312	00:08:29	00 08:29:16	0	18	42
Conditions 29	863.33	3338	00:09:05	00 09:05:10	0	16	44
Conditions 30	381.85	4497	00:11:53	00 11:52:47	0	39	21
Conditions 31	420.31	4329	00:11:09	00 11:09:29	0	34	26
Conditions 32	588.10	4063	00:10:42	00 10:41:43	0	28	32
Conditions 33	393.25	4380	00:11:38	00 11:38:00	0	38	22
Conditions 34	410.05	4398	00:11:35	00 11:34:34	0	38	22
Conditions 35	577.64	3992	00:11:11	00 11:11:14	0	30	30



Graph 9 - Genetic Algorithm Training Results - Average with outliers removed

6.3. Neural Network Performance with regard to Part Recognition

The previous section discussed the performance of each test Condition with regard to the GAs performance. What should be noted is that a test condition with very good performance with regard to GA performance (low error, quick population convergence and low computational complexity) does not necessarily mean good performance with regard to part recognition.

A test condition that resulted in a very low error might actually cause the ANN to over fit, resulting in bad generalization and inability to identify part input patterns other than those used for training. Another test condition, which used more training data sets, might have resulted in a high error value but in turn has an increased ability to classify and generalize part patterns not previously presented to it.

In this section the performance of the various test Conditions, using the connection weight values they optimized, are tested with regard to ANN part recognition.

Test results are classified as one or more of the following conditions:

- Identification OK.
 - The test part is correctly identified and the appropriate ANN output node has an HIGH (1) output.
 - Example: Part A is placed on the conveyor system. The ANN evaluates the captured data points and identifies the part as Part A. The output neuron for Part A has a HIGH (1) output.
 - There can be only one OK identification per test run.

- Identification NOK.
 - The test part is not identified and the appropriate ANN output node has an LOW (0) output.
 - Example: Part A is placed on the conveyor system. The ANN evaluates the captured data points and does not identify the part as Part A. The output neuron for Part A has a LOW (0) output.
 - There can be only one NOK identification per test run.

- Neural Network Misfire.
 - The test part is incorrectly identified as a different part and one, or more, of the incorrect ANN output nodes has a HIGH (1) output.
 - Example: Part A is placed on the conveyor system. The ANN evaluates the captured data points and identifies the part as Part C. The output neuron for Part C has a HIGH (1) output.
 - There can be up to 5 Misfires per test runs (Part A can be incorrectly classified as Part B, C, D, E and F).

The weight values calculated using each test Condition is first evaluated for accuracy using a PLC simulator. The best performing Conditions are then tested on physical hardware.

6.3.1. PLC Simulator

In Section 3.1.4 PLC Simulator, the need for a simulation of the ANN is explained. The results of these simulation tests are discussed in this section.

6.3.1.1. Result Overview

Detailed results of the ANNs ability to correctly identify part data presented to it whilst loaded with optimized weight conditions from each test conditions is tabled below.

Table 18 - PLC Simulator Results Conditions 1 to 11

		Part A Data	Part B Data	Part C Data	Part D Data	Part E Data	Part F Data
Cond. 1	Identification OK	49	38	48	47	48	49
	Identification NOK	1	12	2	3	2	1
	Neural Network Misfire	26	0	1	2	6	0
Cond. 2	Identification OK	49	45	49	45	47	46
	Identification NOK	1	5	1	5	3	4
	Neural Network Misfire	5	0	66	67	6	0
Cond. 3	Identification OK	49	38	49	47	48	49
	Identification NOK	1	12	1	3	2	1
	Neural Network Misfire	59	0	1	2	4	0
Cond. 4	Identification OK	50	50	49	47	47	50
	Identification NOK	0	0	1	3	3	0
	Neural Network Misfire	14	0	8	8	17	0
Cond. 5	Identification OK	50	50	49	47	46	46
	Identification NOK	0	0	1	3	4	4
	Neural Network Misfire	10	0	1	17	8	2
Cond. 6	Identification OK	49	46	47	46	43	44
	Identification NOK	1	4	3	4	7	6
	Neural Network Misfire	16	0	20	3	10	0
Cond. 7	Identification OK	49	34	49	48	48	49
	Identification NOK	1	16	1	2	2	1
	Neural Network Misfire	0	8	1	6	6	0
Cond. 9	Identification OK	49	33	49	48	47	50
	Identification NOK	1	17	1	2	3	0
	Neural Network Misfire	30	27	35	24	16	0
Cond. 10	Identification OK	0	0	0	0	0	0
	Identification NOK	50	50	50	50	50	50
	Neural Network Misfire	0	0	0	0	0	0
Cond. 11	Identification OK	0	0	0	0	0	0
	Identification NOK	50	50	50	50	50	50
	Neural Network Misfire	0	0	0	0	0	0

Table 19 - PLC Simulator Results Conditions 12 to 23

		Part A Data	Part B Data	Part C Data	Part D Data	Part E Data	Part F Data
Cond. 12	Identification OK	0	0	0	0	0	0
	Identification NOK	50	50	50	50	50	50
	Neural Network Misfire	0	0	0	0	0	0
Cond. 13	Identification OK	50	31	49	47	47	50
	Identification NOK	0	19	1	3	3	0
	Neural Network Misfire	5	0	1	13	17	8
Cond. 14	Identification OK	49	33	49	47	47	50
	Identification NOK	1	17	1	3	3	0
	Neural Network Misfire	0	4	8	3	7	0
Cond. 15	Identification OK	49	34	49	47	50	49
	Identification NOK	1	16	1	3	0	1
	Neural Network Misfire	3	32	2	5	12	1
Cond. 16	Identification OK	49	34	49	47	49	49
	Identification NOK	1	16	1	3	1	1
	Neural Network Misfire	1	24	39	15	6	35
Cond. 17	Identification OK	49	34	49	47	50	50
	Identification NOK	1	16	1	3	0	0
	Neural Network Misfire	22	57	3	5	14	0
Cond. 18	Identification OK	49	31	49	47	50	49
	Identification NOK	1	19	1	3	0	1
	Neural Network Misfire	29	28	0	16	16	0
Cond. 21	Identification OK	49	36	49	47	48	50
	Identification NOK	1	14	1	3	2	0
	Neural Network Misfire	24	31	2	2	9	0
Cond. 22	Identification OK	49	34	49	50	48	50
	Identification NOK	1	16	1	0	2	0
	Neural Network Misfire	25	0	1	2	46	2
Cond. 23	Identification OK	49	31	49	47	50	49
	Identification NOK	1	19	1	3	0	1
	Neural Network Misfire	20	15	49	24	12	34

Table 20 - PLC Simulator Results Conditions 24 to 35

		Part A Data	Part B Data	Part C Data	Part D Data	Part E Data	Part F Data
Cond. 24	Identification OK	49	41	49	47	48	49
	Identification NOK	1	9	1	3	2	1
	Neural Network Misfire	12	3	1	3	9	2
Cond. 26	Identification OK	49	32	49	47	50	50
	Identification NOK	1	18	1	3	0	0
	Neural Network Misfire	10	12	46	2	11	1
Cond. 27	Identification OK	49	33	49	47	48	50
	Identification NOK	1	17	1	3	2	0
	Neural Network Misfire	1	47	45	7	30	0
Cond. 28	Identification OK	50	42	49	47	48	49
	Identification NOK	0	8	1	3	2	1
	Neural Network Misfire	11	3	50	7	9	1
Cond. 29	Identification OK	49	41	48	47	48	50
	Identification NOK	1	9	2	3	2	0
	Neural Network Misfire	0	16	90	11	5	0
Cond. 30	Identification OK	49	38	49	47	48	49
	Identification NOK	1	12	1	3	2	1
	Neural Network Misfire	2	29	2	2	15	1
Cond. 31	Identification OK	49	34	49	47	47	49
	Identification NOK	1	16	1	3	3	1
	Neural Network Misfire	38	0	3	7	7	0
Cond. 32	Identification OK	50	34	49	50	47	49
	Identification NOK	0	16	1	0	3	1
	Neural Network Misfire	12	0	1	6	7	39
Cond. 33	Identification OK	49	32	49	47	47	49
	Identification NOK	1	18	1	3	3	1
	Neural Network Misfire	27	8	2	5	13	0
Cond. 34	Identification OK	49	34	49	47	47	50
	Identification NOK	1	16	1	3	3	0
	Neural Network Misfire	6	3	1	2	10	0
Cond. 35	Identification OK	49	34	49	49	50	50
	Identification NOK	1	16	1	1	0	0
	Neural Network Misfire	11	13	1	15	9	0

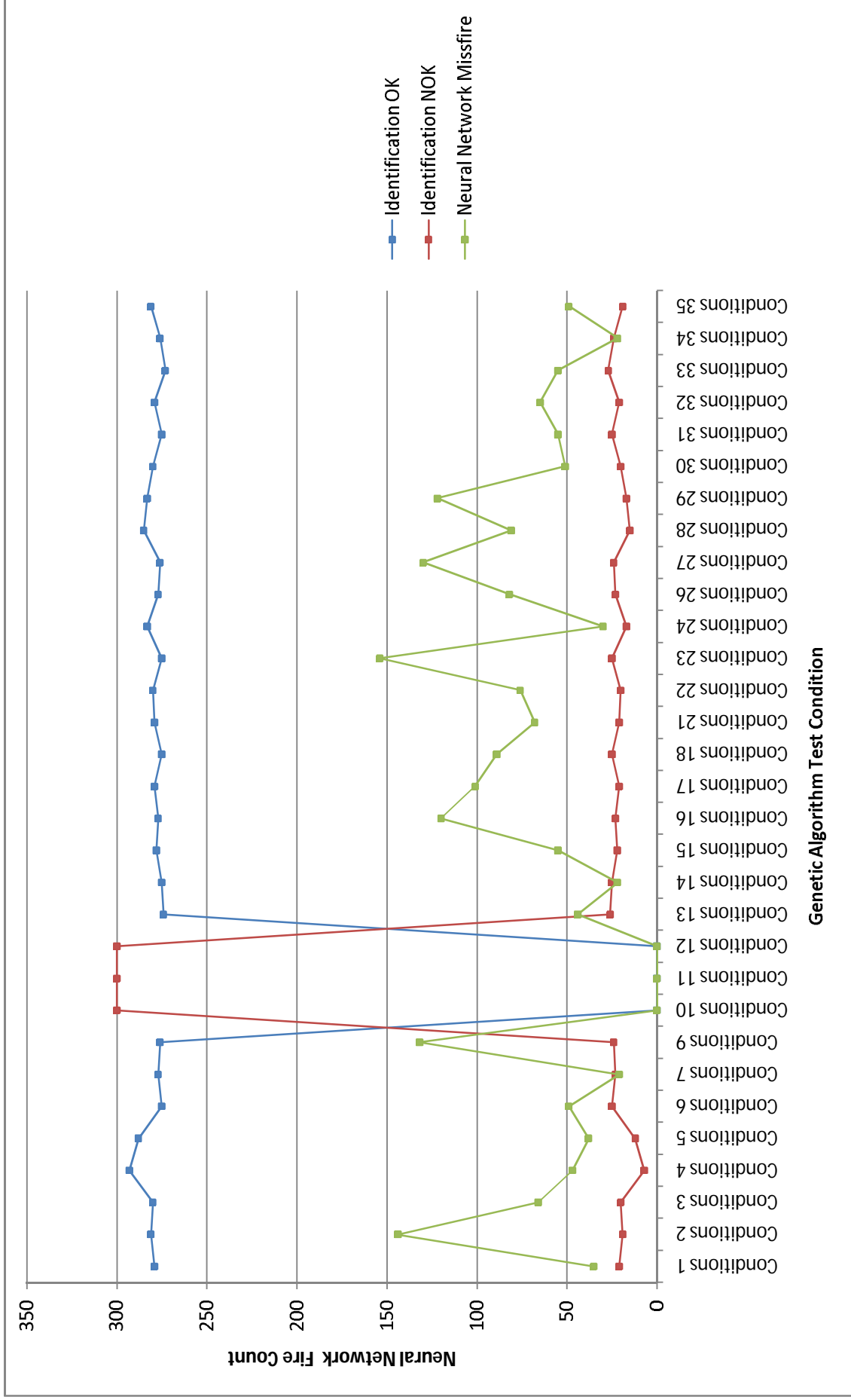
As previously explained in Section 3.3.4 *PLC Simulator*, 50 pre-captured data sets of each of the 6 parts is presented to the PLC Simulator. The PLC Simulator contains a duplicate of the ANN in the PLC software.

The total amount of OK, NOK and Miss Fires possible during this testing is:

- OK Identification.
 - $300 = 50 \text{ test runs} \times 6 \text{ parts} \times 1 \text{ correct identification.}$
- NOK Identification.
 - $300 = 50 \text{ test runs} \times 6 \text{ parts} \times 1 \text{ incorrect identification.}$
- ANN Miss fires.
 - $1500 = 50 \text{ test runs} \times 6 \text{ parts} \times 5 \text{ incorrect identification.}$

Table 21 - PLC Simulation Performance

	Identification OK	Identification NOK	Neural Network Misfire
Conditions 1	279	21	35
Conditions 2	281	19	144
Conditions 3	280	20	66
Conditions 4	293	7	47
Conditions 5	288	12	38
Conditions 6	275	25	49
Conditions 7	277	23	21
Conditions 9	276	24	132
Conditions 10	0	300	0
Conditions 11	0	300	0
Conditions 12	0	300	0
Conditions 13	274	26	44
Conditions 14	275	25	22
Conditions 15	278	22	55
Conditions 16	277	23	120
Conditions 17	279	21	101
Conditions 18	275	25	89
Conditions 21	279	21	68
Conditions 22	280	20	76
Conditions 23	275	25	154
Conditions 24	283	17	30
Conditions 26	277	23	82
Conditions 27	276	24	130
Conditions 28	285	15	81
Conditions 29	283	17	122
Conditions 30	280	20	51
Conditions 31	275	25	55
Conditions 32	279	21	65
Conditions 33	273	27	55
Conditions 34	276	24	22
Conditions 35	281	19	49



Graph 10 - PLC Simulation Performance

When looking at the test results tabled in *Table 18 (page 102)*, *Table 19 (page 103)* and *Table 20 (page 104)* as well as *Graph 10 (page 106)* it can be seen that the majority of the conditions performed reasonably well, with the exception of the very bad performance of conditions 10, 11 and 12. The bad performance of Conditions 10, 11 and 12 can be explained by the large error value that resulted from their training conditions (refer to Section 6.2.6 *Overview of all test conditions*). These three outliers are removed from further analysis.

The majority of Conditions resulted in good, high OK identification counts as well as good, low NOK identification counts. The standard deviation of OK Identification across the test Conditions is 4.458 (1.486%) and the standard deviation of NOK Identification is 4.458 (1.486%). This means that performance of all the conditions with respect to OK and NOK identification are very similar.

This is however not the case when we look at the *misfire* counts of the various test Conditions as can be seen in *Graph 10 (page 106)*. The standard deviation is 38.855 (12.952%), a very large variance in performance.

In order to accurately gauge the performance of each Condition, the misfire count was deducted from the OK count. These results are tabled below in *Table 22 (page 108)* and *Graph 11 (page 109)*.

Two Conditions have been selected to perform part recognition testing using hardware:

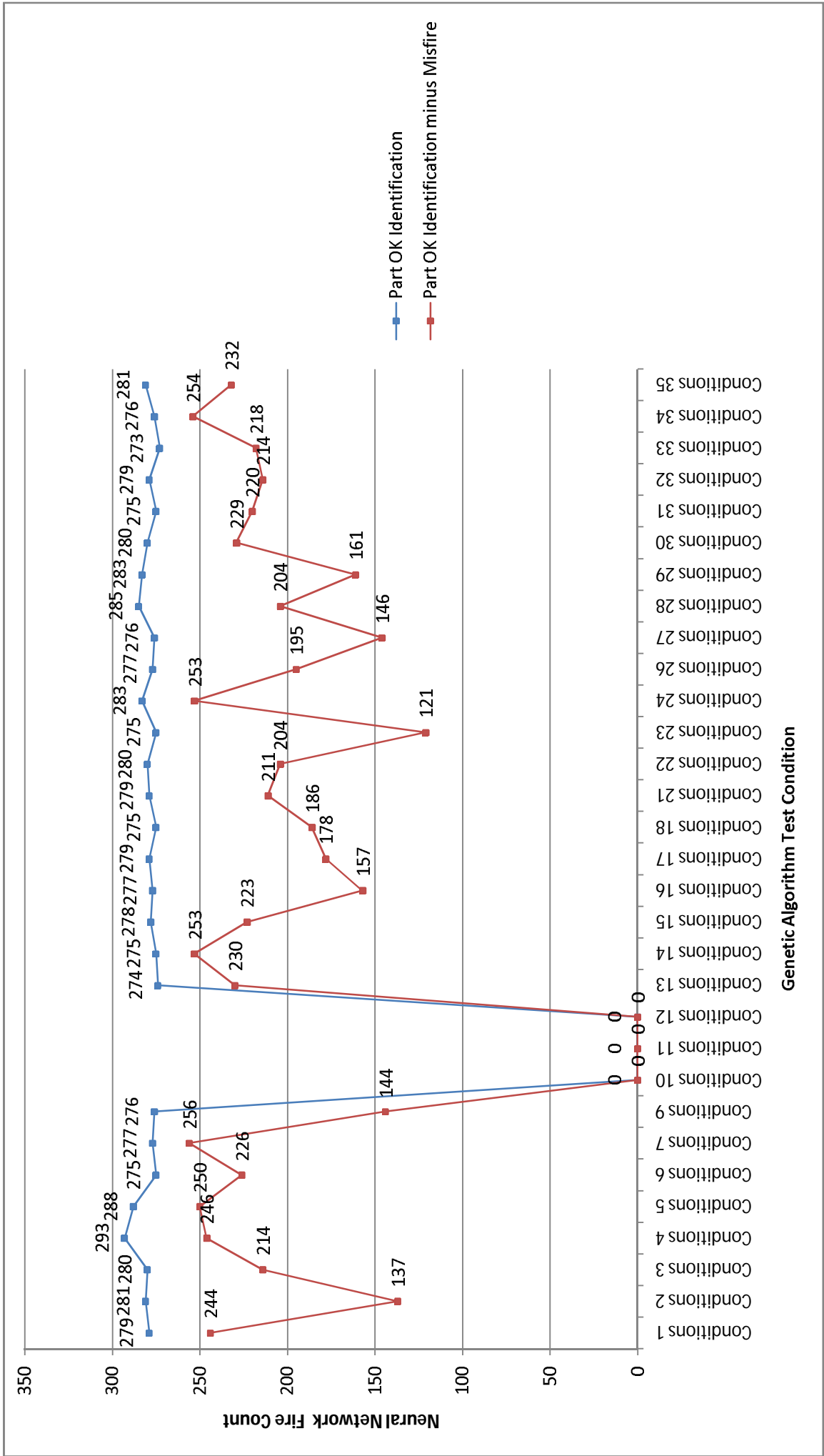
- Best OK part Identification .
 - Conditions 4 with an OK Identification of 293.

- Best part identification minus misfire.
 - Conditions 7 with an OK Identification minus Misfire of 256.

The details and results of these tests are discussed in the following two sections.

Table 22 - Part OK Identification minus Neural Network Misfires

	Identification OK	Neural Network Misfire	ID OK minus NN Misfire
Conditions 1	279	35	244
Conditions 2	281	144	137
Conditions 3	280	66	214
Conditions 4	293	47	246
Conditions 5	288	38	250
Conditions 6	275	49	226
Conditions 7	277	21	256
Conditions 9	276	132	144
Conditions 10	0	0	0
Conditions 11	0	0	0
Conditions 12	0	0	0
Conditions 13	274	44	230
Conditions 14	275	22	253
Conditions 15	278	55	223
Conditions 16	277	120	157
Conditions 17	279	101	178
Conditions 18	275	89	186
Conditions 21	279	68	211
Conditions 22	280	76	204
Conditions 23	275	154	121
Conditions 24	283	30	253
Conditions 26	277	82	195
Conditions 27	276	130	146
Conditions 28	285	81	204
Conditions 29	283	122	161
Conditions 30	280	51	229
Conditions 31	275	55	220
Conditions 32	279	65	214
Conditions 33	273	55	218
Conditions 34	276	22	254
Conditions 35	281	49	232



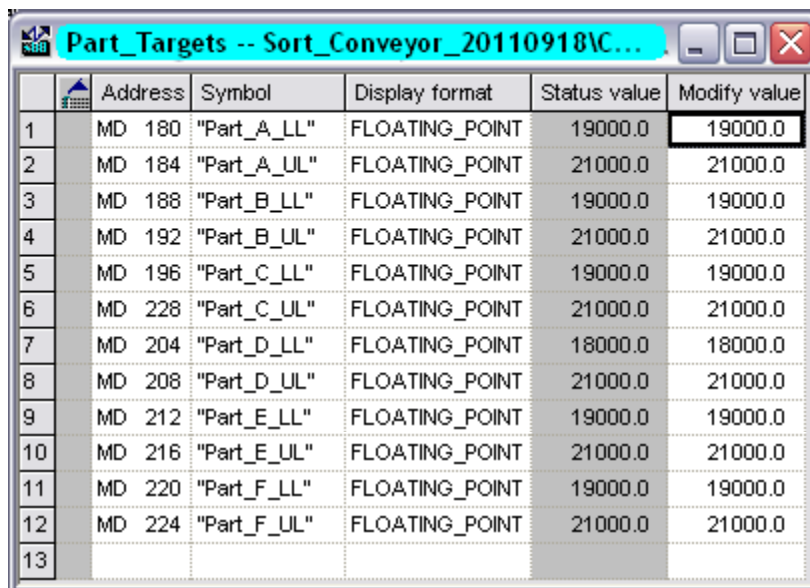
Graph 11 - Part OK Identification minus Neural Network Misfires

6.3.2. Hardware Testing

The two best performing Conditions from the PLC Simulator testing are tested on the hardware system setup. This is done to ensure that the test results are accurate and that a solution for a real-life automated part recognition system has been found during this research.

The test procedure is as follows:

- a) The optimized weights, using GA control parameters from Conditions 4, are loaded into the associated variables in the PLC using Siemens Step 7 Variable Tables (Siemens AG, 2009). See *Figure 44 (page 111)*.
- b) The Upper and Lower limits for the output neuron activation function are loaded into the associated variables in the PLC using Siemens Step 7 Variable Tables (Siemens AG, 2009). See *Figure 43 (page 110)*.
- c) The part to be identified is loaded onto the conveyor system. Time, X-axis and Y-axis values are automatically captured as the part passes the various sensors.
- d) The ANN calculates the outputs of the various Neurons. The part is classified. The results are shown in the Output Data Variable table and recorded. See *Figure 45 (page 111)*. In this particular case the part is correctly identified as part A.
- e) Step c) and d) is repeated 10 times for each of the 6 parts. A total of 60 test runs.



	Address	Symbol	Display format	Status value	Modify value
1	MD 180	"Part_A_LL"	FLOATING_POINT	19000.0	19000.0
2	MD 184	"Part_A_UL"	FLOATING_POINT	21000.0	21000.0
3	MD 188	"Part_B_LL"	FLOATING_POINT	19000.0	19000.0
4	MD 192	"Part_B_UL"	FLOATING_POINT	21000.0	21000.0
5	MD 196	"Part_C_LL"	FLOATING_POINT	19000.0	19000.0
6	MD 228	"Part_C_UL"	FLOATING_POINT	21000.0	21000.0
7	MD 204	"Part_D_LL"	FLOATING_POINT	18000.0	18000.0
8	MD 208	"Part_D_UL"	FLOATING_POINT	21000.0	21000.0
9	MD 212	"Part_E_LL"	FLOATING_POINT	19000.0	19000.0
10	MD 216	"Part_E_UL"	FLOATING_POINT	21000.0	21000.0
11	MD 220	"Part_F_LL"	FLOATING_POINT	19000.0	19000.0
12	MD 224	"Part_F_UL"	FLOATING_POINT	21000.0	21000.0
13					

Figure 43 - Simatic Step 7 Part Targets Table

	Address	Symbol	Display format	Status value	Modify value
1	DB10.DBD 52	"Part_A_Weights".V1_AY	FLOATING_POINT	1.438733	1.438733
2	DB10.DBD 56	"Part_A_Weights".V2_AY	FLOATING_POINT	1.786202	1.786202
3	DB10.DBD 60	"Part_A_Weights".V3_AY	FLOATING_POINT	2.132313	2.132313
4	DB10.DBD 64	"Part_A_Weights".V4_AY	FLOATING_POINT	0.04285055	0.04285055
5	DB10.DBD 68	"Part_A_Weights".V5_AY	FLOATING_POINT	0.2383607	0.2383607
6	DB10.DBD 72	"Part_A_Weights".V6_AY	FLOATING_POINT	0.5632356	0.5632356
7	DB10.DBD 76	"Part_A_Weights".V7_AY	FLOATING_POINT	0.1156624	0.1156624
8	DB10.DBD 80	"Part_A_Weights".V8_AY	FLOATING_POINT	0.9819065	0.9819065
9	DB10.DBD 84	"Part_A_Weights".V9_AY	FLOATING_POINT	1.501188	1.501188
10	DB10.DBD 88	"Part_A_Weights".V10_AY	FLOATING_POINT	0.1621789	0.1621789
11	DB10.DBD 92	"Part_A_Weights".V11_TIME_AY	FLOATING_POINT	16.29492	16.29492
12	DB10.DBD 0	"Part_A_Weights".V1_AX	FLOATING_POINT	0.3051932	0.3051932
13	DB10.DBD 4	"Part_A_Weights".V2_AX	FLOATING_POINT	0.128727	0.128727
14	DB10.DBD 8	"Part_A_Weights".V3_AX	FLOATING_POINT	0.0656073	0.0656073
15	DB10.DBD 12	"Part_A_Weights".V4_AX	FLOATING_POINT	0.004646748	0.004646748
16	DB10.DBD 16	"Part_A_Weights".V5_AX	FLOATING_POINT	0.1396606	0.1396606
17	DB10.DBD 20	"Part_A_Weights".V6_AX	FLOATING_POINT	2.071166	2.071166
18	DB10.DBD 24	"Part_A_Weights".V7_AX	FLOATING_POINT	0.4321278	0.4321278
19	DB10.DBD 28	"Part_A_Weights".V8_AX	FLOATING_POINT	0.5709586	0.5709586
20	DB10.DBD 32	"Part_A_Weights".V9_AX	FLOATING_POINT	1.357714	1.357714
21	DB10.DBD 36	"Part_A_Weights".V10_AX	FLOATING_POINT	0.3433841	0.3433841
22	DB10.DBD 40	"Part_A_Weights".V11_TIME_AX	FLOATING_POINT	2.588822	2.588822
23	DB10.DBD 108	"Part_A_Weights".AY_BIAS_WEIGHT	FLOATING_POINT	0.007498952	0.007498952
24	DB10.DBD 104	"Part_A_Weights".AX_BIAS_WEIGHT	FLOATING_POINT	0.592715	0.592715
25					

Figure 44 - Simatic Step 7 Part A Weight Table

	Address	Symbol	Display format	Status value	Modify value
1	DB10.DBD 48	"Part_A_Weights".AX_RESULT	FLOATING_POINT	32835.05	
2	DB10.DBD 100	"Part_A_Weights".AY_RESULT	FLOATING_POINT	67455.03	
3	MD 54		FLOATING_POINT	19967.67	
4	Q 124.0		BOOL	true	
5	DB11.DBD 48	"Part_B_Weights".BX_RESULT	FLOATING_POINT	52081.92	
6	DB11.DBD 100	"Part_B_Weights".BY_RESULT	FLOATING_POINT	75380.73	
7	MD 160		FLOATING_POINT	16069.07	
8	Q 124.1		BOOL	false	
9	DB12.DBD 48	"Part_C_Weights".CX_RESULT	FLOATING_POINT	39576.47	
10	DB12.DBD 100	"Part_C_Weights".CY_RESULT	FLOATING_POINT	53656.43	
11	MD 164		FLOATING_POINT	15104.75	
12	Q 124.2		BOOL	false	
13	DB13.DBD 48	"Part_D_Weights".DX_RESULT	FLOATING_POINT	72451.68	
14	DB13.DBD 100	"Part_D_Weights".DY_RESULT	FLOATING_POINT	55413.71	
15	MD 168		FLOATING_POINT	13040.49	
16	Q 124.3		BOOL	false	
17	DB14.DBD 48	"Part_E_Weights".EX_RESULT	FLOATING_POINT	75024.27	
18	DB14.DBD 100	"Part_E_Weights".EY_RESULT	FLOATING_POINT	28666.7	
19	MD 172		FLOATING_POINT	15276.98	
20	Q 124.4		BOOL	false	
21	DB15.DBD 48	"Part_F_Weights".FX_RESULT	FLOATING_POINT	86711.59	
22	DB15.DBD 100	"Part_F_Weights".FY_RESULT	FLOATING_POINT	41555.52	
23	MD 176		FLOATING_POINT	10335.88	
24	Q 124.5		BOOL	false	
25					

Figure 45 - Simatic Step 7 ANN Output Table

6.3.2.1. Best Part Identification – Conditions 4

Weights optimized by the GA under control parameters of Conditions 4 are tabled below.

Table 23- Conditions 4 Optimized Weights

Conditions 4						
	Part A	Part B	Part C	Part D	Part E	Part F
Weight X: 1	1.4387327	0.0097524	0.2603602	0.4979276	0.5105519	0.4183017
Weight X: 2	1.7862022	0.0055107	0.0234618	0.2428633	0.001404	0.1288028
Weight X: 3	2.1323127	0.0607025	0.2075251	0.6724553	0.4549259	1.2384017
Weight X: 4	0.0428505	0.4852304	1.0198115	0.2298516	0.2796867	0.7706845
Weight X: 5	0.2383607	1.8668086	1.6945628	0.1181348	0.2212333	1.9065886
Weight X: 6	0.5632357	0.051487	0.4807401	1.3293235	0.005709	0.2147844
Weight X: 7	0.1156624	0.2119084	0.7072252	0.0160606	0.0044002	0.4134106
Weight X: 8	0.9819065	0.8147115	0.8319214	0.6974159	0.598667	0.0413434
Weight X: 9	1.5011884	0.0294201	0.3955211	0.7501354	0.7876091	0.0644553
Weight X: 10	0.1621789	0.9573333	0.3926012	0.2048469	0.1055694	0.8433537
Weight X: Time	16.29492	0.1211996	0.8519506	2.7119938	5.3153731	1.1385356
Weight Y: 1	0.3051932	1.2163038	0.0194841	0.7313727	0.3508102	0.5501317
Weight Y: 2	0.128727	1.1044052	0.0443169	0.1619364	1.3810257	1.1492997
Weight Y: 3	0.0656073	0.0569903	0.0137175	0.0672034	0.3038722	0.7274533
Weight Y: 4	0.0046467	0.0227241	0.2867397	0.0656559	0.11208	0.496789
Weight Y: 5	0.1396606	1.5757907	1.7380725	0.2525135	0.3169387	0.2479892
Weight Y: 6	2.0711657	0.7380132	0.0983576	0.0469844	0.3168938	0.6027826
Weight Y: 7	0.4321278	0.843519	2.5786192	0.209078	0.0031125	0.1217318
Weight Y: 8	0.5709586	2.089403	1.2214732	3.2237293	1.4730358	0.1364813
Weight Y: 9	1.3577144	0.0176705	0.1896182	0.5017415	1.0886345	0.8588952
Weight Y: 10	0.3433842	0.0105622	0.078258	2.1101647	2.3149367	0.69286
Weight Y: Time	2.5888222	2.6281954	1.3554803	2.9460632	6.2420023	1.0050278
Weight X: Bias	0.007499	0.4974125	0.3142632	0.122114	0.021169	0.0413651
Weight Y: Bias	0.592715	0.0659691	0.0697927	0.2107958	0.2589287	0.1587502

The results of the hardware tests using these optimized weights are tabled below:

Table 24 - Part Recognition Test 1 Result Summary

	Part A	Part B	Part C	Part D	Part E	Part F
Part Identification OK	10	10	10	10	10	10
Part Identification NOK	0	0	0	0	0	0
Neural Network Misfire	1	0	0	0	3	0

Detailed results and outputs for each neuron layer during the part A part recognition test can be seen in *Table 26 (page 114)*. The various tabled values relate to the different neuron layers as depicted in the ANN architecture, *Figure 18 (page 45)*.

Complete results for all parts can be found in Appendix i – Hardware Test Best Part ID Results.

Table 25 - ANN Output Mapping

	Neuron Architecture Component
Part A: X Neuron Result	Y_1
Part A: Y Neuron Result	Y_2
Part A: Final Neuron Result	X_1
Part A: Activation Function	O_1
Part B: X Neuron Result	Y_3
Part B: Y Neuron Result	Y_4
Part B: Final Neuron Result	X_2
Part B: Activation Function	O_2
Part C: X Neuron Result	Y_5
Part C: Y Neuron Result	Y_6
Part C: Final Neuron Result	X_3
Part C: Activation Function	O_3
Part D: X Neuron Result	Y_7
Part D: Y Neuron Result	Y_8
Part D: Final Neuron Result	X_4
Part D: Activation Function	O_4
Part E: X Neuron Result	Y_9
Part E: Y Neuron Result	Y_{10}
Part E: Final Neuron Result	X_5
Part E: Activation Function	O_5
Part F: X Neuron Result	Y_{11}
Part F: Y Neuron Result	Y_{12}
Part F: Final Neuron Result	X_6
Part F: Activation Function	O_6

Table 26 - Part Recognition Test 1 Part A

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	32829.12	32835.05	32795.98	32761.16	32760.56	33146.73	32924.48	32771.13	32242.5	32742.25
Part A: Y Neuron Result	64076.36	67097.73	72439.37	73965.02	79660.48	44207.09	49801.21	66986.15	61694.53	70784.37
Part A: Final Neuron Result	19938.82	19964.99	19981.89	19972.69	20015.05	19978.07	19888.29	19926.27	19573.26	19937.63
Part A: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part B: X Neuron Result	68085.67	68079.21	67923.52	67785.4	67782.49	69605.56	68570.67	67810.42	68566.02	67670.75
Part B: Y Neuron Result	27064.36	38537.18	36090.13	25200.22	28825.48	34047.67	37733.25	29633.89	25971.87	35415.12
Part B: Final Neuron Result	17953.7	23660	22432.53	17006.64	18809.7	21527.55	23292.53	19213.65	17441.97	22080.1
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Part C: X Neuron Result	31501.99	35128.75	35126.25	31601.78	31565.08	35358.77	31923.07	31859.1	31357.65	31968.47
Part C: Y Neuron Result	41469.12	51952.47	50297.61	39107.17	45073.35	44657.36	49066.02	43958.11	39750.5	50553.75
Part C: Final Neuron Result	15230.83	18778.48	18258.24	14495.52	16367.91	16501.95	17647.64	16037.95	14680.65	18118.35
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	48282	49002.59	48908.98	48178.61	48170.21	49647.8	48526.72	48243.09	47960.66	48183.29
Part D: Y Neuron Result	30052	31610.85	33856.24	31128.53	34268.8	25119.47	27039.19	31053.4	29238.62	33087.24
Part D: Final Neuron Result	13847.42	14189.67	14444.13	13957.08	14338.78	13532.99	13531.1	13961.5	13680.35	14197.25
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	63593.43	64798.28	64753.38	63343.25	63328.9	66127.72	64227.48	63476.33	60497.6	63513.36
Part E: Y Neuron Result	20311.63	21871.1	22375.44	21337.56	23963.17	16587.28	18228.42	21241.02	20336.94	22742.45
Part E: Final Neuron Result	16896.14	17241.12	17240.17	16853.08	16904.95	17473.5	17016.21	16885.49	16095.08	16926.86
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	58452.27	64531.06	64460.65	58240.3	58176.7	66308.09	59702.62	58777.52	50027.4	58990.98
Part F: Y Neuron Result	48120.03	58399.27	57726.32	48438.25	56334.07	46520.43	51390.46	50885.16	49242.8	57978.8
Part F: Final Neuron Result	11269.8	12660.01	12621	11249.31	11565.83	12450.75	11603.58	11435.82	9978.794	11763.13
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

6.3.2.2. Best ID minus Misfire – Conditions 7

Weights optimized by the GA under control parameters of Conditions 4 are tabled below.

Table 27 - Conditions 7 Optimized Weights

Conditions 7						
	Part A	Part B	Part C	Part D	Part E	Part F
Weight X: 1	0.2896508	0.2129917	0.2236633	0.643736	0.0169259	0.5605942
Weight X: 2	0.650961	0.1069884	0.626496	0.8100904	0.0197989	0.1417316
Weight X: 3	0.9843429	0.9260315	0.8527593	0.3480191	0.8189701	0.2089701
Weight X: 4	0.4254546	2.3402791	0.2324546	0.3252092	0.398593	0.7814011
Weight X: 5	0.1222026	0.9013037	1.4420261	0.7933724	0.6577362	0.1912165
Weight X: 6	0.0277485	0.6566761	1.0158664	1.0780145	0.1490603	0.3502728
Weight X: 7	1.6750733	0.8314261	0.7332261	0.0130953	0.0297126	1.9709427
Weight X: 8	1.2747416	1.3343229	1.3303655	2.083558	1.5719614	0.4228703
Weight X: 9	0.5438748	0.2269571	1.3068371	0.3746177	0.0739984	0.1440837
Weight X: 10	1.6090906	2.3017007	0.5151782	3.1789466	0.1780303	2.4219242
Weight X: Time	1.2089423	0.6805524	0.3862353	2.9120112	0.0062237	8.1856592
Weight Y: 1	0.8503171	0.9416137	0.0537304	0.779761	0.8765325	0.9806908
Weight Y: 2	1.9380724	0.2007521	0.0831067	0.7266394	0.4548965	0.5592834
Weight Y: 3	2.4752832	0.6398264	0.8024611	0.8585403	1.3770302	1.4760956
Weight Y: 4	0.0298566	0.2120624	0.5255398	0.0855927	0.534863	0.6747847
Weight Y: 5	0.6498714	0.3456385	0.3076583	0.5919261	0.7083291	0.0406149
Weight Y: 6	0.5299015	0.1035231	0.0830483	0.1116966	0.6829214	1.0781444
Weight Y: 7	0.2243483	1.7012191	0.9437965	1.8418689	0.0110735	0.1151873
Weight Y: 8	1.0865299	0.7949521	0.659018	0.8569685	0.7437888	0.6190317
Weight Y: 9	0.0126419	0.5166775	0.4612235	1.7198872	1.0240057	1.2695798
Weight Y: 10	0.9786541	0.0052335	0.3749318	0.8667028	0.4896893	1.3077484
Weight Y: Time	1.2937972	1.9861542	0.5885561	5.0744873	0.1531034	5.9662042
Weight X: Bias	0.0339961	0.1188403	0.1874316	0.1188764	0.0390083	0.0279585
Weight Y: Bias	0.166574	0.1365311	0.1275464	0.0890678	0.1887221	0.1057995

The results of the hardware tests using these optimized weights are tabled below:

Table 28 - Part Recognition Test 2 Result Summary

	Part A	Part B	Part C	Part D	Part E	Part F
Part Identification OK	10	6	10	10	10	10
Part Identification NOK	0	4	0	0	0	0
Neural Network Misfire	0	1	0	1	1	0

Complete results for all parts can be found in Appendix ii – Hardware Test Best Part ID minus Misfire Results.

6.3.2.3. Summary

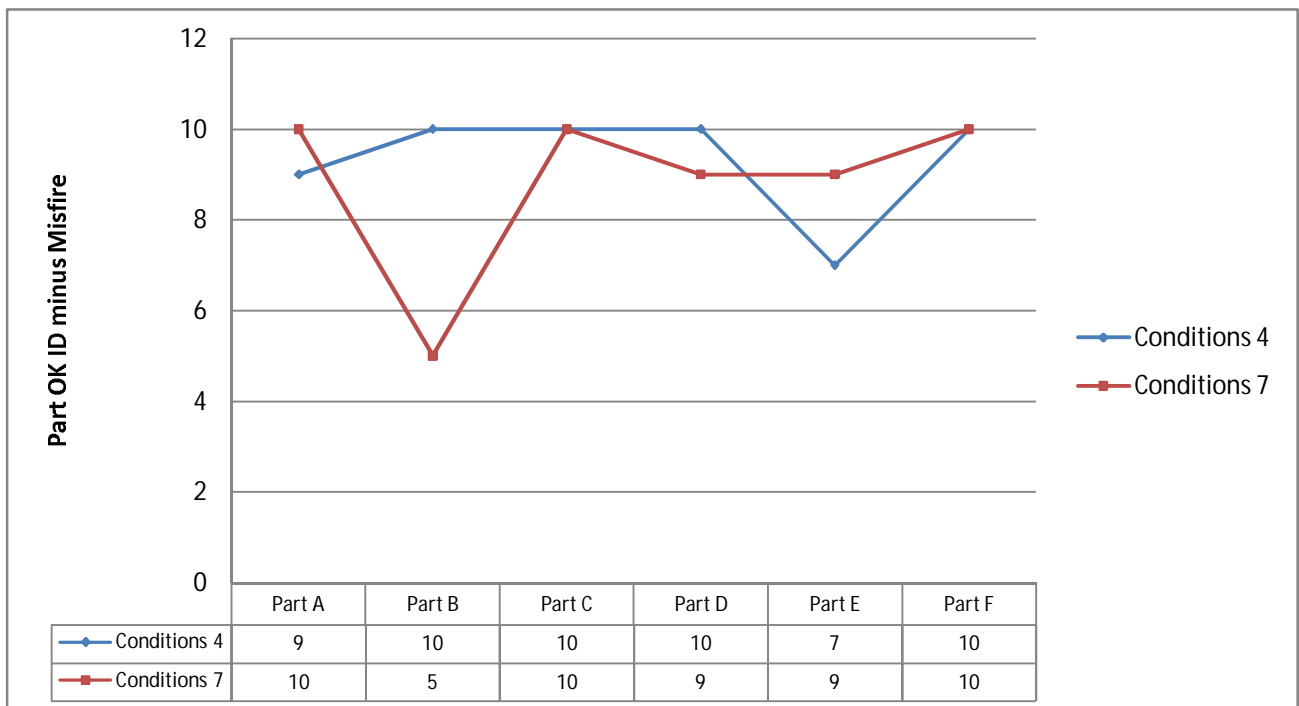
As previously discussed in Section 6.3.1 *PLC Simulator*, the ANN misfire count has to be subtracted from the part OK identification to present the results as accurately as possible.

The results from the two hardware tests are charted below in *Graph 12 (page 116)*. In order to give a better graphical representation of each Conditions results, the linear trend for each was added in *Graph 13 (page 117)*. It is clear that Conditions 4 resulted in more accurate part identification.

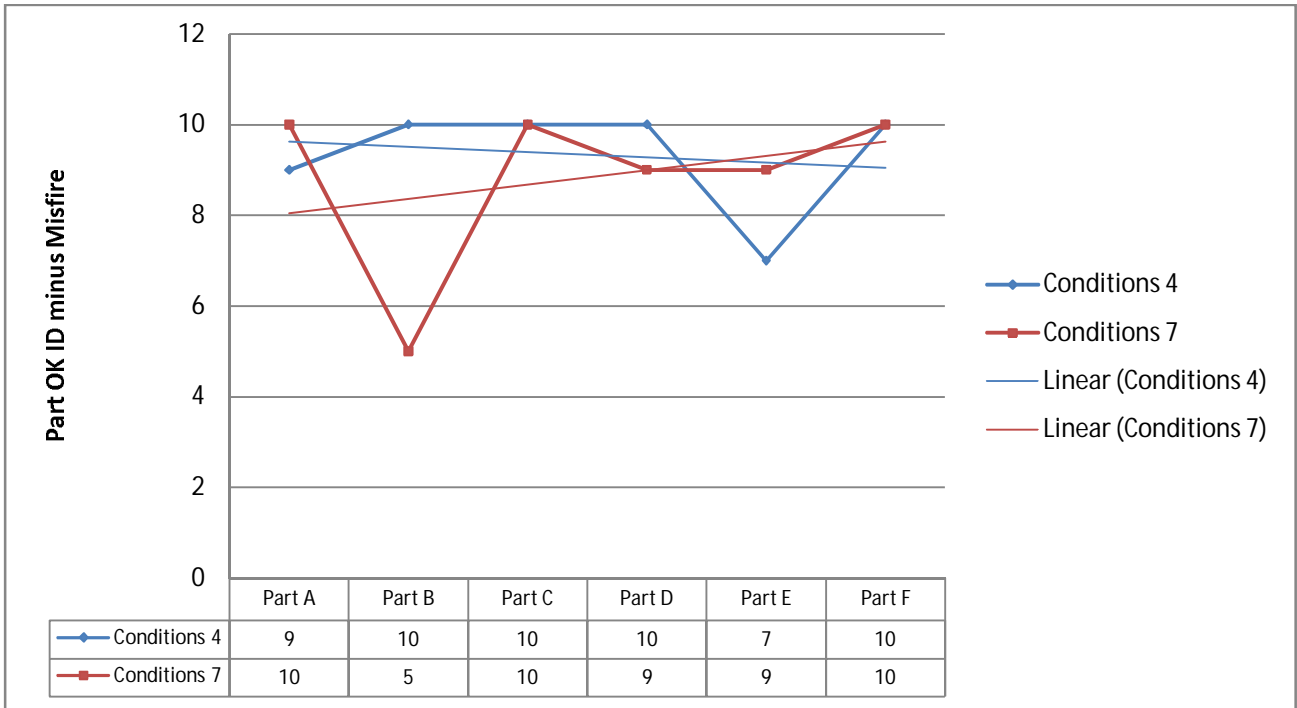
As each of the 6 parts was run through the testing procedure 10 times, there was a possibility of a total of 60 OK Part Identifications (60 = 10 test runs x 6 parts), 60 NOK Part Identifications (60 = 10 test runs x 6 parts) and 300 ANN Misfires (300 = 10 test runs x 6 parts x 5 misfires).

The weights optimized by the GA under the control parameters set for test Conditions 4 yielded the following when utilized for ANN based automated part recognition:

- 60 instances of OK part identifications. This can be translated into 100% accuracy in part identification.
- 0 instances of NOK Part Identification. This can be translated into 100% accuracy in part identification.
- 4 instances of ANN Misfire. This can be translated into 1.333% neuron misfires.



Graph 12 - Hardware Test Results



Graph 13 - Hardware Test Results including Linear Trend

7. Chapter 7 - Conclusion

In this research an understanding of current part recognition methods, GAs, ANNs and their various training methods was gained by literature study, software development and supporting experimental research.

The software and architecture of an existing ANN was adapted to accommodate the new GA based training method and a larger part recognition capacity. A GA was designed and developed to train the before mentioned ANN in order to successfully recognize industrial parts on the experimental test setup.

Supporting experimental test confirmed that the GA successfully trained the ANN to recognize the trained parts whilst not over fitting the ANN to allow generalization in part recognition.

A detailed analysis was performed on the influence the various GA control parameters have on both the performance of the GA and on the intelligent automated part recognition ability of the ANN.

This research identified the ideal control parameters for a GA in order to train an ANN for part recognition.

By developing and implementing an intelligent evolutionary GA based training methodology the original ANN has been proved to be significantly improved when looking at the following factors:

- The need for trail-and-error based hard coded connecting weight has been eliminated.
- The need for trail-and-error based hard coded fuzzy logic has been eliminated.
- The accuracy of the ANN output when compared to the target output for each part has been greatly improved.
- The classification, or identification, ability has been significantly improved. Previously only three predefined parts could be identified. The GA based training method allows any six parts, and possibly more, to be trained for recognition.

The developed research platform, especially the GA software, can be applied to similar optimization applications by only creating a new fitness function.

Further future development and expansion could include the following:

- Increase of the dimensional data capturing rate, although this will require a more powerful processor than currently in use.
- Addition of more dimensional sensors to the experimental test setup.
- Implementation of a higher level GA to further optimize the ideal GA control parameters.
- Optimization of the developed ANN architecture by utilizing GP principles.

Bibliography

- Andries, P. E. (2007). *Computational Intelligence: An Introduction*. Pretoria: John Wiley & Sons, Ltd.
- Batchelor, B. G., & Whelan, P. F. (2002). *Intelligent Vision Systems for Industry*. Springer-Verlag.
- Can, B., & Harvey, C. (2012). A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. *Computers & Operations Research* , 424-436.
- Casasent, D., & Psaltis, D. (1976). Position, rotation, and scale invariant optical correlation. *Applied Optics* , 1795-1799.
- Chen, Y., Wu, P., & Wu, Q. (2009). Higher Order Neural Networks for Stock Index Modeling. *Artificial Higher Order Neural Networks for Economics and Business* , 113-131.
- Darwin, C. (1859). *On the Origins of Species*. London: John Murray.
- Das, S., & Saha, B. (2009). Data Quality Mining using Genetic Algorithm. *International Journal of Computer Science and Security* , 105-112.
- DeCoster, J. (1998). *Intoductory Statistics Notes*. Retrieved December 20, 2011, from Intoductory Statistics Notes: <http://www.stat-help.com/notes.html>
- Demutgal, M., Unal, M., Tansel, I., & Yazicioglu, O. (2011). Fault diagnosis on bottle filling plant using genetic-based neural network. *Advances in Engineering Software* , 1051-1058.
- Dorsey, R., Johnson, J., & Mayer, W. (1994). A genetic algorithm for the training f feedforward neural networks. *Advances in artificial intelligence in economics, finance and management* , 93-111.
- FESTO AG & Co. (2002, October 8). 165327_Fibre_Optic_Device_SOEG_L.pdf. *FESTO SOEG-L-Q30-P-A-S-2L* . Esslingen, Germany.
- FESTO AG & Co. (2007, July). pdf_en_soex_en.pdf. *Products 2007 - Opto-electronic sensors* . Esslingen, Germany.
- Gao, X., & Ovaska, S. (2002). Genetic Algorithm Training of Elman Neural Newtork in Motor Fault Detection. *Neural Computing and Applications* , 37-44.
- Gupta, J., & Sexton, R. (1999). Comparing backpropagation with a genetic algorithm for neural network training. *Omega: The International Journal of Management Science* , 679-684.
- Haupt, R. L. (2004). *Practical Genetic Algorithms: Second Edition*. Hoboken, New Jersey: Wiley-Interscience.
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Pearson Education, Inc.
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Patparganj: Pearson Education Hall.
- Hsu, Y., Arsenault, H. H., & April, G. (1982). Rotation-invariant digital pattern recognition using circular harmonic expansion. *Applied Optics* , 4012-4015.

- Hui, L. C., Lam, K.-Y., & Chea, C. W. (1997). Global Optimization in Neural Network Training. *Neural Computing and Applications*, 58-64.
- Khosravi, A., Mazloumi, E., Nahavandi, S., Creighton, D., & Van Lint, J. (2011). A genetic algorithm-based method for improving quality of travel time prediction intervals. *Transportation Research Part C*, 1364-1376.
- Knowles, A., Hussain, A., Deredy, W. E., & Lisboa, P. G. (2009). Higher Order Neural Networks with Bayesian Confidence Measure for the Prediction of the EUR/USD Exchange Rate. *Artificial Higher Order Neural Networks for Economics and Business*, 48-59.
- Koza, J. R., & Rice, J. P. *GENETIC GENERATION OF BOTH THE WEIGHTS AND*. Stanford.
- Lee, K.-H. (2011). Exploring generation of a genetic robot's personality through neural and evolutionary means. *Data & Knowledge Engineering*, 923-954.
- Li, Y., & Häußler, A. (1996). Artificial evolution of neural networks and its application to feedback control. *Artificial Intelligence in Engineering*, 143-152.
- Liatsis, P., Hussain, A., & Milonidis, E. (2009). Artificial Higher Order Pipeline Recurrent Neural Networks for Financial Time Series Prediction. *Artificial Higher Order Neural Networks for Economics and Business*, 164-189.
- Melanie, M. (1999). *An Introduction to Genetic Algorithms*. Cambridge, Massachusetts: A Bradford Book The MIT Press.
- Michalewicz, Z., & Fogel, D. B. (2000). *How to Solve It: Modern Heuristics*. New York: Springer.
- Montana, D. J., & Davis, L. (1989). Training Feedforward Neural Networks Using Genetic Algorithms. *Machine Learning*, 762 - 767.
- Nelson Mandela Metropolitan University. (2009, July 9). *Nelson Mandela Metropolitan University*. Retrieved December 08, 2011, from Volkswagen South Africa - DAAD International Chair in Automotive Engineering: <http://www.nmmu.ac.za/vwsadaad>
- Paretto, P. (1994). *An Introduction to the Modeling of Neural Networks*. Cambridge: Cambridge University Press.
- Perelmuter, G., Carrera, E. V., Vellasco, M., & Pacheco, M. (1995). Recognition of Industrial Parts Using Artificial Neural Networks. *Proceedings of the 5th International Conference on Education, Practice, and Promotion of Computational Methods in Engineering Using Small Computers*, (pp. 481-486). Macau, China.
- Shi, D., Tan, S., & Ge, S. S. (2009). Automatically Identifying Predictor Variables for Stock Return Prediction. *Artificial Higher Order Neural Networks for Economics and Business*, 60-78.
- SICK AG. (2008, February 4). TI_DT20Hi_en.indd.pdf. *DT20 Hi Distance Sensor*. Waldkirch, Germany.
- Siemens AG. (2011, June). Instruction list S7-300 CPUs and ET200 CPUs. *6ES7398-8FA10-8BA0*. NÜRNBERG, Germany.

Siemens AG. (2011, March). S7-300 Automation System Module Data. 6ES7398-8FA10-8BA0 . NÜRNBERG, Germany.

Siemens AG. (2011). Simatic IPC Brochure. *brochure_simatic_industrial_pc_en.pdf* .

Siemens AG. (2009). *SIMATIC S7: Programming 1*.

Siemens AG. (2009). *SIMATIC S7: Programming 2*.

Singh, G., & Srivastava, L. (2011). Genetic Algorithm-Based Artificial Neural Network for Voltage Stability Assessment. *Advances in Artificial Neural Systems* , 1-9.

Toshiba Corporation. (2011). *Satellite Pro L650-1C0*. Retrieved December 11, 2011, from Toshiba Computers Europe: <http://za.computers.toshiba-europe.com/innovation/product/Satellite-Pro-L650-1C0/1091605/printFriendly/true/toshibaShop/false/>

Venables, A., & Tan, G. (2007). A 'Hands on' Strategy for Teaching Genetic . *Journal of Information Technology Education* .

Zhang, M. (2009). *Artificial Higher Order Neural Networks for Economics an Business*. Hershey: Information Science Reference.

Zhang, M. (2009). Ultra High Frequency Trigonometric Higher Order Neural Networks for Time Series Data Analysis. *Artificial Higher Order Neural Networks for Economics and Business* , 133-163.

Appendices

i. Hardware Test Best Part ID Results

Table 29 - Part Recognition Test 1 Part B

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	80182.1	100543	80103.35	99861.3	100691	100452.3	100296.9	79913.08	100416.7	100242.9
Part A: Y Neuron Result	35908.97	43100.77	37622.2	38511.87	39441.76	41425.79	36950.44	35881.63	43838.54	42111.94
Part A: Final Neuron Result	47794.42	59916.59	47760.59	59478.09	59976.84	59850.22	59724.57	47634.76	59847.24	59731.29
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	148804.6	147488.1	148971.3	147438	148064.5	147642.9	147773.7	148827.7	147648.9	147646.6
Part B: Y Neuron Result	19248.3	20166.55	19726.17	19815.07	20577.73	20338.48	20341.73	18794.13	20600.94	20413.13
Part B: Final Neuron Result	19390.84	19760.75	19639.54	19582.61	20003.29	19856.47	19866.72	19166.45	19987.42	19893.85
Part B: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part C: X Neuron Result	117888.6	119350.2	117851.5	119299.7	119565.6	119338.4	118742.2	117821.5	119330.2	119305.9
Part C: Y Neuron Result	26770.42	27461.93	28034.08	26713.14	27937.07	27342.82	27574.94	27039.61	27870.21	27754.58
Part C: Final Neuron Result	16640.72	16960.04	17035.25	16721.2	17124.4	16921.79	16953.13	16720.64	17086.95	17048.92
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	105192.6	111371.2	104835.6	111371.3	111442.4	111670.7	111381	104750.7	111409.2	111345
Part D: Y Neuron Result	19440.67	22988.22	20513.16	20832.81	21428.64	22121.66	20586.36	20204.38	23381.57	22616.26
Part D: Final Neuron Result	24548.14	26283.77	24603.86	26020.59	26108.34	26241.08	25992.54	24548.25	26339.81	26232.82
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	99712.19	114479.7	99774.9	114226.3	114565.4	114681.6	115080.3	99458.79	114424.1	114284.8
Part E: Y Neuron Result	12738.94	16248.73	12886.15	13999.17	14645.54	15255.22	13710.83	12171.63	16228.8	15594.98
Part E: Final Neuron Result	26088.02	29986.05	26107.37	29872.82	29974.29	30017.28	30087.84	26010.39	29971.24	29921.75
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	85828.95	98034.38	86005.51	97654.31	98064.64	98083.55	98860.17	85748.45	98087.97	97978.03
Part F: Y Neuron Result	23029.55	23261.65	24573.54	22676.33	23378.86	23385.08	22673.69	23798.66	23889.32	23497.86
Part F: Final Neuron Result	14577.99	16525.2	14669.88	16440.65	16534.85	16538.11	16631.97	14597.02	16559.67	16526.03
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 30 - Part Recognition Test 1 Part C

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	103659.3	102754.4	102103.9	107117.3	103174.3	102707	102866.5	103875.9	103970.3	102793.5
Part A: Y Neuron Result	55552.56	56136.74	55950.01	55245.09	56043.89	56409.68	55952.59	56081.75	56097.47	56088.85
Part A: Final Neuron Result	61856.98	61325.06	60938.09	63904.32	61573.25	61298.97	61390.09	61989.36	62045.41	61347.85
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	145274.9	143785	142937.3	149090	144640.7	143622	144195.9	144144.5	144572.2	143755.2
Part B: Y Neuron Result	27233.46	27558.78	27831.94	26724.13	27429.65	27566.55	27557.53	27249.07	27123.48	27567.11
Part B: Final Neuron Result	23129.91	23193.44	23273.39	23128.24	23185.66	23186.55	23219.92	23063.1	23028.84	23195.62
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	119798.6	118631.2	118407.8	123021	119285.3	118626.9	119042.8	119189.8	119066.5	119179.3
Part C: Y Neuron Result	36958	37327.76	37397.75	36564.36	37215.97	37354.32	37294.43	37079.42	36981.54	37123.65
Part C: Final Neuron Result	19975.6	20010.33	20016.74	20076.8	20020.85	20018.38	20028.59	19971.27	19931.91	19984.44
Part C: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part D: X Neuron Result	142786.3	141743	141490.5	146774.9	141968.5	141640.7	141900.9	142681.9	142186.6	142593.3
Part D: Y Neuron Result	29242.64	29676.74	29561.18	29174.22	29480.37	29762.82	29613.01	29562.6	29471.45	29607.74
Part D: Final Neuron Result	33669.69	33502.77	33435.43	34502.13	33526.34	33491.73	33528.28	33686.75	33571.23	33673.6
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	147496.3	146339.1	145926.3	151660.8	146886.6	146255.3	146589.8	147121.1	147034.1	147105.1
Part E: Y Neuron Result	18320.31	18542.91	18332.04	18299.68	18528.66	18601.14	18477.56	18444.62	18330.47	18368.51
Part E: Final Neuron Result	38578.84	38283.92	38172.57	39656.71	38425.4	38263.45	38347.47	38484.33	38459.38	38478.57
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	105847.2	104775.4	104271.8	108685.1	105484.8	104611.8	105135.6	105016.6	105469.4	104848.2
Part F: Y Neuron Result	36895.11	37403.91	37678.86	36320.47	37268.47	37562.43	37350.37	37134.66	37034.95	37435.44
Part F: Final Neuron Result	18329.44	18180.33	18111.77	18756.19	18287.35	18160.92	18235.31	18207.48	18275.24	18193.2
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 31 - Part Recognition Test 1 Part D

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	37624.96	39595.5	52727.36	39762.91	51532.17	53098.06	53425.2	33661.84	39391.61	51410.19
Part A: Y Neuron Result	162459.7	169617.1	169538.8	171839.2	166982.8	170896.6	166923.9	170242	167832.6	154560.6
Part A: Final Neuron Result	23519.16	24740.8	32523.66	24856.69	31796.09	32753.56	32917.68	21228.51	24606.57	31630.63
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	81510.3	81389.77	85638.41	82410.3	84388.13	86259.07	86965.93	79789.68	80216.18	84091.88
Part B: Y Neuron Result	62908.14	59531.49	59424.02	57509.49	59598.79	57999.6	60361.93	58202.22	59902.16	57404.53
Part B: Final Neuron Result	36668.45	34980.91	35207.73	34042.46	35212.18	34540.15	35761.83	34214.16	35087.87	34101.19
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	52262.26	51865.8	52450.71	52405.36	51826.18	52765.92	53447.68	51407.87	51325.11	52430.64
Part C: Y Neuron Result	90110.62	84533.79	85667.76	83691.36	82893.79	84910.81	92386.91	82745.13	84079.89	81510.3
Part C: Final Neuron Result	31965.97	30185.71	30582.9	29958.62	29667.55	30367.02	32764.06	29591.64	30005.33	29274.96
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	49860.21	49607.17	49612.96	49937.46	49338.18	49863.94	49975.52	49848.73	49186.21	49075.11
Part D: Y Neuron Result	71090.04	71532.8	73495.17	75058.43	65899.73	75786.15	77855.59	69945.9	67481.96	64727.52
Part D: Final Neuron Result	19191.42	19192.14	19433	19692.29	18447.56	19765.66	20041.89	19049.28	18608.74	18248.97
Part D: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part E: X Neuron Result	65209.7	65088.1	66934.5	65725.3	66132.95	67173.02	67374.2	64524.22	64412.22	65536.38
Part E: Y Neuron Result	38826.14	40658.96	40132.41	40671.98	40865.07	40137.92	39861	41127.27	41270.11	36310.59
Part E: Final Neuron Result	17706.57	17713.88	18180.82	17879.15	17988.79	18242.7	18288.93	17577.79	17551.82	17737.91
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	63510.44	63525.43	67016.3	64338.38	65809.43	67486.32	67751.81	62304.58	62643.07	65095.38
Part F: Y Neuron Result	115155.9	112249.3	112794.6	110626.3	111488.3	111688.1	115083.9	110125.3	111927.3	107277.1
Part F: Final Neuron Result	14845.73	14727.88	15304.61	14789.8	15058.99	15333.46	15516.07	14446.21	14574.49	14771.43
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 32 - Part Recognition Test 1 Part E

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	65659.66	62277.44	65418.19	65990.1	60777.58	65504.54	65236.84	65891.98	65182.48	65850.04
Part A: Y Neuron Result	155839	146767.5	150622.6	155964.7	152189.1	151222.3	157144.7	156424.3	163373.6	156897.5
Part A: Final Neuron Result	40086.09	38013.38	39903.86	40282.89	37165.04	39959.53	39845.28	40228.19	39859.77	40206.88
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	107934	102465.4	108234.9	108393.1	99793.85	108241.6	108144.5	109003.7	107789.7	108710.7
Part B: Y Neuron Result	70906.21	68554.83	68833.23	66665.45	68880.45	68215.38	69834.27	68425.67	69903.27	72073.73
Part B: Final Neuron Result	42389.94	40859.57	41378.66	40310.82	40845.3	41071.77	41870.63	41226.65	41881.55	43021.91
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	88960.55	70909.67	88337.7	89199.87	63413.21	88738.23	88724.3	89570.37	88500.69	89520.57
Part C: Y Neuron Result	106002	99924.14	101438.8	99726.59	98802.63	100255.6	103892.5	100961.9	102688.9	106962.9
Part C: Final Neuron Result	39521.33	36351.46	38043.8	37565.9	35475.81	37699.94	38841.89	37979.96	38448.04	39862.38
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	55336.23	54171.98	55586.67	55476.79	53544.88	55593.82	55584.86	55734.63	55461.86	55676.16
Part D: Y Neuron Result	78619.51	71733.61	73734.83	77600.88	73736.06	77705.23	80039.35	76943.41	79883.77	78635.15
Part D: Final Neuron Result	21265.19	20178.91	20721.5	21170.43	20291.24	21207.84	21490.98	21144.5	21446.06	21338.76
Part D: Activation Function	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	73667.61	74018.9	73952.09	73832.54	73929.01	73865.3	73818.5	74250.07	73739.27	73843.29
Part E: Y Neuron Result	41841.81	41513.49	40374.12	40684	40972.73	40075	41386.86	41138.43	42514.14	42073.98
Part E: Final Neuron Result	19960.41	20044.41	20003	19978.6	20009.69	19974.19	19989.84	20096.33	19993.19	20010.81
Part E: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part F: X Neuron Result	77881.88	77360.88	78195.51	78154.3	76909.19	78039.22	77961.06	78591.38	77859.91	78140.45
Part F: Y Neuron Result	119263.7	114890.6	115622.9	113617.4	116121.3	115259.6	118378.8	116217	121058.8	120216.9
Part F: Final Neuron Result	17297.12	17033.52	17196.31	17106.81	17012.72	17156.47	17273.09	17283.73	17367.89	17377.6
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 33 - Part Recognition Test 1 Part F

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	108778.1	109669.3	109254.6	109285.3	109485.4	109310.9	109613.3	109308.7	109684.4	109044.9
Part A: Y Neuron Result	118789.2	129954	116111.8	116263.2	121226.6	117699.3	128877.9	122178.6	121102.8	109509.2
Part A: Final Neuron Result	65365.2	65977.16	65627.55	65646.87	65802.7	65672.84	65935.91	65705.12	65919.73	65453.78
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	155128.5	154650.9	156110.2	155021.6	154105.7	155793.7	154840.6	155101.8	155108.3	155671.3
Part B: Y Neuron Result	55354.29	40421.11	49499.06	40751.71	40162.02	46961.11	39519.27	43643.23	40584.77	45240.47
Part B: Final Neuron Result	37767.59	30308.14	34919.89	30497.04	30143.3	33636.6	29872.06	31940.6	30419.71	32772.66
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	127511	128243.1	128315.8	127884.9	128208.8	128245	128043.8	127885.3	128235.5	128061.8
Part C: Y Neuron Result	61656.41	45954.25	56455.4	47786.89	39852.31	53931.53	45167.8	50645.2	47246.93	51358.83
Part C: Final Neuron Result	28275.68	23392.16	26697.36	23943.09	21472.15	25899.25	23131.1	24841.38	23797.87	25077.96
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	145706.8	147278.9	146324.3	146209.5	147123.7	145970	146537.2	146348	146955.3	145252.4
Part D: Y Neuron Result	62593.91	52648.84	59108.11	53544.06	43205.19	57807.84	52241.19	56164.42	52884.57	55048.41
Part D: Final Neuron Result	38357.97	37474.94	38062.48	37358.83	36289.02	37829.02	37268.8	37708.02	37435.5	37340.78
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	151425.3	152568.9	151984.7	151892.6	152393	151499.5	152085.1	151818	152281.8	151091
Part E: Y Neuron Result	52750.67	40131.5	50031.69	44303.8	27811.04	48686.35	40569.74	46201.28	42534.89	46771.81
Part E: Final Neuron Result	40325.04	40354.01	40412.31	40267.21	40047.65	40258.22	40238.02	40288.08	40330.54	40111.91
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	111736.7	111475.1	112316.4	111711.9	111157.4	111806.7	111598.8	111830.3	111678.8	111830.9
Part F: Y Neuron Result	54868.23	55925.18	54346.93	54586.51	58973.73	55476.16	54646.39	55330.52	55598.78	52801.09
Part F: Final Neuron Result	20007.85	20010.05	20078.32	19992.27	20085.71	20044.12	19976.78	20041.84	20028.89	19937.3
Part F: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

ii. Hardware Test Part ID minus Misfire Results

Table 34 - Part Recognition Test 2 Part A

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	109862.6	110427.5	111777.3	110976.1	112369	109861.3	110199.2	111800.2	117837.4	113881
Part A: Y Neuron Result	47148.39	45960.88	39332.91	45787.54	38102.21	48146.96	48733.67	40086.12	36845.1	33979.32
Part A: Final Neuron Result	19903.11	19956.84	19956.36	20042.34	20013.08	19936.85	20013.09	19985.79	20881.25	20124.78
Part A: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part B: X Neuron Result	49433.3	49540.89	52402.45	51276.59	52700.05	49431.31	50747.97	52565.31	52048.23	51943.33
Part B: Y Neuron Result	72686.34	70390.82	72980.09	83615.46	68409.72	73848.79	82489.18	71112.24	64457.09	68058.49
Part B: Final Neuron Result	15387.25	15129.14	15827.54	16937.74	15325.03	15525.13	16731.72	15627.8	14766.3	15179.97
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	32821	32934.35	39480.59	37139.89	40064.82	32820.41	36322.16	39884.2	34256.29	36958.76
Part C: Y Neuron Result	49747.18	48684.9	53100.61	54218.06	51021.04	50241.48	52104.59	54802.68	44394.04	50001.21
Part C: Final Neuron Result	13510.39	13325.75	14988.34	14899.24	14673.08	13602.96	14398.81	15358.84	12690.11	14085.76
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	71356.96	71559.87	73002.32	72332.98	73281.91	71351.88	71914.79	73068.06	74884.63	73527.29
Part D: Y Neuron Result	52235.63	50118.38	51964.04	55843.64	51336.66	52079.01	54520.14	53805.26	44746.59	49227.85
Part D: Final Neuron Result	12565.19	12331.57	12679.45	13081.03	12629.77	12546.12	12886.45	12904.18	11989.12	12400.94
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	68182.42	68416.91	75375.32	72769.23	76089.05	68182.27	71688.35	75786.09	72230.27	73529.56
Part E: Y Neuron Result	28287.59	28105.08	27834.82	30177.83	26047.8	28852.91	30470.8	28115.62	24437.57	25329.27
Part E: Final Neuron Result	13970.98	14008.11	15310.78	14910.35	15375.77	13993	14717.79	15399.25	14584.72	14864.71
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	78097.3	78357.84	87051.84	83809.52	87916.98	78091.34	82528.77	87570.08	82598.49	84561.62
Part F: Y Neuron Result	40501.72	38827.16	39306.85	44099.61	38506.23	40122.41	43840.11	39548.23	35002.56	36395.71
Part F: Final Neuron Result	9395.025	9375.771	10309	10099.97	10378.15	9383.789	9957.208	10370.58	9717.5	9964.149
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 35 - Part Recognition Test 2 Part B

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	156449.5	155310.3	156631.7	156226.9	158579.2	154936.1	155060.6	156147.3	156611.9	156005
Part A: Y Neuron Result	36811.88	38363.56	39586.91	39292.84	38175.79	39644.72	38069.95	39379.59	38479.12	38986.17
Part A: Final Neuron Result	27311.89	27174.87	27436.57	27359.15	27713.01	27156.09	27123.29	27348.84	27395.62	27311.76
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	105374.7	105096.5	98664.96	98017.84	106502.9	104669.1	105004.7	98170.02	105323.8	97936.01
Part B: Y Neuron Result	43029.09	43958.63	44799.94	43711.48	44600.52	44801.72	44261.38	44070.03	44798.22	43300.21
Part B: Final Neuron Result	19500.51	19572.99	18794.87	18577.17	19841.29	19614.84	19596.43	18640.55	19703.81	18517.12
Part B: Activation Function	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
Part C: X Neuron Result	77595.23	77358.9	71457.86	70951.28	78467.45	77062.23	77289.23	71088.43	77584.84	70885.99
Part C: Y Neuron Result	38975.54	39033.51	39723.05	38132	40009.63	39193.6	39073.98	38089.88	40184	37734.99
Part C: Final Neuron Result	17202.24	17182.96	16559.54	16196.72	17507.31	17175.13	17181.66	16206.32	17427.42	16113.98
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	150426	150361	126930.5	126292.8	152067.6	149747.4	150047.8	126872.4	150364.7	126322.4
Part D: Y Neuron Result	45585.13	45122.24	45548.87	42442.6	46625.26	44519.48	46211.97	42457.18	46589.08	41853.97
Part D: Final Neuron Result	18817.1	18756.29	16720.09	16294.04	19086.96	18629.98	18857.94	16347.39	18930.99	16226.7
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	126837.1	126367.6	112366.6	111907.3	128555.3	125877.5	126178.3	112209.6	126914.9	111852.3
Part E: Y Neuron Result	21303.31	21147.66	21297.8	19080.14	21647.07	20954.78	21414.64	19459.98	21753.53	19112.67
Part E: Final Neuron Result	24767.97	24673.29	22036.86	21863.66	25105.64	24573.28	24647.98	21935.54	24800.21	21854.55
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	138742.2	138369.7	120330.8	119742.3	140714.2	137693.2	138135.6	120218.6	138984.7	119814.4
Part F: Y Neuron Result	31905.28	34022.56	35323.31	36860.6	33475.25	35925.4	33528.07	36724.61	33682.52	36515.59
Part F: Final Neuron Result	15570.88	15590.67	13718.52	13699.25	15823.42	15572.3	15552.07	13745.84	15646.23	13697.23
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 36 - Part Recognition Test 2 Part C

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	164824.7	164051.2	164300.2	165212.1	165224.8	164117.8	164344.3	163964.5	165290	164422.1
Part A: Y Neuron Result	45599.2	46051.88	45808.84	45562.12	46496.46	46422.19	45960.34	46738.51	46591.58	46443.63
Part A: Final Neuron Result	29005.71	28892.26	28925.48	29068.98	29102.87	28915.94	28937.96	28901.16	29116.96	28967.35
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	103516.8	103058.8	103127.5	103789.9	104157.3	103178.3	103379	103157.4	103761.1	103407.2
Part B: Y Neuron Result	58763.45	59667.59	59345.71	58736.19	60121.41	60204.09	59340.78	60501.88	60027	60115.99
Part B: Final Neuron Result	21116.73	21161.65	21132.77	21150.78	21365.55	21241.71	21166.52	21274.25	21300.24	21262.49
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	81547.02	81232.41	81324.49	81715.27	82245.16	81557.58	81485.57	81440.34	81861.17	81686.45
Part C: Y Neuron Result	50881.4	51130.12	51035.55	50852.77	50999.29	51343.39	51071.74	51333.91	51093.6	51187.45
Part C: Final Neuron Result	19937.81	19944.3	19938.32	19953.9	20048.95	20025.75	19965.65	20009.02	20017.65	20012.96
Part C: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part D: X Neuron Result	161290.7	160391.5	160487.1	161476.2	162286.8	160950.7	160847.9	160620.7	161329.5	161060.5
Part D: Y Neuron Result	57067.54	57867.24	57377.94	56985.14	58321.26	58481.81	57743.22	58831.71	58689.95	58462.66
Part D: Final Neuron Result	21149.78	21164.76	21115.1	21156.51	21387.54	21287.63	21190.66	21299.83	21346.11	21295.12
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	130053.1	129450	129538.8	130438	130344.8	129403.8	129609.3	129295.5	130185	129698.1
Part E: Y Neuron Result	24101.02	24204.56	24179.6	24098.86	24247.36	24253.1	24151.37	24296.53	24123.77	24171.08
Part E: Final Neuron Result	25484.03	25374.25	25390.04	25556.59	25544.79	25367.43	25402.24	25348.69	25509.81	25419.76
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	154288.3	153303.2	153354.1	154527	154480.9	153367.4	153564.5	153170.1	154041.5	153600.8
Part F: Y Neuron Result	42361.02	43081.35	42699.59	42289.26	43515.29	43504.54	42909.58	43816.82	43716.97	43551.82
Part F: Final Neuron Result	17507.98	17423.89	17418.61	17531.23	17560.63	17442.52	17446.75	17430.38	17519.78	17468.54
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 37 - Part Recognition Test 2 Part D

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	112426.3	112336.5	117288.2	113820	111263.6	114445.5	117396.7	110150.8	117645.1	108531.5
Part A: Y Neuron Result	85587.88	81893.25	90665.14	86886.91	84051.45	95840.48	99639.77	88298.88	92652.84	80819.48
Part A: Final Neuron Result	21636.96	21496.4	22619.42	21913.28	21391.04	22321.85	22942.6	21350.08	22746.45	20826.07
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
Part B: X Neuron Result	53281.53	53683.93	54539	53497.82	53200.33	53454.5	54382.18	51721.67	54685.22	52184.41
Part B: Y Neuron Result	138917.1	124789.6	142021.1	135141.3	124195.7	149599.7	150292.9	145766	142153.2	120268.9
Part B: Final Neuron Result	23783.54	22159.57	24324.1	23364.35	22022.96	25076.68	25285.72	24384.5	24359.77	21417.59
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	41117.05	41257.31	42252.09	41282.78	41122.73	41522.05	42198.35	40427	42269.61	40122.33
Part C: Y Neuron Result	118338.3	110852.6	118569	117109	113522.6	122466.6	119201.5	111774.9	118828.4	106021.8
Part C: Final Neuron Result	27424.66	26039.5	27612.69	27215.39	26522.79	28250.1	27724.38	26106.46	27663.54	24989.3
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	75840.82	76309.95	77332.81	76018.13	75723.84	75928.8	77203.08	73719.03	77525.63	74357.7
Part D: Y Neuron Result	110951.6	109647.9	110296.5	111700	110096.3	112701.4	109549.7	106463.8	109951.8	104326.8
Part D: Final Neuron Result	19944.5	19831.29	19999.51	20049.25	19832.4	20160.34	19899.18	19222.02	19975.7	19024.86
Part D: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part E: X Neuron Result	80454.58	79185.11	86091.09	82283.77	78643.06	84141	86030.63	81030.66	86261.52	76655.9
Part E: Y Neuron Result	57188.59	52953.17	56454.18	55752.28	53370.59	57956.64	56506	56055.02	56279.64	52086.38
Part E: Final Neuron Result	17414.39	17009.59	18449.47	17703.57	16923.58	18140.05	18440.08	17478.89	18474.83	16498.46
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	85768.77	83391.38	93799.81	88932.44	82814.72	91755.8	94011.48	88436.92	93996.93	80997.28
Part F: Y Neuron Result	69937.07	67402.48	77236.85	72381.1	68611.87	84189.3	89355.13	77557.47	79501.99	64920.63
Part F: Final Neuron Result	11029.63	10707.24	12083.4	11432.68	10680.04	12061.53	12444.6	11524.97	12167.59	10384.56
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 38 - Part Recognition Test 2 Part E

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	130023.6	128214.7	130852.8	130574.1	130401.2	130795.3	131003.3	136598.2	130906.3	128939.4
Part A: Y Neuron Result	95504.05	87157.53	89633.15	91013.7	92491.02	93154.27	92020.69	92161.15	93287.35	85561.61
Part A: Final Neuron Result	24905.32	24320.27	24843.85	24844.36	24865.79	24953.99	24950.1	23886.83	24976.99	24386.71
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	77997.27	61872.74	78672.06	78116.83	78010.2	78236.55	78197.91	81915.7	78229.71	62985.78
Part B: Y Neuron Result	146764.6	143521.3	141893.3	140568.4	145205.8	148227.3	142298.2	140588.6	144952.4	141039.5
Part B: Final Neuron Result	28090.61	25503.68	27603.83	27370.58	27907.13	28297.1	27587.21	27891.63	27906.97	25360.7
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	56231.03	47176.41	56634.95	56397.07	56446.84	56393.33	56489.17	59068.58	56528.54	47836.5
Part C: Y Neuron Result	125512.7	117927	119376.2	121601.1	125736.3	124972.3	122849.6	118003.1	125950.8	120596
Part C: Final Neuron Result	30697.11	28120.43	29598.46	29985.13	30766.56	30616.53	30230.88	29651.5	30817.18	28704.88
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	103527.5	86039.35	104307.3	103671.5	103568.4	103835.7	103845.8	108231.3	103842.9	87281.76
Part D: Y Neuron Result	117139.3	110573.4	111044	115244	118686	115568.9	115428.4	108175.8	117343.2	114258.5
Part D: Final Neuron Result	23146.05	20807.89	22490.92	22933.57	23333.57	22986.83	22971.02	22499.46	23198.38	21356.63
Part D: Activation Function	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	94189.44	94011.02	94449.72	94451.29	94320.14	94446.88	94510.95	98753.97	94630.8	94344.41
Part E: Y Neuron Result	56433.62	56585.43	54644.39	53778.8	57020.39	57161.49	55431.65	55578.18	56569.49	55968.54
Part E: Final Neuron Result	19977.01	19949.26	19956.33	19922.86	20024.56	20053.99	19998.6	20805.07	20065.6	19988.11
Part E: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Part F: X Neuron Result	103419.9	102263.3	103743.2	103792.6	103615.4	103713.7	103777.1	108245.1	103848.7	102656.7
Part F: Y Neuron Result	90846.96	81417.63	85281.05	87206.13	87445.58	89652.91	87186.09	84331.27	88973.64	80073.03
Part F: Final Neuron Result	13481.72	13095.72	13360.31	13419.36	13407.31	13479.42	13417.16	13810.06	13474.71	13099.75
Part F: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Table 39 - Part Recognition Test 2 Part F

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Part A: X Neuron Result	176036.6	175695	178188	175762.3	175999.2	175903.4	178603.2	176310.2	177126.1	178814.3
Part A: Y Neuron Result	88876.57	96216.23	100800.1	97083.18	97817.02	96014.34	109636.6	95266.2	96824.31	92829.61
Part A: Final Neuron Result	32344.59	32537.2	33108.3	32577.88	32642.3	32565.06	33477.88	32607.38	32796.27	32941.67
Part A: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part B: X Neuron Result	110008.6	110019.4	111856.9	109797.4	110168.3	110064.3	111337.6	110410.9	111062.5	111869.2
Part B: Y Neuron Result	104162.6	101977.2	104407.8	112779.3	105864.1	103854.2	126329.6	103261	103721.5	99914.19
Part B: Final Neuron Result	27398.3	27140.07	27679.8	28393.48	27622.33	27369.27	30214.1	27346.09	27489.78	27147.45
Part B: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part C: X Neuron Result	86508.01	86937.53	88206.01	86358.21	86767.08	87024.9	87658.9	87008.03	87765.65	88428.71
Part C: Y Neuron Result	82470.5	85122.31	93940.34	90163.46	90718.66	86059.73	103506.7	86920	82452.86	80234.14
Part C: Final Neuron Result	26491.36	27043.18	28857.75	27914.16	28070.37	27230.02	30581.01	27389.12	26648.46	26317.18
Part C: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part D: X Neuron Result	168630.2	169357.1	171908.1	168379.9	169035.4	169543.9	170856.4	169613.6	171099.4	172382.6
Part D: Y Neuron Result	139373.7	148999.5	151474.4	152623	150930.3	149008.9	173190.5	147180.4	151796.5	145077.4
Part D: Final Neuron Result	31587.75	32796.78	33318.2	33140.49	32997.65	32814.53	35806.05	32603.37	33284.45	32600
Part D: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part E: X Neuron Result	138341.3	138578.8	140665.7	138171.3	138609.6	138732.2	140355.7	139003	139835.8	140953.9
Part E: Y Neuron Result	23504.61	21222.33	24142.87	32700.32	23902.03	21890.33	48856.02	21872.44	21855.07	21084.04
Part E: Final Neuron Result	27024.93	26980.73	27488.5	27351.57	27091.08	27035.73	28394.02	27086.14	27242.64	27423.57
Part E: Activation Function	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Part F: X Neuron Result	161952.8	162471.5	164671.8	161690.3	162096.5	162644.1	164274.8	162653.5	164004.8	165304.8
Part F: Y Neuron Result	101520.2	101976.3	101697.5	105278.9	104134.6	103376.1	109458.4	102408.5	103728.5	99904.83
Part F: Final Neuron Result	19972.88	20040.51	20265.5	20050.2	20061.17	20097.9	20440.49	20071.85	20251.72	20282.36
Part F: Activation Function	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

V. Training Data

Table 41 - Training Data Part A

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	10112	9600	10368	10368	9856	9600	10368	9600	9984	10240	4480	5632	8576	8448	5120	4224	9216	8320	11008	5632
Point X:2	8704	9984	9856	8704	9344	9600	8832	9344	8832	9856	4352	8064	8448	9600	5120	4096	9472	6784	9472	6400
Point X:3	11136	12544	11648	11392	14976	15488	11776	14720	13696	12032	4608	13952	10624	13312	11264	4864	11136	8704	9984	13952
Point X:4	15616	15488	17920	16384	17280	17152	16512	16768	17536	17920	12800	12672	14336	11264	15232	12672	14976	16512	15104	15872
Point X:5	11648	8448	10752	11264	6912	6144	13184	6784	8192	9472	10368	4992	8320	6016	6784	11136	9472	14976	17536	6400
Point X:6	6144	5120	5248	6016	4608	4352	5632	4480	4608	5376	5760	3456	5632	4480	4480	7552	6016	6272	6144	4352
Point X:7	4224	3712	3840	3840	3712	3712	3712	3712	3584	3584	3840	3456	3584	3584	3584	3968	3584	3968	4096	3584
Point X:8	3712	3712	3584	3712	3712	3584	3712	3712	3584	3712	3584	3584	3584	3712	3712	3584	3584	3712	3584	3712
Point X:9	3712	3712	3584	3712	3584	3712	3584	3840	3584	3584	3584	3456	3712	3712	3712	3584	3584	3712	3584	3584
Point X:10	3712	3712	3584	3712	3712	3712	3712	3712	3584	3712	3712	3584	3584	3712	3712	3584	3456	3712	3584	3584
Point Y:1	17536	17664	17664	17664	17664	17664	17536	17664	17664	17664	18560	17920	17664	17792	18176	18560	17664	17792	17536	18048
Point Y:2	17664	17664	17664	17792	17664	17664	17792	17664	17664	17664	18560	17920	17792	17664	18176	18560	17664	17792	17664	17920
Point Y:3	17664	17536	17536	17536	17536	17536	17536	17536	17536	17536	18432	17664	17664	17664	18048	18560	17664	17792	17664	17792
Point Y:4	17536	17408	17408	16896	4992	4864	17536	4864	4864	17408	18432	4864	5120	4864	4864	5120	17408	17664	17536	4864
Point Y:5	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	5120	4864
Point Y:6	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:7	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:8	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:9	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:10	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Time	9	8	9	8	8	8	8	8	8	8	8	8	8	9	8	8	8	8	8	8

Table 42 - Training Data Part B

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	4736	3840	5760	4736	6656	5376	8192	7552	7424	4736	6144	4224	8192	4224	3712	6528	5504	4480	3840	7552
Point X:2	3584	3584	3840	3712	4096	4096	6144	4480	4224	3712	3840	3712	6016	3584	3584	3968	3840	3712	3584	4352
Point X:3	3840	3584	3584	3584	3584	3712	4096	3712	3712	3712	3584	3584	4096	3712	3712	3712	3584	3584	3584	3584
Point X:4	3712	3584	3712	3712	3712	3584	3712	3712	3712	3840	3584	3456	3584	3584	3584	3584	3584	3712	3584	3584
Point X:5	3712	3456	3456	3584	3584	3584	3840	3712	3712	3712	3584	3584	3840	3584	3712	3584	3584	3584	3584	3584
Point X:6	3968	3584	3712	3712	3712	3968	3840	3840	3584	4224	3584	3712	3712	3968	3840	3584	3712	3840	3584	3840
Point X:7	7296	6144	5376	5888	4736	6144	4096	4608	3968	7552	4480	6400	3840	7168	8192	4480	4864	5888	5888	4480
Point X:8	7808	7936	8192	8192	8192	8192	6912	8192	7680	7680	8192	8192	4480	8064	6400	8192	8192	8192	7680	8064
Point X:9	4352	3584	4864	4352	5888	4992	7936	6912	7680	3840	6144	4224	8192	4096	3840	5888	5248	4224	3712	7424
Point X:10	3712	3456	3584	3712	3584	3712	3968	3712	3968	3712	3712	3712	6272	3712	3712	3456	3584	3584	3584	3584
Point Y:1	19584	19328	19712	19200	19712	19712	19328	19584	19200	19840	19072	19200	19200	19200	19200	19456	19328	19200	19584	19712
Point Y:2	19968	19328	19840	19200	19712	19840	20224	19712	19200	20352	19456	19328	19328	19840	19712	19584	19456	19456	20224	19712
Point Y:3	19584	19840	19840	19328	19840	19840	20096	20224	19456	19840	19200	19328	19968	19456	19200	19968	19456	19200	19712	20224
Point Y:4	19584	19328	19840	19200	19840	19840	20224	19840	19456	19840	19328	19200	19456	19456	19200	19584	19456	19200	19712	19840
Point Y:5	19584	19328	19840	19328	19840	19840	19968	19840	19584	20096	19328	19200	19584	19968	19840	19584	19584	19328	19712	19840
Point Y:6	19584	19328	19968	19456	20096	19968	19840	20352	19712	19840	19456	19712	20352	19456	19200	20096	19584	19328	19840	20352
Point Y:7	19584	19200	19840	19328	19840	19840	19712	19840	19584	19840	19584	19200	19840	19456	19200	19584	19584	19328	19840	19968
Point Y:8	19584	19200	19840	19328	19840	19840	19712	19968	19712	19840	19584	19200	19968	19712	19456	19584	19584	19328	19840	19840
Point Y:9	19584	19328	19968	19328	19968	19968	19584	20096	19968	16000	6912	19200	20352	5120	4992	19712	19840	19456	7936	20096
Point Y:10	4864	4864	4864	4864	4864	4864	19456	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Time	9	8	9	10	10	10	10	10	11	9	11	9	11	10	9	9	10	10	9	10

Table 43 - Training Data Part C

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	6272	5376	5376	6272	5376	5248	5248	5248	5248	5376	5376	5504	4992	4736	5376	6144	4864	5120	6016	5120
Point X:2	6272	5760	5760	6272	5888	6144	6016	6016	6016	5632	5632	6016	5376	5504	5632	6272	5248	5504	6144	5376
Point X:3	6272	6272	6272	6272	6272	6272	6272	6272	6272	6272	6144	6144	5760	6272	6016	6144	5632	6016	6272	6016
Point X:4	6272	6272	6272	6272	6272	6272	6272	6272	6272	6272	6144	6272	6144	6272	6144	6144	6016	6144	6144	6144
Point X:5	6400	6272	6272	6400	6272	6272	6272	6272	6272	6272	6272	6144	6144	6144	6272	6144	6144	6272	6144	6272
Point X:6	6400	6272	6272	6400	6272	6272	6272	6272	6272	6272	6272	6272	6144	6144	6144	6016	6272	6144	6016	6272
Point X:7	6272	6400	6400	6272	6400	6400	6400	6400	6400	6272	6272	6272	6144	6144	6144	6144	6144	6144	6144	6272
Point X:8	6272	6400	6400	6272	6400	6400	6400	6400	6400	6400	6272	6272	6144	6144	6016	6144	6144	6144	6144	6272
Point X:9	6272	6272	6272	6272	6272	6272	6272	6272	6272	6272	6400	6144	6144	6144	6144	6016	6016	6016	6144	6400
Point X:10	5760	6272	6272	5888	6272	6272	6272	6272	6272	6272	6400	6144	6144	6144	6144	5120	6144	6144	5888	6144
Point Y:1	19584	19968	19968	19584	19840	15232	19840	19840	19840	19840	19840	19712	19840	9856	19968	19840	19712	19840	19712	19840
Point Y:2	19456	15488	14720	19456	17024	16000	19584	19200	18048	14592	13824	19584	19968	13312	19968	19968	19712	19840	19712	19840
Point Y:3	19328	19712	19712	19328	19584	19456	19456	19456	19456	19712	19584	19584	4864	19968	19840	19712	19712	19712	19840	19840
Point Y:4	19072	19456	19456	18944	19328	19456	19456	19456	19328	19456	19456	19584	19712	19968	19840	19200	19712	19712	19456	19584
Point Y:5	18432	19456	19328	18304	19328	19328	19328	19328	19328	19328	19456	19456	19712	19968	19840	19200	19840	19712	18944	19584
Point Y:6	18304	19200	19072	18304	18944	19072	18688	18688	18816	19200	19328	18944	19712	19456	19584	19584	19840	19584	19072	19584
Point Y:7	18560	18432	18432	18560	18432	18304	18304	18304	18304	18432	18688	18816	19328	18944	19072	19712	19456	19072	19456	18944
Point Y:8	18560	18304	18304	18560	18304	18304	18176	18304	18304	18304	18304	18944	18944	18944	19072	19712	19328	19072	19584	18816
Point Y:9	18432	18560	18560	18432	18560	18560	18560	18560	18560	18560	18176	19200	18944	19328	19456	19456	19456	19328	19584	18944
Point Y:10	18048	18560	18560	18048	18560	18560	18560	18560	18432	18560	18560	19328	19328	19328	19584	19200	19968	19584	19328	19200
Time	12	12	12	12	12	12	12	12	12	12	8	8	8	8	8	8	8	8	8	8

Table 44 - Training Data Part D

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	23808	24320	21504	19200	20608	25216	19200	23296	20480	20480	21120	23552	19328	20608	20608	20224	20352	24192	22272	20480
Point X:2	28288	28672	28032	28288	26624	27776	27008	27776	26752	26752	25472	28032	27264	26624	26624	25600	27264	28032	28160	24448
Point X:3	28032	29056	28544	28928	28544	27904	28288	28160	28288	28288	28800	28800	28032	28672	28672	28160	28672	28544	27648	28032
Point X:4	19072	18048	22528	27392	28160	22656	27136	19712	26112	26112	30848	18944	25472	24192	24192	28544	24320	17536	23424	29184
Point X:5	19584	18432	18048	18432	17408	18304	17792	18432	17152	17152	15104	17920	17792	17152	17152	16768	17664	18176	18816	19072
Point X:6	19072	18176	17408	18304	18176	18304	18048	18176	18304	18304	16896	18048	18432	17920	17920	17920	17920	18048	18176	16512
Point X:7	6016	6528	6784	15232	17024	6272	16768	9600	14976	14976	14592	7808	15488	11520	11520	17024	15616	6272	9216	15360
Point X:8	3584	3584	3712	3712	3712	3456	3840	3584	3712	3712	3584	3584	3712	3712	3712	3968	3584	3584	3712	3840
Point X:9	3840	3584	3584	3584	3712	3584	3712	3584	3584	3584	3584	3584	3712	3584	3584	3712	3712	3712	3840	3712
Point X:10	3584	3712	3712	3584	3584	3584	3712	3712	3712	3712	3712	3712	3584	3584	3584	3712	3584	3584	3712	3712
Point Y:1	16640	17152	17280	17152	17152	16768	17024	17152	16896	16896	17280	17280	16896	17152	17152	17024	17152	17280	17152	17664
Point Y:2	17024	17408	17792	17664	17792	17280	17664	17536	17408	17408	17408	17664	17536	17536	17536	17664	17664	17536	17664	18304
Point Y:3	16640	17152	17536	17536	17664	17152	17664	17408	17280	17280	17408	17536	17408	17408	17408	17408	17536	17408	17536	18048
Point Y:4	16384	17152	17536	17280	17408	16896	17408	17152	17152	17152	16896	17408	17152	17280	17280	17280	17280	17280	17280	18048
Point Y:5	15616	16128	16768	17152	17024	16128	17280	16256	16512	16512	17024	16512	16640	16640	16640	16896	16512	16384	16512	17792
Point Y:6	4992	5120	9984	16256	16256	5632	16128	14592	16000	16000	16128	6912	16128	16128	16128	16128	16128	6912	14976	16512
Point Y:7	4864	4864	4864	4864	4992	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:8	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:9	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:10	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Time	5	4	5	4	4	5	5	5	4	4	4	5	4	4	4	4	4	5	5	4

Table 45 - Training Data Part E

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	24192	23168	23168	24704	24320	23296	24064	24448	23808	23296	23296	23168	23168	23936	23168	23680	23168	23680	23424	23936
Point X:2	23552	21376	22528	23296	24320	22400	23296	23424	21248	20480	20352	22772	24320	24832	22784	23296	21504	21376	22784	21376
Point X:3	23168	22528	21376	21888	24832	21248	20992	24320	21504	23040	22272	22528	22528	23168	22528	22272	21632	23040	21632	22016
Point X:4	24320	25600	23424	22784	25088	24192	22144	23680	25472	25856	24832	23168	22784	22656	22144	24960	25472	25088	23552	23296
Point X:5	23040	25472	23936	23680	24064	24064	22528	22656	24448	24448	23936	23552	23808	23168	24064	23552	25344	21760	23552	23040
Point X:6	20096	19200	20352	22656	17408	18944	21632	17664	19328	16512	18560	21760	23168	22400	23680	21248	23168	13824	19072	13952
Point X:7	13056	10240	13184	12672	11648	10752	13952	9856	13696	8960	10752	13312	13952	17920	19200	15104	11392	8576	12800	7168
Point X:8	5376	4992	4736	4992	5120	4864	5888	5120	4992	4736	4992	4992	6400	7040	6656	5376	4992	4992	4608	4608
Point X:9	4352	4608	4608	4736	4992	4608	4480	4864	4608	4096	4608	4480	4608	4736	4608	4736	4608	4608	4608	4480
Point X:10	3712	3712	4096	3968	3840	3840	3840	3584	3840	3584	3712	3712	3712	4608	4224	3968	4096	3840	3968	3712
Point Y:1	17408	17792	17792	17664	18048	17792	17792	18176	17920	18048	17792	18176	17792	17664	17536	18048	17920	18048	17920	18048
Point Y:2	18176	18176	18176	18048	18176	18176	18048	18176	18176	18304	18176	18560	18176	18048	18048	18176	18176	18304	18176	18304
Point Y:3	18304	18176	18304	18304	18176	18304	18304	18304	18304	18304	18176	18560	18304	18176	18432	18304	18304	18304	18304	18176
Point Y:4	18176	18048	18176	18304	18176	18176	18176	18048	18176	18176	18048	18560	18176	18176	18176	18176	18176	18048	18176	18048
Point Y:5	17792	17792	17920	17920	17664	17920	17920	17792	17792	17664	17792	18304	17920	18048	18048	17920	17920	17664	17920	17664
Point Y:6	17280	17024	17408	17408	16768	17280	17664	16768	17152	16768	17024	17664	17408	17536	17536	17280	17408	16640	17280	16768
Point Y:7	16384	16384	16512	16256	6912	16384	16384	5888	16384	6144	8960	16768	16512	16512	16512	16384	16256	6016	16384	4992
Point Y:8	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:9	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Point Y:10	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864	4864
Time	5	4	4	5	4	4	4	4	5	4	4	4	4	4	5	4	5	4	5	4

Table 46 - Training Data Part F

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7	Data Set 8	Data Set 9	Data Set 10	Data Set 11	Data Set 12	Data Set 13	Data Set 14	Data Set 15	Data Set 16	Data Set 17	Data Set 18	Data Set 19	Data Set 20
Point X:1	26752	27520	27648	27264	27776	27136	27264	26752	26880	27776	27392	26880	24960	27136	27136	26240	27264	27904	27136	26880
Point X:2	23296	13056	10368	17408	5504	25216	22400	4096	23296	12288	11648	24576	3712	19200	4992	3968	25088	27648	9344	25344
Point X:3	3840	3840	3712	3840	3712	3968	3712	3840	4224	3712	3840	3840	3712	3840	3840	3840	3712	7424	3840	3968
Point X:4	3712	3712	3840	3840	3712	3840	3584	3840	3584	3584	3712	3840	3840	3840	3840	3968	3584	3712	3840	3712
Point X:5	3840	3968	3840	4224	3968	3840	3968	3968	3584	3968	3968	3840	3840	3968	3968	3968	3840	3840	3968	3712
Point X:6	3840	3840	3968	4096	3712	3840	3968	3712	3584	3840	3840	3968	3968	3840	3840	3840	3840	3968	3840	3968
Point X:7	3840	3840	3840	3968	3840	3712	3840	3840	3584	3712	3840	3712	3968	3712	3840	3968	3584	3840	3712	3712
Point X:8	3840	4224	4608	7424	4224	3968	3840	11008	3584	3840	4224	3840	16640	3840	9856	17408	3968	3840	4864	3712
Point X:9	18176	26624	26880	27008	27008	5248	14464	27264	16128	25344	25600	17024	27520	24320	27520	27392	15488	3968	27136	12544
Point X:10	27776	27136	26880	27136	26880	27776	27520	26752	26624	26880	27136	27520	27520	27008	27264	27776	27520	24576	26880	27136
Point Y:1	18944	19328	19456	19840	19584	19200	19328	19584	19200	19456	19456	19072	19712	19200	19584	19712	19200	18688	19584	19072
Point Y:2	19712	19840	19968	20352	19968	19712	19840	20096	19840	19968	19968	19584	20096	19840	20096	20224	19712	19328	19968	19712
Point Y:3	20096	20224	20352	20608	20224	20096	20096	20352	20096	20352	20352	20096	20352	20096	20480	20352	20096	20096	20352	20096
Point Y:4	20352	20480	20480	20864	20608	20480	20352	20352	20608	20480	20608	20352	20480	20352	20736	20224	20480	20224	20480	20352
Point Y:5	20608	20736	20864	20992	20480	20480	20608	20608	20736	20480	20736	20480	20608	20352	20608	20608	20352	20608	20608	20480
Point Y:6	20480	20608	20608	20992	20608	20480	20480	20480	20608	20480	20480	20608	20480	20608	20608	20480	20608	20480	20480	20736
Point Y:7	20480	20480	20480	20736	20352	20480	20480	20352	20608	20352	20480	20480	20352	20480	20480	20352	20480	20480	20352	20608
Point Y:8	20352	20224	20352	20480	20096	20352	20352	20096	20352	20096	20224	20352	20096	20096	20096	19968	20352	20352	20096	20480
Point Y:9	19968	19840	19840	20224	19712	20096	19840	19584	20096	19712	19840	19968	19584	19840	19584	19456	19968	20096	19712	20096
Point Y:10	19456	19200	19200	19584	19456	19456	19456	19072	19584	19200	19328	19584	19456	19456	18944	19200	19328	19712	19072	19584
Time	13	13	13	13	12	12	13	12	13	13	13	12	12	12	13	12	12	13	12	12

vi. PLC Simulator Source Code

Refer to enclosed CD.

vii. Intelligent Neural Network Training utilizing Genetic Algorithm Source Code

Refer to enclosed CD.

viii. Step 7 Part Recognition PLC Source Code

Refer to enclosed CD.