

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

The Effects of Using Problem Knowledge in a Neural Network for Image Processing Tasks

A thesis presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science at Massey University, Palmerston North, New Zealand.

Kapila Sanjeeva Gunetilleke

2001

To the memory of my father Dr. K. G. Gunetilleke

ABSTRACT

This thesis is concerned with aspects of computational intelligence. Computational intelligence is a new paradigm of artificial intelligence based on biological intelligence. Computational Intelligence explores the potential for biologically inspired intelligent adaptive machines and behavior. A relatively new and important sub discipline within the field of computational intelligence, is that of neural networks. Neural networks are networks of artificial neurons with a high degree of interconnectivity. The networks capture and accumulate knowledge as the pattern of weights in the interconnections between neurons. Neural networks are iteratively trained to perform tasks by “learning” from examples. These networks are often slow to train. One of the reasons for this is that the starting weights of the network are conventionally not related to the problem being solved. In this thesis a methodology is explored that maps problem knowledge to the starting weights of a fuzzy neural network window filter (FuNNWF). The FuNNWF architecture was developed from the combination of fuzzy logic and artificial neural networks for image processing tasks. The effects of the use of problem knowledge on FuNNWF training are investigated. The problem knowledge mapping procedure is extended from boolean rule mapping to conditional rule mapping, which allows better representation of the problems. Four real world image processing problems are investigated using the new weight initialization methodology. The experiments reported in this thesis demonstrate that the use of problem knowledge improves the robustness and convergence of the neural network. It is also shown that the use of the methodology is most effective on network training when the training data is noisy, unreliable and ambiguous.

ACKNOWLEDGEMENTS

I would like to thank my two supervisors Prof. Bob Hodgson and Dr. Bob Chaplin for their help, encouragement, understanding and patience. I have been very fortunate to have them as my supervisors, who have encouraged high standards in research and writing at the same time as providing considerable insight into the fascinating area we have been studying. I would also like to thank them for their friendship and sense of humour.

I would like to thank my mother and grandfather who always encourage me to pursue further studies. Finally I would like to thank my wife Samani for her support, encouragement and the considerable amount of “family time” that she sacrificed to enable me to complete this thesis.

CONTENTS

Abstract	i
Acknowledgements	iii
Contents	v
List of Figures	xiii
List of Publications	xvii
CHAPTER 1	1
1 INTRODUCTION	1
1.1 Scope of research	3
1.2 Thesis overview	4
1.3 Contents by chapter	5
CHAPTER 2	7
2 BACKGROUND	7
2.1 Fuzzy logic	7
2.1.1 Example of a fuzzy membership function	8
2.1.2 Example of a rule	9
2.2 Artificial Neural Networks.....	10
2.2.1 Neural network weight initialization.....	10
2.2.2 Architecture of a Fuzzy Neural Network	12
2.2.3 FuNN training methods.....	13
2.2.4 The training algorithm	14
2.3 Image processing.....	18
2.3.1 The window filter.....	18
2.4 The fuzzy neural network window filter	20
2.4.1 Structure of the FuNNWF.....	21
2.5 Putting it all together	21
2.5.1 Problem knowledge	21
2.6 Environment.....	22
2.6.1 Software	22

2.6.2	Hardware.....	23
2.6.3	Simulations	23
2.7	Summary	24
CHAPTER 3		25
3	RULE MAPPING PROCEDURE	25
3.1	Problem knowledge.....	26
3.2	Mapping to FuNN architecture	28
3.3	Rule insertion procedure	29
3.3.1	Boolean logic rule mapping	30
3.3.2	Conditional rule mapping	33
3.4	General derivation of rules	41
3.4.1	Rule: Input > Constant	41
3.4.2	Rule: Input = Constant	44
3.4.3	Multiple inputs to a Rule node.....	45
3.4.4	K out of M expressions.....	45
3.4.5	Rule: Input i > Input j	46
3.4.6	Non conclusive logic.....	47
3.5	The quality factor.....	47
3.6	Discussion	47
CHAPTER 4		49
4	USE OF A FUNN TO SOLVE FULLY DETERMINED PROBLEMS	49
4.1	The XOR Problem	49
4.1.1	Training and test data generation	50
4.1.2	Rules representing the XOR problem	52
4.1.3	The error function	52
4.1.4	The MSE error	53
4.1.5	Fuzzy neural network structure used for XOR problem	53
4.1.6	Experiment to determine if a FuNN could emulate the XOR problem without training using boolean logic rules.	53
4.1.7	Experiment to determine if a FuNN could emulate the XOR problem without training using conditional rules.	55
4.1.8	Experiment - Operational performance test 1 for a range of parameters on the XOR problem using boolean logic rules	56
4.1.9	Experiment – Operational performance test 2 for a the same parameters but a smaller range on the XOR problem using boolean logic rules	58
4.1.10	Experiment – Operational performance test 3 for a range of parameters on the XOR problem using conditional rules.....	59
4.1.11	Experiment – Operational performance test 4 for the same parameters on a smaller range on the XOR problem using conditional rules.....	60
4.2	Partial Rule Sets.....	60
4.2.1	Training and test data.....	61
4.2.2	Fuzzy neural network structure used for this problem	61
4.2.3	Experiment 1 - Problem A = B & C & (not D) – complete rule set.....	62

4.2.4	Experiment 1a - Problem $A = B \& C \& (\text{not } D)$ – incomplete rule set	63
4.2.5	Experiment 2 – Problem $A = B \& (\text{not } C) \& D$ – complete rule set	65
4.2.6	Experiment 2a – Problem $A = B \& (\text{not } C) \& D$ – incomplete rule set	66
4.2.7	Experiment 3 – Problem $A = (\text{not } B) \& C \& D$ – complete rule set	68
4.2.8	Experiment 3a – Problem $A = (\text{not } B) \& C \& D$ – incomplete rule set	69
4.2.9	Experiment 4 – Problem $A = B \& C \& (\sim D)$ using only a partial rule set	71
4.3	Discussion and Conclusions	72
CHAPTER 5		75
5	FULLY UNDERSTOOD NON DETERMINED PROBLEM	75
5.1	Overview of the toy problem.....	75
5.1.1	A detailed description of the problem.....	75
5.1.2	Description of a toy.....	76
5.1.3	The training images	77
5.2	Network training method	80
5.2.1	The training data	80
5.2.2	Rule development	80
5.2.3	The output image	81
5.3	Toy type classification.....	81
5.3.1	Membership functions for fuzzification process.....	82
5.3.2	Membership functions for defuzzification process	82
5.3.3	Error measure.....	83
5.3.4	Experiment to determine how a network initialized with rules performs compared to one initialized without rules.....	83
5.3.5	Experiment to measure the performance of a network over 100 runs when initialized with rules	85
5.3.6	Experiment to measure the performance of a network initialized with random numbers over 100 runs.	86
5.3.7	Experiment to determine if adding rules developed to remove false positives would improve the performance of the network.	86
5.3.8	Experiment to determine if changing the quality factor for individual rules improved the classification performance of the network.	87
5.3.9	Experiment to determine the effects of adding small random numbers to a network initialized with rules over 100 runs.....	88
5.3.10	This experiment is a continuation of the previous experiment. A network with the same structure was initialized with random number and trained.	89
5.3.11	Experiment is to determine the network with the best performance from several networks initialized with rules.	89
5.3.12	This is a continuation of the previous experiment. The quality factor for the rules was changed and the experiment was repeated.....	91
5.3.13	This is a continuation of the previous experiment to measure the performance of the network without rules.	92
5.3.14	Experiment to determine how a bad rule changes with network training	92
5.3.15	This Experiment is a continuation of the previous experiment to determine how a bad rule changes with network training	95
5.3.16	Experiment to measure the operational performance of a network initialized with rules using a larger number of membership functions.....	96
5.4	Toy orientation classification	97
5.4.1	Membership functions for fuzzification process.....	98
5.4.2	Defuzzification membership functions	98
5.4.3	Experiment to compare the performance of a network initialized with rules and random numbers.....	99

5.4.4	Experiment to determine network performance with a new set of fuzzification membership functions and extended rules.....	100
5.4.5	A statistical experiment to determine the performance of a network.....	102
5.4.6	Experiment to determine the operational performance of a population of networks tested on several new images.....	103
5.4.7	Experiment to determine the incremental and cumulative weight change during training.....	103
5.4.8	This is a continuation of the previous experiment to determine the effects of adding free nodes to the network.....	106
5.4.9	Experiment to determine the effects of adding a bad rule to the network.....	107
5.4.10	Experiment to determine the range of operational performances shown by a population of networks when initialized with rules.....	109
5.5	Classification of Rules.....	110
5.5.1	A strong rule.....	110
5.5.2	A weak rule.....	110
5.5.3	A weak incorrect rule.....	110
5.5.4	A strong incorrect rule.....	110
5.5.5	A bad rule.....	110
5.6	Conclusions.....	111
5.6.1	Specific Conclusions.....	111
5.6.2	General Conclusions.....	114
CHAPTER 6.....	115	
6	WANE EDGE DETECTION.....	115
6.1	The wane edge detection problem.....	115
6.2	Problem knowledge for wane edge detection.....	118
6.2.1	Problem knowledge expressed in high level terms.....	118
6.2.2	The size and shape of the window filter.....	119
6.3	Training data.....	119
6.4	The membership functions.....	120
6.5	The error measure.....	121
6.5.1	The error measure function.....	121
6.6	The experiments.....	123
6.6.1	Experiment to detect the edges using a 3x3 window filter.....	124
6.6.2	Experiments to compare the effectiveness of wane edge detection for a range of window sizes.....	127
6.6.3	Experiment to determine if the network could detect one wane edge only using a 9x9 window filter.....	133
6.6.4	Use of a 9x9 window filter to detect the left wane edge.....	137
6.6.5	Network training experiment that involves changing the number of rules and training times.....	139
6.6.6	Experiment to determine if a FuNN using a 9x9 window filter with a smaller rule set could detect the left wane edge.....	142
6.6.7	Experiment to detect the left wane edge using conditional rules.....	144
6.6.8	Experiment to determine the performance of a FuNN seeded with different random numbers. 100 runs were made.....	147
6.6.9	Experiment to detect the right wane edge.....	149
6.6.10	Generating the final image containing both left and right wane edges.....	150
6.6.11	Using the trained network to detect the wane edges on other planks of wood.....	151

6.6.12	Experiment to test alternative training algorithms	154
6.6.13	Experiment to determine the quality factor for the rules.....	158
6.7	Conclusions.....	161
CHAPTER 7		163
7	TREE CROWN DETECTION.....	163
7.1	Introduction.....	163
7.1.1	The training images	165
7.1.2	Window filter size.....	166
7.1.3	The membership functions.....	167
7.1.4	The rules	167
7.2	Experiments.....	169
7.2.1	Experiment to develop a neural network to detect tree crowns.....	169
7.2.2	Experiment to investigate the used of a hierarchical network structure to separate blobs of clustered trees into blobs corresponding to individual trees.....	173
7.2.3	Experiment to determine the tree counts for multiple networks initialized with random numbers.....	176
7.2.4	The new training images.....	177
7.2.5	Experiment to determine the tree count when the network was trained using the supplied ground truth based training images.....	179
7.2.6	Experiment to determine the tree count made by the neural network using modified ground truth data.....	182
7.2.7	Experiment to determine the tree counts for 100 networks initialized with random numbers.....	184
7.2.8	Experiment to investigate the hypothesis that the rules compensate for inaccuracies in the training data	186
7.3	Conclusions.....	190
CHAPTER 8		191
8	IRIS EDGE DETECTION.....	191
8.1	Introduction.....	191
8.1.1	Iris isolation	192
8.1.2	Use of a standard filter to detect the iris edge	194
8.2	The training images	196
8.3	Development of a figure of merit.....	197
8.4	The experiments.....	198
8.4.1	Experiment to determine if a FuNN could detect the iris edge.....	198
8.5	The new training images	203
8.6	Generation of the new membership functions.....	204
8.6.1	Membership functions for fuzzification process.....	205
8.6.2	Membership functions for defuzzification process	206
8.7	The experiments.....	206
8.7.1	Experiment to find the iris edge using a FuNN initialized with rules.....	206

8.7.2	Experiment to find the iris edge using a FuNN using a larger window than in experiment 8.7.1	209
8.7.3	Experiment to find the iris edge using a FuNN using a greater number of free nodes compared to experiment 8.7.2.....	212
8.7.4	Experiment to train a network to detect the iris edge near the eyelashes	215
8.7.5	Use of two neural networks to detect the iris edge	217
8.7.6	Operational test for iris detection problem using two neural networks.....	222
8.7.7	Experiment to determine if the least cost path algorithm in the post process procedure could be replaced by a neural network.....	225
8.7.8	Experiment to determine the weight change of the rules during training	228
8.7.9	Experiment to determine if removing rule 2 and 7 from the rule set from experiment 8.7.8 would improve the network performance	230
8.8	Conclusions.....	233

CHAPTER 9 235

9 CALCIFICATION DETECTION IN MAMMOGRAMS 235

9.1	Introduction.....	235
9.2	The training data	236
9.2.1	Window filter size.....	239
9.3	Rule development.....	240
9.3.1	The texture based method	240
9.3.2	Spatial Gray Level Dependence Matrix (SGLDM)	241
9.3.3	Neural network training method	244
9.3.4	The values for the texture measures	244
9.4	The experiments.....	246
9.4.1	Experiment to determine if the FuNN could detect microcalcifications in a mammogram	247
9.4.2	Experiment to compare network performance	249
9.4.3	Experiment to determine the network performance of a large population of networks	251
9.4.4	Experiment to determine if the calcification detection rate of the network could be improved	252
9.5	Discussion and conclusions.....	254

CHAPTER 10 257

**10 REDUCING THE FUNN TO A CONVENTIONAL MLP NETWORK ..
..... 257**

10.1	Introduction.....	257
10.2	Funn to MLP mapping	258
10.2.1	Fuzzification layer equivalence	259
10.2.2	Defuzzification layer equivalence.....	261
10.2.3	Yam and Chow's weight initialization method.....	263
10.3	The experiments.....	265
10.3.1	Experiment to determine the network performance of a MLP network initialized with rules.....	265
10.3.2	Experiment to determine the effects on MLP training when a modified version of Yam and Chow's initialization method was used.....	272

10.4	Conclusions.....	273
CHAPTER 11		275
11	SUMMARY AND CONCLUSIONS	275
11.1	Summary	275
11.2	Conclusion	277
11.3	Future work.....	279
APPENDIX A		283
REFERENCES		309
BIBLIOGRAPHY		315

LIST OF FIGURES

Figure 2.1 A membership function for young.....	8
Figure 2.2 Membership function for an example rule.....	9
Figure 2.3 Fuzzy neural network architecture.....	14
Figure 2.4 Operation of a window filter.....	19
Figure 2.5 Example of filter operation. Image (a) is the original image. Image (b) was generated by passing a common edge detection filter over image (a). Image (c) was generated by passing an embossing filter over image (a).....	19
Figure 2.6 Neural network window filter.....	20
Figure 2.7 Fuzzy neural network window filter structure.....	21
Figure 3.1 (a) Image containing circular objects and other geometric shapes, all the objects are outlined for clarity. (b) – (e) clusters of objects extracted from (a).....	26
Figure 3.2 Fuzzy neural network section.....	34
Figure 3.3 Rule node output function.....	35
Figure 3.4 Fuzzy function for Input>Constant.....	41
Figure 3.5 Fuzzy function for Input = Constant.....	44
Figure 4.1 XOR training data – 200 pairs.....	51
Figure 4.2 XOR test data – 600 pairs.....	51
Figure 4.3 FuNN 2-4-4-2-1 architecture for XOR problem.....	53
Figure 4.4 Operational error graph for XOR problem.....	57
Figure 4.5 Operational errors for different saturation values.....	59
Figure 4.6 Operational error graph for $A=B \& C \& (\sim D)$	64
Figure 4.7 Operational error graph for $A=B \& (\sim C) \& D$	67
Figure 4.8 Operational error graph for $A=(\sim B) \& C \& D$	70
Figure 4.9 Operational performance of $A=B \& C \& (\sim D)$ for a network using a partial rule set with high certainty.....	72
Figure 5.1 Toys that can be classified by type and orientation.....	76
Figure 5.2 Image containing 25 individual toys of different type and orientation.....	77
Figure 5.3 Training images.....	78
Figure 5.4 Window filter structure.....	81
Figure 5.5 Incremental weight change graph for experiment 5.3.14.....	93
Figure 5.6 Cumulative weight change for experiment 5.3.14.....	94
Figure 5.7 Incremental weight change graph for experiment 5.3.15.....	95
Figure 5.8 Toy orientation problem defined by the placement of the 2x2 square around the 3x3 larger square.....	97
Figure 5.9 Incremental weight change of the network ($Q=0.5$) over 175 epochs for experiment 5.4.7. (A) is the incremental weight change for the rules and (B) is the incremental weight change for the free nodes in the network.....	104
Figure 5.10 Cumulative weight change for the network ($Q=0.5$) over 175 epochs for experiment 5.4.7. (A) is the cumulative weight change for the rule nodes and (B) is the cumulative weight change for the free nodes.....	105
Figure 5.11 Incremental weight change of the network ($Q=0.5$) over 175 epochs for experiment 5.4.8. (A) is the incremental weight change for the rules and (B) is the incremental weight change for the free nodes in the network.....	106
Figure 5.12 Cumulative weight change for the network ($Q=0.5$) over 175 epochs for experiment 5.4.8. (A) is the cumulative weight change for the rule nodes and (B) is the cumulative weight change for the free nodes.....	107
Figure 5.13 Incremental weight change when ($Q=0.5$) for experiment 5.4.9.....	108
Figure 5.14 Cumulative weight change for the network ($Q=0.5$) over 175 epochs for experiment 5.4.9. (A) is the cumulative weight change for the rule nodes and (B) is the cumulative weight change for the free nodes.....	108
Figure 6.1 Image of a plank with manually marked defects.....	116

Figure 6.2 The images show the output generated by processing the original image with a vertical edge detector and thresholding. Image (a) is the output generated by processing figure 6.1 with a common edge detection filter. Image (b) was generated by thresholding image (a)..	117
Figure 6.3 Training images for wane problem.	119
Figure 6.4 The training data was generated using the rectangular sections in the image.	120
Figure 6.5 Image showing the wane edge, bark and useable area on the plank of wood.	122
Figure 6.6 Neural network output for experiment 0. Image (a) is the output generated by the network initialized with rules and image (b) is the output generated by the network initialized with random numbers.	126
Figure 6.7 Representation of If I1=HI Then O1=LO.	131
Figure 6.8 Network output for experiment 6.6.3	135
Figure 6.9 The post process procedure, image (a) generated by thresholding the network output, image (b) generated by thinning the thresholded output and image (c) generated by scanning the thinned image from left to right to detect the first occurrence of a high intensity pixel..	137
Figure 6.10 Results of training the network for 200 epochs for experiment 6.6.4 (a) network output (b) network output thresholded (c) thresholded image thinned (d) thinned image scanned for left wane edge.	138
Figure 6.11 Results after training t network for 800 epochs for experiment 6.6.4 (a) output image (b) output image thresholded (c) thresholded image thinned (d) thinned image scanned for left wane edge.	139
Figure 6.12 Output images for experiment 6.6.5.	141
Figure 6.13 Output images for experiment 6.6.6.	143
Figure 6.14 Training performance graph for experiment 6.6.7.	146
Figure 6.15 Output images for experiment 6.6.7. Image (a-1) output image from network with rules. Image (a-2) generated by post processing (a-1). Image (b-1) output image from network without rules and image (b-2) generated by post processing (b-1).	147
Figure 6.16 The output images for experiment 6.6.9. (a) the output image generated by the network, (b) the output image thresholded, (c) thresholded image thinned, (d) thinned image scanned for right wane edge.	150
Figure 6.17 (a) shows the left and right wane combined, (b) shows the combined wanes superimposed on the input image.	151
Figure 6.18 Plank 1 used to test the trained FuNN. (a) input image, (b) left and right wane detected using trained FuNN, (c) detected left and right wane superimposed on the input image.	152
Figure 6.19 Plank 2 used to test the trained FuNN. (a) input image, (b) left and right wane detected using trained FuNN, (c) detected left and right wane superimposed on the input image.	153
Figure 6.20 Plank 3 used to test the trained FuNN. (a) input image, (b) left and right wane detected using trained FuNN, (c) detected left and right wane superimposed on the input image.	153
Figure 6.21 Graph showing the error between the output and target image during training for the network initialized with rules.	157
Figure 6.22 Graph showing the error between the output and target image during training for the network initialized with random numbers.	158
Figure 6.23 Network performance for experiment 6.6.13 when 6 rules and 1 free node was used.	159
Figure 6.24 Network performance for experiment 6.6.13 when 6 rules and 2 free nodes were used.	160
Figure 6.25 Network performance for experiment 6.6.13 when 6 rules and 4 free nodes were used.	160
Figure 7.1 Aerial photograph of a commercial Pinus Radiata plantation.	164
Figure 7.2 The training images. Image (a) is a sub image from the original aerial photograph. Image (b) is the pre-processed image and image (c) is the target image.	166
Figure 7.3 Image of a tree superimposed on the 11x11 window.	166
Figure 7.4 Image of two trees close to each other. Rule 3 was generated from this image.	168
Figure 7.5 Image of two trees close to each other. Rule 4 was generated from this image.	168
Figure 7.6 Image of two trees horizontally close to each other. Rule 5 was generated from this image.	169
Figure 7.7 Image of two trees vertically close to each other. Rule 6 was generated from this image.	169
Figure 7.8 Output images generated by the trained networks for experiment 7.2.1. Image (a) is the output image generated by the network initialized with rules and image (b) is the output generated by the network initialized with random numbers.	170

Figure 7.9 Training performance graph for network initialized with and without rules for experiment 7.2.1.....	171
Figure 7.10 Hierarchical neural network structure.....	173
Figure 7.11 Output images generated by the trained networks for experiment 7.2.2. Image (a) is the input image, image (b) is the output image generated by the network initialized with rules and image (c) is the output generated by the network initialized with random numbers.	175
Figure 7.12 Training performance graph for networks initialized with and without rules for experiment 7.2.2.....	175
Figure 7.13 Tree count distribution for network initialized with random numbers for experiment 7.2.3.....	177
Figure 7.14 The commercial plot.	178
Figure 7.15 The ground truth data for the commercial plot.	178
Figure 7.16 The new training images for the tree crown detection problem. Image (a) is the input image and image (b) is the target image.....	179
Figure 7.17 Image (a) is the output image generated by the hierarchical neural networks for experiment 7.2.5. Image (b) is the target image.	180
Figure 7.18 The image shows a sub-image from the commercial plot. The boxes and circles mark the tree positions given by the ground truth data.....	181
Figure 7.19 New training image for tree crown detection problem. (a) input image and (b) target image.....	183
Figure 7.20 Output image generated by hierarchical neural network for experiment 7.2.6.	184
Figure 7.21 Tree count distribution for experiment 7.2.7.	185
Figure 8.1 Image of an eye used for the iris edge detection problem.....	192
Figure 8.2 Image (a) is the linearised eye image using a polar to rectangular transform. Image (b) shows the pupil and iris edges of the eye.	194
Figure 8.3 Processing the input image using a vertical edge detection filter. Image (a) is the input image and image (b) is the processed image.	195
Figure 8.4 Training images for iris detection problem. Image (a) is the input image and image (b) is the target image.....	196
Figure 8.5 Image to demonstrate the error calculation.....	197
Figure 8.6 Input image showing areas of interest.....	199
Figure 8.7 Shows the window filter placed over a section of the eye.	200
Figure 8.8 Output image generated by processing the input image with the trained neural network.	202
Figure 8.9 Training images for iris detection problem. Image (a) is the original eye image, image (b) was generated by passing an illumination correction process on (a) and is used as the input image and image (b) is the target image.....	204
Figure 8.10 Input image displaying areas of interest to generate fuzzification membership functions.	205
Figure 8.11 Output generated from the trained network for experiment 8.7.1.....	208
Figure 8.12 Output image generated by the trained network initialized with rules for experiment 8.7.2.....	211
Figure 8.13 Network output for experiment 8.7.3. Image (a-1) was generated by the network initialized with rules, image (a-2) was generated by thresholding and thinning image (a-1). Image (b-1) was generated by the network initialized with no rules and image (b-2) was generated by thresholding and thinning image (b-1).....	214
Figure 8.14 Profile plots of different sections along the iris edge.....	215
Figure 8.15 The training images and network output for experiment 8.7.4. Images (a) and (b) were the training images and image (c) was the network output.	217
Figure 8.16 Segmented input image showing section 1 and section 2.	218
Figure 8.17 Image (a) is section 1 from the input image and image (b) is the output generated by the network.....	218
Figure 8.18 Image (a) is the input image and image (b) is the output generated for section 2.....	219
Figure 8.19 Image (a) is the combined output. Image (b) was generated by post processing image (a). Image (b) was superimposed on the input image to generate image (c).	220
Figure 8.20 Image (a) is the combined output from the two networks. Image (a) was post processed to generate image (b). Image (b) was superimposed on the input image to generate image (c).	221
Figure 8.21 New eye images for operational test.	223
Figure 8.22 Eye images in figure 8.21 post processed.....	223

Figure 8.23 Eye images in figure 8.22 processed using network one and network two..... 224

Figure 8.24 Images in figure 8.23 post processed. 224

Figure 8.25 Post processed network output overlaid over input images. 225

Figure 8.26 Input image showing areas of interest..... 226

Figure 8.27 Image (a) is the input image and image (b) is output image generated by processing image (a) using the trained network. 228

Figure 8.28 The cumulative weight change graph for the network when $Q=0.5$. The weight change (rules) graph shows the cumulative weight change for the rule nodes and the weight change (free) graph shows the cumulative weight change for the free nodes. 230

Figure 8.29 The results of the cumulative weight change for experiment 8.7.9 using a quality factor of 0.5. The weight change (rules) graph shows the cumulative weight change for the rule nodes and the weight change (free) graph shows the cumulative weight change for the free nodes. 231

Figure 9.1 The two mammographic views taken for each breast. 237

Figure 9.2 CC mammographic view of the right breast from case 1220 from the DDSM. 237

Figure 9.3 MLO mammographic view of the right breast from case 1220 from the DDSM. 238

Figure 9.4 An image of a clustered microcalcification. 238

Figure 9.5 Training images for microcalcification detection problem. Image (a) is the input image and image (b) is the target image. 239

Figure 9.6 Input generation for FuNN for mammogram problem..... 244

Figure 10.1 The proposed FuNN to MLP mapping. 259

Figure 10.2 The proposed equivalent MLP network for the FuNN. (a) shows a section from the FuNN and (b) shows the equivalent MLP section..... 260

Figure 10.3 Proposed mapping for defuzzification section. (a) is the output layer of the FuNN and (b) is the equivalent output layer for the MLP network. 261

Figure 10.4 Training performance of MLP network for the Sobel filter emulation problem..... 269

Figure 10.5 Training performance of MLP network for wane edge detection problem..... 270

Figure 10.6 Training performance of MLP network for the tree crown detection problem. 271

Figure 11.1 Extended input layer of a FuNNWF. 279

PUBLICATIONS

Publications prepared during the course of the research for this thesis are listed below. Refereed publications are highlighted:

1. Gunetileke S., Chaplin R. I., Siroki M., Hodgson R. M., The Development of Neural Network Technologies for Image Processing, Proceedings of the 5th Annual New Zealand Engineering and Technology Postgraduate Conference, Palmerston North, New Zealand, November 1998, pp 103-104.
2. **Chaplin R. I., Siroki M., Hodgson R. M. & Gunetileke S., The Development of Mapping Techniques to Incorporate Image Processing Problem Specific Rules into Neural Networks, Proceedings of the Image & Vision Computing New Zealand International Conference (IVCNZ '98), Auckland, New Zealand, November 1998, pp 357-362 (Refereed publication).**
3. **Chaplin R. I., Hodgson R. M. & Gunetileke S., Towards the Use of Problem Knowledge in Training Neural Networks for Image Processing Tasks, Proceedings of IEE Seventh International Conference on Image Processing and its Applications, Manchester, UK, July 1999, pp 62-66 (Refereed publication).**
4. **Chaplin R. I., Hodgson R. M. & Gunetileke S., Automatic Wane Detection in the Images of Planks using a Neural Network, Proceedings of the 5th International Symposium on Signal Processing and its Applications (ISSPA '99), Brisbane, Australia, August 1999, pp 657-660 (Refereed publication).**

5. **Chaplin R. I., Hodgson R. M. & Gunetileke S., Seeding an ANN From Problem Knowledge for Image Processing Tasks, Proceedings of the Image & Vision Computing New Zealand International Conference (IVCNZ '99), Christchurch, New Zealand, August 1999, pp 277-282 (Refereed publication).**
6. Chaplin R. I., Hodgson R. M. & Gunetileke S., Towards the Use of Problem Knowledge in Training Neural Networks for Image Processing Tasks, 6th National New Zealand Postgraduate Conference, Christchurch, New Zealand, November 1999.
7. **Chaplin R. I., Hodgson R. M. & Gunetileke S., Seeding an ANN from Problem Knowledge for Image Processing Tasks, *in* Emerging Knowledge Engineering and Connectionist-Based Information Systems, Proceedings of the ICONIP/ANZIIS/ANNES '99 International Workshop, Dunedin, New Zealand, November 1999, pp 83-87 (Refereed publication).**
8. Gunetileke S., Chaplin R. I. & Hodgson R. M., A Tutorial on how to Insert Problem Knowledge into a Fuzzy Neural Network, Proceedings of the 7th Annual New Zealand Engineering and Technology Postgraduate Conference, Palmerston North, New Zealand, November 2000, pp 411-416.
9. **Gunetileke S., Chaplin R. I. & Hodgson R. M., The Use of Problem Knowledge to Improve the Robustness of a Fuzzy Neural Network, *in* Neural Networks for Signal Processing X, Proceedings of the 2000 IEEE Signal Processing Society Workshop, Sydney, Australia, December 2000, pp 682-691 (Refereed publication).**
10. **Chaplin R. I., Gunetileke S. & Hodgson R. M., Initializing Neural Networks with A-Priori Problem Knowledge, *in* Neural Networks for Signal Processing X, Proceedings of the 2000 IEEE Signal Processing Society Workshop, Sydney Australia, December 2000, pp 165-174 (Refereed publication).**