A Domain Independent Approach to 2D Object Detection Based on the Neural and Genetic Paradigms

A thesis submitted for the degree of
Doctor of Philosophy

Mengjie Zhang, B.E., M.E.
Department of Computer Science
Faculty of Applied Science
RMIT University
Melbourne, Victoria, Australia

Supervisors: Dr. Victor Ciesielski
Dr. James Thom

Augest 2000

Abstract

The development of traditional object detection systems usually involves a time consuming investigation of good preprocessing and filtering methods and a hand-crafting of different programs for the extraction and selection of important image features in different problem domains. To avoid these problems, this thesis describes a domain independent approach to multiple class, translation and rotation invariant object detection problems without any preprocessing, segmentation and specific feature extraction. The approach is based on learning/adaptive methods – neural networks, genetic algorithms and genetic programming. Rather than using specific image features, raw image pixel values or pixel statistics are used as inputs to the learning/adaptive systems.

Six object detection methods have been developed and tested. These are (1) the basic approach which uses multilayer feed forward networks trained by the back propagation algorithm, (2) a centred weight initialisation method which improves the performance of the basic method, (3) a method which uses a genetic algorithm to train neural networks, (4) a method which uses a genetic algorithm to refine network weights obtained in the method of the genetic algorithm for network training, (5) a method which uses a genetic algorithm to refine network weights obtained in the basic approach and (6) a method which uses genetic programming to build the object detector.

These methods have been tested on three databases which represent detection problems of increasing difficulty: an easy database of circles and squares, a medium difficulty database of coins and very difficult database of retinal pathologies.

For detecting the objects in all classes of interest in the easy and the medium difficulty problems a 100% detection rate with no false alarms was achieved. On the retina problem the best performance achieved was 100% detection with a 588% false alarm rate. The centred weight initialisation algorithm improved the detection performance over the basic approach on all three databases. Also, refinement of weights with a genetic algorithm significantly improved detection performance on the three databases. The genetic programming based method was particularly effective on the retina pictures. However, the method of using mutlilayer feed forward networks trained by the genetic algorithm did not improved the detection performance on any of the three databases obtained in the basic approach.

The goal of domain independent object recognition was achieved for the detection of relatively small regular objects in larger images with relatively uncluttered backgrounds. Detection performance on irregular objects in complex, cluttered backgrounds is comparable to traditional computer vision methods.

Preface

Preliminary work of neural networks for multiple class object detection, or the basic approach described in chapter 4, was published in the Proceedings of the Annual RMIT Computer Science Postgraduate Students' Conference (CSPSC'97) in December 1997 [217]. The centred weight initialisation method which improved the network training and object detection performance in pixel based neural networks was published in the Proceedings of the Twenty Second Australasian Computer Science Conference (ACSC'99) in February 1999 [220]. An extended version of the paper was given in a technical report in May 1998 [218]. Chapter 5 gives the details of the weight specification method. A paper of using genetic algorithms to improve the accuracy of object detection was published in the Proceedings of the third Pacific-Asia Knowledge Discovery and Data Mining Conference, Knowledge Discovery and Data Mining – Research and Practical Experiences, in April 1999 [35]. A two phase approach of using back propagation algorithm and genetic algorithms to train and refine neural networks for object detection was published in the Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99), Lecture Notes in Computer Science, in August 1999 [223]. An extended version of the paper was presented in the the Annual RMIT Computer Science Postgraduate Students' Conference (CSPSC'98) in December 1998 [219]. Chapter 6 describes the details of the approach. The method of using genetic programming paradigm in multiple class object detection problems was published in the Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99), Lecture Notes in Artificial Intelligence, in December 1999 [221]. An extended version of the paper was presented in the Annual RMIT Computer Science Postgraduate Students' Conference (CSPSC'99) in December 1999 [222]. Chapter 7 extends this work.

The relevant information taken from some of the papers is also contained in the homepage of

`http://goanna.cs.rmit.edu.au/~mengjie/papers.html`

The work in this thesis has not been published elsewhere, except as noted above.

Mengjie Zhang
July 2000, Melbourne, Australia

Declaration

I certify that

- except where due acknowledgement has been made, the work is that of the candidate alone;
- the work has not been submitted previously, in whole or in part, to qualify any other academic award;
- the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Mengjie Zhang

July 2000, Melbourne, Australia

Acknowledgements

I would like to thank my supervisors, Dr. Victor Ciesielski and Dr. James Thom for their invaluable guidance, support and continual encouragement. Victor led me to think how to establish a series of research problems and how to apply learning and adaptive methods to these problems, and provided very helpful feedback along the project. Without his supervision, this thesis would not be possible to finish. James gave me a lot of useful information and discussions about performance measurement and also the presentation of the results. I am also grateful to my consultant, Zhi-Qiang Liu, for a number of discussions on image processing and image understanding techniques.

A special thank must go to my dear wife, Xiaoying Gao, whose entire understanding and support is absolutely important at the beginning of the project. During the course, she gave me continuous support, particularly on domestic aspects.

I am very grateful for the financial support throughout my candidature from the Australia government and RMIT university. The Australian government awarded me the IPRS scholarship (formerly named OPRS), which covers the tuition fees of the course. RMIT provided the living allowance during the project.

Thanks also go to Chris Kamusinski, who provided the retina pictures, and Alex Rosenberg, who gave me very useful support of technical writing in English.

Many thanks must go to the departmental research committee, especially Justin Zobel, Lin Padgham, Zahir Tari, who gave me encouragement during the annual meeting, and also Rita Healy, who gave the full administration work during the course and led me to be familiar with the department when I just came here.

I am thankful for the enjoyable conversions with some other postgraduate research students over technical and non-technical matters, particularly Zhiqi Shen, Simon Ch'ng, Santha Sumanasekara, James Brusey, Surya Nepal, Ken Gardiner, Andy Song, Tom Loveard and Saluka Kodituwakku.

My gratitude goes to my parents and my parents in law who have always encouraged me throughout this project.

Thanks must also go to my former supervisors Professor Pusheng Kuang and Professor Qing Yang for their support of my studying in Australia. Many thanks must also go to my home country, my home university, and the Chinese consulate in Melbourne.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation/Introduction

As more and more images are captured in electronic form the need for programs which can find objects of interest in a database of images is increasing. For example, it may be necessary to find all tumours in a database of x-ray images, all cyclones in a database of satellite images or a particular face in a database of photographs. The common characteristic of such problems can be phrased as "Given $subpicture_1, subpicture_2 ... subpicture_n$ which are examples of the object of interest, find all pictures which contain this object and the locations of all of the objects of interest". Examples of this kind include target detection problems [60, 202] where the task is to find, say, all tanks, trucks or helicopters in a picture. Unlike most of the current work in the object recognition area, where the task is to detect only one object of a single class or to find several objects of a single class in a large picture [60, 157, 161], the goal of this thesis is to detect multiple objects of a number of different classes in a database of large pictures.

The object recognition task using traditional image processing and computer vision methods [13, 23, 48, 214] usually involves the following subtasks: *preprocessing*, *segmentation*, *feature extraction* and *classification*. The main goal of preprocessing is to remove noise or enhance edges. Segmentation aims to divide an image into coherent regions. Feature extraction is concerned with finding transformations to map patterns to lower-dimensional spaces for pattern representation and to enhance class separability. The output of feature extraction, which often is a vector of feature values, is then passed to the classification step. Using these vectors, the classifier determines the distinguishing features of each object class, such that new vectors are placed in the correct class within a predetermined error tolerance. In general, the algorithms or methods for preprocessing and segmentation vary from one problem domain to another. To obtain good performance, some "important" specific features have to be manually determined (selected and extracted) and the classifier has to be built or selected for the specific domain. This thesis focuses on the development of a domain independent method without preprocessing and segmentation for multiple class object detection.

Most work which has been done for object detection and recognition problems uses multiple independent stages[29, 155]. In the multiple stage systems, the current stage uses the results of the previous stage as inputs, rather than the original source data (pictures). The final results rely too much upon the results of each stage. If some objects are lost in one of the early stages, it is very difficult or impossible to recover them in a later stage. To avoid these disadvantages and use the potential benefits of learning-based methods (section 2.2, page 19), this thesis concentrates on the investigation of a single stage, learning/adaptive approach.

In recent years, neural and genetic learning paradigms (neural networks, genetic algorithms and genetic programming) have attracted attention as very promising methods of solving automatic target recognition and detection problems [24, 73, 170, 172, 177, 184, 186, 198, 206]. A wide variety of problem domains have been shown to be amenable to being treated with these learning and adaptive techniques due to the enormous flexibility of the representation afforded. In particular, neural and genetic systems offer potentially powerful learning and adaptive ability and collective computational properties by mimicing human behaviour and evolutionary processes. The automatic, multiple class object detection task is one of the most difficult problems in computer vision and automatic target recognition [155, 161]. While there have been many other recent developments in machine learning such as support vector machines [143, 169], ensembles of classifiers [85] and Gaussian processes [61], we have chosen to work with neural and genetic learning paradigms because they have been successfully applied to automatic object classification tasks and we want to investigate their potential ability to multiple class object detection problems.

In terms of the input patterns of the learning paradigms for object detection and recognition, two main approaches have been used for these kinds of problems – feature based and pixel based. In feature based approaches, various features such as brightness, colour, size and perimeter are extracted from the sub-images of the objects of interest and used as inputs [29, 157, 207]. These features are usually different and specific for different problem domains. In pixel based approaches [36, 91, 172], the pixel values (or pixel level features) are used directly as inputs. Extracting and selecting good specific features for different domains is very time consuming and programs for feature selection and extraction often need to be hand-crafted. These features are not the focus in this thesis. Instead, the main focus will be on the use of pixel data and some domain independent, pixel level image features.

1.2 Goals of the Thesis

This thesis aims to investigate a learning/adaptive, domain independent approach to multiple class, translation invariant and limited rotation invariant object detection problems. The key terms are defined and described as follows.

Definition 1.1 *Object Detection*
Object detection in this thesis refers to the detection of small objects in large pictures. It consists of both object classification, which gives the classes of the objects of interest, and object localisation, which gives the positions of all the objects of interest in the large pictures. This is identical to the term *target detection.*

Definition 1.2 *Multiple Class*
In automatic object detection, in most cases, all the objects of interest are considered as one class of interest. This is usually called the *object* versus *non-object* or *target* versus *non-target* or *object* versus *background* detection problem. In contrast, the *multiple class* detection problem refers to the case where there is more than one class of objects which must be detected. The latter case is the focus in this thesis, which is generally more difficult than the former case.

Definition 1.3 *Translation Invariance, Rotation Invariance and Size Invariance*
Translation invariance means that the objects of interest can be detected no matter where they are located. Rotation invariant detection means that the objects of interest can be detected no matter what angles of orientation they have in the large pictures. Size invariance means that objects can be detected no matter what size they are. The main consideration in this thesis is translation and rotation invariance, but not size invariance.

Definition 1.4 *Domain Independent*
Domain independence means that the methods will work virtually unchanged on a large number of problem types.

Definition 1.5 *Learning and Adaptive*
Learning and adaptive methods can automatically learn possible (candidate) solutions such as

rules, features, programs and networks for a specific task without using any prior knowledge for the given domain. In other words, learning and adaptive methods have to be told precisely *how to learn*, but not *how to directly perform* the task required in solving a specific problem. This is quite different from the traditional object detection/recognition methods which usually perform these tasks by extracting specific image features based on specific domain knowledge. The learning/adaptive techniques used in this thesis are neural networks, genetic algorithms and genetic programming.

As described above, most approaches developed for automatic object recognition or detection normally use specific features extracted from a specific domain as inputs to the classifiers or detectors. These extracted features vary from one problem domain to another. This usually involves the hand-crafting of different programs for object detection in different domains. The goal of this thesis is to avoid the problem of hand-crafting by (1) using *pixel input*, where the raw image pixel data will be used as input of these methods; (2) using a set of domain independent, pixel level features to form *a general pre-existing feature library*, from which the genetic programming process is expected to automatically select just those features relevant to a specific domain.

1.2.1 Hypotheses

To achieve the goal of domain independent object detection, the following specific hypotheses will be explored on a sequence of detection problems of increasing difficulty to determine the strengths and limitations of the different approaches which are introduced in this thesis.

1. Can a basic pixel based neural network approach be developed and applied to multiple class object detection problems? The investigation of the basic approach will be described in chapter 4.

2. Can a centred weight initialisation algorithm be developed to improve the network training and the detection performance over the basic approach? The centred weight initialisation of neural networks for object detection will be described in chapter 5.

3. Can a genetic algorithm, with the fitness of mean squared error, result in an improvement of the neural network training and object detection performance over the backward error propagation algorithm in the basic approach? The genetic algorithm for network training and object detection will be described in chapter 6.

4. Can network refinement with a genetic algorithm, with the fitness based on the detection rate and the false alarm rate, improve the detection performance of the neural networks? The investigation of network refinement based on genetic algorithms will be described in chapter 6.

5. Can the genetic programming learning paradigm be applied to multiple class object detection and produce an improvement in detection performance over the basic approach on the same detection problems? The investigation of genetic programming for multiple class object detection will be described in chapter 7.

1.3 Contributions of the Thesis

The thesis makes the following major contributions:

1. This work has shown how to solve multiple class object detection problems by a single learnt program. Most work in object detection only addresses two class problems, that is, *object* versus *non-object*, or *object* versus *background*. Unlike most current research in this area which uses different programs in multiple independent stages to solve the localisation problem and the classification problem, this work only uses a single trained program (a neural network or an evolved program) for both object classification and object localisation in multiple class object detection problems.

2. This work has shown how object detection systems can be developed with image pixel data input and how traditional specific feature extraction and selection can be avoided. Most research in object detection using traditional image processing and computer vision techniques contains multiple tasks such as preprocessing, segmentation, feature extraction and object classification. The development of these detection systems usually involves a time consuming investigation of good preprocessing and filtering methods, and especially a hand-crafting of different programs for the extraction and selection of specific, important image features for a particular problem domain. This work, however, directly uses the pixel based data (raw image pixel data or domain independent, pixel statistics data) as input to the learning systems in which the features relevant to a particular domain can be automatically learnt through a learning or adaptive process. In this way, traditional preprocessing, segmentation, specific feature extraction and selection are avoided.

3. This work has shown how to develop object detection systems based on neural networks trained by the backward error propagation algorithm and how the detection performance can be improved by use of the specialised initialisation of the network weights and by use of genetic algorithms for network refinement.

4. This work has shown how to extend genetic programming to multiple class object detection. As a relatively new learning paradigm, genetic programming has not previously been applied to multiple class object detection problems. It is evident that genetic programming has a great potential to be applied to difficult problems in the real world. In this thesis, a genetic programming based approach is successfully developed for multiple class object detection by constructing a terminal set, a function set and a fitness function. Twenty domain independent, pixel level image features form the terminal set and the four standard arithmetic operators construct the function set. The objects of interest are classified according to the program classification map. The fitness function is developed based on the detection rate and the false alarm rate of an evolved program. The approach can be successfully applied to different object detection problems of increasing difficulty in which genetic programming learning process can automatically select the features only relevant to a particular problem domain.

1.4 Structure of the Thesis

The remainder of this thesis is organised as follows. After this introduction, chapter 2 presents a detailed review of the literature. Starting from the basic concepts of object detection and machine learning, an overview of neural networks, genetic algorithms and genetic programming is presented. Related work on neural networks, genetic algorithms and genetic programming relevant to object detection is then discussed. Chapter 2 also contains a brief discussion of traditional image processing and computer vision techniques for object detection.

To test the approaches developed and described in this thesis, chapter 3 presents three image databases. These databases are designed to be of increasing difficulty. The analysis of the these databases is also described.

Chapter 4 presents a basic neural network approach for object detection problems. In this approach, the raw image pixel data is used as input to the neural networks. The performance of the basic approach will be used as the baseline for the comparisons with the other methods described later in this thesis.

Chapter 5 describes a new approach to weight initialisation, the *centred weight initialisation algorithm*, on the basis of the pixel based neural network approach. A comparison between this new approach and the basic approach is also given.

To investigate the power of the genetic algorithm for object detection problems, chapter 6 presents a two phase approach to the use of genetic algorithms for object detection problems. In the first phase, the network is trained on the cutouts (sample objects) by a genetic algorithm. The fitness is based on the mean squared error of the cutouts. In the second phase, the weights of trained networks are refined on the entire training images using a second genetic algorithm. The fitness function in this case is based on the detection rate and false alarm rate. A comparison of the new detection methods introduced in this chapter and the basic approach is discussed.

Chapter 7 presents a genetic programming based approach to multiple class object detection problems. In this approach, the evolved programs use a domain independent, pixel level feature set computed from a square input field large enough to contain each of the objects of interest. They are applied, in a moving window fashion, over the large pictures in order to locate the objects of interest. The fitness function is based on the detection rate and the false alarm rate. A comparison between this approach and the basic neural network approach is also presented.

Chapter 8 concludes the main body of the thesis with a comparison of the best results produced with the methods introduced in this thesis. The advantages and disadvantages of these methods are presented. Suggested future work is also discussed.

Chapter 2

Literature Review

In this chapter, we first give a review of the problem domain, including object detection concepts, performance evaluation and examples of object detection work based on traditional image processing and vision techniques. Then we look at the machine learning paradigms and strategies of neural networks, genetic algorithms and genetic programming. Finally, we present a survey of learning and adaptive methods for object detection problems.

2.1 Overview of Object Detection

2.1.1 Object Detection

As can be seen in definition 1.1 (page 3), the term *object detection* used in this thesis integrates both the object classification task and the object localisation task. The former task involves the determination of the classes to which small objects in large pictures belong, and the latter task involves the location of the centres of all the objects in each class of interest. This problem is similar to the commonly used term *automatic target recognition* or *automatic object recognition* and some tasks of image analysis. Here, we give a brief review of such relevant research.

Most research on object detection involves four stages: *preprocessing, segmentation, feature extraction* and *classification* [26, 48, 66, 155], as shown in figure 2.1. The preprocessing stage aims to remove noise or enhance edges. In the segmentation stage, the images are usually divided into a number of coherent regions and "suspicious" regions which might contain objects of interest. As a critical component, a stable, representative feature extraction system should be developed to extract key features for a specific problem domain in the feature extraction stage. The result of feature extraction is normally a feature vector. Finally, a classifier needs to be developed or selected, which will use these features to distinguish the classes of the objects of interest. The algorithms or methods for these stages are generally domain dependent, particularly when using traditional image processing and computer vision techniques. Learning paradigms such as neural networks or genetic programming approaches have usually been applied to the classification stage.

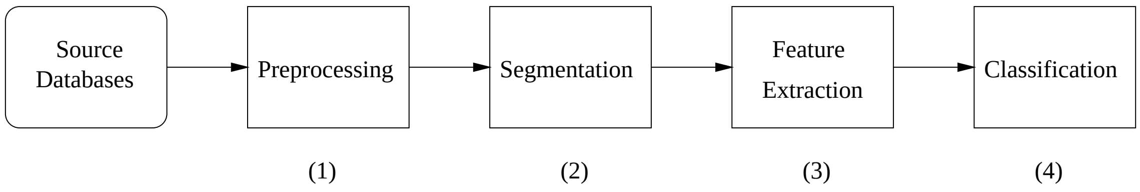


Figure 2.1: A typical procedure for object detection: four stages.

2.1.2 Main Aspects of Object Detection

We classify object detection problems according to three different criteria: the number of classes of interest, the input of the detector/classifier and the number of independent stages used in the detection system.

One Class versus Multiple Class

Regarding the number of object classes of interest in a single picture, there are two main types of detection problems:

- **One class object detection problem**, where there are multiple objects in each picture, however they belong to the same (single) class of interest. One special case in this category is that there is only one object of interest in each source picture. In nature, these problems contain a two class classification problem: *object* versus *non-object*, also called *object* versus *background*. Examples are detecting small targets in thermal infrared images [172] and detecting a particular face in photograph images [119].

- **Multiple class object detection problem**, where there are multiple object classes of interest each of which has multiple objects in each picture. Detection of handwritten digits in postal code images [114] is an example of this kind.

In general, multiple class object detection problems are more difficult than one class detection problems. This thesis is focused on detecting multiple objects for each of the multiple classes in a set of pictures, which is particularly difficult. Most research in object detection which has been done so far belongs to the one class object detection problem, or even only object classification problems (section 2.1.4 on page 18, section 2.5 on page 49, section 2.6 on page 64 and section 2.7 on page 67).

Feature Based versus Pixel Based

In terms of the input of classifiers/detectors, there are also two major approaches, as follows:

- **Feature based approach**, which uses specific image features as inputs to the selected or developed detectors or classifiers. The features, which are usually highly domain dependent, are extracted from the images or segmented images (object and non-object samples). This approach generally involves time consuming hand-crafting of the appropriate feature extraction programs. In a lentil grading and quality assessment system [207], for example, features such as brightness, colour, size and perimeter are extracted and used as inputs to a neural network classifier.

- **Pixel based approach**, where the raw pixel data of the image are directly used as inputs to the detector or classifier. In this way, the hand-crafting of features can be removed. Instead, this approach usually needs learning and adaptive techniques to learn features for the detection task. Directly using raw image pixel data as input to neural networks for detecting vehicles (tanks, trucks, cars, etc.) in infrared images [60] is such an example.

  A special case here is that detectors or classifiers use a number of simple pixel based features as inputs, which are either extracted from the images or selected from a preexisting feature library. Such features are domain independent, for example *mean* and *variance*, and are quite different from the specific features for a particular domain. In nature, these features still belong to the pixel level and we call them *domain independent, pixel level features.* These kinds of features are also called *pixel statistics* in [82]. Thus detection methods based on such features can be still considered as pixel based approaches.

In the recent literature, most work has concentrated on the specific feature based approach, while only a few papers used the pixel based approach (sections 2.1.4 on page 18, section 2.5

on page 49, section 2.6 on page 64 and section 2.7 on page 67). To avoid the hand-crafting problem and to investigate a domain independent approach, this thesis uses pixel based data (raw image pixel data or pixel statistics) as inputs to neural networks, genetic algorithms and genetic programming for multiple class object detection problems. More details of the learning and adaptive techniques are presented in section 2.2 (page 19), section 2.3 (page 24), section 2.4.1 (page 37) and section 2.4.2 (page 42).

Multiple Stage versus Single Stage

According to the number of independent stages used in the detection procedure, we divide the detection methods into two types:

- **Multiple stage detection**, which uses multiple independent stages for object detection. Traditionally, there are four typical stages, *preprocessing, segmentation, feature extraction* and *object classification*, as shown in figure 2.1. In general, each independent stage needs a program to fulfill that specific task and accordingly multiple programs are needed for object detection problems. Success at each stage is critical to achieving good final detection performance. Detection of trucks and tanks in visible, multi-spectral infrared and synthetic aperture radar images [202] and recognition of tanks in cluttered images [29] are two examples.

- **Single stage detection**, which uses only a single stage to detect the objects of interest in large pictures. There might be more than one phase in such a system, however these phases are not totally independent. Typically, the goal of a latter phase is to refine or continuously improve the program produced in a former phase. A key characteristic of these kinds of methods is that there is only a single program produced for the whole object detection procedure. Detecting tanks in infrared images [209] and detecting small targets in cluttered images [172] based on a single neural network are examples of these kinds.

Most recent work on object detection problems is based on multiple stages. This thesis focuses on using a single program (a neural network or a computer program evolved by genetic programming) for object detection problems.

2.1.3 Performance Evaluation

This section presents an overview of the most commonly used performance measures for object classification and detection: *accuracy*, *true positive fraction and false positive fraction*, *detection* *rate and false alarm rate*, and the related *ROC curves.*

Accuracy

For an object detection/classification system, *accuracy*, also called *percent correct*, is defined as the number of objects which are correctly detected/classified as a percentage of the total number of desired objects in the database, as shown in equation 2.1.

$$Accuracy = \frac{N_{detected}}{N_{desired}} \times 100\% \tag{2.1}$$

$N_{detected}$ denotes the number of objects correctly detected or classified by the detection or classification system, and $N_{desired}$ represents the number of desired objects in the data set.

It is clear that this measure has the fairly obvious limitation in that it does not reveal the relative frequencies of false positive and the false negative errors [127], which usually are important for object detection/classification problems.

TPF, FPF and Standard ROC Curves

True positive fraction (TPF) and false positive fraction (FPF) are frequently used in disease diagnostic systems [127]. The terms *true positive fraction* and *true negative fraction* (TNF) are synonymous with "sensitivity" (the fraction of patients actually having the disease in question that is correctly diagnosed as "positive") and "specificity" (the fraction of patients actually without the disease that is correctly diagnosed as "negative"), respectively [123, 126]. In a complementary way, *false negative fraction* (FNF) and *false positive fraction* represent the conditional probabilities or frequencies with which actually positive (having the disease) and actually negative (non-disease) patients are diagnosed incorrectly [123, 126]. To give a clear view of these terms, we present them in table 2.1.

		Actual Situation	
		Disease	Non-Disease
Diagnosed Situation	Disease	True Positive Fraction (TPF)	False Positive Fraction (FPF)
	Non-Disease	False Negative Fraction (FNF)	True Negative Fraction (TNF)

Table 2.1: TPF, FPF, FNF and TNF.

According to these definitions, $FNF = 1 - TPF$, and $FPF = 1 - TNF$. Because of the interrelationships among these measures, it is necessary only to indicate a single pair, either TPF and TNF or TPF and FPF are employed.

If one attempts to use a pair of these measures to compare the performance of two or more diagnostic systems, an ROC curve, or Receiver Operating Characteristic curve (also known as Relative Operating Characteristic curve), can be applied [127].

Standard ROC curves conventionally take the *FPF* as the x axis, and the *TPF* as the y axis. A typical standard ROC curve is shown in figure 2.2 (a).

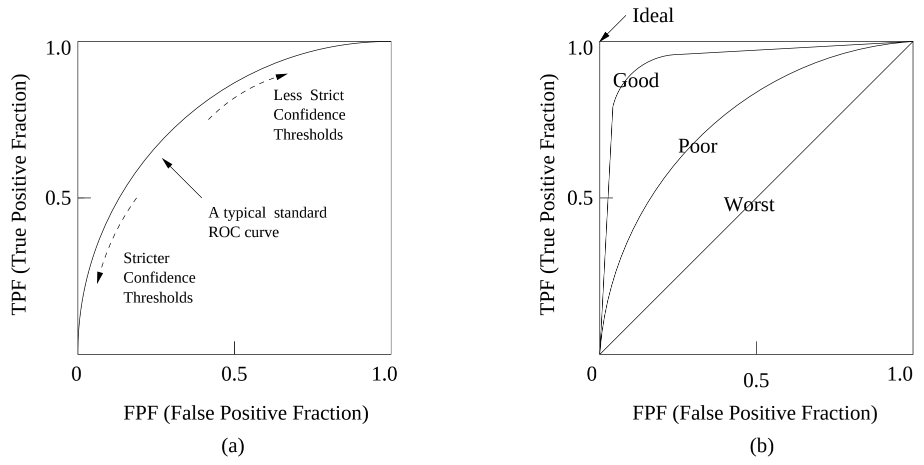


Figure 2.2: Standard ROC curves. (a) A typical standard ROC curve; (b) The quality of classifiers with ROC curves: ideal, good, poor and the worst cases.

In the ROC curve, different confidence thresholds correspond different points, which represent different pairs of TPF and FPF. A higher ROC curve (such as the "good" case in figure 2.2 (b)) would indicate greater discrimination capacity, because a larger value of TPF at each value of FPF – or, equivalently, a smaller value of FPF at each value of TPF – can be achieved on a higher curve if an appropriate confidence threshold is used. Similarly, a lower ROC (such as the "poor" case in figure 2.2 (b)) would indicate less classification capacity. In addition, two extremes of such a measure correspond to the worst case and the best (or the "ideal") case. The worst classification or diagnostic system, on the one hand, is usually defined as that which has no discrimination between positives and negatives. A positive will have equal chance of being interpreted as a positive or a negative, and vice versa. This means that true positive and false positive fractions are equal, or TPF = FPF, which corresponds the straight line between (0, 0) and (1.0, 1.0), that is, the diagonal in figure 2.2 (b). The ideal system, on the other hand, represents perfect interpretation which follows a TPF = 1.0 for all values of FPF. This corresponds to the top-left corner in figure 2.2 (b).

Detection Rate, False Alarm Rate and Extended ROC Curves

The standard ROC curve measure has been widely applied to classification systems [123, 127, 142]. However, it has some limitations for object detection systems: the total number of objects of interest and non-objects must be known and relatively close. In object detection problems, the number of non-objects is close to the total number of pixels of the background. It is very common that there are only a few objects of interest (typically 1-20) and there are a large number of non-objects (in a picture with *1000*$\times$*1000* pixels, the number is about 10^6). In this case, the FPF (the number of non-objects incorrectly classified as actual objects as the proportion of the total number of non-objects) will be always very small and accordingly the results obtained by different detection systems will always be very close to the ideal. This makes it very difficult to compare these systems. To solve this problem, *detection rate, false alarm rate* and *extended ROC curves* are used to evaluate the performance for object detection problems [172].

Detection Rate. The detection rate (DR) refers to the number of small objects correctly reported by a detection system as a percentage of the total number of known (or desired, actual) objects in the database. For multiple class object detection systems, if we have the terms in table 2.2, then the detection rate for a single class and the overall detection rate (for all the

Terms	Description
n	Total number of pictures in image database
m	Total number of object classes of interest
DR_i	Detection rate for class i
DR	Overall detection rate (for all the classes)
FAR_i	False alarm rate for class i
FAR	Overall false alarm rate (for all the classes)
$N_{known}(i,j)$	Number of the actual known objects for the ith class in the jth picture in database
$N_{reported}(i,j)$	Number of the objects reported by a detection system for class i in picture j
$N_{true}(i,j)$	Number of the objects correctly reported by a detection system

Table 2.2: Terms used in calculating detection rate and false alarm rate in a multiple class object detection system.

classes of interest) can be computed according to equation 2.2 and equation 2.3.

$$DR_i = \frac{\sum_{j=1}^{n} N_{true}(i,j)}{\sum_{j=1}^{n} N_{known}(i,j)} \times 100\% \tag{2.2}$$

$$DR = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} N_{true}(i,j)}{\sum_{j=1}^{n} \sum_{i=1}^{m} N_{known}(i,j)} \times 100\% \tag{2.3}$$

False Alarm Rate. The false alarm rate (FAR) here refers to the number of non-objects incorrectly reported as actual objects by a detection system as a percentage of the total number of desired known objects in the database. Similarly to the detection rate, the false alarm rate for the *ith* class and the overall false alarm rate are illustrated in equation 2.4 and equation 2.5.

$$FAR_i = \frac{\sum_{j=1}^{n} N_{reported}(i,j) - \sum_{j=1}^{n} N_{true}(i,j)}{\sum_{j=1}^{n} N_{known}(i,j)} \times 100\% \tag{2.4}$$

$$FAR = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} N_{reported}(i,j) - \sum_{j=1}^{n} \sum_{i=1}^{m} N_{true}(i,j)}{\sum_{j=1}^{n} \sum_{i=1}^{m} N_{known}(i,j)} \times 100\% \tag{2.5}$$

It is important to note that –

- The main goal of object detection is to obtain a high detection rate and low false alarm rate. There is, however, a tradeoff between them for a detection system. Trying to improve the detection rate often results in a increase in false alarm rate, and vice versa.

- While the definition of detection rate is similar to that of true positive fraction, the false alarm rate is different from false positive fraction in classification or diagnostic systems. The ranges of detection rate (or true positive fraction) and false positive fraction are both between 0 and 1.0. The false alarm rate is also called false alarms per object or *false alarms/object* [172], which may be greater than one or 100% in difficult object detection problems.

- Detecting objects in pictures with very cluttered backgrounds is an extremely difficult problem where false alarm rates of 200-2000% (that is the detection system suggests that there are 20 times as many objects as there really are) are common [161, 172].

Extended ROC Curve. Extended ROC curves use the false alarm rate as the x axis and the detection rate as the y axis. A sample extended ROC curve is presented in figure 2.3 (a). Similarly to standard ROC curves, a higher extended ROC curve represents greater detection capacity. There is also an ideal case, which refers to the situation that all the objects of interest can be correctly detected with no false alarms at all, corresponding to the top-left corner in figure 2.3 (b). Unlike the standard ROC curve which has the worst case corresponding to the line between (0, 0) and (1.0, 1.0) (or the diagonal in figure 2.2 (b), page 14), a very poor extended ROC curve can be very low (below the diagonal), even very close to the x axis.

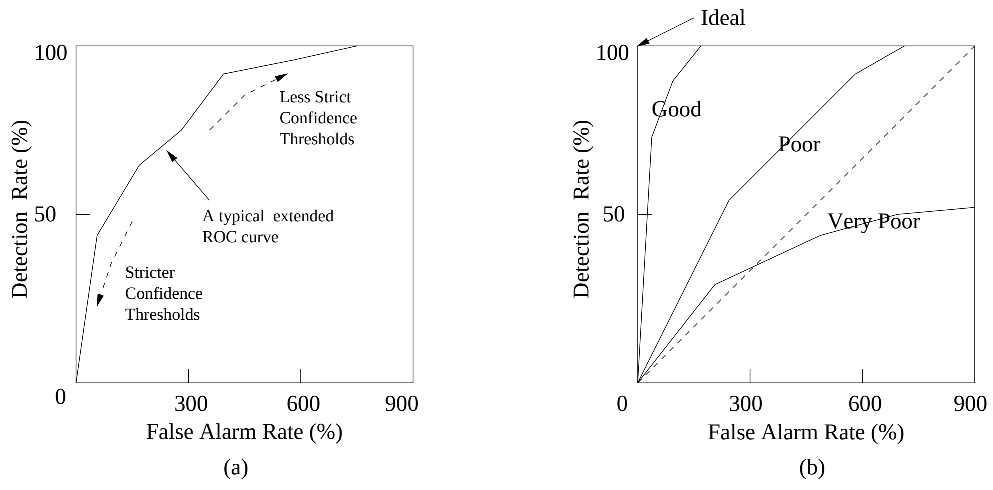


Figure 2.3: Extended ROC curves. (a) A typical extended ROC curve; (b) The quality of detectors with extended ROC curves: ideal, good, poor and very poor cases.

Important Note

It is very important to note that most research which has been done in this area so far only presents the results of the classification stage (only the final stage in figure 2.1, on page 10) and assumes that all other stages have been properly done. However, the results presented in this thesis are the performance for the whole detection problem (including the localisation and classification). Thus, it is unfair to simply compare the results presented in this thesis with those obtained by other systems which are developed only for the classification problem. It is not a fair comparison either if one compares the performance presented in this thesis with that only for one class detection problems.

2.1.4 Examples of Conventional Object Detection

This subsection gives some sample work using traditional techniques for object detection.

Brunelli and Poggio [23, 24] developed and implemented two algorithms for face recognition. The first algorithm is based on the computation of a set of geometrical features such as nose width and length, mouth position and chin shape. The second is based on almost-grey-level template matching. The algorithms were tested on a common database of 188 images, four for each of 47 people (26 males and 21 females). About 90% classification accuracy was achieved by using geometrical features and perfect recognition was obtained by using template matching. In [23] the authors presented the details of the features and experiments.

Samaria and Harter [167] present a set of experiments of using continuous density Hidden Markov Models for human face identification. Through these experiments, different parameterisations were assessed using their success rates in identifying 200 images from a database of 40 people. The results indicated that a large overlap in the sampling resulted in better recognition performance and as the overlap becomes noticeable the effect of the window height is limited.

Yla-Jaaski and Ade [213] present a method of segmenting grey-value images into objects and of recognising the detected objects. Starting from edge maps, the method extracted axial descriptions of symmetrical shapes. Initially, a piecewise linear approximation of the binary edge map was obtained. From any two of the resulting linear segments, a Linear Segment Pair (LSP) was formed and several of its attributes were computed. These attributes were used to select or reject the LSPs through symbolic rules and coarse numeric thresholds. Grouping the LSPs into couples was governed by additional attributes and rules. The applications to shape recognition, object recognition, and stereo correspondence were presented. The authors claimed that the approach was useful for a broad range of images. It was applied to a robot vision system which was capable of manipulating three dimensional, overlapping, real-world objects in close to real time.

2.1.5 A Broad View of Neural and Genetic Learning Methods for Object Detection

In this section, we present a very broad view of neural and genetic learning methods for object detection related problems, as shown in figure 2.4. The details of object detection related work using neural networks, genetic algorithms and genetic programming can be seen in sections 2.5 (page 49), 2.6 (page 64) and 2.7 (page 67).

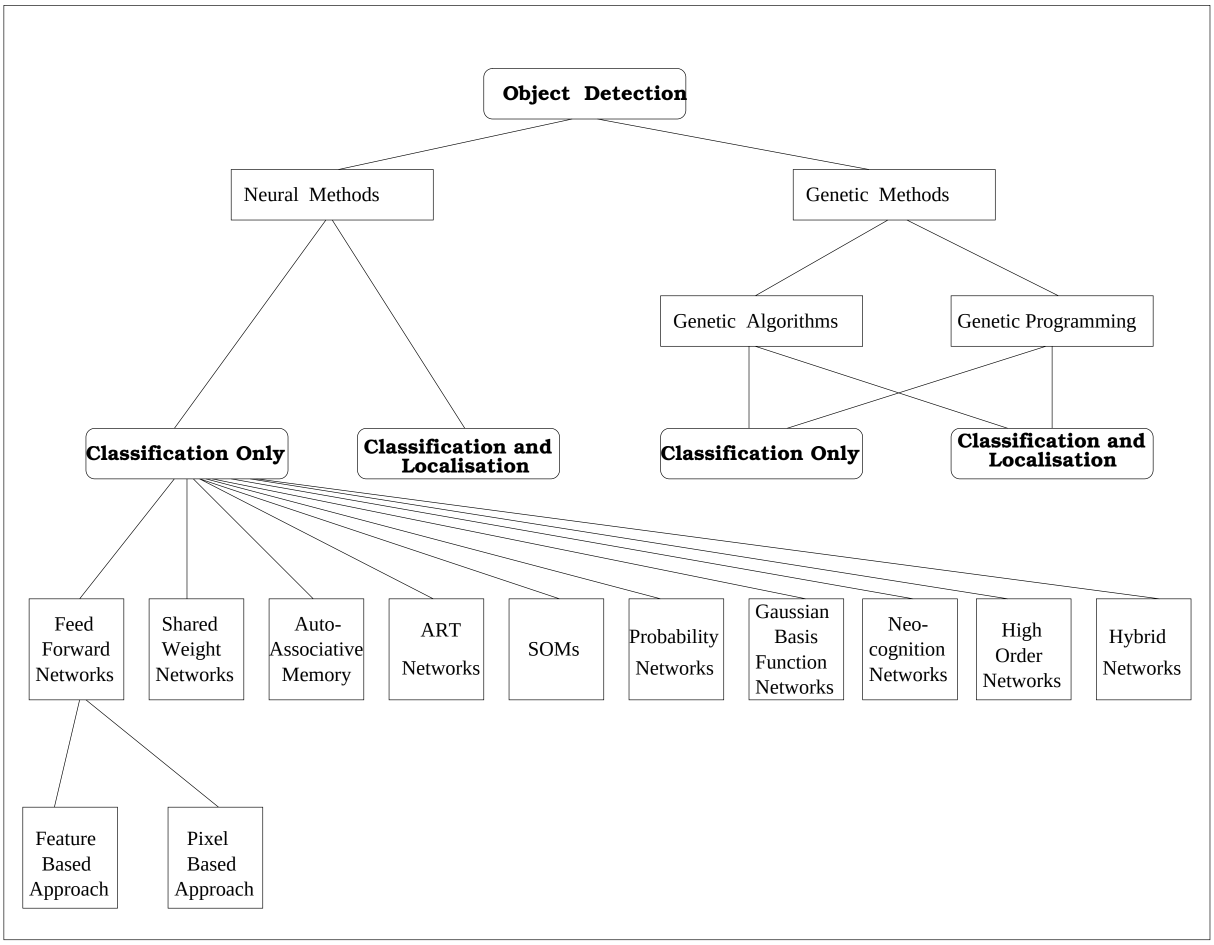


Figure 2.4: A broad view of neural and genetic learning methods for object detection

2.2 Overview of Machine Learning

The goal of this thesis is to investigate a learning/adaptive approach for object detection problems. Since machine learning is a large area, it is not necessary to review all of the aspects of machine learning here. Instead, we only review the basic concepts, learning strategies and learning paradigms that are directly related to the thesis.

2.2.1 Basic Concepts and Definitions

The ability to learn is a fundamental trait of intelligence. In general, the main goal of machine learning is to learn or discover some kind of knowledge or features from a data set, and to use them on unseen data set. But what is the definition of machine learning? A precise definition of learning is difficult to formulate. We give some commonly accepted descriptions or definitions of machine learning as follows.

Friedberg [53, page 2] and [54, page 285] framed the central issue of machine learning as follows:

> If we are ever to make a machine that will speak, understand or translate human languages, ... we must reduce these activities to a science so exact that we can tell the machine precisely how to go about doing them or we must develop a machine that can do things without being told precisely how ... . We could teach this machine to perform a task even though we could not describe a precise method for performing it, provided only that we understood the task well enough to be able to ascertain whether or not it had been done successfully. ... In short, although it might learn to perform a task without being told precisely how to perform it, it would still have to be told precisely how to learn.

Michalski, Carbonell and Mitchell [129, pages 1, 28] described the machine learning as follows:

> Learning is a many-faceted phenomenon. Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organisation of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation. ... The study and computer modelling of learning processes in their multiple manifestations constitutes the subject matter of machine learning. ... Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

Mitchell [132, pages xv, 1] further stated that

> Machine learning is the study of computer algorithms that improve automatically through experience.

Jain, Mao and Mohiuddian [90, pages 34, 35] described learning process as follows:

> To understand or design a learning process, one must first have a model of the environment, or determination of a learning paradigm. Second, one must understand how the learning system works, or the definition of the learning rules. A learning algorithm usually refers to the procedure in which the learning rules are used for improving the learning system.

Banzhaf, Nordin, Keller and Francone [15, page 9] claimed that

> Machine learning is a process that begins with the identification of learning domain and ends with testing and using the results of the learning. ... ... The key parts of this process are the "learning domain", the "training set", the "learning system", and "testing" the results of the learning process.

2.2.2 Training Set and Test Set

Definition 2.1 *Training Set*
A collection of input patterns or instances from which the rules or features are induced is usually called a training data set, or a training set for short.

Definition 2.2 *Test Set*
A collection of input patterns or instances which were never used or *unseen* when the rules or features were learnt is known as a test data set, or a test set for short.

In the usual case, a training set is used for finding or learning rules or features, while a test set is for measuring the performance of these learnt rules or features.

2.2.3 Generalisation Ability

One of the major advantages of a learning system is the ability to learn/extract the useful features from the training data set and to apply these features to the test data. The generalisation ability depends on how well the learning system has modelled the relationships in the training set. If the training set contains all the possible relationships between all the cases, then the learned program, once trained, should give good performance on the test data. There are two important issues here: *overtraining* or *overfitting*, which will be discussed in "Early Stopping" in section 2.3.3 (page 29), and the *training set size*, which will be discussed in section 2.3.5 (page 32).

2.2.4 Three Learning Strategies

There are three main learning strategies: *supervised, unsupervised*, and *hybrid* [90].

Supervised Learning

In supervised learning, or learning with a teacher, the learning system is provided with a correct answer (desired output) for each input pattern. The learning process is continued until the system produces answers as close as possible to the known desired correct answers.

Reinforcement learning is a variant of supervised learning in which the learning system is provided with only a critique or a clue about the correct answers, rather than the desired outputs themselves.

Unsupervised Learning

Unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It usually explores the underlying structure in the data, or correlations between patterns in data, and organises patterns into categories from these correlations.

Hybrid Learning

Hybrid learning combines both supervised and unsupervised learning. Part of the solutions (network weights, architecture, or computer programs) are determined through supervised learning, while the others are obtained through unsupervised learning.

In this thesis, the focus will be on using the first strategy, that is, supervised learning.

2.2.5 Main Learning Paradigms

There is no commonly agreed taxonomy of machine learning paradigms. It is, however, possible to summarise the major paradigms:

- **The Connectionist Paradigm**. Connectionist learning systems are also called *artificial neural networks* (ANNs or NNs) or *parallel distributed processing systems* (PDPs) [166]. Neural networks are algorithms for cognitive tasks, such as learning and optimisation [134], which are based on concepts derived from research of the human brain. A network is generally constructed from nodes, links, weights, biases, and transfer function. The network weights are automatically updated through network training by the learning algorithm, which can be selected or developed based on the architecture of the network. The main concepts of neural network learning are described in section 2.3 (page 24).

- **The Genetic Paradigm**. Genetic learning systems originally refer to *genetic algorithms* (GAs) [64], which are search algorithms based on the mechanism of natural selection and natural genetics. They usually use *bit strings*, which are generally called *chromosomes*, to represent candidate solutions. During the evolutionary process, the genetic operators, *selection, crossover* and *mutation* are applied in order to generate fitter solutions. An overview of genetic algorithms is presented in section 2.4.1 (page 37).

  Since the 1990s, *genetic programming* (GP) [101] has become another important genetic learning paradigm. Unlike genetic algorithms which use the bit strings as inputs and outputs, genetic programming uses a terminal set and a function set as inputs and evolved programs as outputs of the learning system. The main idea of genetic programming learning paradigm is presented in section 2.4.2 (page 42).

- **The Case Based Learning Paradigm**. Case based learning systems, also known as *instance based learning*, represent knowledge in terms of specific cases or experiences and rely on flexible matching methods to retrieve similar cases and apply them to new situations. One of the common schemes of this paradigm is *nearest neighbour learning*, which finds the stored case nearest to the current situation according to some distance measure and then uses it for classification or prediction.

- **The Induction Learning Paradigm**. Induction learning systems are characterised by deriving or inducing a general rule from a set of examples [149], and usually employ decision trees, condition-action rules, or similar knowledge structures. Nodes in a decision tree involve testing a specific attribute, which usually compares the value of an attribute with a constant. Some (sub) trees, however, compare two or more attributes. Leaf nodes give a classification that applies to all examples which reach the leaf or a probability distribution over all possible classifications. To test the learnt decision tree, unknown examples can be applied by routing the tree according to the values of the attributes tested in successive nodes. When a leaf is reached examples are classified according to the class assigned to the leaf.

- **The Analytic Learning Paradigm**. Analytic learning systems represent knowledge as rules in logic form, but typically employ a performance system that solves multi-step problems using some search process. A common technique is to represent knowledge as Horn clauses, then to phrase problems as theorems and to search for proofs. Learning in this framework uses background knowledge to construct explanations (or proofs), then compiles the explanations into more complex rules that can solve similar problems. Most work on analytic learning has focused on improving the efficiency of search, but some has dealt with improvement of the classification accuracy.

In this thesis, we focus on the first two paradigms, that is, neural networks, genetic algorithms and genetic programming.

2.3 Overview of Neural Networks

Artificial neural network research has experienced three periods of extensive activity [90]. The first peak appeared in the 1940s, when McCulloch and Pitts [125] introduced a binary threshold unit as a computational model for an artificial neuron and described a logical calculus of neural networks. The second occurred in the 1960s due to Rosenblatt's *perceptron* convergence theorem [158] and Minsky and Papert's work which showed the limitations of a simple perceptron [131]. Minsky and Papert's results greatly decreased the enthusiasm of many researchers, especially those in the computer science community [90]. This resulted in a long "pause" in neural network research for almost 20 years. The third period came from the early 1980s and since then neural network research has received considerable renewed interest. The major work includes Hopfield's energy approach [80] in 1982 and the backward error propagation learning algorithm for multilayer perceptrons (multilayer feed forward networks) popularised by Rumelhart et al. [166] in 1986[1]. With the development of various of types and applications, artificial neural network research has gradually become one of the two main strands (*symbolism* versus *connectionism*) in the area of artificial intelligence.

Many different types of neural networks have been developed and applied to object detection and vision problems (section 2.5, page 49). These include multilayer feed forward networks, recurrent networks, Hopfield networks, ART networks and self organising maps. The various types of networks and neurons are too numerous to describe here. The neural networks used in this thesis are multilayer feed forward networks with the nodes having a logistic activation function. An overview of the main terms and concepts of these kinds of neural networks is given in the rest of this section.

[1]The backward error propagation algorithm was first proposed by Werbos [203] in 1974.

2.3.1 Terminology

A *neuron* (also called a *unit* or *node*) is the basic computational cell in a neural network. A node can have several *links* carrying signals into or out of it. Links are also referred to as *connections.* Each link has an associated *weight.* The weight can be looked upon as a factor which strengthens or weakens a signal which is fed into the link. In addition, a node usually has also an associated *bias*, which can be viewed as a constant input to the node from a "virtual" unit in the network. An artificial neural network in general is a collection of such units linked to each other in some manner. A typical unit or node is presented in figure 2.5.

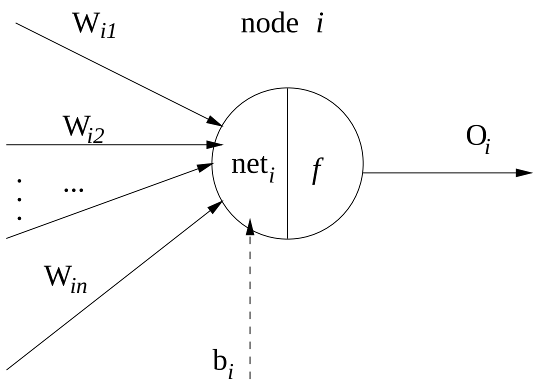


Figure 2.5: A typical network node or neuron

In figure 2.5, the links, input weights, bias and the activation function (or transfer function, f) are presented. The input of the node i, net_i, is computed according to equation 2.6.

$$net_i = \sum_{k=1}^{n} W_{ik} \cdot O_k + b_i \tag{2.6}$$

Here n is the number of the nodes connecting into node i, W_{ik} is the weight on the connection to node i from node k, O_k represents the activation (output) of node k, b_i stands for the bias of node i. If node i is an input node, then $O_k = 1$ and $b_i = 0$, that is, input nodes usually have zero bias and the net input of an input node is the value of the input pattern at that node. Input nodes have their activations set to a value determined by the input pattern. The activation at non-input nodes is usually determined by applying an activation function[2] to the net input to

[2]The term *output function* can be different from the *activation function.* The term *transfer function* often denotes the combination of activation function and output function. Here we do not distinguish these terms. In other words, the three terms are considered as the same in this thesis.

that node. The most commonly used activation function, also used in this thesis, is the *logistic* or *sigmoid* function. According to this idea, the output of the node i, O_i, is presented in equation 2.7.

$$O_i = f(net_i) = \frac{1}{1 + e^{-\beta \cdot net_i}} \tag{2.7}$$

where β is a constant which can be used to change the shape of the curve of function f. The default value of β is 1.0, which is used in this thesis.

Neural networks also distinguish their nodes as being input nodes, hidden nodes and output nodes. The input nodes receive signals from outside the network, for example, features or attributes for a problem domain. The output nodes collectively hold the results of the neural computation, for instance, the labels of the classes. The nodes between the input and output nodes are called intermediate or hidden nodes. There is only one input layer and one output layer in multilayer feed forward networks, however there might be more than one hidden layer in the network architecture. If there is more than one hidden layer, the layers, from the input layer to the output layer, are usually called the first hidden layer, the second hidden layer, etc. In this way, each layer except the input layer has incoming connections from the previous layer, and each layer except the output layer has the connections outgoing to the next layer. Figure 2.6 shows a typical three layer feed forward neural network.

In figure 2.6, there is only one hidden layer. The theoretical results provided by Irie and Miyake [87] and Funahashi [59] have proved that any continuous mapping can be approximated by a network with a single hidden layer, which means that one hidden layer is sufficient for any practical purpose and there is no need for more than one hidden layer. This architecture is used in this thesis. The collection of the inputs applied to the input nodes at one time constitutes an *input pattern* and the corresponding outputs produced form an *output pattern.* In supervised learning there is also a *target pattern* for each input pattern. Comparing the network actual outputs and the target patterns can decide whether the network has learnt the tasks.

2.3.2 Network Training

In object classification, the main goal of the network training is to learn weights to distinguish different classes of objects and the background. The main idea of the network training is as follows.

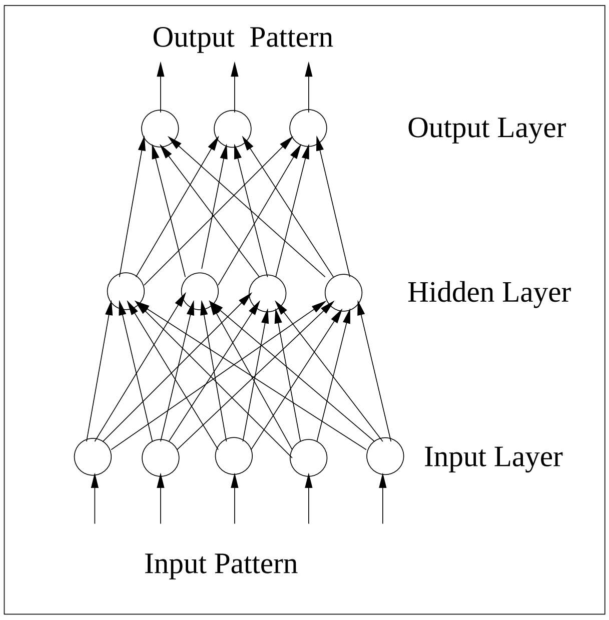


Figure 2.6: A typical three layer feed forward neural network.

Each input pattern in the training set is presented to the network which produces the *actual output.* This is compared with the target output. If the actual output matches (to some desired level of precision) the target output, the neural network is said to have been trained. Otherwise, the internal weights and biases of the network nodes need to be adjusted in such a way as to produce the target outputs. An *epoch* is a *cycle* in training which consists of presenting all the input patterns in the training set, passing the signals through the network and calculating the outputs, and adjusting the weights of the network if the actual outputs and the target outputs do not match. Typically the network weights have to be repeatedly adjusted over a number of epochs until a certain criterion is reached. This is called a *termination strategy.* Termination strategies are discussed in section 2.3.3 (page 29).

The Backward Error Propagation Algorithm

There are several algorithms for training multilayer feed forward neural networks, such as the backward error propagation algorithm, the back percolation algorithm, and the quickprop algorithm [216]. Among these, the *backward error propagation* (*BP*) is the most commonly used one. It is well known that BP networks can perform well for general, relatively simple classification problems. In this thesis, we will investigate whether these kinds of networks can do a good job for a variety of object classification and detection problems. The details of the training algorithm are described by Rumelhart et al. [76, 165]. Only a brief overview is presented here.

Training a feed forward neural network with the BP algorithm consists of the following procedure:

- *Forward propagation phase*: An input pattern is presented to the network. The input is then propagated forward into the network until activation reaches the output layer.
- *Backward propagation phase*: The output is then compared with the target output. The error, i.e. the difference (*delta*) δ_j between the output o_j and the desired output t_j of a target output node j is then used together with the output o_i of the source node i to compute the necessary changes of the weight w_{ij}, or Δw_{ij}. To compute the deltas of the weights into the hidden nodes, for which no teaching inputs are available, the deltas of the following layer, which have already been obtained, are used. In this way the errors (deltas) are propagated backward.

The weight changes in the BP algorithm are computed according to equation 2.8 and equation 2.9 [165, 216]:

$$\Delta w_{ij} = \eta \delta_j o_i \tag{2.8}$$

$$\delta_j = \begin{cases} o_j(1-o_j)(t_j-o_j) & \text{if node } j \text{ is an output node} \\ o_j(1-o_j)\sum_k \delta_k w_{jk} & \text{if node } j \text{ is a hidden node} \end{cases} \tag{2.9}$$

where:

i: index of a predecessor to the current node j with link w_{ij} from i to j;

j: index of current node;

k: index of a successor to the current node j with link w_{jk} from j to k.

η: learning rate;

δ_j: error of node j;

t_j: teaching input (or the target output, or the desired output) of node j;

o_i: output of the preceding node i.

Variations of the BP Algorithm

Momentum: The BP algorithm is basically a gradient descent scheme. In practice, several variations can be made to its actual implementation. A common one is the so-called *Backprop-Momentum* [165, 216] in which the momentum component is introduced and calculated from the delta values for the previous epoch, as shown in equation 2.10.

$$\Delta w_{ij}(t+1) = \eta\delta_j o_i + \alpha\Delta w_{ij}(t) \tag{2.10}$$

Here:
$\Delta w_{ij}(t+1)$: weight change in the epoch;
$\Delta w_{ij}(t)$: weight change in the previous epoch;
α: a constant specifying the momentum.

However, the use of momentum is somewhat controversial [189]. In this thesis, momentum is not used.

Online learning: At what stage are the network weights updated when training patterns are presented? There are, in general, two possibilities [42, 114]. One is *online learning* (sometimes called the *stochastic gradient procedure*), where the weight changes are applied to the network after each training pattern. The second is *off-line learning* (sometimes called the *true gradient procedure*) in which the weight changes are accumulated for all training patterns and the sum of all changes is applied after one full epoch. Online learning was found to converge much faster than the off-line learning on image data [42, 114] and is used in this thesis.

Fan-in: Another variation of network training is weight initialisation and weight changes modified by the *fan-in* factor [42, 116, 164]. The fan-in is the number of elements that either excite or inhibit a given node of the network. The weights are divided by the number of inputs of the node to which the connection belongs before network training and the size of the weight change of a node is updated in a similar way during network training. In this thesis, the weight change of a node is divided by the fan-in.

2.3.3 Termination Strategies

There are a number of different ways of determining when to stop training. The main ones are:

1. The *epoch/cycle control* strategy, where the training will keep going until the training epochs/cycles reach a user defined number.

2. The *error control* strategy, which uses the mean squared error (MSE) as termination

criterion. When the MSE of the training set is smaller than a user defined value, training will be stopped. Sometimes, the total sum squared error (TSS) is also used for this purpose. The definitions of the TSS and the MSE are presented in section 2.3.4 (page 30).

3. The *proportion control* strategy. When the proportion of the number of patterns correctly classified among the number of total training set reaches a pre-defined percentage, the training will be terminated. A classification is regarded as correct if the output node with the largest activation value (actual output activation value) among all the output nodes is identical to that whose target output value is 1.0 (or 0.9). Otherwise, this pattern is incorrectly classified.

4. The *early stopping* strategy. This strategy is used to avoid overfitting. *Overfitting*, also called *overtraining*, means the trained network can achieve high accuracy on the training set but produces low accuracy or a high error rate on the test set. This is mainly due either to too many epochs performed in network training or too many hidden nodes used in the network architecture. To obtain good generalisation, a third data set, the *validation set*, can be introduced [216] and the number of hidden nodes (and/or hidden layers) should be carefully determined. The determination of the number of hidden nodes will be described in section 4.4.3 (page 97). The validation set is separated from the training set, but it is used for network training. A typical sample of network training with a training set and a validation set is presented in figure 2.7.

   In this figure, the lower curve and the upper curve show the network learning on the training set and the validation set respectively. The training should be stopped at the minimum of the validation set error, that is, when the mean squared error (MSE) reaches 0.13 at about 250 epochs. At this point the network generalises best and overtraining can be avoided.

5. The *user control* strategy, where the user/network trainer forces the training to stop in case he/she thinks there is no need to continue the training.

2.3.4 Performance Measurement

The main goal of network training is to optimise the generalisation of the network by minimising the network error. The error measured during network training or testing can be the total sum squared error (TSS), the mean squared error (MSE) or the root mean squared error

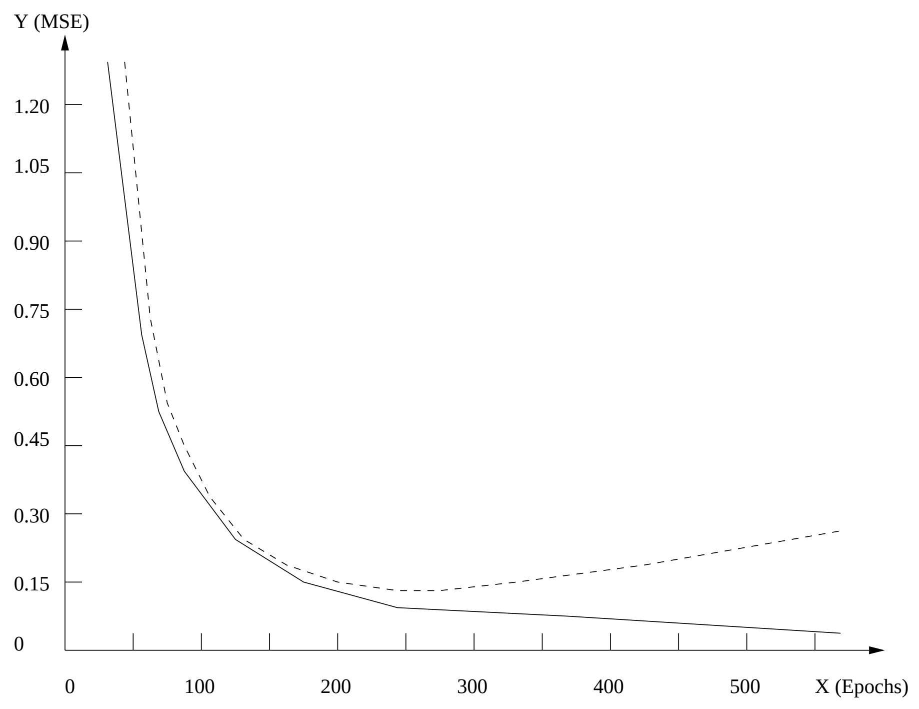


Figure 2.7: A typical network training procedure with a training set and a validation set

(RMSE). These errors are computed according to equation 2.11, equation 2.12 and equation 2.13 respectively [94, 93, 166].

$$TSS = \frac{1}{2}\sum_{p=1}^{n}\sum_{i=1}^{m}(t_{pi} - o_{pi})^2 \tag{2.11}$$

$$MSE = \frac{TSS}{n} = \frac{1}{2n}\sum_{p=1}^{n}\sum_{i=1}^{m}(t_{pi} - o_{pi})^2 \tag{2.12}$$

$$RMSE = \sqrt{\frac{2TSS}{n \cdot m}} \tag{2.13}$$

where:

p: the index of the patterns in the data set (training set or test set);

n: the total number of the patterns;

i: the index of the output nodes of the network;

m: the total number of the output nodes;

t_{pi}: the target output for the ith output node for the pth pattern;

o_{pi}: the actual output of the ith output node for the pth pattern.

2.3.5 Number of Examples in Training Set

As mentioned earlier, one important issue for generalisation of a learning system is the number of examples in the training set. In this subsection, we briefly discuss PAC results, heuristic guidelines and a number of successful neural systems that use small training sets to train large networks.

PAC Learning

Probably Approximately Correct (PAC) Learning [9, 78, 95] addresses the relationship between the number of training examples and the accuracy with which a classifier can be learnt.

A classifier (h) can be considered "good" if it has an error rate smaller than some small number ϵ. It is bad if the error rate is greater than ϵ.

A learning algorithm can be considered "good" if it has a low probability (less than δ) of producing a bad classifier (equation 2.14).

$$P[error(h) > \epsilon] < \delta \tag{2.14}$$

According to equation 2.14, different degrees of "goodness" will correspond to different values of ϵ and δ. The smaller ϵ and δ, the better the learnt concept will be.

PAC theory seeks to establish a relationship between m, the number of examples and ϵ and δ. Given that we want to achieve some level of accuracy in a learnt neural network, the PAC learning model gives the number of examples needed in the training set as

$$m \geq \max[\frac{1-\epsilon}{\epsilon}\ln\frac{1}{\delta}, \frac{V.C.dim - 1}{32\epsilon}] \tag{2.15}$$

where m denotes the number of examples in the training set and $V.C.dim$ is the VC dimension of the network. The *Vapnik-Chervonenkis (VC) dimension* is a measure of the learning capacity of a learning system as determined by its complexity [9, 201].

The VC dimension of a single-layer (of weights) feed forward neural network is simply the number of its weights. For a multilayer feed forward network, the lower bound is the number of weights in the network [9, 18, 201].

According to equation 2.15, for some given ϵ, δ and the network architecture, the number of training examples for a network can be obtained. Table 2.3 shows some examples of a 196-3-4 network with different ϵ and δ. In this table, W is the number of weights and N is the number of nodes in the network.

ϵ	0.05	0.05	0.05	0.01	0.01	0.01
δ	0.1	0.01	0.001	0.1	0.01	0.001
W	600	600	600	600	600	600
N	203	203	203	203	203	203
$V.C.dim$	600	600	600	600	600	600
m	9.6×10^6	9.6×10^6	9.6×10^6	5.4×10^7	5.4×10^7	5.4×10^7

Table 2.3: Number of training samples required by PAC learning for a network of 196-3-4.

As can be seen from table 2.3, training a 196-3-4 network requires millions of examples according to the PAC learning theory, many more than are generally available in practice.

Heuristic Guidelines

A number of heuristic guidelines for determining the size of the training set have been proposed for a given size of the network.

Baum and Haussler [18, page 153] stated that: The number of training examples for a network is approximately the number of weights in the network times the inverse of the accuracy parameter, ϵ. If one wants to achieve an accuracy level of 90%, according to this heuristic guideline, about 10 times as many training examples as the weights in the network would be required. For a feed forward network of 196-3-4, 6000 ($= 600 \times 10$) would be needed for training the network.

Bartlett [17, page 3] claimed that "for pattern classification applications, the VC-bounds seem loose; neural networks often perform successfully with training sets that are considerably smaller than the number of network parameters." In other words, the number of training examples required for network learning can be smaller than the number of weights in the network.

Lawrence et al. [111, page 2] suggested that the optimal number of weights in a network can be much larger than the number of training examples for image recognition such as face recognition problems.

Wasserman [201, page 224] thinks that learning theory is far from complete and that neural network engineers must rely on experience, common sense and creativity to decide the size of the training set according to the problem domain.

Large Networks with Small Training Set

As discussed earlier, for a learning system to obtain good generalisation performance, a large number of training examples are required according to the learning theory. However, many practical neural network researchers have rejected these results on the basis of experience and many networks have been successfully trained with far fewer training examples [115, 201]. In particular, Anthony [9] claimed that smaller training set than predicted by PAC learning should suffice. The main reason is that the original assumptions in PAC leaning and VC dimension are not in general accurate in practice [77, 78]. A number of practical applications have proven this point.

Dunston et al. [47] present a neural network model for an automated control application. In this approach, the architecture of the network used is 20-7-6-1, a multilayer feed forward network, which has 188 weights. The network was successfully trained with 96 examples in the training set.

Ciesielski and Zhu [36] describe a neural network system for bacterial growth detection. One of the networks used here was a three layer feed forward network with the architecture of 400-10-1. In this network, the number of weights was 4010. The network was successfully trained on 263 training examples by the backward propagation algorithm and the trained network resulted in very good results (99.32% correct classification).

Shirvaikar and Trivedi [172] use neural networks to detect small targets in high clutter backgrounds. The network with the architecture of 91-2-2-1 has 188 weights. Two groups of training set were used to trained the network by the backward propagation algorithm with a piecewise linear transfer function. The first training set consisted of 14 examples (7 targets and 7 background examples) and the second had 13 examples (1 target and 12 backgrounds).

LeCun et al. [43, 114, 116] describe a variation of multilayer feed forward network – a shared weight neural network architecture for handwritten digit recognition. The network has one input layer, three hidden layers and one output layer. The number of nodes in these layers from input to output direction is 256, 768, 192, 30 and 10 respectively. In this network, there were in total 1256 nodes, 64660 links/weights. Considering the weight sharing feature, the number of independent parameters was 9760. This network was successfully trained on 7291 examples by the backward propagation algorithm.

2.3.6 Main Parameters

Table 2.4 gives a brief description of the main parameters used for network training.

Parameter Name	Description
Random_range(r)	The random range ([-r, r]) of the initial weights.
Learning rate(η)	The constant in true gradient descent. The bigger the learning rate, the larger the changes in the weights.
Momentum(α)	A constant which determines the effect of past weight changes on the current direction of movement in weight space.
Critical error (ce)	The desired error (TSS, MSE or RMSE) for stopping network training. If the actual error is equal to or less than this error, the training will be terminated.
Correct_Percentage ($Percent$)	The desired percentage of patterns correctly classified/learnt in the training set. If the actual correct percentage of the training patterns is equal to or greater than this pre-defined value, the training stops.

Table 2.4: Main parameters applied to network training.

2.3.7 Tackling a Problem with Neural Networks

Assuming that a generalised tool based on the backward error propagation algorithm for training multilayer feed forward networks is available, the task of using these kinds of neural networks for any given problem becomes one of determining a suitable network architecture and a set of suitable parameters. More specifically, the following questions need to be considered before the experiments can be performed:

- How to properly arrange the data for network training and for measuring the results?
- What is the number of output nodes?
- How many input nodes are needed?

- How many hidden layers are needed and how many nodes in each hidden layer?
- What values can be given for the parameters and variables for controlling the training process, for example, learning rate, range of initial weights, momentum and number of epochs?
- What termination strategies need to be applied during network training and how many runs do we perform for the problem?
- At what stage are the network weights updated when training patterns are presented?

2.3.8 Current Issues in Neural Networks

The major research issues in neural networks include [46, 224]: development of new suitable network architectures for real world problems [31, 38, 65, 92, 144, 152, 224], development of more efficient learning algorithms [224], integration of statistical models, symbolic representation and symbolic reasoning into connectionist systems [62, 74, 75, 178, 179], extraction of symbolic rules from trained networks [7, 185, 191, 192], and learning theory in neural networks such as PAC leaning and bounds on the number of training examples [9, 32, 95, 118].

2.4 Overview of Evolutionary Computation

Evolution is the primary unifying principle of modern biological thought. Since the late 1980s evolutionary computation has received significant attention, although the origins can be tracted back to 1950s [10]. Evolutionary computation thought has extended beyond the study of human life: evolution has become an optimization process that can be simulated or learned using a computer and put to good engineering purpose [50].

Evolutionary computation has been generally grouped into three broad avenues: genetic algorithms (with links to genetic programming and classifier systems), evolutionary strategies, and evolutionary programming [10, 50]. In this thesis, we only use genetic algorithms and genetic programming. In the remainder of this section, we mainly review genetic algorithms and genetic programming, and only give a brief overview of current state-of-the-art issues in the whole evolutionary computation area.

2.4.1 Overview of Genetic Algorithms

The first genetic algorithm was developed by Holland at the University of Michigan beginning in the early 1960s. In the early 1970s, Holland formulated the "Schema Theorem". This in turn formed the basis of his landmark book *Adaptation in Natural and Artificial Systems* [79]. Since then, the genetic algorithm has gradually become one of the main learning paradigms.

In this subsection, we only review the main idea and genetic operators. The details of genetic algorithms can be found in Goldberg[64] and Holland [79].

The Main Idea

A genetic algorithm is a kind of search and learning method based on Darwinian natural selection theory. The technique involves generating a random initial population with a given number of individuals, each of which represents a potential solution to a given problem. Potential solutions are encoded into *chromosomes*, usually bit strings of some arbitrary length. Each individual's fitness as a solution to the problem is computed and evaluated against some designated criteria. Members of the population are then selected for reproduction based on their fitness and a new generation of potential solutions is generated from the offspring of the most fit individuals. The process of evaluation, selection, recombination is iterated until a certain criterion is reached.

The individuals in a population are evaluated by a fitness function. The fitness function should be able to assess the performance of an individual with reference to the problem for which it is a potential solution. The fitness function plays a very important role in the evolutionary process. If the fitness function is not properly set, the evolutionary learning would most likely fail. The fitness function in general varies with problem domains, since different problems have different goals.

The main goal of selection is to have some part of the individuals' genetic material propagated through to the next generation of potential solutions. A number of selection methods have been developed. Among them the *biased roulette wheel* mechanism [64] is the most commonly used one. In this method each individual in the current population has a slot on a roulette wheel proportional in size to that individual's fitness. The roulette wheel is spun once for each parent required. The winning individuals are paired for reproduction and recombination.

Recombination is achieved by one of the combination operators. There are two main genetic operators: *mutation* and *crossover*, which are used to produce the new offspring. The concepts of mutation and crossover are presented in section 2.4.1 (page 38).

The Operators

Mutation is the random modification of the selected individuals. The main goal of mutation is to maintain the diversity in the population. An example of a single bit mutation is shown in figure 2.8.

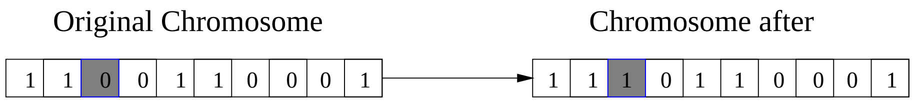


Figure 2.8: An example of a single bit mutation.

Crossover operators approximate sexual reproduction in biology and are the combination of the genetic material from two selected individuals to produce offspring. Examples of crossover operators are: single point, two point, and uniform crossover. Single point crossover is achieved by randomly choosing a single point at which to separate, swap, and rejoin the bit strings. In two point crossover, two points are chosen at random and the segments of the bit string between the two points are exchanged and form the two new children. These two methods can suffer from the problems of destroying good schemata or incompletely combining the schemata.

Uniform crossover [44] is implemented by generating a bit mask equal in length to the chromosomes being manipulated, with the value of each bit being determined with some arbitrary probability. For each bit of the mask which has “1”, the corresponding bit of the parent chromosomes is swapped before propagation to the offspring; and for each mask bit which has “0”, the corresponding parent bits are propagated to the offspring unchanged. An example of uniform crossover is presented in figure 2.9. Uniform crossover can combine all possible schemata, even if it might be quite disruptive of schemata of any length.

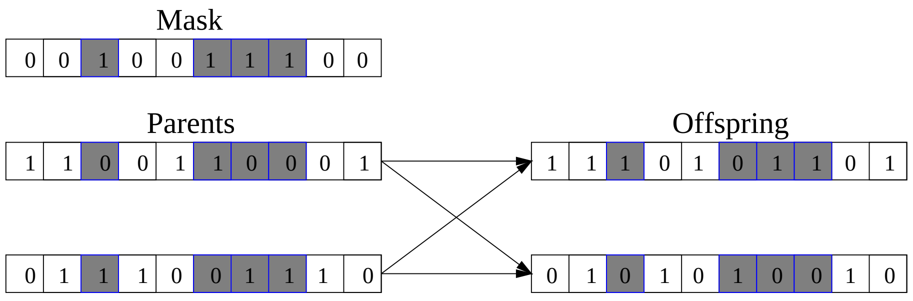


Figure 2.9: An example of uniform crossover

It is noted that there are also other types of crossover, such as *shuffle crossover* and *segmented crossover*, which are not described here since they are not used in this thesis.

Main Parameters

The main parameters that are associated with using genetic algorithms are described in table 2.5.

Parameter Name	Description
POPULATION_SIZE	The number of chromosomes in the population
CROSSOVER_RATE	The number of chromosomes used for crossover as a percentage of the total number of chromosomes in the population
MUTATION_RATE	The percentage of chromosomes in the population applied to the mutation operator
NUM_GENERATIONS	The maximum number of generations the genetic algorithm evolutionary process will run

Table 2.5: Main parameters applied to genetic algorithms.

Tackling a Problem with Genetic Algorithms

Assuming that a generalised genetic algorithm "engine" (including the representation of the solutions) to perform the evolutionary processes is available, the task of using genetic algorithms for any given problem becomes one of determining a suitable chromosome encoding, a fitness function and a set of suitable parameters. More specifically, the following questions need to be considered before the experiments can be performed:

- How to properly arrange the data for evolutionary learning and for the measurement of results?
- How to represent the problem to match the internal representation of the genetic algorithm engine?
- What is the fitness function?
- What values can be given for the parameters and variables for controlling the evolutionary process, for example, population size and number of generations?

- When to terminate a run and how many runs do we perform for the problem?
- What genetic operators, at what frequencies, are going to be applied?

Genetic Algorithms for Evolving Neural Networks

In this thesis genetic algorithms are used to train and refine neural networks (chapter 6). This subsection briefly reviews related work in this area and describes some results achieved by different training algorithms.

Related work in this area can be grouped into three approaches:

1. Evolving weights in fixed networks. In this approach, the neural network architecture and learning parameters are pre-defined. The genetic algorithm is used to train the given network by evolving the weights and biases. In the genetic algorithm, the network weights and biases are encoded into a chromosome. Each chromosome is an individual member of a population and a population often has several hundred chromosomes. During the evolutionary process, the genetic operators of selection, crossover and mutation are applied to these individuals. After each generation of the process the chromosomes are applied to the network, that is, the weights and biases are set from the values which the chromosomes represent. The error rate on the training patterns is then computed and used as the fitness function for the genetic algorithm.

   There have been a large number of papers published on this approach. For example, Whitley and Hanson [204] used genetic algorithms to find appropriate weights and biases for a network for the minimal adder problem described in Rumelhart [166]. Montana and Davis [133] used a genetic algorithm with floating point chromosomes to evolve network weights for their sonar signal data and produced results competitive with the back propagation algorithm. Korning [100] presents a way of using genetic algorithms to efficiently train a multilayer feed forward network (with an architecture of 6-15-1) for his two class ("wanted" vs "unwanted") classification problem.

   The major advantage of this approach is that the "local minima" problem in which gradient descent algorithms often result can be avoided. However, if the network is very large, the chromosomes which encode the weights and biases will be very long and the computational cost will be very high.

2. Evolving network architectures. In this approach, genetic algorithms are used to evolve not only the network weights and biases, but also the network architecture or topology as well.

Potter [148] describes a method which applies a genetic algorithm to the evolution of a cascade-correlation architecture and produces results comparable with the standard quick-prop learning method. Syed [182] investigates modifying genetic algorithm parameters to evolve both network weights and the network architecture for recurrent neural networks to solve the XOR problem. Balakrishnan and Honavar [12] present a nice overview and classification of the research in the area of evolutionary design of neural network architectures.

3. Evolving other factors in neural networks. This includes evolving learning parameters [19], learning rules [52], delta values [34, 108] and input features [22].

Alander [5] gives an indexed bibliography of genetic algorithms and neural networks. A good survey for evolving neural networks with evolutionary algorithms can be found in [212].

Krishnan, Riley and Ciesielski [34, 108] developed the 2DELTA-GANN algorithm for training feed forward and recurrent networks. In this algorithm, instead of having the genetic algorithm evolve all weights and biases of a network, only the changes of the weights and biases (deltas) are evolved. In [34], they compare the results for training networks using different algorithms. The feed forward networks include the standard XOR network, the 4-2-4 encoder network and a digital to analogue converter network. The network training algorithms compared are the back propagation algorithm, Whitley's GA-NN algorithm [204] and their own new 2DELTA-GANN method. The recurrent networks tested include a SINE network and a network for the sunspot data. They are trained by the new 2DELTA-GANN method and the recurrent back propagation algorithm. In terms of network accuracy, for feed forward networks tested, the new 2DELTA-GANN method out-performed the Whitley's GA-NN method for all but the XOR network and compared very well with back propagation. In terms of training time, the new 2DELTA-GANN method out-performed the Whitley GA-NN for the 4-2-4 encoder network but was much slower than the back propagation algorithm for the digital to analogue converter network. For training the SINE simple recurrent network, the new 2DELTA-GANN method out-performed recurrent back propagation in terms of both network accuracy and training time. For the sunspot network, however, recurrent back propagation out-performed the 2DELTA-GANN method in terms of training time, but the new method compared quite favourably in terms of network accuracy.

As can be seen from the literature, the networks evolved by genetic algorithms are relatively small and the process runs quite well. For the object detection problems investigated in this thesis, however, the networks are very large. For example, in a feed forward network with an architecture of 484-5-5, there will be 494 nodes and 2455 weights/biases. This thesis will investigate whether genetic algorithms can be used to train and refine such large networks in chapter 6.

2.4.2 Overview of Genetic Programming

Genetic programming is a relatively recent technology based on the use of Darwinian evolution in the generation of computer programs. The process starts with a randomly generated population of programs. Each program is executed and its degree of success in achieving its task is measured and assigned as its fitness. Programs with good fitness are then selected for mating. In the mating process two parents are chosen and randomly selected sub-trees are swapped giving two children of a new population. In general, individuals in the new generation will be fitter than those in the current generation. The process terminates when a solution is found or the best individual does not improve over the course of a few generations.

Main Characteristics

There are a number of differences between the standard GA and GP [15, 69, 184].

1. While the standard genetic algorithms use strings to represent solutions, the forms evolved by genetic programming are trees or tree-like structures. The standard GA bit strings use a fixed length representation while the GP trees can vary in length. While the GAs use a binary alphabet to form the bit strings, the GP uses alphabets of various sizes and content depending on the problem domain. These trees are made up of internal nodes and leaf nodes, which have been drawn from a set of primitive elements that are specific to the problem domain.

2. The term *genetic programming* comes from the notion that computer programs can be represented by a tree-structured genome. Computer programming languages, such as *Lisp*, can be represented by formal grammars which are tree based, thus it is actually relatively straight forward to represent program code directly as trees. These trees are randomly constructed from a set of primitive functions and terminals (section 2.4.2 on page 43).

3. With the change of the representation from GA to GP comes a change in the dynamics of the evolutionary search process. The search process is governed by the interaction of the

selection methods, genetic operators, the fitness function and the primitive functions and terminals. In general, the selection methods used in GP can be the same as those in GAs, but the genetic operators and the fitness function are usually different. These issues will be described in more detail in the remainder of this section.

Programs

We first describe the representation of the evolved programs and then present the ways of program generation.

Program Representation. Much of the GP work was done using LISP or LISP-like representations of the programs. A sample computer program for the algebraic equation $(x - 1) - x^3$ can be represented in LISP as the S-expression `(- (- x 1) (* x (* x x)))`. The tree representation is shown in figure 2.10.

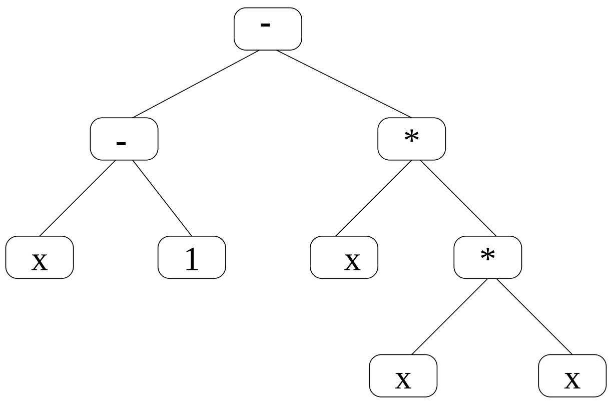


Figure 2.10: A simple tree representation for a sample LISP program.

The programs are constructed from *a terminal set* and *a function set* which vary according to the problem domain. Terminals and functions are also called *primitives*, and the terminal set and the function set are combined to form a *primitive set* [69].

Functions. Functions form the root and the internal nodes of the tree representation of a program. In general, there are two kinds of functions used in genetic programming. The first class refers to general functions, such as the four arithmetic operations and other standard functions like trigonometric functions, as shown in equation 2.16.

$$F = \{+, -, *, \%, \sin, \cos, \exp, \mathrm{rlog}\} \tag{2.16}$$

The function % is protected division such that dividing by 0 returns 0. The second class comprises specific functions which vary with the problem domain.

Terminals. Terminals have no arguments and form the leaves of the parse tree. Typically, terminals represent the inputs to the GP program, the constants supplied to the GP program, or the zero-argument functions with side-effects executed by the GP program [15]. In any case, a terminal returns an actual numeric value without having to take an input.

It is important to note that the selection of the functions and terminals is critical to success. The terminal set and the function set should be selected so as to satisfy the requirements of closure and sufficiency [15, 101]. In other words, each function in the function set should be able to handle gracefully all values it might receive as input (closure), and the functions and terminals should be powerful enough to be able to represent a solution to the problem (sufficiency). A bad selection could result in very slow convergence or even not being able to find a solution at all.

Program Generation. There are several ways of generating programs to appear in the initial population and after mutation, including *full, grow* and *ramped half-and-half* [101]. In the full method, functions are selected as the nodes of the program until a given depth of the program tree is reached. Then terminals are selected to form the leaf nodes. This ensures that full, entirely balanced trees are constructed. When the grow method is used, nodes are selected from either functions or terminals. If a terminal is selected, the branch with this terminal is terminated and the generation process moves on to the next non-terminal branch in the tree. In the ramped half-and-half method, both the full and grow methods are combined. Half of the programs generated for each depth value are created by using the grow method and the other half using the full method.

Fitness Cases

The patterns or examples in other learning paradigms such as neural networks are called *fitness cases* in genetic programming [15, 101]. Accordingly, there are also two different sets of fitness cases: training cases for learning and test cases for performance evaluation.

Fitness and Selection

Fitness Function. Fitness is the measure of how well a program has learnt to predict the output from the input during simulated evolution. The fitness of a program generated by the evolutionary process is computed according to the fitness function. The fitness function should be designed to give graded and continuous feedback about how well a program in a population performs on the training set. Like the primitives, the fitness function plays a very important role in the evolutionary process and varies with the problem domains.

There are several types of fitness functions: *continuous fitness, standardised fitness* and *normalised fitness* [15].

- A *continuous fitness function* is a manner of calculating fitness in which small improvements in how well a program has learned the problem domain are related to small improvements in the measured fitness of the program and larger improvements in how well a program has learned the problem domain are related to larger improvements in its measured fitness. Such continuity is an important property of a fitness function since it allows the evolutionary process to improve programs iteratively.

- *Standardised fitness* is a transformed fitness function in which the fitness of the best individual is assigned a value of zero. For a particular problem, if a lower value of continuous (raw) fitness is better, standardised fitness equals the continuous fitness for that problem; if a greater value of continuous fitness is better, standardised fitness equals the maximum possible value of the raw fitness minus the observed raw fitness. Thus, standard fitness has the administrative feature that the best fitness is always the same value of zero, regardless of what problem one is working on.

- *Normalised fitness* is a transformed fitness function where the fitness is computed from the standard fitness (or adjusted fitness [101]). The normalised fitness has three desirable characteristics: the range of the fitness is between zero and one; better individuals in the population have larger fitness value; and the sum of the normalised fitness values is one.

Selection. Fitness selection determines which evolved program will be used for the genetic operators to produce new individuals for the next generation during the evolutionary process. Two of the most commonly used selection methods are *proportional selection* and *tournament selection.*

In the proportional selection method [101] an individual in a population will be selected according to the proportion of its own fitness to the total sum of the fitness of all the individuals in the population. Programs with low fitness scores would have a low probability of having any genetic operators applied to them and so would most likely be removed from the population. Programs which perform particularly well in an environment will have a very high probability of being selected.

The tournament selection method [15, 101] is based on competition within only a subset of the population against each other, rather than the whole population. A number of programs is selected randomly according to the tournament size and a selective competition takes place. The better individuals in the tournament are allowed to replace the worse individuals. In the smallest possible tournament, two individuals can compete. The winner is allowed to reproduce with mutation and the result is returned to the population, replacing the loser of the tournament.

Genetic Operators

There are three fundamental genetic operators in genetic programming, known as *reproduction, crossover* and *mutation*. In general, a percentage of individuals in a population will have one or more genetic operators applied to them.

Reproduction is the basic engine of Darwinian theory [101], and is the simplest among these operators. It involves just simply copying the selected program from the current generation to the new generation. In general, reproduction allows good programs to survive into the next generation of the population.

Mutation operates only on a single selected program and introduces new genetic code into a population in the new generation. The most common form of mutation removes a random subtree of a selected program, then puts a new subtree in the same place. This can be done by the *grow* method, the *full* method or the ramped half-and-half method (page 44). Figure 2.11 (a) shows an example of mutation for a selected program.

Crossover takes advantage of different selected programs within a population, attempting to integrate the useful attributes from them. The crossover operator combines the genetic material of the two selected parents by swapping a subtree of one parent with a subtree of the other, and introducing two newly formed programs into the population in the next generation. An example of crossover for two selected programs is presented in figure 2.11 (b).

It is noted that some researchers argue that mutation is useless while some others insist that a high mutation rate would help the GP process converge [55, 68, 101]. This is most likely to be domain dependent and one can make a decision according to the problem domain and the environment.

Main Parameters

The main parameters applied to genetic programming are described in table 2.6.

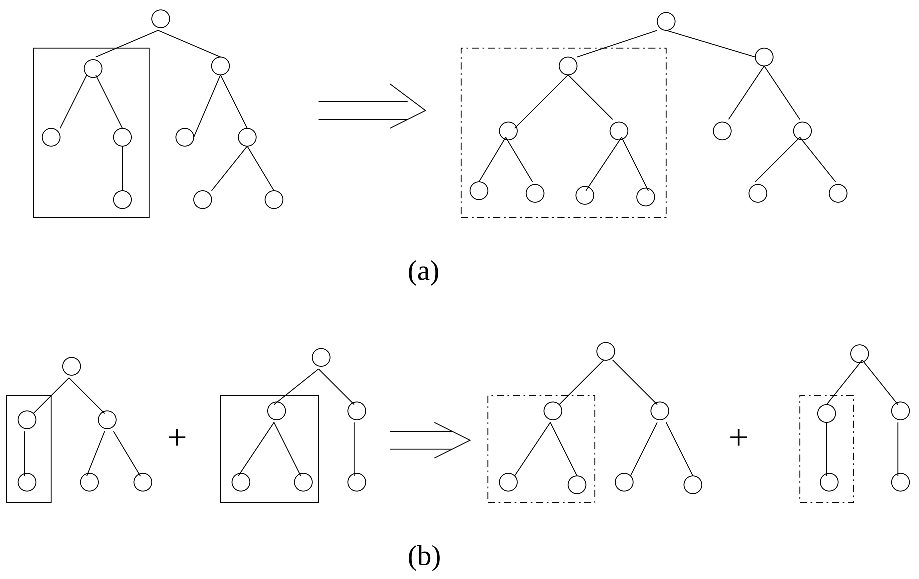


Figure 2.11: Effect of genetic operators in genetic programming. (a) Mutation in GP: Replaces a random subtree; (b) Crossover in GP: Swaps two random subtrees.

Tackling a Problem with Genetic Programming

Assuming that a generalised genetic programming "engine" to perform the evolutionary processes is available, the task of using genetic programming for any given problem becomes one of determining the appropriate set of functions and terminals, a suitable fitness function, and a set of suitable parameters. More specifically, the following questions need to be considered before the experiments can be performed:

- What is the set of terminals used in the program trees?
- What kind of functions can be used to form the function set to represent the program tree?
- What is the fitness measure?
- What values can be given for the parameters and variables for controlling the evolutionary process, for example, population size and number of generations?
- When to terminate a run?
- How do we know the result is good enough?
- What genetic operators, at what frequencies, are going to be applied?

Parameters	Description
POPULATION_SIZE	The number of programs in the population
ELITISM_PCNT	The percentage of each generation created by reproduction
CROSS_RATE	The percentage of each generation created by crossover
MUTATION_RATE	The percentage of each generation created by mutation
INITIAL_MAX_DEPTH	The maximum depth programs can be in the initial population
MAX_DEPTH	The maximum depth programs can be during the evolutionary process
MAX_GENERATIONS	The maximum number of generations the GP evolutionary process will run

Table 2.6: Main parameters applied to genetic programming.

Each of these factors can affect the evolutionary process, and consequently the changes of finding (learning) a successful solution. The interaction of genomic representation and the fitness function define the nature of the search space. Such interactions are usually extremely complex, and the space is extremely high-dimensional. The selection methods and genetic operators together define the balance between *exploration* and *exploitation* [69], that is, they determine how much effort is devoted to exploring parts of the space that show promise and how much effort goes into looking for new parts of the space that may ultimately be more rewarding.

Also some other practical issues also need to be considered, such as run-time and memory overheads. In such cases, some degree of parallelism in execution might be beneficial, such as some powerful parallel hardware. Within the limitation of the conditions, some controls might have to be taken to limit the maximum size of a program tree, even some control parameters. More details of the relevant issues can be found in [15, 101, 103].

2.4.3 Current Issues in Evolutionary Computation

Since early 1990s, evolutionary computation algorithms including genetic algorithms, genetic programming, classifier systems, evolutionary programming and evolutionary strategies have been successfully applied to the following areas: biocomputing [101, 141], cellular programming [173], game playing [30], job-shop scheduling [110], non-linear filtering [45] and time-tabling [40]. In this subsection, we briefly review the major research issues in genetic algorithms, genetic programming, as well as other areas of evolutionary computation.

The major research issues in genetic algorithms include theoretical foundations [10, 50, 147], interactions between representations [10, 50, 51], selection mechanisms and genetic operators [10, 51, 147], and complex adaptive systems to solve difficult problems in the real world [10, 50, 51, 147, 174].

The following topics constitute major issues in genetic programming: investigation of general rules for determination of crossover versus mutation rates for different problems [15, 104, 106]; evolution of programs with iteration rather than tree structures only [15, 104]; definition of meaningful fitness functions for very complex, dynamic problems [15, 104]; dealing with intron removal in the evolved programs [104, 105]; automatically defined function/subroutine finding [15, 104]; combinations of genetic programming with neural networks [105]; combinations of genetic programming with symbolic rules [104, 105]; and self-adaptation of parameters [104, 106].

The following topics constitute major current issues in other evolutionary algorithms: theoretical foundations [10]; biological modelling [147, 174]; combinations of evolutionary computation and computational intelligence such as neural and fuzzy systems [10, 147]; evolutionary robotics and evolvable hardware [174]; and formal characterisation of the application domain and limits of evolutionary computation [10, 147].

2.5 Neural Networks for Object Detection

Since the late 1980s, the use of neural networks in a number of computer vision areas including object classification and detection has been investigated in a variety of application domains. These domains include military applications (detection or classification of tanks, planes and trucks), human face recognition, agricultural product classification, handwritten character recognition and medical image analysis. The types of the neural networks used include multilayer feed forward networks [165], self organising maps [99], higher order networks [63, 146] and ARTs [27, 28]. In this section, we review this work in the categories of object classification and object detection.

2.5.1 Object Classification

There have been many reports of using neural networks for object classification. A review of this work is presented in this section, according to the types of networks. It is important to note that no identification or localisation of objects is done in the systems presented in this section. The classifier works on pictures that have been segmented by another process or by hand.

Feed Forward Networks with Preprocessing and/or Feature Extraction

In the previous work presented here, the networks are feed forward neural networks (or multilayer perceptrons, multilayer neural networks and backward propagation networks) and specific image features are extracted through preprocessing and/or feature extraction as input to these networks. Note that some approaches with hybrid techniques are also included in this section if there is one or more feed forward networks in these systems.

Classification of Mammograms: Verma [200] proposes a neural network based technique that extracts suspicious areas containing microcalcifications in digital mammograms and classifies them into two classes depending on whether they contain benign or malignant clusters. The centroids and radii of suspicious areas are provided by an expert radiologist. A feed forward network with a single hidden layer is used to classify these areas. The network is trained by the backward error propagation algorithm with momentum and a direct solution method based algorithm. There are 105 extracted areas containing 76 malignant and 29 benign microcalcifications and 90 are used for training and 15 for testing. Both training algorithms result in a best classification rate of 86.6%, however, training based on the direct solution method is much faster than backward error propagation. The networks use 1257 features as inputs, but the author does not mention exactly what they are.

Sonar Target Recognition: Roitblat et al. [157] use two backward propagation neural networks, a probabilistic neural network and an expert system for the sonar recognition of targets embedded in sediment. Three experiments are presented in which the system is used to discriminate between small stainless-steel cylinders and cylinders of the same size made of hollow aluminium, foam-filled aluminium, and coral rock embedded in resin. Each of the targets is buried in mud at a depth of several centimetres. The system was highly effective at recognising these objects. In this system a high level of preprocessing including the Fast Fourier Transform [39] was used to prepare data prior to classification.

Classification of Missiles, Planes and Helicopters: Howard, Padgett and Liebe [81] introduce a multi-stage neural network for object classification. The multi-stage neural network contains two neural networks which are trained in a serial fashion. Both networks process 45 input nodes (image features), one hidden layer with 30 nodes, and one output node. The first network is initially trained on a pattern set created from projecting a set of prototype images onto the eigenvector set [89]. The input images consist of a combination of regions of interest containing targets against background, targets without background and background without targets. The network output values used are -1.0 for non-targets and 1.0 for targets. A high threshold is used for the classification to ensure a low false alarm rate. Once the first network is trained, the objects which were incorrectly classified are assembled into another training set. The second network is trained with the set of misclassified targets and the set of background images. The standard backward error propagation algorithm is used to train the two networks. The method is tested on the images obtained by using scale target models of objects of missiles, planes and helicopters against different backgrounds and settings of luminance, translation, scale, azimuth and elevation. The approach resulted in a 95% detection rate with no false alarm rate.

Maneuver Target Recognition: Wong and Sundareshan [210] apply a neural network based approach with the integration of a multi-layered neural network and a Kalman filter to the data fusion and tracking of complex target maneuvers. The types of maneuvers include longitudinal accelerations, coordinated turns and non-coordinated turns, which need to be classified by neural networks. The network is trained and optimised by the simplex algorithm, which the authors claim can guarantee a globally optimal solution. This algorithm employs concepts from simplex optimisation and is implemented by splitting the 3-layer neural network into two portions – a linear portion and nonlinear portion. The connections between the input layer and the hidden layer form the nonlinear portion, while those between the hidden layer and the output layer constitute the linear portion. The simplex optimisation method is used to find the optimal weights in the nonlinear portion, while a linear least squares minimisation is used to determine the optimal weights in the linear portion of the network. The approach is tested on a combination of maneuvers by the target in quick succession and a sharp 360 degree turn maneuver. The results show a superior performance of this approach to the interacting multiple model algorithm [16]. In this approach preprocessing is applied and features are extracted as input to the networks for classification.

Classification of Tanks and Trucks on Laser Radar Images: Troxel, Rogers and Kabrisky [193] present an approach to the use of multi-layer perceptron neural networks in classifying segmented objects invariant to position, rotation and scale. Objects to be classified are multi-function laser radar data of tanks and trucks at various aspect angles. In this approach, candidate targets are first segmented through preprocessing (by using a doppler segmenter [163] and a range segmenter [190]). Each segmented target is then compared with stored templates representing the different classes. The template and the image are transformed into the magnitude of the Fourier transform with log radial and angle axis to form the feature space. These features are used as input to networks for classification. The network is trained by a back propagation algorithm. This approach achieves a promising performance with an accuracy of near 100% on this laser radar database.

Underwater Target classification: Huang et al. [84] use neural networks to classify underwater mines and mine-like targets from the acoustic backscattered signals. The system consists of three stages: a feature extractor using wavelet packets, a feature selection scheme and a backward propagation neural network classifier. The backward propagation neural network is used to perform the discrimination between targets and non-targets based on a reduced set of features. The data set consists of the backscattered signals for seven frequency bands and six different objects: 2 mine-like targets and 4 non-targets. The targets are organised at 72 aspect angles from 0-355 degrees with 5 degree increments. This network (9-8-2) achieved 85% correct classification accuracy on 2160 samples (720 targets and 1440 non-targets) at a 10% false alarm rate. This was better than an alternative method based on a combination of a matched filter and spectrogram correlation and transformation [25] which achieved 75% correct classification at the same false alarm rate.

Mine and Mine-like Target Detection: Miao et al. [128] introduce an approach to the recognition of arbitrarily scattered surface-laid mines and mine-like targets from multi-spectral imagery data of a minefield. The system consists of six channels which use different neural network architectures for feature extraction and classification of targets in six different optical bands. Auto-associative networks [14, 72] are used for feature extraction and the multilayer feed forward networks for object classification. The images are obtained from a multi-spectral video camera with the range from near UV (400nm) to near IR (900nm). The training data set is chosen primarily from the sub-images contained in selected target blocks. The three layer backward propagation neural network achieves better performance than the standard maximum likelihood classification scheme [66, 89] and the overall system produces some promising results for the classification of mines and mine-like targets. Note that high level preprocessing including contrast mapping schemes [89] was applied prior to feature extraction.

Handwritten Character Recognition: Verma [199] proposes a feature extraction technique in conjunction with neural networks to classify cursive segmented handwritten characters. A heuristic and neural network based algorithm is used to segment the characters. The proposed technique extracts global features from the segmented characters and passes them into the neural networks for classification. The neural networks can still recognise characters even if these characters are rotated 90 degrees and a little distorted. The handwritten character data was obtained by using a HP flat bed scanner and then converted into monochrome bitmap (binary) form. The segmented character size is 30×30 pixels. The neural networks using the architecture of 156-16-26, 156-25-26 and 156-30-26 are trained by a modified backward error propagation algorithm. The best performance obtained by the method is 100% recognition rate on the training set and 65% on the test set. The author claims that this is better than other approaches on similar handwriting recognition problems even if the data used are different.

Agricultural Product Recognition: Winter et al. [207, 208] use feed forward backward propagation neural networks for agricultural product recognition and vision problems. The neural network based method is used to discriminate popcorn kernels which can be popped from those that can not [208]. The results show that the system using simple gray-scale colour and morphological features can separate the poppable and the hard-to-pop popcorn kernels in a commercial sample set with a 75% accuracy. They also apply neural networks to the classification of commercial samples of lentils into three classes: “good”, “discoloured” and “broken and peeled” [207]. Twenty one features, that is, eight morphological and thirteen colour features of a sample are extracted, and then used as input to the neural networks. The method results in 94% accuracy in classification of “good”, 97% for “discoloured” and 95% for “broken and peeled”.

Multispectral Remote Sensing Data Classification: Lee and Landgrebe [117] apply a decision boundary feature extraction algorithm to feed forward neural networks. The authors first define the decision boundary in the networks, then propose a procedure for extracting all the necessary features for classification from the decision boundary. The data used here includes remotely sensed data, multi-source data and simulated data specially designed to test the robustness of the algorithm. For the multispectral remote sensing data from a helicopter-mounted field spectrometer, there are three different classes, which consist of 1209, 1146 and 1103 object samples respectively. Five hundred randomly selected samples are used as training set and the rest are used for test. Seventeen features are selected from sixty by the proposed algorithm. The networks are trained using different architectures based on the number of input nodes varying from 2 to 17 features. The results show that if more than 2 features are used, the neural networks based on the boundary feature extraction algorithm always produce better performance than those with principal component analysis and discriminant analysis [56]. The neural network approach is also superior to the k nearest neighbourhood classifier [56] on the same data.

Feed Forward Networks with Pixel Based Input

This section presents previous work of object classification based on feed forward networks with inputs of pixel based data, including raw pixel data and pixel statistics (pixel level, domain independent features).

Bacterial Growth Detection: Ciesielski and Zhu [36] describe a neural network based system for detecting bacterial growths on microbiology plates. Experiments are performed with three layer (only one hidden layer) feed forward neural networks of different architectures trained by the backward error propagation algorithm. The network with architecture 400-10-1 gave the best performance of 99.32% correct classification. The inputs are the intensity values of the pixels in a 20×20 square around a growth position. This pixel based neural network approach is compared with a number of image processing and computer vision methods such as thresholding, 1d entropy, 2d entropy, region growing and template matching for the same classification problem [37]. The results show that neural networks using raw pixel data are superior with respect to classification accuracy, execution time and development time.

Tank and Helicopter Classification: Ranganath, Kerstetter and Sims [151] introduce a self partitioning neural network (SPNN) approach for object classification. The SPNN can partition the target vectors into an appropriate number of groups and train one subnetwork to recognise the targets in each group. A fusion network combines the outputs of the subnetworks to produce the final response. This approach can automatically determine the number of subnetworks needed without excessive computation. The subnetworks are three layer feed forward networks with only one hidden layer and one node in the output layer. They are topologically identical. The authors claim that the method is robust and capable of self organisation to overcome the ill effects of non-cooperating targets in the training set. The SPNN approach improved the classification accuracy and reduced the training time of the backward propagation neural networks significantly. The trained self partitioning network is also capable of incremently learning new training vectors. Note that 40 domain independent, pixel level features, such as the average grey level of each line and each row of the object, are used as input to the neural networks, rather than specific features for this problem domain.

To give a clearer view, we summarise the work of object classification using feed forward neural networks in table 2.7.

Network Input	Applications/Problems	Authors	Year	Source
Specific Features	Classification of Mammograms	Verma	1998	[200]
	Sonar Target Recognition	Roitblat et al.	1995	[157]
	Classification of Missiles, Planes and Helicopters	Howard et al.	1998	[81]
	Maneuver Target Recognition	Wong et al.	1998	[210]
	Classification of Tanks and Trucks on Laser Radar Images	Troxel et al.	1988	[193]
	Underwater Target classification	Huang et al.	1998	[84]
	Mine and Mine-like Target Detection	Miao et al.	1998	[128]
	Handwritten Character Recognition	Verma	1998	[199]
	Agricultural Product Recognition	Winter et al.	1996	[207, 208]
	Multispectral Remote Sensing Data Classification	Lee et al.	1997	[117]
Pixel Based	Bacterial Growth Detection (Classification)	Ciesielski et al.	1992	[36]
	Tank and Helicopter Classification	Ranganath et al.	1995	[151]

Table 2.7: Object classification based on feed forward neural networks.

Other Networks

Besides feed forward neural networks, other types of networks have also been applied to object classification. These networks include shared weight neural networks [114, 116, 165], auto-associative memory networks [1, 98, 99, 196], ART networks [27, 28], probability networks [188] (or probabilistic neural networks [176]), Gaussian basis function networks [140], neocognitron networks [57, 58], higher order networks [63, 73, 146], and hybrid/multiple neural networks [29]. Since they are not used in this thesis, we only summarise the related work of these networks according to the kinds of networks, the applications or problems, the first authors, published year and the source of these approaches, as shown in table 2.8.

Kind of Network	Applications/Problems	Authors	Year	Source
Shared Weight Neural Networks	Handwritten optical character recognition	Soulie et al.	1993	[175]
	Zip code recognition	LeCun et al.	1989	[114, 116]
	Digit recognition	de Redder et al.	1996	[42, 43]
	Lung nodule detection Microcalcification classification	Lo et al.	1995	[121]
Auto-associative Memory Networks	Face categorisation, recognition, identification, classification	Abdi Valentin et al. Valentin et al. Valentin et al. Valentin et al.	1988 1994 1994 1996 1996	[1] [196] [198] [195] [197]
ART Networks	Vehicle recognition	Bernardon et al.	1995	[20]
	Tank recognition	Fogler et al.	1992	[156]
Serf-Organising Maps	Handwritten alphabet and digit character recognition	Chigawa et al.	1991	[33]
		Nakayama et al.	1992	[136]
Probability Neural Networks	Cloud classification	Tian et al.	1998	[188]
	Radar target 'detection'	Kim et al.	1992	[96, 97]
Gaussian Basis Function Networks	3D hand gesture recognition	Ahmad et al.	1993	[3]
Neocognitron Networks	Bend point and end point recognition	Fukushima et al.	1998	[58]
High Order Neural Networks	2D and 3D helicopter (F18) recognition	Spirkovska et al.	1994	[177]
	Apple sorting (classification)	Hecht-Nielsen	1992	[73]
	Recognition of bars, triangles and squares	Cross et al.	1995	[41]
Hybrid/Multiple Neural Networks	River identification(classification)	Liu et al.	1998	[120]
	Classification of “hot” and “cold” objects	Casasent et al.	1995	[29]
	Face recognition	Lawrence et al.	1997	[112, 113]

Table 2.8: Object classification based on other kinds of neural networks.

2.5.2 Object Detection

As defined on page 3, object detection includes both classification and localisation. There have been a number of reports on the use of neural networks in object detection problems. Typically, they belong to the *one-class object detection* problems (section 2.1.2, page 10), where the objects in a single class in large pictures need to be detected. Work in *multiple class object detection* based on a single network or in one stage has not been reported so far. The network types for one class object detection problems include multilayer feed forward networks, shared weight neural networks, probabilitistic decision-based neural networks and hybrid (or multiple) neural systems.

Feed Forward Networks

Shirvaikar and Trivedi [172] use a feed forward neural network as a filter to detect small targets in cluttered backgrounds in thermal infrared images. In this approach, feature extraction is eliminated and raw grey levels are utilised as input to the network. Two different methods are applied to the design of the two training data sets: a direct use of actual image data with seven object and seven background samples and a model-based method with one object and 12 background samples. The size of the small targets detected here is 9×9 pixels. The neuron transfer function is also modified from the sigmoid function to a piecewise linear function and the network is trained on these cutouts (either object or background) by the backward error propagation algorithm. The trained network is used as a filter like a moving window to locate the targets of interest. The network is convolved with the full input image to produce output at each pixel and the process produces a grey level filtered image, where the filter response is supposed to be high for target pixels and low for the background pixels. The locations of the detected targets can be obtained by applying a threshold on the filtered image. The results show that the overall performance of the neural network filter is much better than that obtained by the size-matched contrast-box filter [171], even if some false alarm rates are still quite high. For three test images presented here, this approach achieves the best false alarm rates of 418%, 1256%, 843% for the actual image data method and 659%, 619%, 243% for the model-based training method, at a detection rate of 100%. There is only one target class of interest in these images.

Shared Weight Neural Networks

Shared weight neural networks are feed forward networks in which the weights in the first sets of layers are applied as linear correlational filters and in the last sets of layers in the ordinary fully connected feed forward fashion [60]. In other words, the network consists of two parts: a feature extraction network followed by a classification network. The feature extraction network can consist of one or more layers, and each layer can have one or more feature maps. Each layer performs feature extraction by template operations over input to that layer. The size(s) of the feature maps are determined by the undersampling rate for the convolution over their input and usually are domain dependent. The feed forward network performs the classification task based on the outputs from the feature extraction network. More details of shared weight neural networks can be found in Rumelhart et al. [165] and LeCun et al. [114, 116].

Gader et al. [60] describe a segmentation free shared weight neural network approach for automatic vehicle detection. A shared weight network in scanning mode used here is an image-to-image transformation. An input scene image is used as input and the target output confidence is used to form the output image, which can be regarded as a correlation plane. The convolution of the input scene image (with masks defined by user) are used to create feature maps. The feature maps are scanned with a moving window large enough to contain the largest target expected. These windows are sub-images that can input to the classification network, which can determine whether a sub-image at a pixel with the window size is a target or background. Two sets of images are considered in this paper. The first set consists of forward looking infrared images of tanks and the second consists of images of cars in a parking lot. There are 35 *256*$\times$*256* input scene images each of which contain only one tank in the first set, 17 are used for training and 18 for testing. The second set consists of 88 *512*$\times$*512* images with 36 of them for training and 52 for testing. The first set of images are directly used for network training and testing, while some preprocessing and postprocessing are applied before and after network training. On the first set, this approach achieves 100% accuracy in both the training set and the test set. Detection on the second data set is difficult, where only 66.67% (10 out of 15) accuracy on the training set and 60% (12 out of 20) for the test set are obtained. Note that not all the images in the test set are used. However, this is much better than that obtained by a minimum average correlation energy matched filter method [124], that is, 30% accuracy (6 out of 20) on the test set. In this approach, image pixels are used as inputs to the neural networks, rather than image specific features. They present a variation of the approach, or a morphological shared weight neural network approach, for the automatic target recognition problems on the same data [209].

Probabilitistic Decision-Based Networks

A probabilitistic decision-based neural network (PDBNN) is a probabilitistic variant of the decision-based network (DBNN) [109]. Compared with multilayer feed forward networks, DBNNs have a different architecture and learning rules. In a DBNN, one subnet is designated to represent one object class, which is usually called "one-class-one-network" property. This is different from the "all-class-in-one-network" property of the feed forward networks. Unlike the feed forward networks which usually use exact target values (such as 1.0 or 0.9 for the expected classes and 0.0 or 0.1 for other classes), DBNNs use decision based learning rules where the teacher only tells the correctness of the classification for each training pattern. Based on the teacher's information, the DBNN uses a distributed and localised weight updating rule. In addition, DBNNs use "hybrid locally unsupervised and globally supervised learning". In other words, there are two phases in the learning scheme: during the locally unsupervised learning phase, each subnet is trained individually and no mutual information across the classes may be utilised. In the globally supervised learning phase, teacher information is introduced to reinforce or anti-reinforce the decision boundaries obtained in the locally unsupervised learning phase.

Probabilitistic decision-based networks follow the original ideas of DBNNs, but have some additional characteristics. First, for face detection problems where the "non-face" class (background) can be considered as the complement of face class, the PDBNN detector uses only one, instead of using two, subnets in the architecture. This subnet represent the face class. Second, a PDBNN uses the positive training patterns (a positive pattern is defined as a virtual pattern which is only slightly perturbed from the original exemplary pattern and if the perturbation exceeds certain threshold this pattern will be considered as a negative pattern) to adjust the subnet parameters by unsupervised learning, and in the globally supervised learning phase it only uses the misclassified patterns for reinforced and anti-reinforced learning. The negative patterns are only used for anti-reinforced learning of the subnet. The decision boundaries are determined by a threshold. Finally, PDBNNs follow probabilitistic constraints, that is, the subnet discriminant functions of a PDBNN are designed to model the log-likelihood functions. Details of DBNNs and PDBNNs can be found in [109, 119].

Lin et al. [119] present an automatic face recognition system which performs human face detection (face vs non-face), eye localisation and face recognition. The probabilitistic decision-based neural network is applied to implement all the modules of the system. The goal of the detection system is to find the location of all faces and segment the face regions. During the detection, a confidence score is produced by the network, indicating the system's confidence on this detection result. If the score is below a certain threshold, then no object (face) is detected. Based on these regions, the eye localisation network will find where the eyes of the faces are and extract some useful features including width of head, distance between eyes, top of head to eyes, and between eye and nose. The face detection and eye localisation actually belong to the *one class object detection* problems in which there is only one class of interest in the detected pictures. These features will form the input of the face recognition network for classification. The system is tested on two public (FERET and ORL) databases and an in-house (SCR) database. In the face detection experiment, 92 annotated images are used for training and 473 images for testing. Before the detectors are applied to the input images, some preprocessing methods are applied, including the scaling of the images to a size of *320*×*240* pixels. The face size is defined as *140*×*100* pixels. To reduce the searching time, the images are further normally down-sized by a factor of 7. In this way, search range approximately becomes *46*×*35* pixels and block size is *12*×*12*, which is equivalent to the size of facial feature vector. Applying the network (with block input) as a template to sweep the scaled images with search step of one pixel, the face regions can be detected. Among all the 473 test faces, 98.5% of the errors are within five pixels and 100% are within 10 pixels in the original high-resolution image. The authors also claimed that working with the low resolution images, the detection system only used 200 ms to detect one face on a SUN Sparc 10 machine. However, the network detection system cannot detect "artificial faces" such as faces on poker cards and hand-drawn faces. The final recognition results on another database (FERET) show that the PDBNN network approach results in better recognition accuracy (99%) and faster processing speed than the traditional decision based neural network (DBNN, 96% accuracy) as well as the multilayer perceptron network (with an accuracy of 87.5%) approaches. It is noted that the final classification was based on the excellent face detection and eye localisation, that is, only 200 persons that were correctly detected among the 304 faces were used for the final classification and the above performance obtained was based on only these 200 faces rather than 304 faces.

Other/Hybrid/Multiple Networks

Waxman et al. [202] describe a computational neural system for target enhancement, detection/segmentation, learning and recognition/classification in visible, multispectral infrared (IR), and synthetic aperture radar (SAR) imagery. Multiple networks are used in this multiple stage approach, including segmentation networks (for object segmentation), centre-surround networks (for object edge enhancement), diffuse-enhance networks (for feature extraction), ART networks (for view or orientation learning and recognition) and aspect networks (for object recognition and evidence accumulation). The visible images tested here are model aircraft (F16, F18, HK-1) pictures captured using a conventional CCD camera. These targets are detected and segmented from the background using a combination of motion and contrast information. The approach achieved 71.1%, 77.1% and 83.9% accuracy of final recognition for F16, F18 and HK-1 objects. The authors also claimed that for all cases in target enhancement, detection/segmentation, learning and recognition, valuable insights can be derived from biological vision systems and translated into neural system architectures and computational networks.

Bosch et al. [21] propose a network based on spiking neurons performing visual object detection/segmentation. The network architecture consists of a feature map, a saliency map, a location map and an attention map. The feature map encodes the grey level input image provided as an external constant input. Weak illuminance results in a low firing frequency whereas a strong illuminance results in a high frequency. The saliency map extracts regions which differ from the background, through a centre/surround receptive field (filter) [130]. Since the contrast level may vary significantly within an object, local excitation in the saliency map accelerates the detection of the whole region. The location map computes the positions of these regions in the image and produces information encoding the priority of each position/location. The attention map generates an attention feedback to all salient regions in a sequential manner and selects the most active salient location by a winner-take-all mechanism [138]. The approach is tested on two input images. The first image contains three ("rock" or "stone") objects against a relatively uniform background. The second contains four vehicles against a non-uniform background (but not cluttered). These objects can be successfully detected by the approach. It is noted that this is a single class detection problem. The authors claim that this model is capable of processing real images, and can be adapted to a varying number of objects and non-constant backgrounds.

Ahmad and Omohundro [2] describe a network for extracting the locations of point clusters using selective attention. This work concentrates on the task of learning whether three clumps of points in a 256×256 image form an equilateral triangle or not. This system consists of an efficient focus of attention mechanism and a cluster detection scheme. The focus of attention mechanism allows the system to select any circular portion of the image in constant time. This is done by using locally tuned receptive fields (in which linear threshold units can give a localised response in a feature space), dynamic receptive fields (where the location and size of the receptive field can be quickly shifted in response to changing demands), and a focus of attention. To select the interesting locations, the focus of attention is decided by the coordinates of the locations. This is independent of any particular criterion (e.g. bottom-up which chooses the brightest image point, or top-down which represents the result of prior expectation). The cluster detector directs the focus of attention to clusters in the image. Once attention has been directed to that location, it is fine tuned to settle exactly on the centre of mass of the cluster. These two mechanisms are used to sequentially extract the relevant coordinates. The distances between the locations are computed to form a set of distance units. A standard feed forward network with a single hidden layer and a single output node produces the final decision/classification, that is, whether three points can form an equilateral triangle or not. In the experiment, a training set consisting of random triangles (approximately 50% of which are equilateral) with Gaussian noise added around each vertex is generated. For each triangle the focus of attention is initialised to cover the entire image plane. With a training set of only 100 triangles the network (output) score is consistently greater than 0.9 for equilateral triangles on an independent test set. The authors neither gave the number of triangles in the test set, nor the network output scores for non-equilateral triangles.

Rogers et al. [155] review the use of neural networks associated with the processing of military data to find and recognise targets. They divide automatic object recognition systems into four stages: selection of the sensors to produce the target measurements; preprocessing of the data and locating of the regions of interest within the data (segmentation); feature extraction and selection from the interesting and segmented regions; and processing of the features for decision making (classification). They also mention that the area of classification is where most automatic object recognition related neural network research has been carried out. They conclude as well that “artificial neural networks have been proven to be an interesting and useful alternate processing strategy, however, are not magical solutions with mystical abilities that work without good engineering” [155, page 1153].

Roth presents a survey of neural network technology for automatic target recognition [161]. He reviews automatic target recognition systems and gives some of highlights of neural network techniques that have the potential for making a significant impact on these systems. His focus is on the neural network technology development in the areas of collective computation, learning algorithms, expert systems and neurocomputer hardware. He concludes that “ultimately, neural network learning could be used for addressing the automatic object recognition needs for an adaptation to target and environment changes, a selection of good target features, and an integration of *a priori* knowledge about the target signatures and backgrounds [with network architectures]" [161, page 40].

A summary of the related work to object detection using neural networks is shown in table 2.9.

Kind of Network	Applications/Problems	Authors	Year	Source
Feed Forward Neural Networks	Target detection in thermal infrared images	Shirvaikar and Trivedi	1995	[172]
Shared Weight Neural Networks	Vehicle detection	Gader et al.	1995	[60]
		Won et al.	1997	[209]
Probabilitistic Decision-Based Neural Networks	Face detection and recognition	Lin et al.	1997	[119]
Other/Hybrid/Multiple Neural Networks	Aircraft detection and recognition	Waxman et al.	1995	[202]
	Vehicle detection	Bosch et al.	1998	[21]
	Face detection	Rowley et al.	1998	[162]
	Triangle detection	Ahmad et al.	1990	[2]
	Detection/extraction of weak targets for Radars	Roth	1989	[160]
	Review of target detection	Rogers et al.	1995	[155]
	Review of target recognition	Roth	1990	[161]

Table 2.9: Object detection based on neural networks.

2.5.3 Comments

This section summarises object classification and detection related work based on neural networks, including feature based and pixel based approaches. In the feature based approach specific features are manually selected and extracted then used as inputs to neural networks. The main advantage of this approach is that the network architectures are relatively small and network training generally takes relatively short time and low computational cost. The main disadvantages include a time consuming investigation of important features and a hand-crafting of feature extraction programs resulting in highly domain specific systems. The pixel based approach, on the other hand, avoids the above disadvantages by directly using image pixels as inputs. The main potential problem of this approach is that the network size is generally quite big, which might lead to relatively long training times. However, this can be ameliorated by the increasingly powerful computer hardware used today. Since this thesis aims to develop a domain independent approach, the pixel based approach is adopted.

2.6 Genetic Algorithms for Object Detection

This section presents a review of using genetic algorithms for object detection and related vision problems.

2.6.1 Classification and Feature Extraction

Bala et al. [11] present a hybrid methodology that integrates genetic algorithms and decision tree learning to evolve useful subsets of discriminatory features for recognising complex visual concepts. A genetic algorithm is used to search the space of all the possible subsets of a large set of candidate discrimination features. Candidate feature subsets are evaluated by using a decision tree learning algorithm (C4.5) to produce a tree, based on the given features using a limited amount of training data. The approach is tested on two different image databases, one involving satellite images (LANDSAT) and the other involving facial images (FERET). The results show that a genetic wrapper with C4.5 resulted in a significant decrease in both classification error and in cost as measured by the number of features used compared with C4.5 on full feature set.

Yang and Honavar [211] present an approach to the multi-criteria optimisation problem of feature subset selection using a genetic algorithm. They compare the classification performance of using neural networks with a subset features selected by the genetic algorithm against a neural network using the full set of features. The real world datasets obtained from the machine learning data repository at the University of California at Irvine [135] were used for the experiment. For the image segmentation experiment 210 patterns were used as the training set and 2100 for the test set. The network with the full feature set achieved 90.5% accuracy while the network with the subset of features selected by the genetic algorithm obtained 91.4% accuracy on the test set. Results on other other pattern recognition data samples gave a similar trend, that is, the performance with the subset of features selected by the genetic algorithm was slightly superior to that with all the attribute features. This indicated that genetic algorithms could offer an approach to solving the feature subset selection problem in inductive learning of neural network pattern classifiers. The details of images were not given in this paper.

Huang and Liu [83] propose a hybrid system based on a genetic algorithm with a Hopfield neural network. The system can recognise patterns formed by the transformation caused by rotation, scaling, or translation, singly or in combination. The method rests on a polygonal approximation technique which extracts appropriate feature vectors of specified dimensions characterising a given shape. These features are used as input to a network classifier for shape recognition. Object recognition is formulated as matching a global model graph with an input scene graph representing either a single object or several overlapping objects. A matrix needs to be defined for combining the state of each neuron in the 2D Hopfield neural model. The matrix is considered as genes of the genetic algorithm. The neural nodes represent the possible matches between the global and scene graphs and the linkages between the neural nodes comprise the constraints. Experiments on recognising hammers, wrenches and several other tools as the two class problems show that this hybrid method is better than the Hopfield approach only [8, 137]. The authors also claim that since genetic algorithms employ the probabilistic transition rule rather than the deterministic descent rule, they can avoid a local minimum, and are superior to a number of traditional optimisation techniques.

2.6.2 Object Detection

There is only a small number of reports of using genetic algorithms for the whole object detection problem (classification and localisation). These are presented in this subsection.

Swets, Punch and Weng[181] describe a technique of using genetic algorithms for object localisation in a complex scene. The training set consists of a labelled set of images which has been previously segmented such that only a single object of interest is contained in each of the images. Furthermore, the object in the image has a standard size, position and orientation; the image dimensions for all the images are identical in the training phase. In this phase, a simple "object mean" image is generated, where each pixel represents the average pixel value of all the training images for that object at that pixel position. In the test phase, no constraints on the input images are made; instead, a sub-image is extracted from the test image and projected to the image dimensions used for each object of interest during the training phase. An image distance is computed between the normalised sub-image extracted from the test image and the learned object mean image. The sub-image with the smallest distance from a particular object mean is taken to be the best sub-image for classification. In this approach, the coordinates of upper left and lower right corners of possible object positions are encoded into the chromosomes (strings) of the genetic algorithm and the distance measure is used as the fitness function. The approach is tested on a crowd image and the object, a specific face, was correctly detected by the genetic algorithm. They presented the details of this work in [180].

Goulermas and Liatsis [67] present a method of using genetic algorithms for fine-tuning the feature space for a Hough transform [86]. A hybrid system is configured, by embedding the Hough transform module into the genetic algorithm, which simultaneously performs feature space fine-tuning and circular shape detection. In this approach, the images are preprocessed before they are applied to the hybrid system and features are extracted for the representation in the chromosomes. Each gene corresponds to a unique feature point and chromosomes are represented as bit strings. The evaluation of a chromosome (fitness function) is based on a circular Hough transform variation [215]. The system was tested with synthetic (an artificially drawn image containing four small and two big overlapped circular disks) and real-world (underwater bubble-image with a high degree of noise) imagery. The hybrid genetic algorithm/Hough transform method produced very accurate detection results while the conventional Hough transform method [86] failed. However, the genetic algorithm system was very slow.

2.6.3 Other Vision and Image Processing Problems

Harvey and Marshall [70] present a method of designing morphological filters for specific tasks using genetic algorithms. The method was tested on noise-reduction problems with a set of training images for which the optimum filter was known. The results applying the genetic algorithm for morphological filter optimisation were quite impressive and the genetic algorithm seemed to have no problem in finding the global optimum sequence and the structuring element.

Tsang [194] presents the development of a genetic algorithm for aligning contours of near planar object shapes. This method was tested on examples of spanner, paper clip, scissor and hammer images and achieved better performance than other object alignment techniques such as dominant point simulated annealing. However, the genetic algorithm method required more computation.

Schaffer, Whitely and Eshelman [168] present a survey of the combination of genetic algorithms and neural networks. The methods were classified as supportive or collaborative by the authors. Supportive methods involve cases where genetic algorithms are used to help neural networks in their search and the neural network components are not determined by the genetic algorithms. Collaborative approaches, on the other hand, involve the evolutionary process being used to manipulate the neural network components and define the network topology.

The previous work related to object detection and vision problems using genetic algorithms is summarised in table 2.10.

Problems	Applications	Authors	Year	Source
Object classification and feature selection	Satellite and facial image	Bala et al.	1997	[11]
	Feature subset selection	Yang and Honavar	1997	[211]
	Shape recognition	Huang and Liu	1997	[83]
Object Detection	Face detection	Swets, Punch and Weng	1995	[181]
		Swets and Punch	1995	[180]
	Circular shape detection	Goulermas and Liatsis	1995	[67]
Other Problems	Morphological filter optimisation	Harvey and Marshall	1996	[70]
	Object alignment	Tsang	1997	[194]
	Survey	Schaffer, Whitely and Eshelman	1992	[168]

Table 2.10: Object detection related work based on genetic algorithms.

2.6.4 Comments

Compared with neural networks there are only a small number of reports of genetic algorithms for object detection. Reports on the use of genetic algorithms to evolve neural networks for multiple class object detection is even fewer and the sizes of the neural networks evolved by genetic algorithms are generally quite small. However, for the object detection problems investigated here, the network architectures are very large. It is still unclear whether genetic algorithms can be used to evolve large networks for multiple object detection problems. This thesis will investigate this issue (chapter 6).

2.7 Genetic Programming for Object Detection

Since the early 1990s, there has been only a small amount of work on applying genetic programming techniques to object classification, object detection, image processing, signal analysis and vision problems. This in part reflects the fact that genetic programming is a relatively young discipline compared with, say, neural networks.

2.7.1 Object Classification

Tackett [183, 184] uses genetic programming to assign detected image features to a *target* or *non-target* category. Seven primitive image features (mean and standard deviation of three windows/sub-images and the blob contrast of the object) and twenty statistical features (moment and intensity based) are extracted and used as the terminal set. The four standard arithmetic operators and a logic function are used as the function set. The fitness function is based on the classification result. The approach is tested on US Army NVEOD Terrain Board imagery, where vehicles such as tanks need to be detected/classified. When classifying feature vectors, the genetic programming method is found to outperform both a neural network classifier and a binary tree classifier on the same data, producing lower rates of false positives for the same detection rates. The best generated program is also found to require less computation at run-time than a neural network, and achieves better performance.

Andre [6] uses genetic programming to evolve functions that would traverse an image, calling upon co-evolved detectors in the form of hit-miss matrices to guide the search. These hit-miss matrices are evolved with a two dimensional genetic algorithm. These evolved functions can be used to discriminate between 2 letters or to recognise single digits.

Koza [103] (in chapter 15) uses a "turtle" to walk over a bitmap landscape. This bitmap is to be classified either as a letter "L", a letter "I", or neither of them. The turtle has access to the values of the pixels in the bitmap by moving over them and calling a detector primitive. The turtle uses a decision tree process, in conjunction with negative primitives, to walk over the bitmap and decide which category a particular landscape falls into. Limitations on resources prevented the discovery of a solution without using ADFs (automatically defined functions), although it is thought that a solution is possible. Using ADFs as local detectors and a constrained syntactic structure, some perfect scoring classification programs were found. Further experiments showed that detectors can be made for different sizes and positions of letters, although each detector has to be specialised to a given combination of these factors.

Teller and Veloso [186] use a genetic programming method based on the PADO language to perform face recognition tasks on a database of face images in which the evolved programs have a local indexed memory. PADO is a special language especially invented for genetic programming which uses iteration as well as functions. Although some encouraging results are obtained, they stress that the face database used, like many others in face recognition tasks, is insufficiently general to be of practical use. They use their genetic programming approach based on the

PADO language to perform a discrimination task between 5 classes of images [187]. The classes of images, consisting of different kinds of shapes placed at various sizes and positions on black backgrounds, are chosen to prove the classification ability of the PADO language such that the distinctions between these classes are conjunctions of abstract features. In addition, the discrimination can be performed on these images with some noise and obstructions added. Without noise, this approach achieves respectable performance of up to 60% correct classification.

Robinson and McIlroy [154] apply genetic programming techniques to the problem of eye location in grey-level face images. The input data from the images is restricted to a 3000 pixel block around the location of the eyes in the face image. This approach produced promising results over a very small training set, up to 100% true positive detection with no false positives, on a three image training set. Over larger sets the genetic programming approach performed less well however, and could not match the performance of neural network techniques.

Winkeler and Manjunath [206] produce genetic programs to locate faces in images. Face samples are cut out and scaled, then pre-processed for feature extraction. The statistics gleaned from these segments are used as terminals in genetic programming which evolves an expression returning how likely a pixel is to be part of a face image. Separate experiments process the grey scale image directly, using low level image processing primitives and scale-space filters.

2.7.2 Object Detection

All the genetic programming based object detection approaches which have been reported so far belong to the *one class object detection* category (section 2.1.2, page 10). In these detection problems, there is only one object class of interest which needs to be detected in the large pictures. Such examples are presented in this subsection.

Howard, Roberts and Brankin [82] present a genetic programming approach to automatic detection of ships in low-resolution synthetic aperture radar imagery. The detector design goals are to maximise detection accuracy across images, to minimise the computational effort during image processing, and to minimise the effort during the design stage. A number of random integer and real constants and pixel statistics, that is, domain independent, pixel level features are used as terminals. The four arithmetic operators and *min* and *max* operators constitute the function set. The fitness is based on the number of the true positives and false positive objects detected by the evolved program. This is actually a one class object detection problem, where only ships need to be detected in the large pictures. A two stage evolution strategy that comprises three distinct steps was applied in this approach. In the first stage genetic programming evolved a detector (it is a classifier in nature) that could correctly classify the chosen target (ship) pixels from the chosen non-target (ocean) pixels. The best generated detector was then applied to the entire image and produced a number of false alarms. In the second stage, a brand new run of genetic programming was tasked to discriminate between the clear targets and the false alarms as identified in the first stage and another detector was generated. This two stage process resulted in two detectors that were then fused using the *min* function. These two detectors return a real number, which if greater than zero denotes a ship pixel, and if zero or less denotes an ocean pixel. The approach was tested on images chosen from commercial SAR imagery, that is, a set of 50m and 100m resolution images of the English Channel taken by the European Remote Sensing satellite. One of the 100m resolution images was used for training, two for validation and two for testing. The training was quite successful with no false alarms, while there was only one false positive in each of the two test images and the two validation images which contained 22, 22, 48 and 41 true objects. The authors concluded that at least as far as accuracy was concerned the results compare very favourably with those on the same problem by other techniques such as Kohonen neural networks.

Isaka [88] uses genetic programming to locate mouth corners in small (50×40) images taken from pictures of faces. Processing each pixel independently using an approach based on relative intensities of surrounding pixels, the genetic programming approach was shown to perform comparably to a template matching approach on the same data.

2.7.3 Other Vision and Image Processing Problems

Lucier, Mamillapalli and Palsberg [122] use four program optimisation methods (boundary checks, variable bindings, fixnum arithmetic, and flattening and/or) in a genetic programming approach for edge detection problems. The approach was tested on three images: a bank image, the Lenna (a woman's face) image, and an F16 helicopter image. The image size was 512×512 pixels. The results showed that the use of fixnum arithmetic was the most effective optimisation, while flattening and/or optimisation was the least effective. These optimisations sped up the evolutionary process by a factor of 17 for the bank and Lenna images and 22 for the F16 images. The times for the evolutionary process to find a good solution with and without these optimisations for the Lenna image were 583 and 35.2 hours respectively.

Koza [101, 102] uses automatically defined functions to show how hierarchically structured solutions can be applied to the San Mateo trail problem. This problem involves programming an artificial ant to follow a trail of food on a grid world. The trail may have gaps of one or two squares where no food is present. The ant is allowed to use basic local sensor information, and can either more forward or turn 90° in response to some condition. Automatically defined functions used in this context are interpreted by Koza as detectors for certain configurations of terrain, returning positive values if the terrain matches some configuration represented by that automatically defined function. Koza shows that using automatically defined functions reduces the average structural complexity of a solution and the number of individuals that have to be processed by the search before a solution is found.

Nordin and Banzhaf [139] apply a linear structured form of genetic programming, based on primitive sets consisting of machine-code instructions, to the programmatic compression of digital signals, both sampled sound signals and 2-D raster images. The genetic programming approach is a simple application of symbolic regression to the input signal, using a squared distance error function to drive the evolutionary process. In the experiment of image compression, pictures consisting of *256*×*256* pixels which were divided into squares of either *16*×*16* or *8*×*8* pixel blocks were used. This size corresponded to the size used by several other image compression techniques such as JPEG [145]. The CPU times for compressing a picture in the experiments on a SUN20 were in the range of 30 minutes to 10 days depending on the requested quality and compression ratio.

A summary of object detection related work based on genetic programming is shown in table 2.11.

2.7.4 Comments

As a relatively young learning/adaptive technique, genetic programming has been only applied to object classification and detection problems in very recent years. The main advantage is that the genetic programming evolutionary process can automatically evolve computer programs for a particular problem and achieve quite reasonable results. However, most work has focused on two class object classification or one class object detection problems. Due to the high computational cost, the technique has previously only been used for relatively easy problems and no work on genetic programming for multiple class object detection problems has been reported. This thesis will focus on the investigation of the use of genetic programming for multiple object detection problems of increasing difficulty, including very difficult problems (chpater 7).

Problems	Applications	Authors	Year	Source
Object Classification	Tank detection (classification)	Tackett	1993	[183]
		Tackett	1994	[184]
	Letter recognition	Andre	1994	[6]
		Koza	1994	[103]
	Face recognition	Teller et al.	1995	[186]
		Winkeler et al.	1997	[206]
	Shape recognition	Teller et al.	1995	[187]
	Eye recognition	Robinson et al.	1995	[154]
Object Detection	Ship detection	Howard et al.	1999	[82]
	Mouth detection	Isaka	1997	[88]
Other Vision Problems	Edge detection	Lucier et al.	1998	[122]
	San Mateo trail problem	Koza	1992	[101]
		Koza	1993	[102]
	Image compression	Nordin et al.	1996	[139]

Table 2.11: Object detection related work based on genetic programming.

2.8 Summary and Discussion

This chapter reviews object detection problems, machine learning paradigms and the basic idea and components of the three learning/adaptive techniques used in this thesis, that is, neural networks, genetic algorithms, genetic programming. An overview of previous work related to object detection and other vision problems using these three learning techniques is also presented.

As presented before, most object detection related work based on neural networks, genetic algorithms and genetic programming which has been reported was focused on one or more of the following issues:

- Pure object classification problems in which no object localisation is done, or one class object detection problems where only one object or multiple objects in a single class of interest need to be detected. It is important to note that detecting multiple class objects of interest in large pictures using a single computer program either learnt by neural systems or evolved by a genetic learning process has not been reported so far.

- Feature based approaches where domain dependent, specific features are previously extracted and used as inputs. This usually involves a time consuming investigation of good image features for a specific domain and hand-crafting of programs for feature extraction and selection.

- Using multiple independent stages which produce multiple programs to find the classes and locations of the objects of interest in large pictures. The current stage usually uses the results of the previous stage as input, rather than the original source data. The final results rely too much on the results of each stage.

To address the current gap in this area, this thesis focuses on the investigation of a learning/adaptive, domain independent approach to multiple class object detection problems. Rather than using traditional image processing and vision methods, we apply neural networks, genetic algorithms and genetic programming techniques. Rather than using specific features for a given database in a particular domain, pixel based data, that is, either raw pixels or domain independent, pixel level statistics are used. Rather than using multiple independent stages, this thesis uses a single learnt program for multiple class object detection problems.

Chapter 3

Image Databases

To investigate the strengths and limitations of the learning/adaptive approaches described in this thesis, three different grey scale databases were used in the experiments. The pictures were selected to provide detection problems of increasing difficulty. The three different detection problems are presented in table 3.1. Example pictures are given in figure 3.1 (page 77), figure 3.2 (page 79) and figure 3.3 (page 82). The key characteristics of these databases and the detection problems are summarised in table 3.2.

Difficulty of object detection task	Database Name	Detection Problems
Easy	Easy pictures (Synthetic)	Detection of filled circles and squares against a uniform background
Medium Difficulty	Coin pictures (real world)	Detection of different sides of different coins against a relatively uniform background
Very difficult	Retina pictures (real world, for medical use)	Detection of haemorrhages and micro-aneurisms against a highly cluttered background

Table 3.1: Three different databases and the corresponding detection problems.

		Easy Pictures	Coin Pictures	Retina Pictures
Entire images (for object detection)	Total number of images in database	10	20	20
	Number of images in detection training set	4	10	12
	Number of images in detection test set	6	10	8
	Size of images (pixels)	700×700	640×480	1024×1024
Object examples (cutouts, for classification only)	Total number of objects	240	200	164
	Number of objects in classification training set	60	100	100
	Number of objects in classification test set	180	100	64
	Max size of objects (Size of input field)	14×14	24×24	16×16
Object Classes	Number of total classes	4	5	5
	Number of classes of interest	3	4	2

Table 3.2: Key characteristics of the three databases and detection problems. (Note: the definition of the terms used in this table can be found in section 4.1.1, page 88.)

3.1 Easy Pictures

The first database consists of several synthetic pictures, the easy pictures, *700× 700* pixels, which were generated to give well defined objects against a uniform background. A sample easy picture is presented in figure 3.1. The pixels of the objects were generated by using a Gaussian generator based on the normal distribution [4] with different means and variances for different classes. All the objects in each class have the same size, but are located at different positions.

There are three classes of small objects of interest in this database: black circles ($class1$), grey squares ($class2$) and white circles ($class3$) against a uniform grey background (class *other*). The three kinds of objects were generated with different intensities. Ten and 5, 180 and 25, 230 and 20, and 140 and 0 were taken as the mean and standard deviation for *class1, class2, class3* and *other*, respectively, as shown in table 3.3.

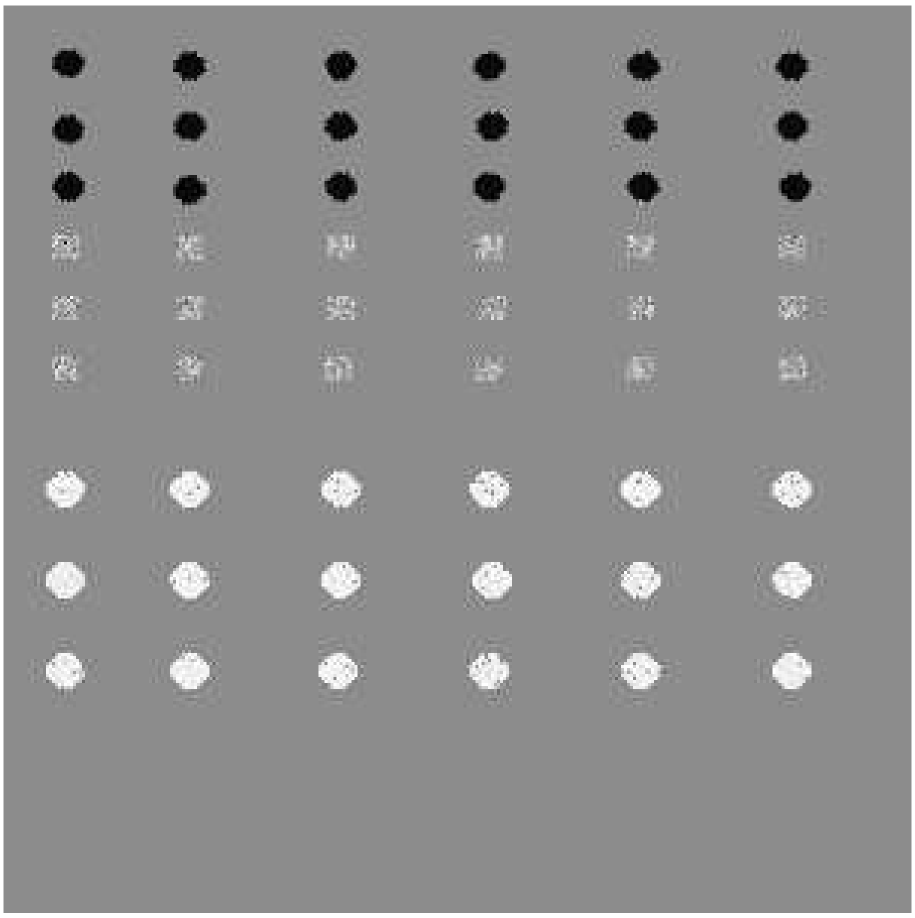


Figure 3.1: A sample easy picture.

Class names	Description	Mean	Standard Deviation
*class*1	Black circles	10	5
*class*2	Grey squares	180	25
*class*3	White circles	230	20
other	Background	140	0

Table 3.3: Description of object classes in the easy pictures.

For neural network related approaches, 240 object examples, also called *cutouts*, were cut out from the four entire images in the detection training set, 60 for each of the four classes. Of these, 60 were randomly selected for network training and other 180 were used for network testing. Network training and testing is an object classification problem where no object localisation procedure is carried out. After network training and testing, the six entire images in the detection test set were used for object detection where both classification and localisation were carried out. For the genetic programming based approach, the entire pictures in the detection training set were directly used for training and the entire pictures in the detection test set were used to measure the object detection performance.

3.2 Coin Pictures

The second database contains 20 coin pictures (640×480 pixels) which were intended to be somewhat harder than the easy pictures and were taken with a CCD camera over a number of days with relatively similar illumination. In these pictures the background varies slightly in different areas of the image and between the images. The brightness of objects also varies in a similar way. The objects to be detected are more complex than those in the easy pictures, but still regular. A typical sample coin picture is shown in figure 3.2. All the objects in each class have a similar size. They are located at arbitrary positions and with rotational invariance in two dimensions.

Each of the pictures contains four object classes of interest, which are the head side of 5 cent coins (class $head005$), the head side of 20 cent coins (class $head020$), the tail side of 5 cent coins (class $tail005$) and the tail side of 20 cent coins (class $tail020$). The background (class $other$) is relatively uniform – not totally uniform because of the different lighting conditions and camera positions. A summary of the object classes is described in table 3.4.

Class names	Description
$head005$	Head side of 5 cent coins
$tail005$	Tail side of 5 cent coins
$head020$	Head side of 20 cent coins
$tail020$	Tail side of 20 cent coins
$other$	Background

Table 3.4: Description of object classes in the coin pictures.

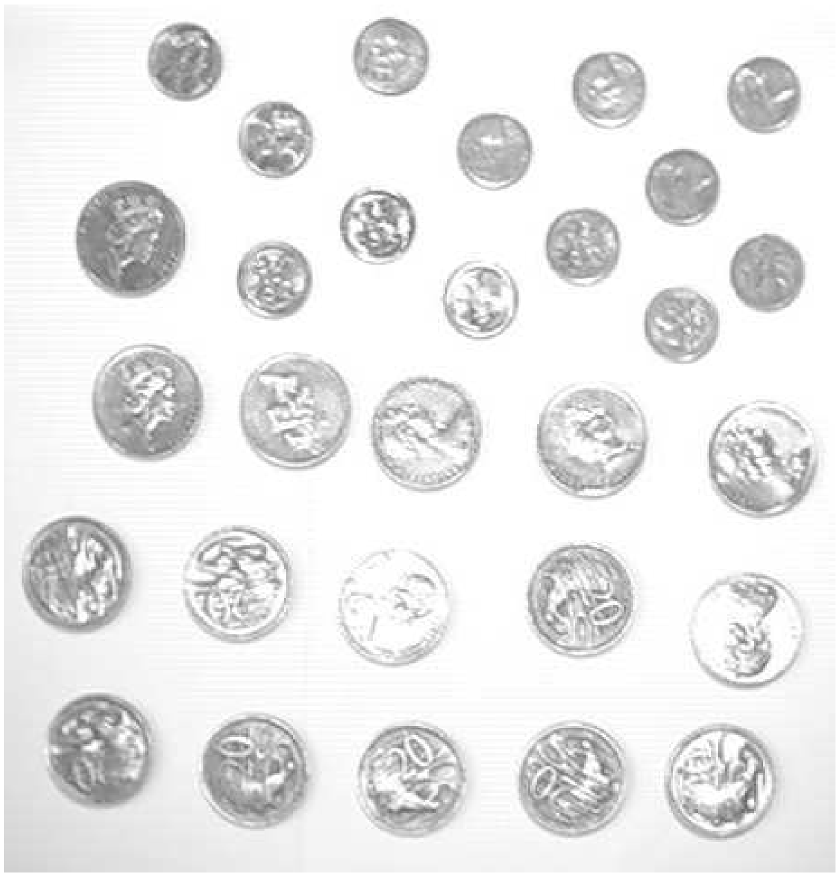


Figure 3.2: A sample coin picture.

We split the 20 coin pictures in the database into two separate data sets: a detection training set which contains 10 entire pictures and a detection test set which contains the other 10 entire images. For neural network related approaches, 200 object examples were cut out from the detection training image set, 40 for each of the five classes. Of these, 100 were randomly selected for network training and the other 100 for network testing, which only involves object classification. Then the trained network was applied to the entire images in the detection test set to measure object detection performance, including both object classification and localisation. For the genetic programming based approach the entire images in the detection training set were directly used by the evolutionary process to generate the learnt computer programs, which were then used to detect the objects of interest in the entire pictures in the detection test set.

3.3 Retina Pictures

The third database consists of 20 large retina pictures (*1024*×*1024* pixels), which were taken by a professional photographer with special apparatus at a clinic. Figure 3.3 (page 82) shows an example of the retina pictures. Compared with the previous two databases and other databases used in automatic target recognition or detection problems such as the recognition of regular, man-made small objects, the detection problems in this database are more difficult. They contain very irregular and complex objects in several classes against a highly cluttered background.

There are two object classes of interest: haemorrhages (class *haem*) and micro-aneurisms (class *micro*). Figure 3.4 (page 83) shows examples of haemorrhages and figure 3.5 (page 84) gives examples of micro-aneurisms. In these two figures, haemorrhage and micro-aneurism examples are labelled using white surrounding squares.

These objects are not only located in different places, but the sizes of the objects in each class are different as well, particularly for the haemorrhages. In addition, there are also other objects of different classes, such as veins (class *vein*, figure 3.6, page 85) with different shapes, and retina "edges" (class *edge*, figure 3.7, page 86). The backgrounds (class *other*) are varied, some parts are quite black, some parts are very bright, and some parts are highly cluttered. These object classes are summarised in table 3.5.

Class names	Description	Number of object samples for network training
haem	haemorrhages	31
micro	micro-aneurisms	33
vein	veins, including large, medium and thin veins	33
edge	retina edges	33
other	highly cluttered background	33

Table 3.5: Description of object classes in the retina pictures.

Twenty retina pictures in the database were split into two image sets: a detection training set which contains 12 pictures and a detection test set which consists of the other 8 pictures. For training the neural networks, 161 object examples of the five classes were cut out from the detection training image set. The numbers of the object examples in the five classes are given in table 3.5. Of these, 100 were used for network training and 61 were used for network testing, which only involves object classification. The learnt program, that is, a trained network was then applied to the detection test set for object detection which involves both object classification and localisation. Similarly to the two previous databases, the entire images in the detection training set were directly used for the training procedure of genetic programming and the entire images in the detection test set were used for the measure of the detection performance.

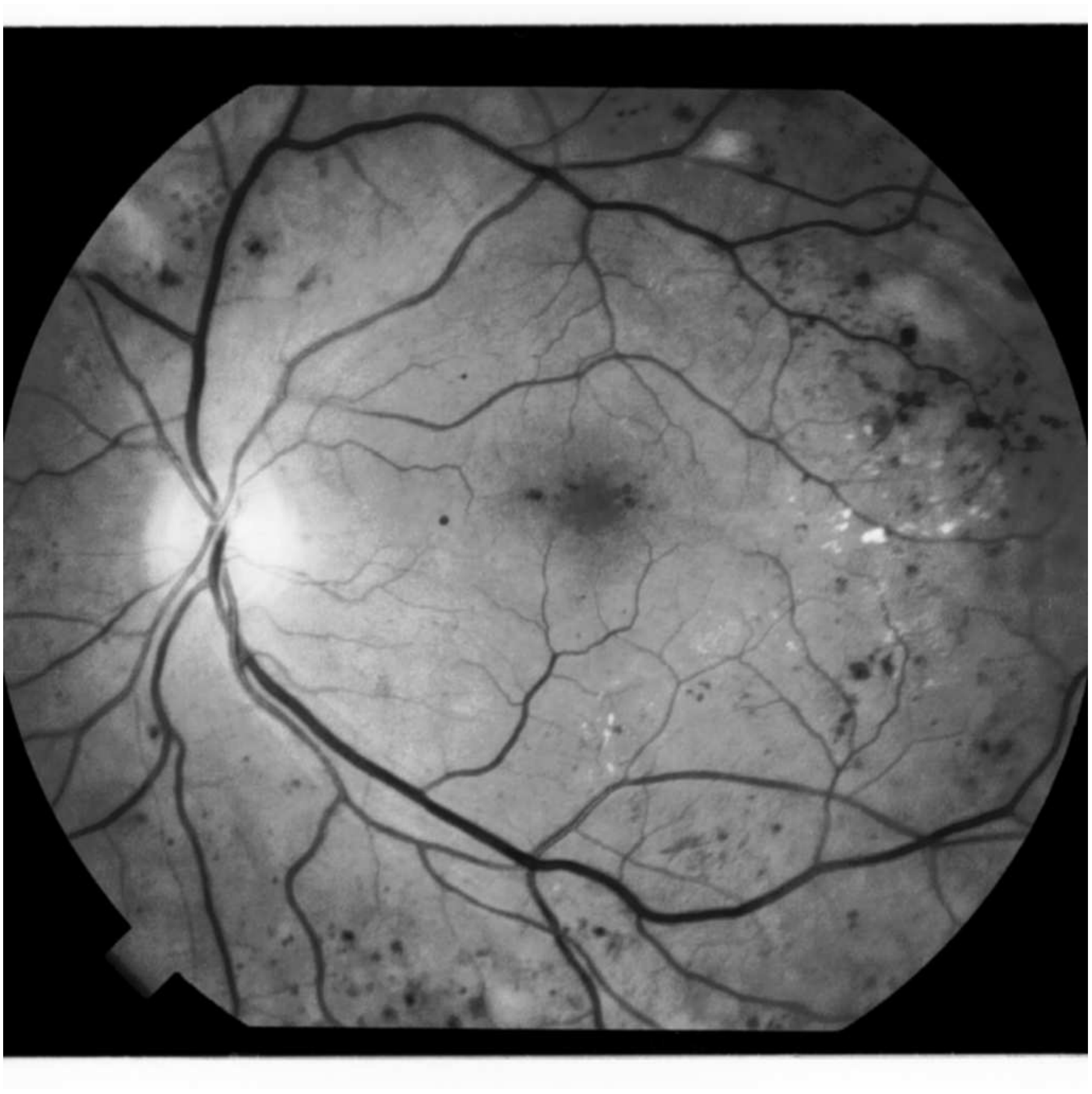


Figure 3.3: A sample retina picture.

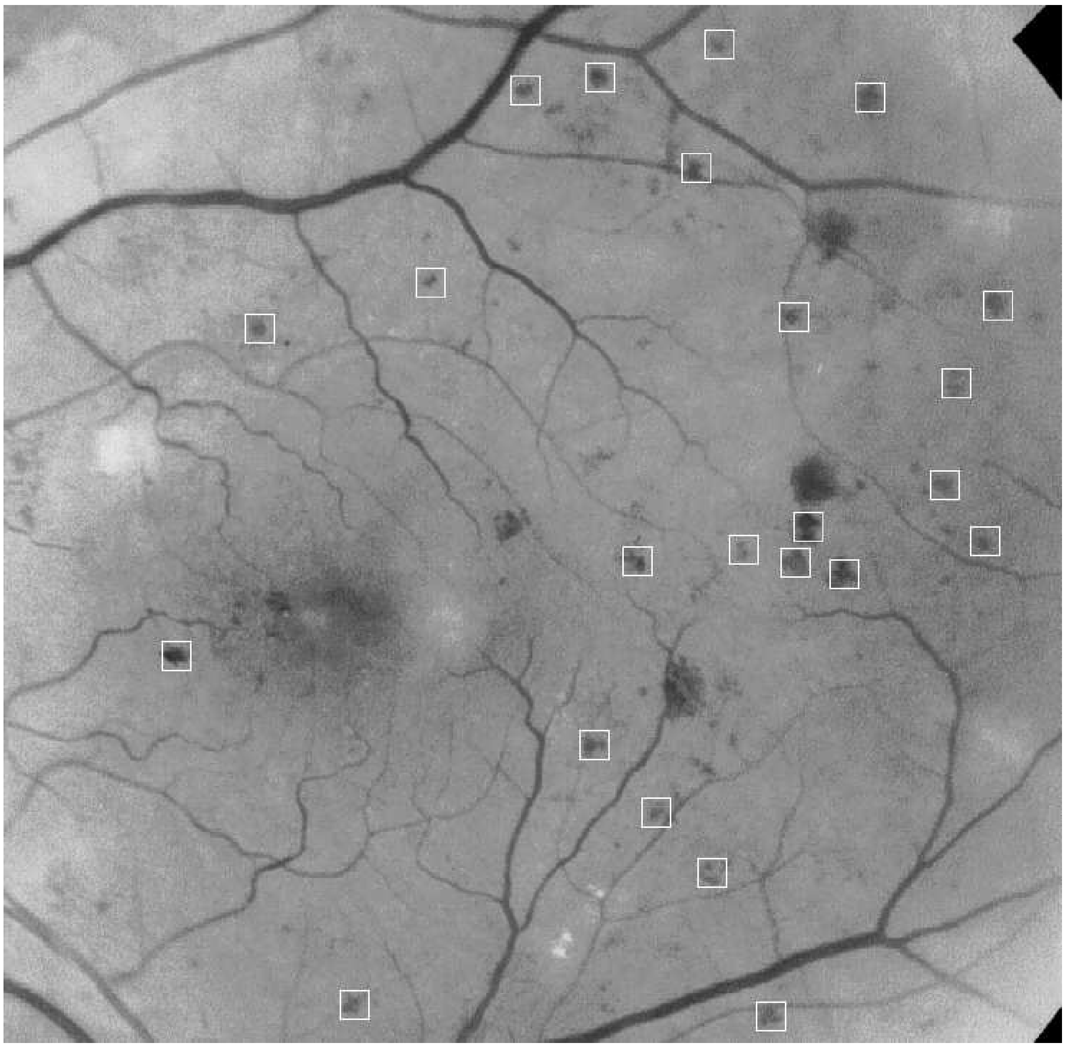


Figure 3.4: Haemorrhage examples.

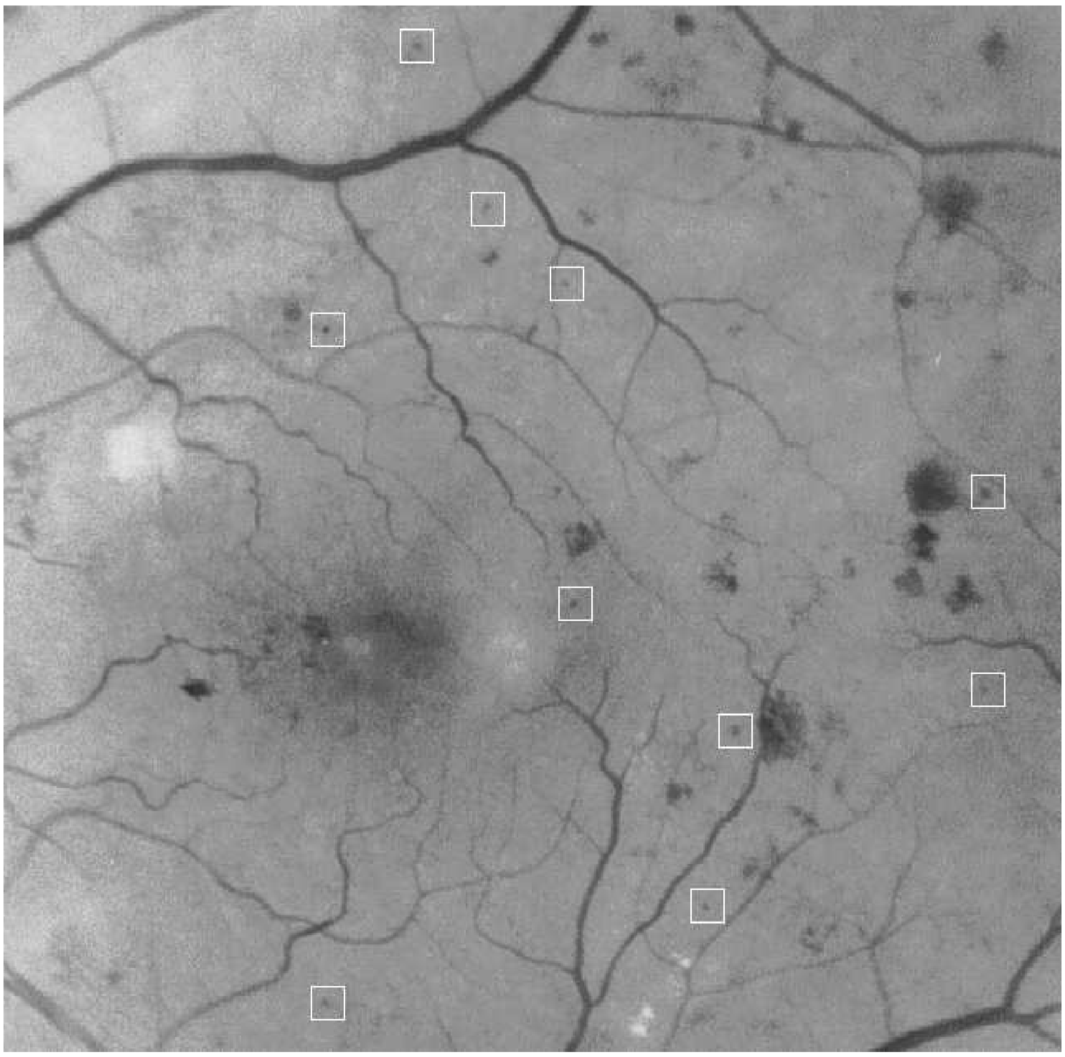


Figure 3.5: Micro-aneurism examples.

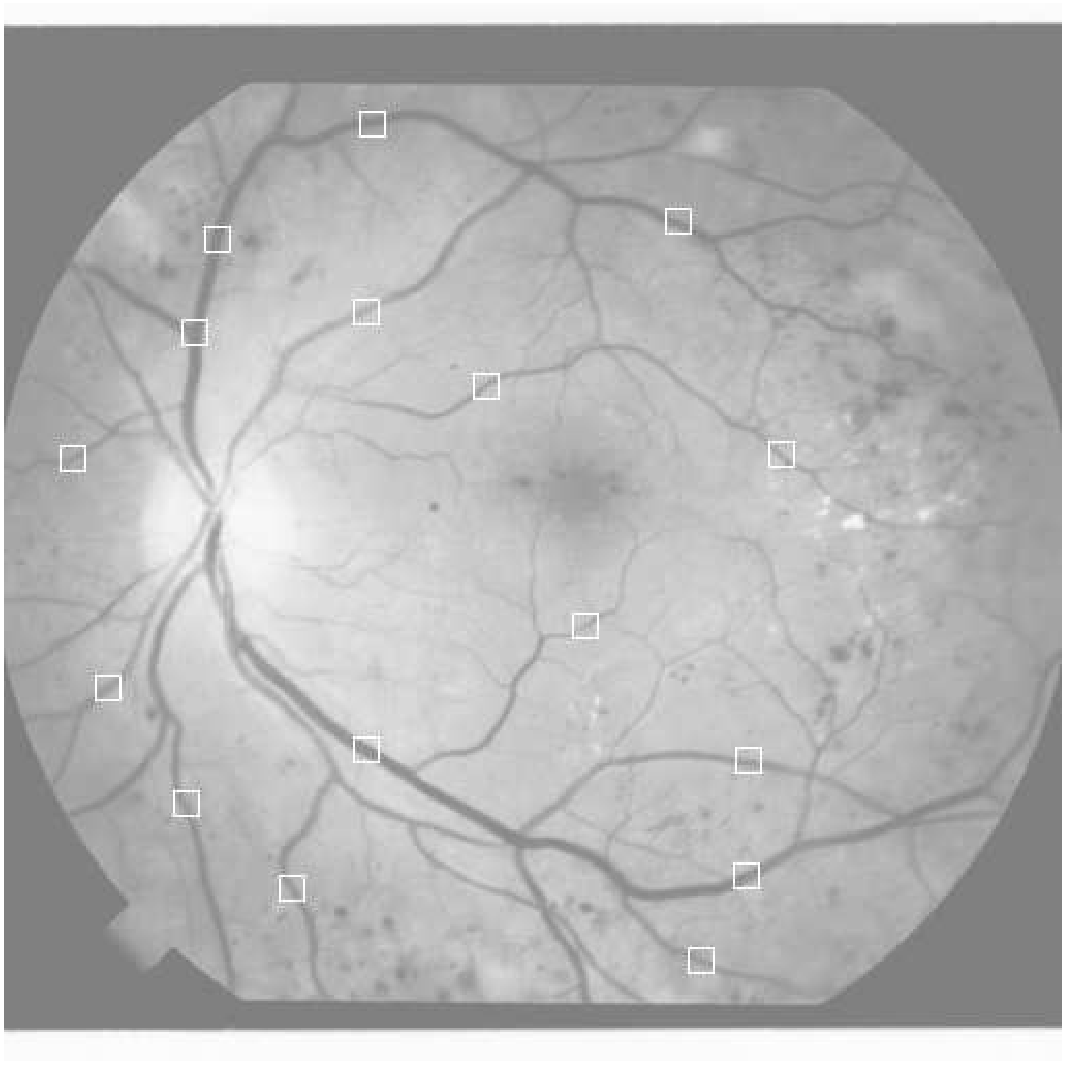


Figure 3.6: Vein examples.

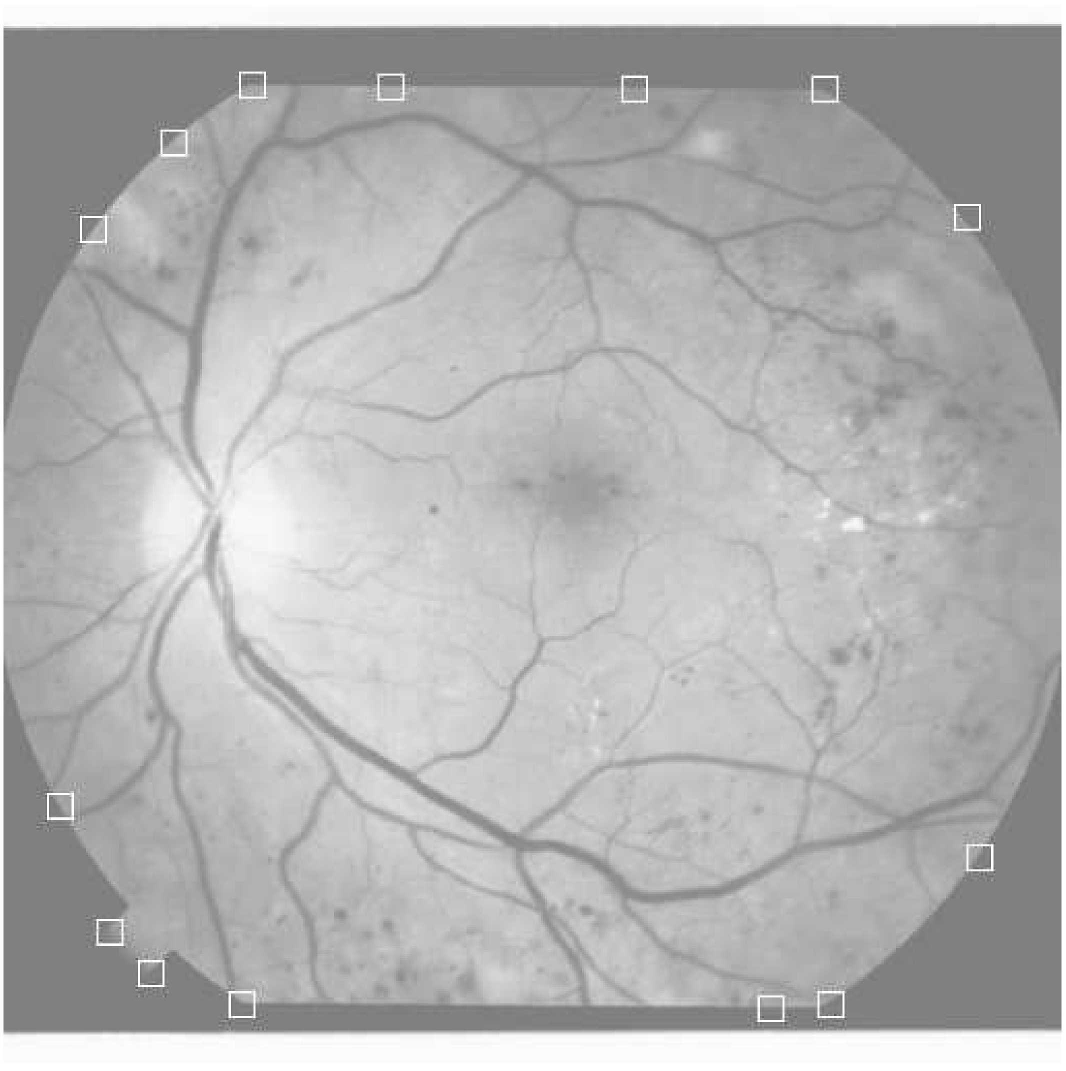


Figure 3.7: Retina edge examples.

Chapter 4

Neural Networks for Object Detection– the Basic Approach

In this chapter, we first give a general overview of our basic approach to the use of multilayer feed forward neural networks for detecting small objects in large pictures. We then present the key ideas and algorithms for this approach, including the generation of different data sets, the determination of the network architecture, network training and testing, finding the centres of the objects and object matching. After presenting the experimental results for network training and testing in object classification and the results for object detection, we summarise the basic approach at the end of this chapter.

4.1 Introduction

Since our goal is domain independent object detection, we use raw pixel data as inputs and avoid the feature extraction and selection which tend to make vision systems domain specific. In this chapter, we describe the development of a pixel based neural network approach to multiple class object detection problems in which the locations of multiple objects in each class of interest must be found.

For presentation convenience, we call the approach described in this chapter *the basic approach.* Its performance will be used as the baseline for the purpose of comparisons with the variations and extensions described in chapters 5, 6 and 7.

4.1.1 Terminology

To avoid confusion, we define the following terms related to the image data sets used in this thesis:

Definition 4.1 *Image Data Set*
An image data set refers to a set of entire images in a database. In this thesis, there are three image data sets used for the experiments, that is, the easy pictures, the coin pictures and the retina pictures, as described in chapter 3.

Definition 4.2 *Detection Training Set*
A detection training set refers to a set of entire images randomly split from the image data set and used to learn a detector, that is, either a neural network or a computer program, for object detection. In this thesis, the detection training set is used to generate a classification data set (see below definition 4.5) for network training and testing, and is also used directly by a genetic algorithm for network refinement (chapter 6) and a genetic programming evolutionary process to generate a computer program (chapter 7).

Definition 4.3 *Detection Test Set*
Similarly to a detection training set, a detection test set also refers to a set of entire images randomly split from the image data set. However, the detection test set is used for measuring object detection performance. The images in this set are *unseen* to the training process and are used to investigate the generalisation ability of the learning methods.

Definition 4.4 *Cutouts*
Cutouts refer to the object examples which are cut out from the detection training set. The cutouts are used to form classification data sets. Sample cutouts from the easy database are shown in figure 4.4 (page 93).

Definition 4.5 *Classification Data Set*
A classification data set refers to a set of cutouts. In this thesis, the classification data sets are used only for object classification, including network training and testing. These data are not used for measuring object detection performance.

Definition 4.6 *Classification Training Set*

A classification training set, also called *classification training data*, refers to a set of cutouts randomly split from the classification data set. This data set is used for network training in object classification.

Definition 4.7 *Classification Test Set*

A classification test set, also called *classification test data*, is similar to the classification training set except that it is used for measuring object classification performance by applying the trained network during network testing.

The relationships between the various data sets are shown in figure 4.1. The terms *full patterns, input patterns* and *output pattern* in this figure will be described in section 4.3.1 (page 93).

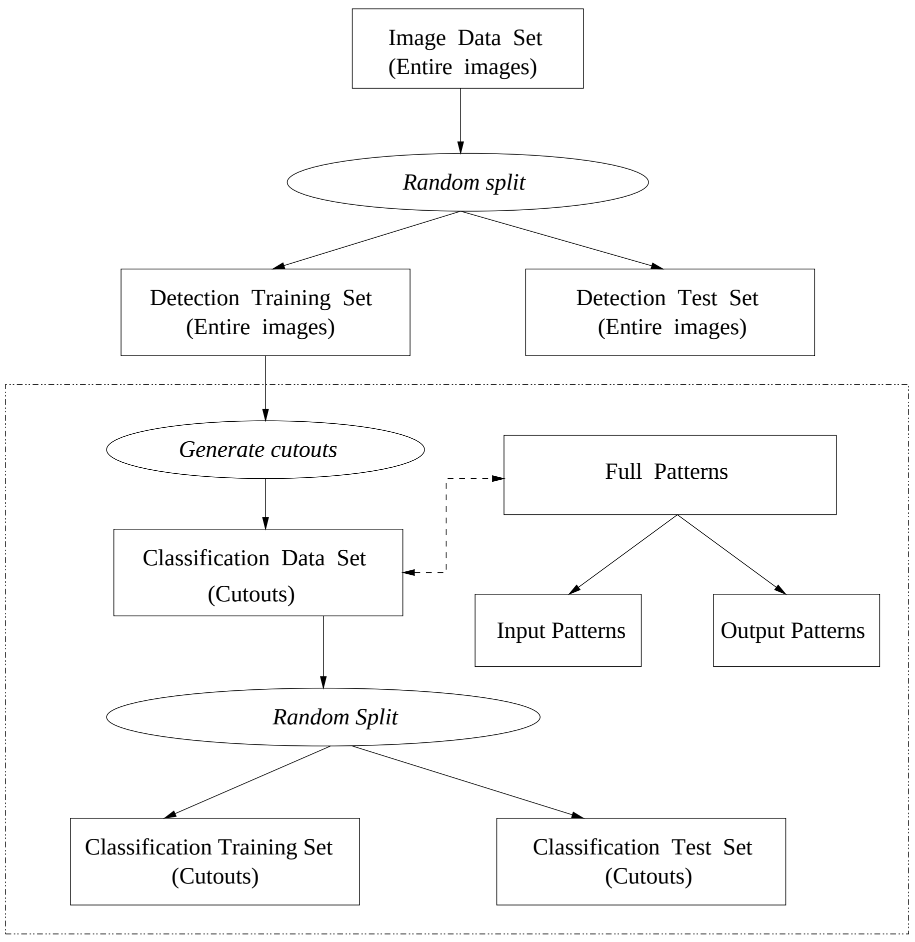


Figure 4.1: Relationships between classification and detection data sets.

4.1.2 Overview of the Basic Approach

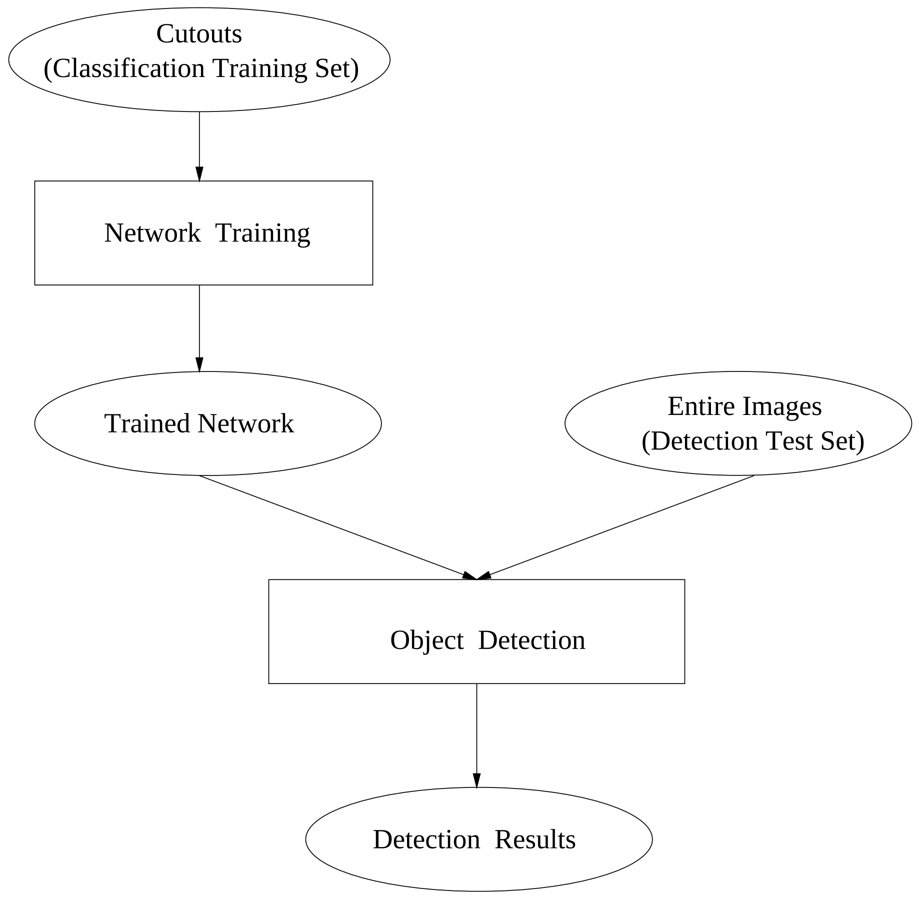


Figure 4.2: An overview of the basic approach.

An overview of the basic approach is presented in figure 4.2. It consists of six main steps, which are briefly described as follows:

1. Assemble a database of pictures in which the locations and classes of all the objects of interest are manually determined. Divide these full images into two sets: a *detection training set* and a *detection test set*. See section 4.2 on page 92 for details.

2. Determine an appropriate size (n) of a square which will cover all objects of interest and form the input field of the networks. Generate a classification data set by cutting out squares of size $n \times n$ from the detection training set. Each of the squares, called cutouts or sub-images, only contains a single object and/or a part of the background. Randomly split these cutouts into a classification training set and a classification test set. These classification data sets are used for object classification, which involves network training

and network testing. Note that all of the backgrounds are also considered as one class, but not a class of interest. See section 4.3 on page 93 for details.

3. Determine the network architecture. A three layer feed forward neural network is used in this approach. The $n \times n$ pixel values form the inputs of a training pattern (definition 4.8 on page 94) and the classification is the output. The number of hidden nodes is empirically determined. See section 4.4 on page 95 for details.

4. Train the network by the backward error propagation algorithm on the classification training data. The network training process will be stopped according to the termination strategy. The trained network is then tested on the classification test set to measure the object classification performance. The classification test data is also used to avoid overtraining the network. See section 4.5 on page 97 for details.

5. Use the trained network as a moving window template [13] to detect the objects of interest in the detection test set. If the output of the network for a class exceeds a given threshold then report an object of that type at the current location. It is important to note that the thresholds for different classes are different. This is done by a network sweeping procedure and an object centre finding algorithm. See section 4.6 on page 98 for details.

6. Evaluate the object detection performance of the network by comparing the classes and locations generated by the network sweeping procedure with the known true classes and locations and calculating the detection rate and the false alarm rate. See section 4.7 on page 101 for details.

A flow diagram for the basic approach is shown in figure 4.3.

4.1.3 Chapter Goals

In this chapter, the following specific research questions will be investigated:

- Will the basic approach work for detecting simple objects in large pictures with a uniform background — the easy pictures?
- Can it be used for regular object detection in large pictures with a relatively uniform background — the coin pictures?
- Is it possible to detect complex objects in large pictures with a highly cluttered background using this approach — the retina pictures?

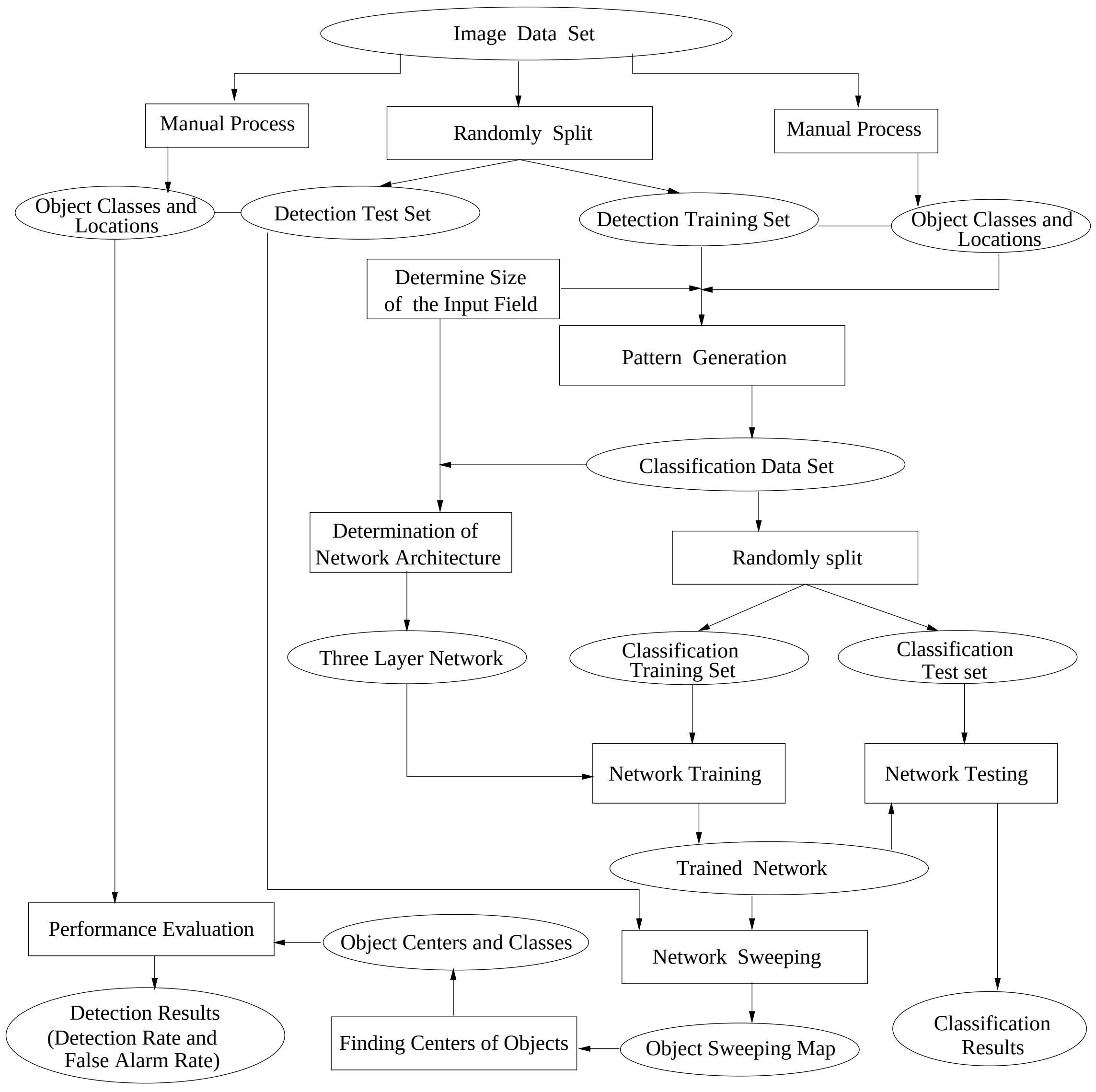


Figure 4.3: Flow diagram of the basic approach.

4.2 Generation of Image Data Sets

In the first step of the approach, the entire pictures in the assembled image database are randomly split into a detection training set and a detection test set. The detection training set is used for the development of a neural classifier and the detection test set is used for the measurement of the object detection performance of the classifier. In these data sets, the classes and locations of the objects of interest are manually determined. The classes and locations in the detection training set are used for the generation of a classification data set consisting of a set labelled $n \times n$ cutouts. These form the training patterns and test patterns (definition 4.10 on page 94). The classes and locations in the detection test set will be used for the comparison with the classes and locations (centres) of the objects reported by the detection procedure (section 4.6 on page 98).

The details of the detection training sets, detection test set and sample images for the three databases used in this thesis can be found in chapter 3.

4.3 Generation of Classification Data Set

4.3.1 Generation of Patterns

Before the network is trained, the input patterns need to be generated. This is done by cutting out a number of sub-images with a square size of $n \times n$ (the input field size) from the detection training image set according to the object locations and classes which have already been determined in step one. Some examples of the cutouts for the detection problem in the easy picture database are presented in figure 4.4.

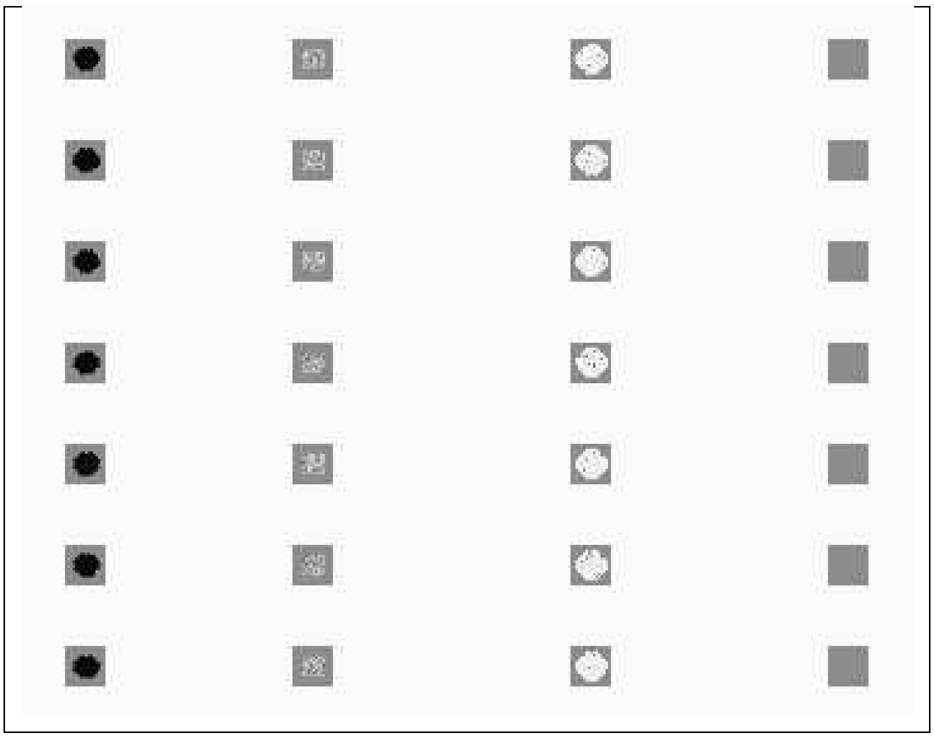


Figure 4.4: Samples of network input patterns for the easy pictures.

To avoid confusion, we give the following definitions:

Definition 4.8 *Input Pattern.*

The set of activations of all input nodes in the network is called an input pattern. As can be seen from figure 4.4, input patterns here refer to a number of cutouts, that is, small object examples plus background, of each class which have been cut out from the detection training set. In the basic approach the pixel values are used directly as inputs to neural networks and the pixel values of such a cutout form an input pattern of the network. In other words, a cutout corresponds to an input pattern. The number of pixels is decided by the size of the cutouts, which have the same size as the input field. Pixel values are scaled from [0, 255] to [0, 1.0].

Definition 4.9 *Output Pattern*

The set of activations of all output nodes in the network is called an output pattern. We use 1.0 to represent the activation of the desired class and 0.0 to represent the activation of non-desired classes. For instance, if there are four output nodes in the neural network which correspond to four object classes in a problem domain, then (1.0, 0.0, 0.0, 0.0), (0.0, 1.0, 0.0, 0.0), (0.0, 0.0, 1.0, 0.0) and (0.0, 0.0, 0.0, 1.0) are the four output patterns for class1, class2, class3 and class4 respectively.

Definition 4.10 *Pattern, Training Pattern and Test Pattern*

The input pattern and its corresponding output pattern constitutes a full pattern, or simply a pattern.

In the classification data set there are a number of patterns which will be used for network training and testing. As discussed in section 4.1.1 (page 88), the patterns are randomly split into a classification training set and a classification test set. The patterns in the classification training set are called training patterns, and the patterns in the classification test set are test patterns.

Definition 4.11 *Input Field*

The input field here refers to a square in the large images. This is used as a moving window for the network sweeping process (see page 98). The size of the input field is the same as that of the cutouts which form the input patterns.

4.3.2 Determination of the Input Field Size

In this approach, the size of the input field is determined according to the following heuristic rule:

The input field should be sufficiently large to characterise the background but should be small enough to contain only a single object of interest.

According to the heuristic rule, the size of the input field can be calculated by equation 4.1.

$$n = size\_of\_object + b \tag{4.1}$$

where *size_of_object* refers to the size of the biggest object which needs to be detected and b stands for a chosen size of the background to surround the object. After preliminary trials of a range of sizes, we set b as *2-4* pixels. In the easy pictures used in the thesis, the largest object size is *12*×*12*, and the input field size for the detection problem in this database is defined as *14*×*14*. Similarly, typical input field sizes used for the coin pictures and the retina pictures are *24*×*24* and *16*×*16* respectively.

4.4 Determination of the Neural Network Architecture

We use three layer feed forward neural networks where there is one input layer, one output layer and one hidden layer. Thus, the design of the network architecture becomes the determination of the number of input nodes, the number of output nodes and the number of hidden nodes. One example of this kind of neural network with its associated input pattern is shown in figure 4.5.

4.4.1 Number of Input Nodes

This approach directly uses the raw pixel data as inputs in order to avoid the hand-crafting of specific features for each domain. As shown in figure 4.5, the values of all the pixels in an input pattern are fed into the network. The number of input nodes should be equal to the number of the pixels in the input pattern. This is in turn decided by the size of the square input field. Thus, the number of input nodes can be calculated by equation 4.2:

$$no\_of\_input\_nodes = n^2 \tag{4.2}$$

where *no_of_input_nodes* denotes the number of input nodes and n is the input field size. Typical numbers of input nodes for the easy, the coin and the retina pictures are *196* ($= 14^2$), *576* ($= 24^2$) and *256* ($= 16^2$) respectively.

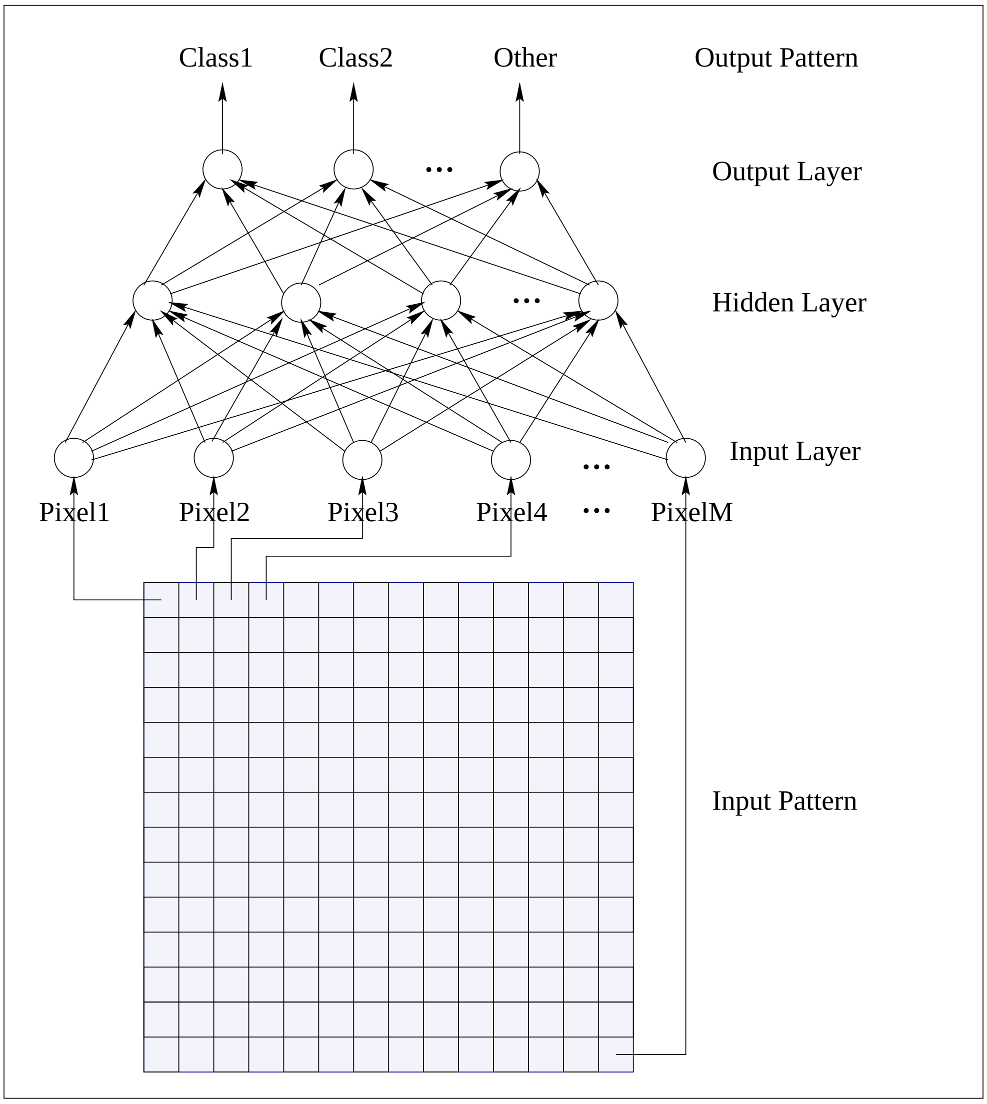


Figure 4.5: A sample neural network architecture with ($14 \times 14 = 196$) pixel input.

4.4.2 Number of Output Nodes

We define the number of output nodes ($no\_of\_output\_nodes$) in the network as the number of object classes contained in the large pictures plus one background class called *other*, as shown in equation 4.3.

$$no\_of\_output\_nodes = number\ of\ classes + 1 \tag{4.3}$$

In other words, the background is considered as one independent class here.

4.4.3 Number of Hidden Nodes

A number of methods have been used for determining the number of hidden nodes in feed forward networks, for example, trial-and-error, evolutionary algorithms and simulated annealing. Since the main focus here is on investigating the strengths and limitations of the basic approach with the back propagation algorithm rather than testing a constructive or stochastic search algorithm, we use the trial-and-error method for determining the number of hidden nodes.

In this method, the number of hidden nodes needs to be empirically determined during network training and testing. It is difficult to give a formula to determine the number of the hidden nodes, however we used the following guideline:

$$\boxed{\text{Use as few as possible}}$$

There are two main reasons for this. One concerns computation time. The more hidden nodes in the network, the longer the time taken in network training and object detection. The other concerns the possibility of overfitting (section 2.3.3, page 29), that is, using too many hidden nodes results in high accuracy on the training set but a high error rate on the test set.

Through empirical experiment under this guideline, we have found that 3–10 hidden nodes are suitable for most domains and that the process is relatively robust with respect to the number of hidden nodes. The details for different detection problems can be seen in section 4.8.1 (page 101).

4.5 Object Classification

4.5.1 Neural Network Training

The main ideas of network training were presented in section 2.3.2 (page 26). As indicated in that section a number of decisions need to be made when training neural networks: selection or development of the training algorithm, selection of the termination criterion for network training and the selection or determination of the relevant parameters. This section describes the first two and the last one will be presented in section 4.8.1 (page 101).

Selection of Training Algorithm

In this approach, we use the modified backward error propagation algorithm with *online learning* and *fan-in*, but with no momentum, to train the networks. The details of backward error propagation algorithm and its variations can be seen in section 2.3.2 (pages 27–28).

Selection of Termination Criteria

In this approach, we mainly used the *proportion control* strategy to terminate the network training process. The *user control* strategy was also used in case the trainer thinks there is no need to continue, however, it is used only for the determination of the network architecture and the parameters. The details of the termination strategies for network training can be seen in section 2.3.3 (page 29).

When the training is terminated, the learned network weights and biases are saved for the use in network testing or subsequent resumption of training.

4.5.2 Network Testing and Classification Criterion

The trained network is then applied to the classification test set. If the test performance is reasonable, then the trained network is ready to be used for object detection. Otherwise, the network architecture and/or the learning parameters need to be changed and the network re-trained, either from the beginning or from a previously saved, partially trained network.

The network classification is considered correct if the largest activation value produced by the feed forward procedure is for the output node which corresponds to the desired class. Otherwise, the classification is incorrect. For example, if the actual activation vector of an input pattern is (0.32, 0.84, 0.45, 0.17) for a four class detection problem and the target output vector is (0.0, 1.0, 0.0, 0.0), then this input pattern is correctly classified by the network.

4.6 Object Detection

In this stage, the trained network is used to detect the classes and locations of the objects of interest in large pictures. Classification and localisation are performed by the procedures — *network sweeping* and *finding object centres.* It is important to mention that the large pictures used here are those in the detection test set, which were not used in any form for network training.

4.6.1 Network Sweeping

After network training is successfully done, the trained neural network is used as a template matcher, and is applied, in a moving window fashion, over the large pictures to detect the objects of interest. The template is swept across and down these large pictures, pixel by pixel in every possible location. We call this procedure *network sweeping.*

After the sweeping is finished, an *object sweeping map* for each object class of interest detected will be produced. An object sweeping map corresponds to a PGM image. Sample object sweeping maps for class *class1*, class *class2* and class *class3* together with the original picture for the easy detection problems are shown in figure 4.6. During the sweeping process, if there is no match between a square in a detecting picture (an original picture in the detection test set) and the template, then the neural network output is 0, which corresponds to black in the sweeping maps; partial match corresponds to grey on the centre of the object, and best match is close to white in the sweeping maps.

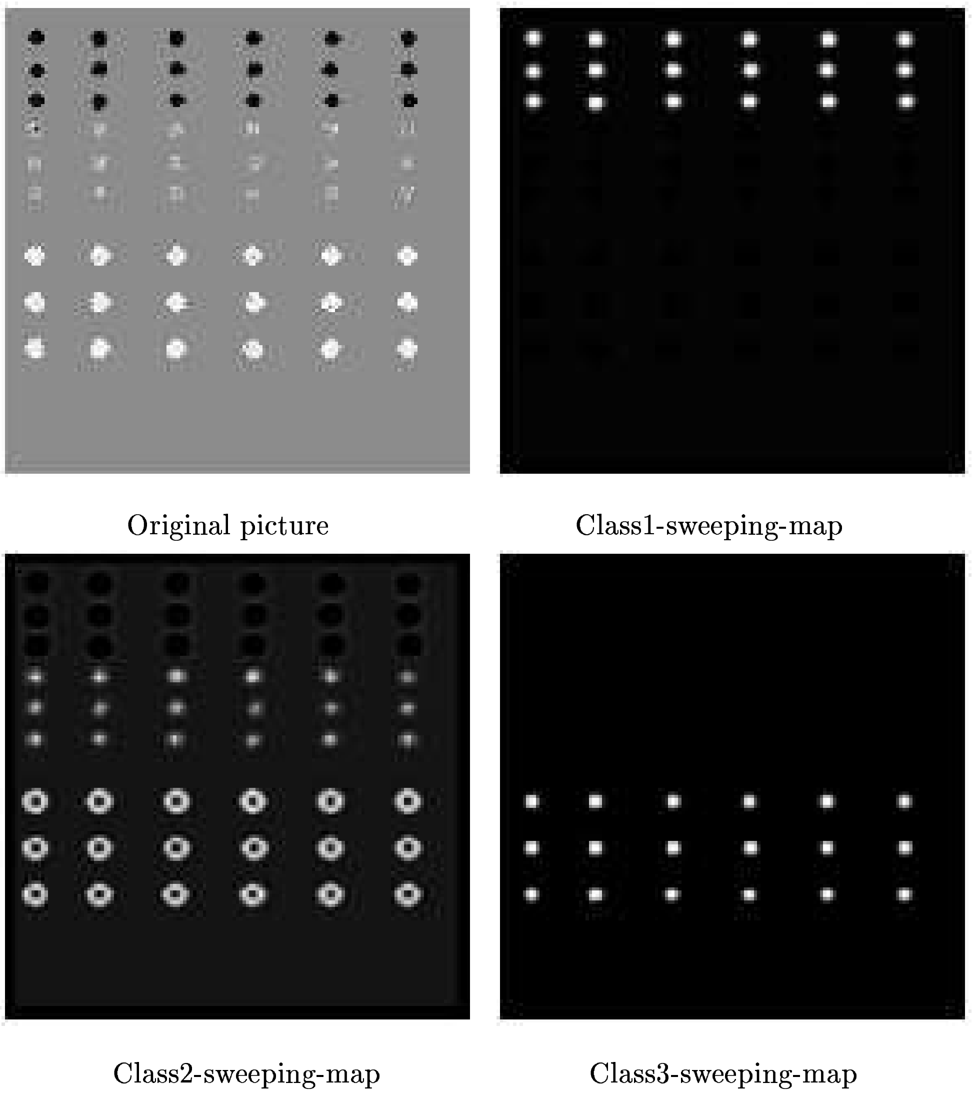


Figure 4.6: Sample object sweeping maps in object detection.

Under this design, each of the object sweeping maps is smaller than the original picture. The width and length of an object sweeping maps are equal to the width and length of the original picture minus the size of the input field. For example, if the original picture size is *700*×*700* and the input field size is *14*×*14* then the size of a object sweeping map is *(700-14)*×*(700-14)* or *686*×*686*. Accordingly, it is possible for this method to detect the objects located in the square with the upper-left corner (7, 7) and the lower-right corner (693, 693), but not possible to detect object located outside this square, that is, the edges of the original pictures. The possible positions here refer to the pixel positions within this square. For presentation convenience, we add some black edge pixels to make the object sweeping maps have the same size as the original picture, as shown in figure 4.6.

4.6.2 Finding Object Centres

In this thesis, the centres of the objects are used to represent the objects themselves. We develop a *centre-finding algorithm* in order to find the centres of all objects detected by the trained network during network sweeping. For each object class of interest in each picture in the detection test set, this algorithm is used to find the centres of the objects in the class based on the corresponding object sweeping map. The centre-finding algorithm is shown in figure 4.7.

For each object sweeping map:

Step 1 Set a *threshold* for the class (figure 4.8 on page 110).

Step 2 Set all of the values in the sweeping map to zero, if they are less than the threshold.

Step 3 Search for the largest value, save the corresponding position (x, y), and label this position as an object centre.

Step 4 Set all values in the square input field of the labelled centre (x, y) to zero.

Step 5 Repeat step 3 and step 4 until all values in the object sweeping map are zero.

Figure 4.7: Centre-finding algorithm

If two or more "object centres" for different classes at the same position are found, the decision will be made according to the network activations at this position. For example, if the centre-finding algorithm finds one object centre for *class2* and one for *class3* at position *(260, 340)* for the easy detection problem and the activations for the three classes of interest and the background at this position are *(0.27, 0.57, 0.93, 0.23)*, then the object for *class3* will be considered as the detected object at this position since the activation for this class is the biggest one (0.93).

4.7 Evaluation

In this approach, we design the two tasks involved in measuring the detection performance. They are *object matching* and *performance measurement.*

4.7.1 Object Matching

The main purpose of object matching here is to find the number of objects correctly reported by the detection system for each object class of interest. This is done by comparing all the object centres reported by the *centre-finding* algorithm with all the known true object centres in the detection test set. Here, we allow a tolerance of 4 pixels in the x and y directions. For example, if the coordinates of a known true object are *(21, 19)* and the coordinates of a detected object are *(22, 21)*, we still consider them as the same object. In this case the matching is successful, even though there is a gap of one and two pixels in the x and y directions respectively.

4.7.2 Performance Measurement

In this thesis, we use the *detection rate* and *false alarm rate* to measure the object detection performance. The details of the detection rate and false alarm rate can be found in section 2.1.3 (page 15). It is important to note that the detection performance here refers to the performance achieved by a trained network on the detection test set.

4.8 Experimental Results

4.8.1 Object Classification Results

This section presents a number of object classification results on the cutouts of the three detection problems, including network training and testing. Network training and test performance on the cutouts is measured by *mean squared error (MSE)* and the number of epochs used for network training. To give a clear view, the classification accuracy for network training and testing is also presented.

It is important to note that the results for object classification are not the the final detection results. There are two main reasons that we present these results here:

1. To establish that the classification networks can be successfully trained with some sets of parameters.

2. To determine what is the best classification on cutouts to use as a reference point for object detection.

In other words, these results are used for the selection of good trained networks and a corresponding set of parameters for object detection. For the easy and the coin pictures, 15 experiments are performed. For the retina pictures, 10 experiments are performed. For all cases, the mean (μ) and the corresponding standard deviation (σ) are presented.

Easy Pictures

In the easy pictures, the biggest object size is *12*×*12* pixels. The size of the input field was defined as *14*×*14*, which is the largest object size plus 2 pixels of background. Accordingly, the number of input nodes is 196 (*14*×*14*). There are three classes of interest in the easy picture database and thus the number of output nodes in the network is 4. A series of numbers of hidden nodes were tested here: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 50, 100. The experiment indicated that 3-10 hidden nodes gave similar results and the performance was best for 4 or 5 hidden nodes. Thus we used the network architectures 196-4-4 and 196-5-4 for further experiments. The network training and testing results with the two networks are presented in table 4.1 and table 4.2. The second column "initial weight format" is given in these tables to keep consistency with results in later chapters.

As can be seen from table 4.1, the first group of the experiments was carried out with a network architecture of 196-4-4. The learning rate (η) used here was 0.5, the momentum (α) was zero (without momentum). The training was terminated when all the training patterns in the classification training set were correctly classified ($percent$ = 100%). The trained network was then applied to the classification test set. After 15 runs of the experiments, the average number of epochs used for network training was 199.40 with a standard deviation of 18.09 (199.40±18.09). The mean squared error on the classification training set was $(5.09 \pm 0.30) \times 10^{-2}$

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	175	5.2	100	5.2	100
2	random	182	5.1	100	5.1	100
3	random	196	5.0	100	5.1	100
4	random	171	6.1	100	6.1	100
5	random	197	5.0	100	5.1	100
6	random	210	5.0	100	5.1	100
7	random	203	5.0	100	5.1	100
8	random	214	4.9	100	5.0	100
9	random	212	5.0	100	5.1	100
10	random	210	5.0	100	5.1	100
11	random	234	4.8	100	4.9	100
12	random	184	5.1	100	5.2	100
13	random	222	4.9	100	5.0	100
14	random	199	5.1	100	5.2	100
15	random	182	5.1	100	5.2	100
$\mu \pm \sigma$		199.40±18.09	5.09±0.30	100±0	5.17±0.27	100±0

Table 4.1: Results of network training and testing for object classification in the easy pictures using the basic approach. (Network architecture: 196-4-4; $\eta = 0.5$; $\alpha = 0$; Stop criterion: *Percent* = 100%; Training set size = 60; Test set size = 180; Repetitions = 15.)

and the mean squared error on the classification test set was $(5.17 \pm 0.27) \times 10^{-2}$. The trained network classified all the patterns in the classification test set into the correct desired classes, that is, test classification accuracy is 100%.

Table 4.2 shows the experimental results with the network architecture 196-5-4. In these experiments, training was terminated when 95% of the training patterns were correctly classified by the network. In this case, the number of epochs required for network training was 177.20±8.68, the mean squared error on the training set was $(5.73 \pm 0.10) \times 10^{-2}$ and the mean squared error on the test set was $(5.75 \pm 0.10) \times 10^{-2}$. The classification rate for the cutouts in the classification test set is 93.58% on average.

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	169	5.6	95.0	5.7	91.7
2	random	178	5.8	95.0	5.8	92.2
3	random	187	5.7	95.0	5.7	92.8
4	random	183	5.7	95.0	5.7	93.3
5	random	182	5.8	95.0	5.8	93.3
6	random	179	5.8	95.0	5.8	94.4
7	random	174	5.6	95.0	5.7	92.8
8	random	164	6.0	95.0	6.0	93.3
9	random	172	5.7	95.0	5.7	94.4
10	random	179	5.7	95.0	5.7	95.0
11	random	182	5.9	95.0	5.9	95.0
12	random	171	5.7	95.0	5.7	93.3
13	random	173	5.7	95.0	5.7	95.0
14	random	167	5.7	95.0	5.7	94.4
15	random	198	5.6	95.0	5.6	92.8
$\mu \pm \sigma$		177.20± 8.68	5.73±0.10	95.0±0	5.75±0.10	93.58±1.06

Table 4.2: Results of network training and testing for object classification in the easy pictures using the basic approach. (Network architecture: 196-5-4; η = 0.5; α = 0; Stop criterion: *Percent* = 95%; Training set size = 60; Test set size = 180; Repetitions = 15.)

Coin Pictures

As described in chapter 3, there are four classes of interest in the coin picture database. The number of output nodes was accordingly defined as 5. The largest size of these objects is around *20*×*20*, thus we used 576 — an input field of *24*×*24*, as the number of input nodes. We empirically determined the number of hidden nodes as 3 and 5. Networks were trained based on these two architectures.

The results with a network architecture of 576-3-5 are shown in table 4.3. The network was trained with a learning rate of 0.5 without momentum. The training process was terminated when all the patterns in the classification training set were correctly classified.

As can be seen from table 4.3, training the network (576-3-5) required 234.6±65.94 epochs;

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	197	3.1		100 3.1	100
2	random	214	2.3	100	2.3	100
3	random	175	2.9	100	2.9	100
4	random	258	2.5	100	2.5	100
5	random	190	2.5	100	2.5	100
6	random	238	2.5	100	2.5	100
7	random	197	2.3	100	2.3	100
8	random	242	1.9	100	1.9	100
9	random	259	2.2	100	2.2	100
10	random	219	3.5	100	3.5	100
11	random	343	1.9	100	1.9	100
12	random	231	2.2	100	2.2	100
13	random	175	3.1	100	3.1	100
14	random	175	2.7	100	2.7	100
15	random	208	3.0	100	2.9	100
$\mu \pm \sigma$		234.6±65.94	2.60±0.47	100±0	2.60±0.46	100±0

Table 4.3: Results of network training and testing for object classification in the coin pictures using the basic approach. (Network architecture: 576-3-5; η = 0.5; α = 0; Stop criterion: *Percent* = 100%; Training set size = 100; Test set size = 100; Repetitions = 15.)

the mean squared error performance on the classification training set was (2.60±0.47)$\times 10^{-2}$, and was (2.60±0.46)$\times 10^{-2}$ on the classification test set. The classification rate on the classification test set was also 100%.

As in the easy pictures, we performed another group of experiments on the coin pictures using a network of 576-5-5. To test the role that the learning parameters play during network training, we set the learning rate as 1.5 without momentum. The training process was terminated at the point when 95% of the training patterns were correctly classified. Under this condition, training took 87.13±5.08 epochs; the mean squared error on the classification training and test sets were (3.33±0.39)$\times 10^{-2}$ and (3.17±0.36)$\times 10^{-2}$ respectively. The details of the network training and

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	90	4.1	95	3.8	94.0
2	random	81	3.1	95	3.0	95.0
3	random	80	3.3	95	3.1	97.0
4	random	89	3.4	95	3.2	93.0
5	random	98	2.9	95	2.8	95.0
6	random	83	2.6	95	2.5	94.0
7	random	94	2.8	95	2.7	95.0
8	random	89	3.6	95	3.6	96.0
9	random	81	3.5	95	3.2	93.0
10	random	88	3.5	95	3.3	96.0
11	random	83	3.6	95	3.4	94.0
12	random	89	3.1	95	2.8	93.0
13	random	89	3.8	95	3.6	95.0
14	random	84	3.4	95	3.2	93.0
15	random	89	3.3	95	3.3	95.0
$\mu \pm \sigma$		87.13±5.08	3.33±0.39	95.0±0	3.17±0.36	94.53±1.25

Table 4.4: Results of network training and testing for object classification in the coin pictures using the basic approach. (Network architecture: 576-5-5; $\eta = 1.5$; $\alpha = 0$; Stop criterion: *Percent* = 95%; Training set size = 100; Test set size = 100; Repetitions = 15.)

testing are presented in table 4.4. The classification accuracy on the classification test set was 94.53% on average.

Retina Pictures

In the retina database, there are two classes of interest (class *haem* and class *micro*) and two additional classes (*vein* and *edge*). The number of output nodes was defined as 5. We used *16*×*16* as the input field size, which corresponded to 256 input nodes in the network. In the experiments, we used 4 and 5 as the number of hidden nodes. Note that 10 runs of experiments on the two networks were performed in the retina picture database due to long computation times.

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	191	9.5	65.0	9.1	67.2
2	random	186	9.6	65.0	9.2	65.5
3	random	196	9.7	65.0	8.7	68.9
4	random	179	9.6	65.0	9.2	65.5
5	random	213	9.1	65.0	9.2	60.7
6	random	195	9.7	65.0	9.3	62.2
7	random	172	9.7	65.0	9.0	68.9
8	random	193	9.4	65.0	9.3	63.9
9	random	219	9.1	65.0	9.1	62.2
10	random	264	9.3	65.0	9.1	65.5
$\mu \pm \sigma$		200.8±26.67	9.44±0.22	65.0±0	9.12±0.18	65.05±2.82

Table 4.5: Results of network training and testing for object classification in the retina pictures using the basic approach. (Network architecture: 256-4-5; η = 1.5; α = 0; Stop criterion: *Percent* = 65%; Training set size = 100; Test set size = 61; Repetitions = 10.)

Unlike the easy and the coin picture databases, it was much more difficult to train the networks for the retina picture data. A network could not be trained to correctly classify all the training patterns of the cutouts for a wide range of parameter values.

The first experiment was done with a 256-4-5 network. A learning rate of 1.5 was used with no momentum; the training process was terminated when 65% of all patterns in the classification training set were correctly classified by the network. The results are presented in table 4.5.

The second experiment is presented in table 4.6. In this case, a 256-5-5 network was used, the learning rate was 0.5 and network training was continued until 75% of the training patterns were correctly classified by the network. Note that the test mean square error was larger than the training square error, which indicated that overtraining occurred. This was confirmed by the classification rate on the classification test set, 71.83% on average, which is less than that on the training cutouts (75.4%).

In this section, a series of network training and testing results for object classification were presented. These results show that:

Expt. No.	Initial Weight Format	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	random	431	6.9	75.4	7.7	72.1
2	random	345	7.2	75.4	9.7	68.9
3	random	645	6.0	75.4	7.7	73.8
4	random	412	6.5	75.4	8.0	68.9
5	random	408	7.3	75.4	9.3	73.8
6	random	373	7.1	75.4	9.3	68.9
7	random	516	5.9	75.4	7.6	73.8
8	random	415	7.2	75.4	8.1	75.4
9	random	445	6.7	75.4	7.7	73.8
10	random	768	6.2	75.4	8.3	68.9
$\mu \pm \sigma$		475.8±132.76	6.70±0.52	75.4±0	8.34±0.79	71.83±2.63

Table 4.6: Results of network training and testing for object classification in the retina pictures using the basic approach. (Network architecture: 256-5-5; $\eta = 0.5$; $\alpha = 0$; Stop criterion: *percent* = 75%; Training set size = 100; Test set size = 61; Repetitions = 10.)

- The degree of difficulty for network training varies with the three detection problems. For the easy and coin pictures, the training patterns were all correctly classified (100% classification accuracy) by the network. For the retina pictures, however, it was not possible to reach such a point.

- Robust: Networks are trained successfully with a wide range of hidden nodes and learning parameters.

- Different starting points such as the random initial weights also produce different results for the same network architecture with the same training patterns using the same learning parameters.

It is important to note that network training and testing is only an intermediate step. The trained networks will be then used as templates to detect the objects of interest in the entire images in the detection test set. The detection results are presented in the next section.

4.8.2 Object Detection Results

This section describes the detection performance of the basic method on the three detection problems using the sweeping method described in section 4.6 (page 98). Each of the 15 trained networks shown in table 4.1 was applied to the detection test set of the easy pictures. Each of the 15 trained networks shown in table 4.3 was applied to the entire images in the detection test set of the coin pictures. Each of the 10 trained networks shown in table 4.5 was applied to the detection test set of the retina pictures. The averages are presented here.

Easy Pictures

As mentioned in the centre-finding algorithm (page 100), various thresholds result in different objects detected, and accordingly different object detection results. The higher the threshold, the fewer the objects that can be detected by the trained network, which results in a lower detection rate but also a lower false alarm rate. Similarly, the lower the threshold is selected, the higher the detection rate and the higher the false alarm rate which are produced. As mentioned on page 16, there is a trade-off between the detection rate and the corresponding false alarm rate.

During the object detection process, thresholds were selected as shown in figure 4.8.

To show the relationship between the detection rate/false alarm rate and the threshold selection, the detection results of one trained network for *class2* in the easy pictures are presented in table 4.7.

Easy Pictures	Object Classes							
(Basic Method)	Class2							
Threshold	0.54	0.56	0.57	0.595	0.625	0.650	0.673	0.700
Detection Rate(%)	100	96.67	93.33	90.00	86.67	83.33	80.00	76.67
False Alarm Rate (%)	90.5	35.1	13.8	11.9	11.2	8.1	5.6	3.7
Threshold	0.725	0.747	0.755	0.800	0.835	0.865	...	0.940
Detection Rate(%)	73.33	70.00	66.67	63.33	60.00	56.67	...	0
False Alarm Rate (%)	3.1	2.5	2.3	1.0	0.25	0	0	0

Table 4.7: Object detection results for *class2* in the easy pictures under different thresholds using one of the 15 trained networks.

For a given object sweeping map for an object class of interest:

1. Initialise a threshold to 0.7.
2. Apply the centre-finding algorithm and object matching process and calculate the detection rate and the corresponding false alarm rate.
3. If the detection rate is less than 100%, decrease the threshold to a new value and repeat step 2 and obtain a new detection rate and a false alarm rate. Repeat this procedure and obtain all the possible detection rates and the best corresponding false alarm rate until the detection rate reaches 100% or the new threshold is less than or equal to 0.20.
4. From the detection point obtained at the threshold of 0.7, if the false alarm rate is greater than zero, increase the threshold and repeat step 2 in order to obtain a new point (a detection rate with its corresponding false alarm rate). Repeat this procedure until either the false alarm rate is zero, or detection rate is zero, or the threshold is greater than or equal to 0.999.
5. To obtain all possible detection rates and their corresponding false alarm rate, the thresholds applied in step 3 and 4 may need to be adjusted. In step 3, the current threshold should be increased if a possible point (detection rate) is lost between the detection rate under the current threshold and that with the previous threshold. The current threshold should be decreased in step 4 if a similar situation happens.

Figure 4.8: Choice of thresholds

Easy Pictures	Object Classes		
(Basic Method)	Class1	Class2	Class3
Detection Rate(%)	100	100	100
False Alarm Rate (%)	0	91.2	0

Table 4.8: Object detection results for the easy pictures using the basic approach. (Network architecture: 196-4-4; Repetitions = 15.)

The overall results for the three classes in the easy pictures are presented in table 4.8. As can be seen from this table, this approach always achieved a 100% detection rate and a corresponding zero false alarm rate for *class1* (black circles) and *class3* (white circles). All of the objects of interest in these two classes were correctly detected without producing any false alarms. However, this is not the case for *class2* (grey squares). Even if a neural network achieved a 100% detection rate under a certain threshold, 91.2% false alarm rate was also produced. The detection results for *class2* are given in table 4.9.

Easy Pictures	Object Classes							
(Basic Method)	Class2							
Detection Rate(%)	100	96.67	93.33	90.00	86.67	83.33	80.00	76.67
False Alarm Rate (%)	91.2	32.3	14.3	11.7	10.5	7.4	5.90	3.10
Detection Rate(%)	73.33	70.00	66.67	63.33	60.00	56.67	...	0
False Alarm Rate (%)	2.80	2.60	2.20	1.20	0.30	0	0	0

Table 4.9: Object detection results for *class2* in the easy pictures using the basic approach. (Network architecture: 196-4-4; Repetitions = 15.)

Coin Pictures

The detection results for the coin pictures are described in table 4.10. In each run, it was always possible to find a threshold for the network output for class *head005* and *tail005* which resulted in detecting all of the objects of these classes with no false alarms. However, detecting class *head020* and *tail020* was a relatively difficult problem with this method. Although all the objects in these two classes were correctly detected (100% detection rate), the neural networks produced

Coin Pictures	Object Classes			
	head005	tail005	head020	tail020
Detection Rate (%)	100	100	100	100
False Alarm Rate (%)	0	0	182	37.5

Table 4.10: Object detection results for the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

some false alarms. The average false alarm rates for the two classes at a 100% detection rate were 182% and 37.5% respectively.

Tables 4.11 and 4.12 show the details of the average detection performance for class *head020* and *tail020* in the coin pictures.

Coin Pictures	Object Classes								
(Basic Method)	head020								
Detection Rate(%)	100	93.75	87.50	81.25	75.0	68.75	62.5	56.25	50.0
False Alarm Rate(%)	182	159	140	114	92.7	67.5	51.4	35.3	21.4
Detection Rate(%)	43.75	37.5	31.25	25.0	18.75	12.5	6.25	0	
False Alarm Rate (%)	10.3	8.50	7.40	6.40	1.0	0.4	0	0	

Table 4.11: Object detection results for class *head020* in the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

Coin Pictures	Object Classes								
(Basic Method)	tail020								
Detection Rate(%)	100	93.75	87.50	81.25	75.0	68.75	62.5	56.25	50.0
False Alarm Rate (%)	37.5	25.0	18.8	12.5	0	0	0	0	0
Detection Rate(%)	43.75	37.5	31.25	25.0	18.75	12.5	6.25	0	
False Alarm Rate(%)	0	0	0	0	0	0	0	0	

Table 4.12: Object detection results for class *tail020* in the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

Retina Pictures

Tables 4.13 and 4.14 describe the average detection performance of the two classes *haem* and *micro* in the very difficult retina pictures. Compared with the performance of the easy and coin pictures, these results are disappointing. The best detection rate for class *haem* was 73.91% with a corresponding false alarm rate of 2859%. Even at a detection rate of 50%, the false alarm rate was still quite high (about 1800%). All the objects of class *micro* were correctly detected (detection rate could reach 100%) with a false alarm rate of 10104%.

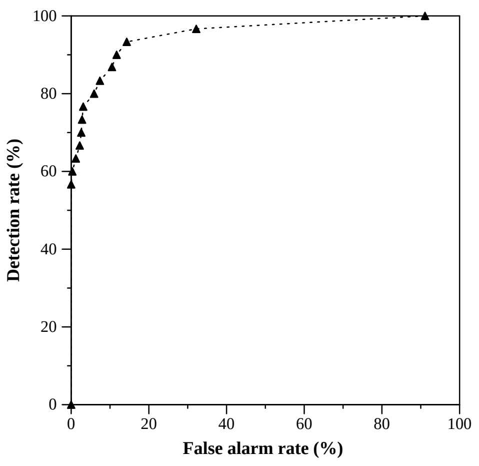
Retina Pictures (Basic Method)	Object Classes								
	haem								
Detection Rate(%)	73.91	69.57	65.22	60.87	56.52	52.17	47.83	43.48	39.13
False Alarm Rate(%)	2859	2758	2698	2529	1984	1872	1568	1151	1095
Detection Rate(%)	34.78	30.43	26.09	21.74	17.39	13.04	8.70	4.35	0
False Alarm Rate(%)	956.1	873.0	682.6	534.8	426.1	373.9	187.0	65.2	0

Table 4.13: Object detection results for class *haem* in the retina pictures using the basic approach. (Network architecture: 256-4-5; Repetitions = 10.)

Retina Pictures (Basic Method)	Object Classes										
	micro										
Detection Rate(%)	100	90.00	80.00	70.00	60.00	50.00	40.00	30.00	20.00	10.00	0
False Alarm Rate(%)	10104	9800	9332	8680	7440	7196	6309	5327	4980	3694	0

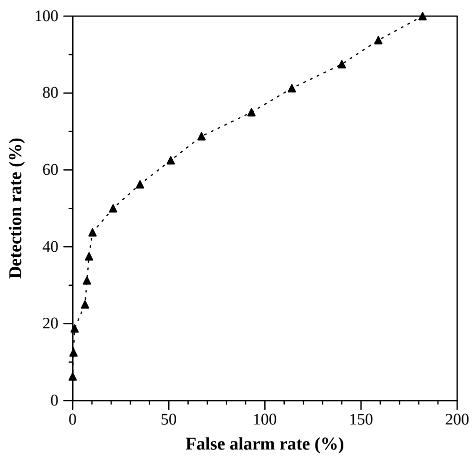
Table 4.14: Object detection results for class *micro* in the retina pictures using the basic approach. (Network architecture: 256-4-5; Repetitions = 10.)

To give an intuitive view, the extended ROC curves (section 2.1.3 on page 13) for class *class2* in the easy pictures, class *head020* in the coin pictures and classes *haem* and *micro* in the retina pictures are presented in figure 4.9 (a), (b), (c) and (d) respectively.



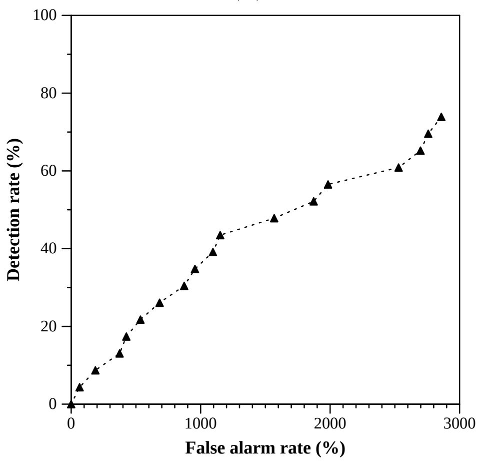
ROC curve for "class2" in the easy pictures

(a)



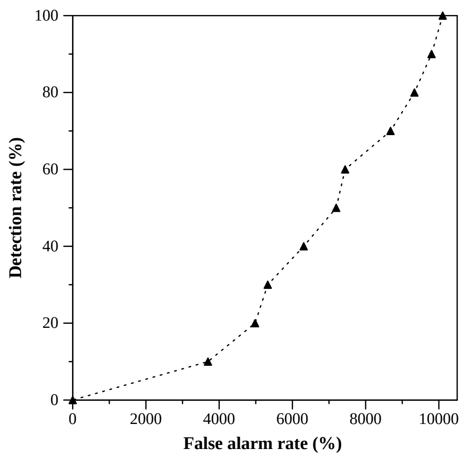
ROC curve for "head020" in the coin pictures

(b)



ROC curve for "haem" in the retina pictures

(c)



ROC curve for "micro" in the retina pictures

(d)

Figure 4.9: Some typical results (extended ROC curves) for object detection in the three databases using the basic approach.

Analysis of Results

As can be seen from the detection results obtained here, it was always possible to detect all objects of interest in the easy and the coin pictures. This reflects the fact that the objects in the two databases are simple or regular and the background is uniform or relatively uniform. While detecting the easy pictures only resulted in a few false alarms, detecting objects in the coin pictures resulted in a relatively higher false alarm rate. This is mainly because the detection problems in the coin pictures are more difficult than in the easy pictures.

Due to the difficulty of the detection problems, the results for the retina pictures are not good. For class *micro*, while all objects were correctly detected, a very high number of false alarms were produced. This is mainly because these objects are irregular and complex and the background is highly cluttered. For class *haem*, it was not possible to detect all objects of interest (the best detection rate was 73.91%). This is mainly due to the size variance of these objects (from 7×7 to 14×14 pixels).

4.9 Discussion

4.9.1 Characteristics

This approach has the following characteristics:

- The raw image pixel data are used as inputs to neural networks, and accordingly it is a domain independent approach.
- Network training uses the backward error propagation algorithm on the cutouts of the objects rather than the entire images.
- The network weights are initialised to small random values at the beginning of the network training.
- The trained networks are applied as a template, in a moving window fashion, over the large pictures in the detection test set to detect (locate and classify) the objects of interest.
- There are multiple object classes of interest in the large pictures rather than an *object* versus *non-object* detection problem.
- Traditional specific feature extraction is avoided.

4.9.2 Linearisation

In the basic approach, we used a three layer feed forward neural network trained by the backward error propagation algorithm. The image pixel values are directly used as inputs to the neural network. However, images have spatial structures and linearising an $n \times n$ image pixels into a vector of inputs might destroy important information in the image.

There have been a number of discussions on this issue in the literature. Some researchers argue that for general object/target recognition problems, input linearisation is not a big problem and a fully connected network can still learn object parts (image local features) by the backward error propagation learning algorithm using a non-linear transfer function (like a non-linear filter) [29, 159].[1] Rogers [155] states that a weighted linear combination of inputs can be passed through a non-linearity to model actions and these kinds of networks can be used for finding regions of interest, extracting features, selecting features, and object classification (page 1153). However, this might not be the best technique for a given difficult problem.

Two general approaches have been reported for further improvement of the linearisation. The first is to constrain the network training algorithm to reflect the image spatial correlation. An example is the weight smoothing method [91], where a smoothing operation is added to the delta updating rule in the back error propagation algorithm to adapt to the image spatial structure. The second is to constrain the network input and the internal architecture to address the linearisation issue. Shared weight neural networks [115, 42] and artificial convolution neural networks [121] are examples of this kind. Both approaches involve constraining weights, either smoothing weights to reflect the adjacent pixel correlations or sharing the same weight from a squared receptive field (an $n \times n$ subimage) for the same feature.

To address this linearisation issue, we first used the backward error propagation algorithm with the non-linear *sigmoid* transfer function and found that the basic approach performed well for the relatively easy detection problems. We will investigate a new weight constraint method to reflect the spatial correlations among neighbouring image pixels (chapter 5) and a non-linear genetic training algorithm for network refinement (chapter 6) for the multiple class object detection problems.

[1] Our results on weight analysis for the basic approach (see figure 5.4 on page 139) also support this idea.

4.9.3 Next Step

It can be seen from the experiments that for detecting grey squares (*class*) in easy pictures and detecting objects in class *head020* and *tail020* in coin pictures the results are not ideal. The performance in detecting class *micro* and class *haem* is very poor. The remainder of the thesis examines ways of improving detection performance. Three lines of investigation are pursued:

1. The results have shown that the detection performance is related to the network training starting point (initial weights). Is there any way to initialise the weights that can give better performance? A centred weight initialisation method is presented in chapter 5.

2. In this approach, the network is trained by the backward error propagation algorithm. Can improved detection performance be obtained if a genetic algorithm replaces the backward error propagation algorithm for network training? This is investigated in chapter 6.

3. In this approach, the network was trained based on the cutouts of the objects of interest and the trained network was directly applied to the entire images in the detection test set. Is there any way to refine the trained network based on the entire images? A two phase approach which uses a genetic algorithm to refine the trained networks is discussed in chapter 6.

4.9.4 Summary

This chapter presented a pixel based approach for detecting multiple class objects in large pictures using multilayer feed forward neural networks. The backward error propagation algorithm was used for the network training on the sub-images which had been cut out from the large pictures in the detection training set. The trained network was then applied, in a moving window fashion, over the entire images in the detection test set to detect the objects of interest. The objects were represented by the coordinates of their centres. A centre-finding algorithm was developed for object detection after the network sweeping. The detection rate and false alarm rate were used as the evaluation criteria. Object detection performance was measured on the large pictures in the detection test set. The experimental results showed that:

- This approach performed very well for detecting a number of simple and regular objects against a relatively uniform background, such as detecting black circles and white circles in the easy pictures.

- It performed poorly on the detection of class *haem* and class *micro* objects in the retina pictures, which suggests that it may not be well suited to detecting complex and irregular objects against a highly cluttered background.
- As expected, the performance degrades when the approach is applied to detection problems of increasing difficulty.

Chapter 5

Centred Weight Initialisation in Neural Networks for Object Detection

On the basis of the basic approach discussed in chapter 4, this chapter introduces a new method of initialising the weights, *centred weight initialisation*, before network training. After describing the details of the method and implementation algorithm, we present a series of results on the three detection problems of increasing difficulty and compare them with those achieved by the basic approach with random initial weights. We then interpret the results through the analysis of the weights in the trained networks. Finally, we summarise the strengths and limitations of the method.

5.1 Introduction

This section first gives the rationale for the centred weight initialisation method, presents the terminology related to the method, and then describes the goals of this chapter.

5.1.1 Rationale of the Method

In the basic approach, the network weights are initialised to small random floating values for network training. As can be seen from chapter 4, based on the basic approach, the results for detecting black and white circles in the easy pictures, and detecting heads and tails of 5 cent coins in the coin pictures are quite good, but the results for detecting other objects are not good at all. For detecting objects in classes *haem* and *micro* in the retina pictures, for example, a high false alarm rate is produced. In this chapter, we introduce a centred weight initialisation method, which is expected to be able to improve both network training accuracy and training time for object classification and object detection. The main considerations are:

- Image data has its own characteristics. In general approaches to pattern classification with neural networks, the inputs of the networks are usually *independent*. Accordingly, the weights are initialised randomly by the standard random weight initialisation method and training usually results in reasonable training speed and reasonable accuracy. However, in image data, adjacent pixels are clearly not independent. Pixels that are adjacent in an object are very likely to have similar intensities or colours. Therefore, in the pixel based neural approach for object detection problems, this fact should be considered, and accordingly the corresponding weights to these pixels which are used as inputs to neural networks could be initialised in a different way for network training.
- In the basic approach, the centres of the small objects are used to represent the locations of the objects. Thus the weights corresponding to the central pixels of the small objects should perhaps be considered more important than those corresponding to the pixels far from the centres of the objects.

For presentation convenience, we define the following terminology:

Definition 5.1 *Random Weight Initialisation and Random Initial Weights*
Random weight initialisation, or *the random weight initialisation method*, refers to the way of randomly initialising network weights for network training. The weights initialised using this method are called random initial weights.

Definition 5.2 *Centred Weight Initialisation and Centred Initial Weights*
Centred weight initialisation, or *the centred weight initialisation method*, refers to a specialised way of initialising network weights in which the central pixels in a square are expected to be more important than those on the perimeter and the network weights are initialised accordingly. The details of this method can be found in section 5.2. The weights initialised using this method are called centred initial weights.

5.1.2 Chapter Goals

In this chapter we investigate whether the centred initial weights:

- will decrease the number of epochs to convergence for network training;
- will improve network accuracy on test data;
- will result in better object detection, that is, higher detection rate and lower false alarm rate; and
- will be effective for problems of increasing difficulty.

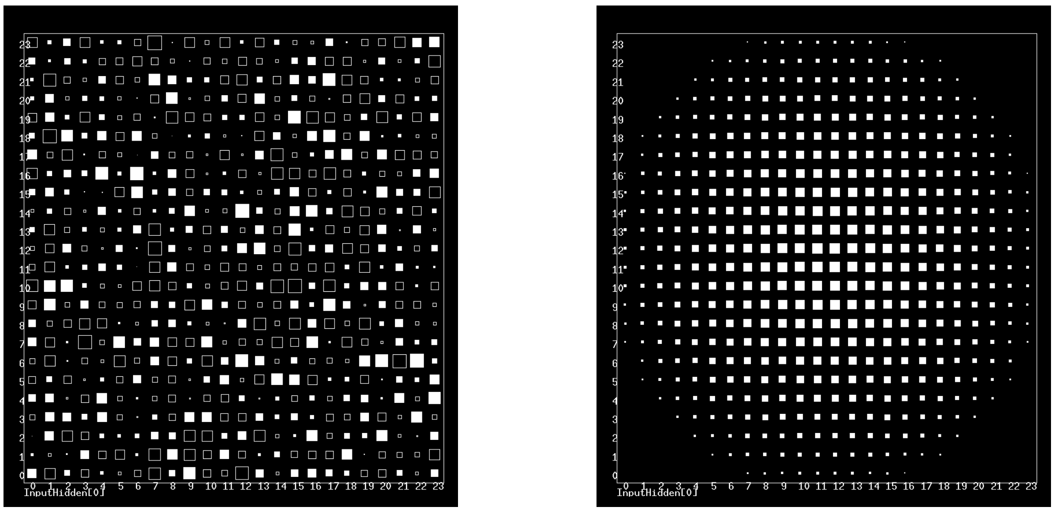
5.2 Centred Weight Initialisation Method

In the basic approach (chapter 4), three layer feed forward neural networks are used and all the weights in a network are initialised with small random floating point numbers between -0.5 and +0.5. For convenience, we call the weights which connect the input nodes and hidden nodes *input-hidden weights*, and call those linking the hidden nodes and output nodes *hidden-output weights*. The input-hidden weights for a particular hidden node can also be expressed as a matrix, or an *input-hidden weight matrix*. The number of such matrices is equal to the number of hidden nodes in the network. A sample of such a matrix with random initial weights is shown in figure 5.1 (a). In this figure the filled squares represent positive weights and the outline squares represent negative weights, while the size of the square is proportional to the magnitude of the weight. To facilitate visualisation the weights are shown as a parallel array to the input field. Thus $weight(i, j)$ in figure 5.1 (a) corresponds to $pixel(i, j)$ in the input field.

5.2.1 Centred Weight Initialisation Method

Figure 5.1 (b) shows a matrix with the centred initial weights. The idea of the centred weight initialisation method is:

- For the input-hidden weights,
  - The weight corresponding to the central pixel of the small object is initialised as the biggest value and the weights corresponding to the pixels close to the corners and edges of the small object take a very small value.
  - The farther the pixels are away from the centre of the object, the smaller the corresponding initial weights.
  - The difference between the two weights corresponding to adjacent pixels in the input field is restricted to a small constant.



(a) Random Initialisation (b) Centred Initialisation

Figure 5.1: Sample initial *input-hidden* weights. (a) Random initial weights; (b) Centred initial weights.

- The hidden-output weights and all the biases are randomly initialised in the same way as those in the basic approach in chapter 4 since there is no direct corresponding relationship between these weights and the network input patterns.
- A small random value produced by a normal random generator [4] is added to every weight in all input-hidden weight matrices. This will make the initial weights in the same positions across the different *input-hidden weight matrices* slightly different.

The weights initialised by this method meet the needs of the characteristics of image data and are consistent with the back propagation algorithm.

5.2.2 Centred Weight Initialisation Algorithm

The algorithm for centred weight initialisation is:

1. Load the network architecture and obtain the number of input nodes ($no\_of\_input\_nodes$), the number of output nodes ($no\_of\_output\_nodes$), and the number of hidden nodes ($no\_of\_hidden\_nodes$).

2. Obtain the training square size (*size_square*) which is equal to the input field size. This can be obtained based on the number of input nodes in the network architecture.

$$size\_square = \sqrt{no\_of\_input\_nodes} \tag{5.1}$$

3. Using the parameter *max_weight* for the central weight obtain the gap between the two neighbouring weights (*weight_gap*) according to equation 5.2.

$$weight\_gap = 2 \times max\_weight \; / \; size\_square \tag{5.2}$$

To initialise all the weights in the first *input-hidden weight matrix*, do steps 4 to 6:

4. Calculate the magnitude of the four central weights corresponding to the four central pixels of the object according to equation 5.3. Because the width and the height (both equal to *size_square*) of a training square that we use are even numbers of pixels, there are four central pixels in the training square.

$$cen\_weight = max\_weight + \epsilon \tag{5.3}$$

ϵ is a small random number generated by a normal distribution [4] as in equation 5.4:

$$\epsilon = normalised\_gaussian(\mu, \sigma) \tag{5.4}$$

In order to make ϵ very small we set μ to zero, and σ to a very small number, such as $weight\_gap/30$.

5. For each weight (*weight(i, j)*) other than the central ones in the input-hidden weight matrix, calculate the distance of the corresponding pixel (*pixel(i, j)*) from the nearest central pixel in the object according to equation 5.5:

$$distance(i, j) = \sqrt{(i - x)^2 + (j - y)^2} \tag{5.5}$$

Here, $distance(i, j)$ stands for the distance between a pixel and the nearest central pixel in the training square, *(x, y)* denotes the coordinates of the nearest central pixel and *(i, j)* is the position of the pixel in the training square, $0 \leq i < size\_square$, $0 \leq j < size\_square$.

6. Calculate the magnitude of each weight in the *input-hidden weight matrix* using the distance from the corresponding pixel to the nearest centre according to equation 5.6:

$$weight(i,j) = max\_weight - distance(i,j) \times weight\_gap + \epsilon \tag{5.6}$$

   If *weight(i, j)* $<$ *0* then $weight(i,j) = 0$.

7. Repeat step 4 to step 6 for each of the other $(no\_of\_hidden\_nodes - 1)$ *input-hidden weight matrices*. As can be seen from the above equations, all the corresponding weights in the different weight matrices are slightly different because of the small random number ϵ.

8. For the initialisation of the hidden-output weights and all the biases, the standard random weight initialisation method is used — random numbers in range *-0.5* to *0.5*.

The algorithm ensures that weights are largest at the centre and decrease uniformly to the perimeter as shown in figure 5.1 (b). While the weights are not truly random they provide a satisfactory starting point for training the network with the back propagation algorithm.

5.3 Experimental Results

5.3.1 Object Classification Results

This section presents a series of experimental results of the *centred weight initialisation* method for network training and testing on the classification data set cut out from the detection training set in the three databases. This only involves network training and testing but not network sweeping. The number of training epochs and the size of the mean squared error on the test set are used as the evaluation criteria. A comparison of the numbers of training epochs and the test mean squared errors of the two weight initialisation methods are presented here. The training and testing classification accuracy is also presented to keep consistency with chapter 4.

Easy Pictures

The first group of network training and testing results for the easy pictures are shown in table 5.1. These results were obtained by the two initialisation methods under the same conditions as table 4.1 — the basic approach with random initial weights: the network architecture is 196-4-4, learning rate (η) is 0.5, there is no momentum (α), network training stop criterion is when all the training patterns are correctly classified, or *percent = 100%*. For each experiment, no matter which method is used, we repeat the network training and testing 15 times, and calculate the *mean(μ)* and *standard deviation (σ)* of the training epochs, the training mean square error, the test mean square error and the classification accuracy. Line 1 of the table shows that with random initial weights, network training is terminated at an average of 199.40 epochs with a standard deviation of 18.09; the average mean squared error on the classification training set is 5.09×10^{-2} with a standard deviation of 0.30×10^{-2}, and is $5.17\times10^{-2}\pm0.27\times10^{-2}$ on the test set. The classification accuracy is 100% on both classification training and test sets. The results of the centred weight initialisation method on the same data with different values of *max_weight* are given in the other lines of the table.

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs ($\mu\pm\sigma$)	Training MSE ($\mu\pm\sigma$) ($\times10^{-2}$)	Training Accuracy ($\mu\pm\sigma$) (%)	Test MSE ($\mu\pm\sigma$) ($\times10^{-2}$)	Test Accuracy ($\mu\pm\sigma$) (%)
1	random			199.40±18.09	5.09±0.30	100±0	5.17±0.27	100±0
2	centred	0.420	0.060	430.80±29.78	4.81±0.10	100±0	4.85±0.11	100±0
3	centred	0.350	0.050	263.00±24.56	5.01±0.07	100±0	5.03±0.05	100±0
4	centred	0.280	0.040	219.47±14.20	4.47±0.12	100±0	4.55±0.11	100±0
5	centred	0.210	0.030	143.80±23.07	4.63±0.17	100±0	4.70±0.15	100±0
6	centred	0.140	0.020	109.87±10.64	5.04±0.11	100±0	5.08±0.12	100±0
7	centred	0.070	0.010	128.87±9.42	5.29±0.10	100±0	5.35±0.07	100±0
8	centred	0.035	0.005	139.33±9.03	5.37±0.12	100±0	5.45±0.11	100±0
9	centred	0.014	0.002	153.40±11.76	5.11±0.06	100±0	5.19±0.04	100±0

Table 5.1: Comparison of the results of network training and testing for object classification in the easy pictures using random and centred initial weights. (Network architecture: 196-4-4; η = 0.5; α = 0; Stop criterion: *Percent* = 100%; Training set size = 60; Test set size = 180; Repetitions = 15.)

As can be seen from table 5.1, many choices of *max_weight/weight_gap* can lead to faster network training than random initial weights. The number of network training epochs under random initial weights is 199.40±18.09, where the test mean squared error is $5.17\times10^{-2}\pm0.27\times10^{-2}$. Using the centred weight initialisation method, for example, when *max_weight/weight_gap* is 0.140/0.020, the number of training epochs is 109.87±10.64. This is 44.90% (1 − 109.87 / 199.40 = 0.4490) faster than that for random initial weights. The standard deviation is 41.18% (1 − 10.64 / 18.09 = 0.4118) less for random initial weights indicating that the process is more stable. At the same time, the test mean squared error is $5.08\times10^{-2}\pm0.12\times10^{-2}$, which is less than the error ($5.17\times10^{-2}\pm0.27\times10^{-2}$) achieved using the standard random weight initialisation method. However, the number of the training epochs based on centred initial weights is not always less than that based on random initial weights. For instance, when the centred initial weight parameter (*max_weight/weight_gap*) is set to "too big" a value (e.g. 0.280/0.040, 0.35/0.05 or 0.42/0.06), the number of training epochs is more than that obtained using random initial weights. With regard to the test performance, it can be found that if the centred initial weight parameter (*max_weight/weight_gap*) is set to "too small" a value (e.g. 0.07/0.01, 0.035/0.005 or 0.014/0.002), the test mean squared error will become larger than the error obtained using the random weight initialisation method (the basic approach). It is evident that for $0.07 < max\_weight < 0.28$ the centred weight method is superior.

Table 5.2 shows the results of centred weight initialisation for the same network and learning parameters as table 4.2 — random weight initialisation in the basic approach.

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs ($\mu\pm\sigma$)	Training MSE ($\mu\pm\sigma$) ($\times10^{-2}$)	Training Accuracy ($\mu\pm\sigma$) (%)	Test MSE ($\mu\pm\sigma$) ($\times10^{-2}$)	Test Accuracy ($\mu\pm\sigma$) (%)
1	random			177.20± 8.68	5.73±0.11	95.0±0	5.75±0.10	93.58±1.06
2	centred	0.420	0.060	463.47±37.54	5.71±0.10	95.0±0	5.72±0.11	95.13±0.27
3	centred	0.350	0.050	285.20±20.60	5.63±0.18	95.0±0	5.65±0.18	95.10±0.35
4	centred	0.280	0.040	208.67±13.47	5.10±0.04	95.0±0	5.15±0.05	95.08±0.42
5	centred	0.210	0.030	131.80±15.47	5.23±0.07	95.0±0	5.27±0.07	95.04±0.39
6	centred	0.140	0.020	105.27± 5.60	5.55±0.10	95.0±0	5.55±0.10	95.02±0.30
7	centred	0.070	0.010	125.20± 6.68	5.78±0.17	95.0±0	5.83±0.19	94.78±0.24
8	centred	0.035	0.005	144.40± 6.57	5.83±0.08	95.0±0	5.87±0.07	94.25±0.37
9	centred	0.014	0.002	149.87± 4.85	5.71±0.03	95.0±0	5.79±0.03	93.45±0.57

Table 5.2: Comparison of the results of network training and testing for object classification in the easy pictures using random and centred initial weights. (Network architecture: 196-5-4; $\eta = 0.5$; $\alpha = 0$; Stop criterion: *Percent* = 95%; Training set size = 60; Test set size = 180; Repetitions = 15.)

Table 5.2 shows a similar pattern to table 5.1. According to table 5.2, the number of epochs used for network training based on random initial weights is 177.20±8.68, where the test mean squared error is $5.75\times10^{-2}\pm0.10\times10^{-2}$. The best number of epochs taken by network training under the centred weight initialisation method is 105.27±5.60 when the *max_weight/weight_gap* was 0.140/0.020. This number is 40.59% (1 − 105.27 / 177.20 = 0.4059) faster than for random initial weights. The standard deviation is 35.48% (1 − 5.60 / 8.68 = 0.3548) less than for random initial weights indicating that the process is more stable. Also the test mean squared error is $5.55\times10^{-2}\pm0.10\times10^{-2}$, which is smaller than that under random initial weights. The centred weight initialisation method under the initial parameter of 0.21/0.03 also results in both faster training speed and better test error than the standard random weight initialisation method. The test classification accuracy also supports this idea. However, if the parameter is "too big" (e.g. 0.42/0.06, 0.35/0.05, etc.) or "too small" (e.g. 0.07/0.01, 0.035/0.005, etc.), either the training speed or the test performance will deteriorate.

The experimental results indicate that for simple object classification in the easy pictures, the centred weight initialisation method under *a certain range* of the centred initial weight parameter can achieve better (faster and more stable) network training speed and better test error than the random weight initialisation method. On this database, the best improvement on the network training speed is 44.90%, with a better test performance at the same time.

Coin Pictures

In order to investigate the network training speed and the test performance of regular object classification in the coin pictures, the network architecture of 576-3-5 is used here. The results are shown in table 5.3. The network and learning parameters used in this table are the same as those in table 4.3.

As can be seen from table 5.3, with the standard random weight initialisation method, the means and standard deviations of the number of epochs needed for training and the test mean squared error are 234.6±65.94 and $2.60\times10^{-2}\pm0.46\times10^{-2}$. Using the centred initial weight method under different initial parameters, different results are obtained. For example, when the *max_weight/weight_gap* is set to 0.048/0.004, the means and standard deviations of the number of training epochs and test mean squared error are 170.9±14.12 and $2.36\pm0.22\times10^{-2}$. This is 27.15% (1 − 170.9/234.6 = 0.2715) faster than for random initial weights. The standard deviation is also less than that for random initial weights which indicates that the process is more stable. Also the test mean squared error is less than for the random weight initialisation method. The centred weight initialisation method using the *max_weight/weight_gap* of 0.048/0.0040 and 0.030/0.0025 show a similar result. However, not all parameter values lead to faster training and

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs $(\mu \pm \sigma)$	Training MSE $(\mu \pm \sigma)$ $(\times 10^{-2})$	Training Accuracy $(\mu \pm \sigma)$ (%)	Test MSE $(\mu \pm \sigma)$ $(\times 10^{-2})$	Test Accuracy $(\mu \pm \sigma)$ (%)
1	random			234.6±65.94	2.60±0.47	100±0	2.60±0.46	100±0
2	centred	0.280	0.023	1143.3±417.6	0.94±0.40	100±0	1.08±0.56	100±0
3	centred	0.210	0.0175	823.3±107.6	1.20±0.48	100±0	1.36±0.51	100±0
4	centred	0.120	0.0100	362.8±21.77	1.97±0.25	100±0	2.04±0.24	100±0
5	centred	0.090	0.0075	266.7±28.70	1.97±0.28	100±0	2.03±0.27	100±0
6	centred	0.060	0.0050	176.8±17.90	2.13±0.27	100±0	2.10±0.25	100±0
7	centred	0.048	0.0040	170.9±14.12	2.39±0.23	100±0	2.36±0.22	100±0
8	centred	0.030	0.0025	182.4±14.89	2.63±0.21	100±0	2.60±0.21	100±0
9	centred	0.024	0.0020	180.7±20.57	2.71±0.33	100±0	2.70±0.33	100±0
10	centred	0.012	0.0010	176.2±13.60	2.73±0.18	100±0	2.71±0.18	100±0
11	centred	0.006	0.0005	186.9±14.81	2.91±0.47	100±0	2.89±0.49	100±0

Table 5.3: Comparison of the results of network training and testing for object classification in the coin pictures using random and centred initial weights. (Network architecture: 576-3-5; η = 0.5; α = 0; Stop criterion: *Percent* = 100%; Training set size = 100; Test set size = 100; Repetitions = 15.)

better test performance. For instance, if *max_weight/weight_gap* is bigger than 0.090/0.0075 or smaller than 0.024/0.002, either the network training epochs or the test error are worse than those achieved using random initial weights. Both weight initialisation methods achieved 100% classification accuracy.

Table 5.4 shows the comparison of centred weight initialisation for the same network and learning parameters as table 4.4 — with random weight initialisation (the basic approach). According to table 5.4, if the centred initial parameter is set to a value bigger than 0.030/0.0025 and smaller than 0.120/0.010, the centred weight initialisation method results in a faster and more stable network training speed and a smaller test mean squared error than the standard random weight initialisation method. In contrast, if "too big" (e.g. equal to or bigger than 0.120/0.010) or "too small" (e.g. equal to or smaller than 0.030/0.0025) centred initial parameters are used, either the number of the network training epochs or the test error obtained becomes worse.

In summary, the results of classifying regular objects in the coin database show that if

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs $(\mu \pm \sigma)$	Training MSE $(\mu \pm \sigma)$ $(\times 10^{-2})$	Training Accuracy $(\mu \pm \sigma)$ (%)	Test MSE $(\mu \pm \sigma)$ $(\times 10^{-2})$	Test Accuracy $(\mu \pm \sigma)$ (%)
1	random			87.13±5.08	3.33±0.39	95.0±0	3.17±0.36	94.53±1.25
2	centred	0.280	0.0230	218.5±26.8	2.75±0.84	95.0±0	2.43±0.76	95.04±0.98
3	centred	0.120	0.0100	93.80±3.57	2.49±0.67	95.0±0	2.37±0.72	96.32±0.76
4	centred	0.060	0.0050	67.13±4.79	2.65±0.59	95.0±0	2.48±0.35	96.12±0.45
5	centred	0.048	0.0040	70.53±1.92	3.19±0.52	95.0±0	3.05±0.52	95.23±0.78
6	centred	0.030	0.0025	69.73±2.63	3.64±0.39	95.0±0	3.52±0.37	95.01±0.89
7	centred	0.024	0.0020	69.47±3.00	4.03±0.53	95.0±0	3.85±0.53	94.51±0.86
8	centred	0.012	0.0010	69.00±1.73	4.12±0.52	95.0±0	3.95±0.51	94.35±0.73
9	centred	0.006	0.0005	74.87±2.33	3.99±0.38	95.0±0	3.84±0.38	94.43±0.68

Table 5.4: Comparison of the results of network training and testing for object classification in the coin pictures using random and centred initial weights. (Network architecture: 576-5-5; $\eta = 1.5$; $\alpha = 0$; Stop criteria: *Percent* = 95%; Training set size = 100; Test set size = 100; Repetitions = 15.)

the centred initial parameter (*max_weight/weight_gap*) is set within *a certain range*, a faster and more stable network training and better test performance can be achieved using centred initial weights than using random initial weights. For this medium difficulty database, the best performance improvement in training speed was 27.15% using the centred weight initialisation method.

Retina Pictures

Two groups of experiments were carried out to investigate the improvement of the centred weight initialisation method on the very complex object classification in the retina pictures. The results are presented in table 5.5 and table 5.6.

Table 5.5 shows the results of centred weight initialisation for the same network and learning parameters as table 4.5 — the basic approach with random weight initialisation. The centred weight initialisation method under some parameters does improve the network training speed and the test performance. When the parameter *max_weight/weight_gap* is 0.280/0.035, the number of training epochs is 161.7±22.05. This is 19.5% ($1 - 161.7/200.8 = 0.195$) faster

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs ($\mu \pm \sigma$)	Training MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)	Training Accuracy ($\mu \pm \sigma$) (%)	Test MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)	Test Accuracy ($\mu \pm \sigma$) (%)
1	random			200.8±26.67	9.44±0.22	65.0±0	9.12±0.18	65.05±2.82
2	centred	0.960	0.120	391.5±229.7	9.61±0.35	65.0±0	9.96±0.21	64.38±2.76
3	centred	0.800	0.100	216.5±25.23	9.42±0.20	65.0±0	9.89±0.10	64.53±1.57
4	centred	0.480	0.060	170.7±18.66	9.67±0.15	65.0±0	9.30±0.17	64.87±2.34
5	centred	0.320	0.040	169.0±19.03	9.31±0.26	65.0±0	9.10±0.25	65.34±1.46
6	centred	0.280	0.035	161.7±22.05	9.04±0.41	65.0±0	8.87±0.13	66.43±1.15
7	centred	0.240	0.030	183.0±24.27	9.01±0.35	65.0±0	8.93±0.24	66.24±1.28
8	centred	0.200	0.025	172.0±25.79	9.27±0.37	65.0±0	9.00±0.13	66.12±1.36
9	centred	0.160	0.020	238.7±62.21	8.76±0.61	65.0±0	8.65±0.28	65.87±1.65
10	centred	0.080	0.010	256.1±19.98	8.41±0.06	65.0±0	8.95±0.28	65.24±1.89
11	centred	0.016	0.002	285.9±27.74	8.33±0.09	65.0±0	8.98±0.35	65.12±2.30

Table 5.5: Comparison of the results of network training and testing for object classification in the retina pictures using random and centred initial weights. (Network architecture: 256-4-5; η = 1.5; α = 0; Stop criteria: *Percent* = 65%; Training set size = 100; Test set size = 61; Repetitions = 10.)

than for random initial weights. The standard deviation is less than for random initial weights indicating the process is more stable. Also the test error is 8.87±0.13, which is less that the error (9.12±0.18) achieved by the random weight initialisation method. However, not all the choices of the parameter (*max_weight/weight_gap*) accelerate network training and lower the test error. If the parameter is “too big” (equal to or bigger than 0.480/0.060) or “too small” (equal to or smaller than 0.160/0.020), either the number of the training epochs or the test mean squared error will be higher.

Table 5.6 shows the results of centred weight initialisation for the same network and learning parameters as table 4.6 — the basic approach with random initial weights. These results show a similar pattern to table 5.5. Network training using the centred weight initialisation method under some values of the initial parameter is faster than that using the random weight initialisation method. For example, using the *max_weight/weight_gap* of 0.48/0.06 the number of epochs is only 354.1±28.36. This is 25.57% (1 − 354.1/475.8 = 0.2557) faster than for

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs ($\mu \pm \sigma$)	Training MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)	Training Accuracy ($\mu \pm \sigma$) (%)	Test MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)	Test Accuracy ($\mu \pm \sigma$) (%)
1	random			475.8±132.76	6.70±0.52	75.4±0	8.34±0.79	71.83±2.63
2	centred	1.600	0.200	1048.7±81.29	7.07±0.29	75.4±0	11.55±0.43	64.30±3.24
3	centred	1.200	0.150	818.8±122.68	7.30±0.54	75.4±0	9.92±0.85	70.24±3.12
4	centred	0.960	0.120	708.3± 52.92	6.85±0.39	75.4±0	8.16±0.53	72.46±2.42
5	centred	0.800	0.100	597.3± 24.24	6.77±0.26	75.4±0	7.98±0.16	73.18±2.75
6	centred	0.480	0.060	354.1± 28.36	7.55±0.42	75.4±0	8.05±0.14	73.14±2.12
7	centred	0.400	0.050	336.7± 38.00	7.38±0.20	75.4±0	8.20±0.11	73.06±2.25
8	centred	0.320	0.040	311.6± 17.02	7.55±0.13	75.4±0	8.33±0.14	71.98±2.48
9	centred	0.280	0.035	342.5± 32.83	7.29±0.18	75.4±0	8.35±0.23	71.76±2.77
10	centred	0.240	0.030	341.2± 22.87	7.27±0.23	75.4±0	8.55±0.40	69.39±2.59
11	centred	0.160	0.020	398.6± 40.30	6.98±0.48	75.4±0	8.52±0.58	69.43±2.98
12	centred	0.080	0.010	443.9± 43.04	6.74±0.56	75.4±0	8.41±0.51	70.32±2.64
13	centred	0.040	0.005	489.2± 72.06	6.51±0.50	75.4±0	8.20±0.45	71.65±2.12
14	centred	0.016	0.002	465.2± 52.70	6.74±0.47	75.4±0	8.33±0.38	71.34±2.73

Table 5.6: Comparison of the results of network training and testing for object classification in the retina pictures using random and centred initial weights. (Network architecture: 256-5-5; $\eta = 0.5$; $\alpha = 0$; Stop criterion: *Percent* = 75%; Training set size = 100; Test set size = 61; Repetitions = 10.)

random initial weights. The standard deviation (28.36) is much less than that (132.76) for random initial weights. The test mean squared error is 8.05±0.14, which is also less than that (8.34±0.79) for random initial weights. The results have a similar pattern if the parameter *max_weight/weight_gap* uses 0.40/0.05 and 0.32/0.04. However, if this parameter is set to "too big" a value (e.g 1.60/0.20, 1.20/0.15) or "too small" a value (e.g. 0.16/0.02, 0.08/0.01), either the training speed or the test performance will become worse.

The results indicate that for the very complex object classification, in pixel based neural systems, the network training speed and the performance on the test set can be improved by using the centred initial weights under *a certain range* of the centred initial parameter. For this very difficult database, the best improvement in training speed is 25.57%.

Summary of training and test performance

In all three databases there is *a range* of values for *max_weight* which can lead to improved network training speed and lower test mean squared error. These results are summarised in table 5.7. On the coins database, for example, the network (576-3-5) shows improved performance for $0.024 < max\_weight < 0.09$. There is an average decrease in training epochs of 27.15% and 19.23% in test mean squared error. There does not appear to be a relationship between problem difficulty and the amount of improvement.

Database	Network Arch.	Range of Centred Initial Parameter (max_wei/wei_gap)	Improvement of Training Speed (μ/σ)	Improvement of Test MSE (μ/σ)
Circles and Squares (Easy)	196-4-4	> 0.07/0.01 and < 0.28/0.04	44.90%/41.18%	9.09%/44.44%
	196-5-4	> 0.07/0.01 and < 0.28/0.04	40.59%/35.48%	8.35%/30.00%
Coins (Medium Difficulty)	576-3-5	>0.024/0.002 and <0.090/0.0075	27.15%/78.59%	19.23%/45.65%
	576-5-5	>0.030/0.0025 and <0.120/0.0100	22.95%/5.70%	21.77%/2.78%
Retina (Very Difficult)	256-4-5	> 0.16/0.02 and < 0.48/0.06	19.5%/17.32%	2.74%/27.78%
	256-5-5	> 0.28/0.035 and < 0.80/0.10	25.57%/78.64%	3.48%/82.28%

Table 5.7: Summary of the improvement in training time and test performance of the centred weight initialisation method over the random weight initialisation.

Unfortunately there does not appear to be a reliable way of choosing the best value for *max_weight*. However, as suggested earlier, the major problem in these kinds of object detection problems is a very high number of false alarms. If the centred weight method can lower this number significantly then a short search for a good *max_weight* is a small price to pay. The next section compares the detection performance of the two weight initialisation methods.

5.3.2 Object Detection Results

After network training on the cutouts, the trained networks are used as templates to sweep the large pictures in the detection test set to detect multiple class objects of interest. This has the same procedure as for the basic approach. This section describes the detection performance of the centred weight networks on the three detection problems described in chapter 3. Similarly to random weight initialisation in the basic approach, each of the 15 trained networks with the centred weight parameter *max_weight* of 0.14 shown in table 5.1 was applied to the entire images in the detection test set of the easy pictures. Each of the 15 trained centred weight networks with *max_weight* of 0.06 shown in table 5.3 was applied to the entire images in the detection test set of the coin pictures. Each of the 10 trained centred weight networks with *max_weight* of 0.28 shown in table 5.5 was applied to the entire images in the detection test set of the retina pictures. The average detection results are presented in this section.

Easy Pictures

When the trained network is applied to the object detection, the procedure is the same as that in the basic approach. After network sweeping, different thresholds applied to the centred-finding algorithm (section 4.6.2, page 100) will result in different detection results. The choice of thresholds in the object detection process can be seen in figure 4.8 (page 110). Table 5.8 shows the relationship between the detection rate and false alarm rate and the threshold selection for class *class2* in the easy pictures using the centred weight initialisation method.

Easy Pictures (Centred Weights)	Object Classes										
	Class2										
Threshold	0.65	0.67	0.68	0.700	0.730	0.770	0.800	0.850	0.860	...	0.910
Detection Rate(%)	100	96.67	93.33	90.00	86.67	83.33	80.00	76.67	73.33	...	0
False Alarm Rate (%)	45.60	1.60	1.05	1.03	0.90	0.35	0.30	0.24	0	...	0

Table 5.8: Object detection results for *class2* in the easy pictures using one of the 15 trained centred weight networks under different thresholds.

The overall results for detecting simple objects in the easy pictures using the centred weight initialisation method are presented in table 5.9. Similarly to the results based on the basic method with random initial weights (table 4.8 on page 111), detecting objects in classes *class1*

Easy Pictures	Object Classes		
(Centred Weights)	Class1	Class2	Class3
Detection Rate(%)	100	100	100
False Alarm Rate(%)	0	46.4	0

Table 5.9: Object detection results for the easy pictures using the centred initial weights. (Network architecture: 196-4-4; $Max\_weight = 0.14$; Repetitions = 15.)

(black circles) and *class3 (white circles)* is relatively straight forward. This can achieve a 100% detection rate without any false alarms (ideal performance). However, this is not the case for detecting class *class2 (grey squares)*, which results in a 46.4% false alarm rate at a detection rate of 100%. The average detection performance for class2 at different thresholds is detailed in table 5.10.

Easy Pictures	Object Classes							
(Centred Weights)	Class2							
Detection Rate(%)	100	96.67	93.33	90.00	86.67	83.33	80.00	76.67
False Alarm Rate(%)	46.4	1.30	1.00	1.00	1.00	0.30	0.30	0.30
Detection Rate(%)	73.33	70.00	66.67	63.33	60.00	56.67	...	0
False Alarm Rate(%	0	0	0	0	0	0	0	0

Table 5.10: Object detection results for *class2* in the easy pictures using centred initial weights. (Network architecture: 196-4-4; $Max\_weight = 0.14$; Repetitions = 15.)

Compared with those obtained using the basic method with random initial weights (table 4.9, page 111), the results for class *class2* here are much better. At a detection rate of 100%, the false alarm rate is 46.4%, which is much better than the 91.2% achieved by the random weight initialisation method. The best detection rate with no false alarms achieved by the centred weight initialisation method is 73.33%, which is much higher than the 56.67% obtained by the basic method under the same conditions. In fact, the centred weight initialisation method has a lower false alarm rate for each detection rate when compared with the random weight initialisation method, that is, the basic approach.

Coin Pictures	Object Classes			
(Centred Weights)	head005	tail005	head020	tail020
Detection Rate (%)	100	100	100	100
False Alarm Rate(%)	0	0	41.4	0

Table 5.11: Object detection results for the coin pictures using centred initial weights. (Network architecture: 576-3-5; *Max_weight* = 0.06; Repetitions = 15.)

Coin Pictures

Detecting objects of interest in the coin pictures gives a similar pattern to the easy pictures. The overall results are presented in table 5.11. Compared with the corresponding results achieved using the basic approach with random initial weights (table 4.10, page 112), these results are better. All the objects in the three classes, *head005*, *tail005* and *tail020* are correctly detected with no false alarms. Detecting objects in class *head020* results in a 41.4% false alarm rate at a detection rate of 100%. This is, however, much better than the corresponding result (182%) obtained using the random weight initialisation method. The relationship between detection rate and false alarm rate under different thresholds for class *head020* with the centred weight initialisation method are presented in table 5.12.

Coin Pictures	Object Classes								
(Centred Weights)	head020								
Detection Rate(%)	100	93.75	87.50	81.25	75.0	68.75	62.5	56.25	50.0
False Alarm Rate(%	41.4	36.1	27.8	23.4	16.6	7.10	4.40	1.10	0
Detection Rate(%)	43.75	37.5	31.25	25.0	18.75	12.5	6.25	0	
False Alarm Rate(%)	0	0	0	0	0	0	0	0	

Table 5.12: Object detection results for class *head020* in the coin pictures using centred initial weights. (Network architecture: 576-3-5; *max_weight* = 0.06; Repetitions = 15.)

Retina Pictures

The results for detecting objects for classes *haem* and *micro* in the retina pictures are presented in table 5.13 and table 5.14. Compared with those for detecting objects in the easy pictures and the coin pictures, these results are disappointing. However, they are much better than

Retina Pictures	Object Classes								
(Centred Weights)	haem								
Detection Rate(%)	73.91	69.57	65.22	60.87	56.52	52.17	47.83	43.48	39.13
False Alarm Rate(%)	1924	1714	1642	1121	1095	1053	1039	992.0	934.0
Detection Rate(%)	34.78	30.43	26.09	21.74	17.39	13.04	8.70	4.35	0
False Alarm Rate(%)	901.0	736.2	555.4	234.1	0	0	0	0	0

Table 5.13: Object detection results for class *haem* in the retina pictures using centred initial weights. (Network architecture: 256-4-5; Max_weight = 0.28; Repetitions = 10.)

Retina Pictures	Object Classes										
(Centred Weights)	micro										
Detection Rate(%)	100	90.00	80.00	70.00	60.00	50.00	40.00	30.00	20.00	10.00	0
False Alarm Rate (%)	2903	2581	2232	1947	1940	1438	560	460	420	311	0

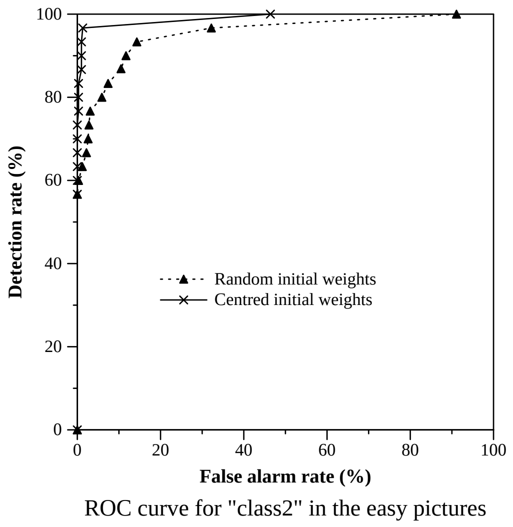
Table 5.14: Object detection results for class *micro* in the retina pictures using centred initial weights. (Network architecture: 256-4-5; Max_weight = 0.28; Repetitions = 10.)

the corresponding results (tables 4.13 and 4.14, page 113) obtained by the basic approach with random initial weights.

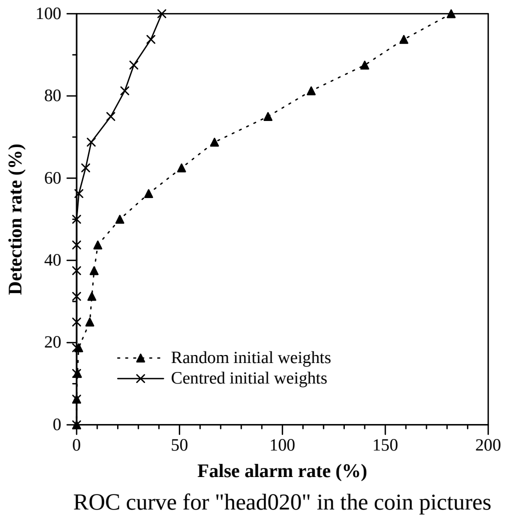
Summary of Object Detection Results

The detection results are summarised in figure 5.2 where the false alarm rate is plotted against the detection rate for the difficult classes. In all cases the centred weight results are superior – the false alarm rate at all levels of detection rate is always lower, in some cases very much so. This suggests that better object detection performance can be achieved on databases of any degree of difficulty using the centred initial weights. These results also suggest that the performance for object detection will deteriorate as the degree of difficulty of the detection problems increases.

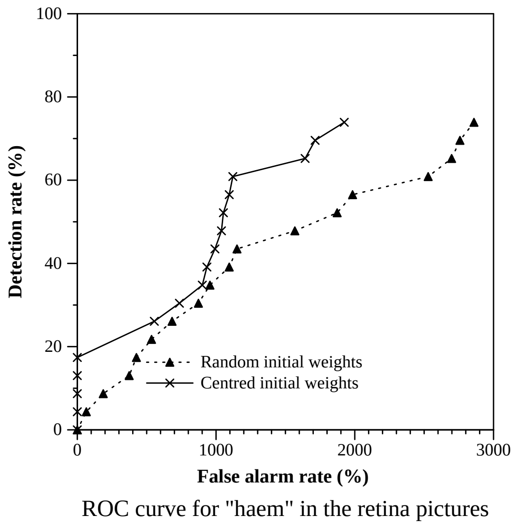
For the easy pictures and coin pictures, network thresholds can be chosen which give good detection and false alarm rates (figure 5.2 (a) and 5.2 (b)). This is not the case for the more difficult retina pictures (figure 5.2 (c) and 5.2 (d)).



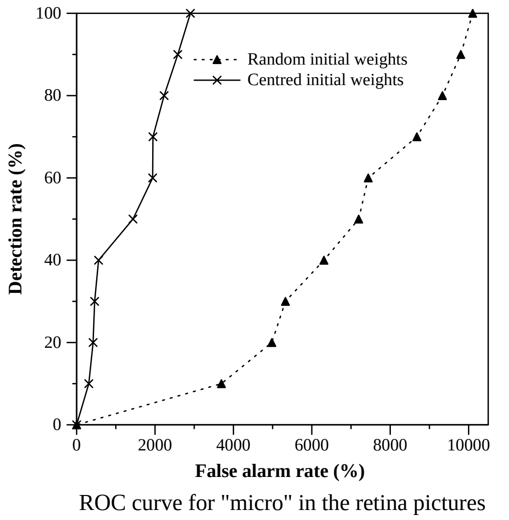
(a)



(b)



(c)



(d)

Figure 5.2: Comparison of results (in ROC curve) for object detection in the three databases between the centred weight initialisation and the random weight initialisation.

5.4 Analysis of Weights

To analyse why centred weight initialisation for object detection is superior to random weight initialisation, this section interprets the network internal behaviour through visual analysis of the weights in the trained networks. For presentation convenience, we use the two trained networks for regular object detection in the coin pictures. Most other networks contained similar patterns.

Figure 5.3 shows the network architecture used in both random and centred weight initialisation methods for object detection in the coin pictures. The weight groups between the input nodes ($24 \times 24 = 576$ in the input field) and hidden nodes and between the hidden nodes and the output nodes are also presented. The weight matrices (a), (b), (c) and (d) shown in figures 5.4 and 5.5 correspond to weight groups (a), (b), (c) and (d) in the network architecture in this figure.

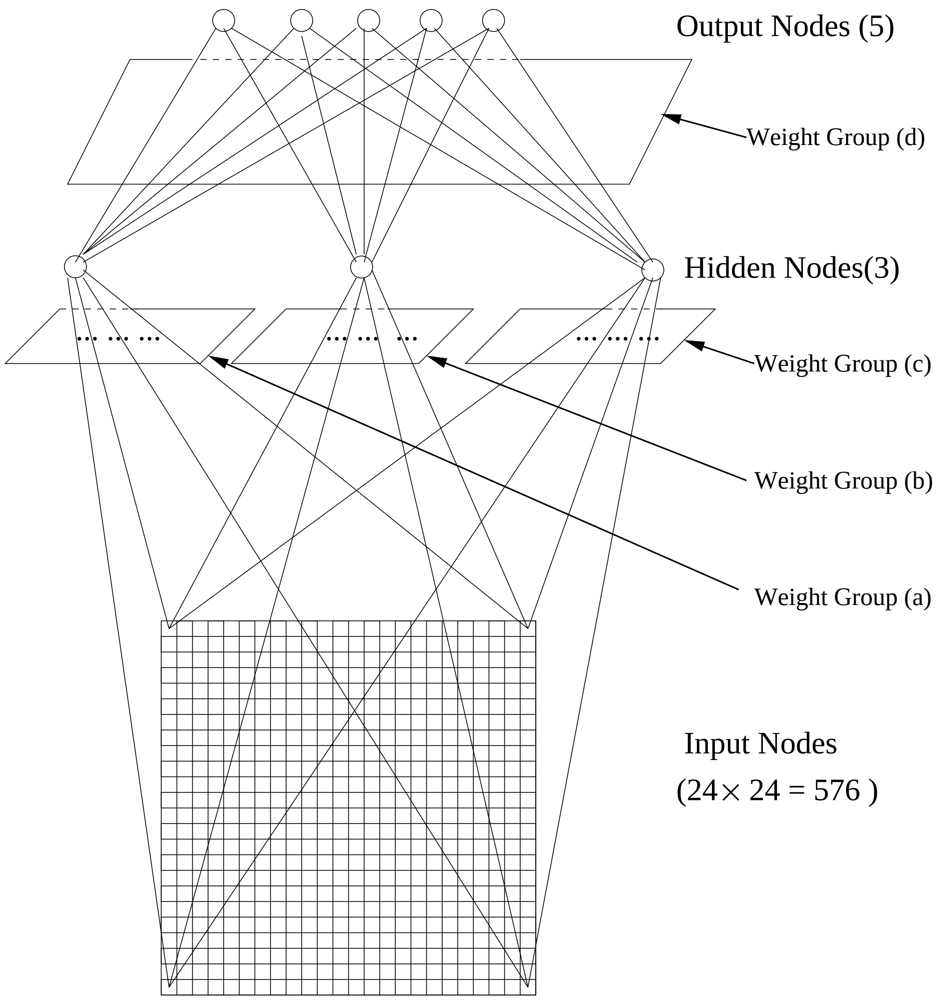
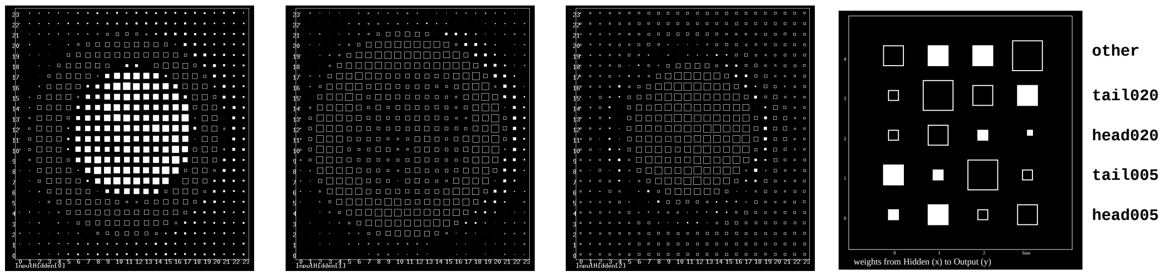


Figure 5.3: Network architecture with four weight groups for object detection in the coin pictures.

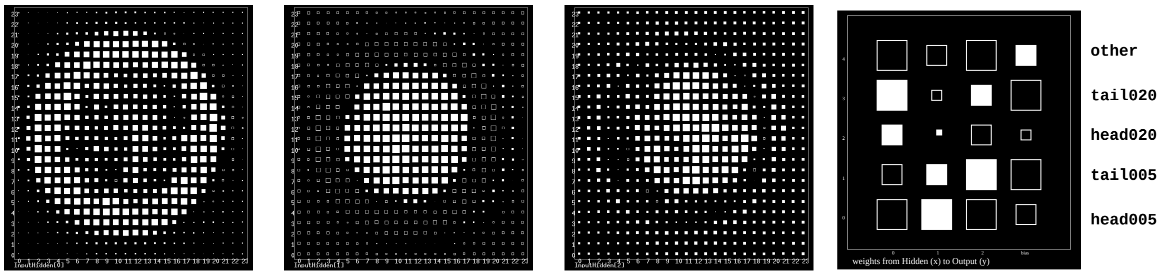
Figure 5.4 shows the weights from a trained 576-3-5 network which has been successfully used in the coin pictures. The network with these weights was trained with the random weight initialisation method. Part (a) shows the weights from the input nodes to the first hidden node, part (b) from inputs to second hidden node and part (c) from the inputs to the third hidden node. The weights are shown in a *24*×*24* square to facilitate visualisation. Figure 5.4 (d) shows the weights from the hidden layer to the five output units and the biases of these output nodes. The five rows in this matrix, in bottom to top order, correspond to the classes *head005, tail005, head020, tail020* and *other.* The four columns, in left to right order, correspond to weights from the first hidden node (associated with weight matrix (a)), the second hidden node (associated with weight matrix (b)) and the third hidden node (associated with weight matrix (c)) to the five output nodes (classes) and the biases of the five output nodes.



(a) (b) (c) (d)

Figure 5.4: Weights in a trained network for object detection in the coin pictures based on random initial weights.

Inspection of the first column of figure 5.4 (d) reveals that weight matrix (a) has a positive influence on *head005* and a very strong positive influence on *tail*005. The same matrix has a negative effect on the other classes. Inspection of the second column reveals weight matrix (b) has a positive effect on the 5 cent coins (classes *head005* and *tail005*) and a negative effect on the 20 cent coins (classes *head020* and *tail020*). Also it has a very strong positive influence on class *other.* This indicates that matrix (b) might be able to discriminate between the 5 cent and the 20 cent coins but might not properly distinguish the 5 cent coins from the background. The third weight matrix in part (c) is relatively difficult to interpret. It does not have a clear feature for discriminating different classes, however it strongly supports the background and has a strong negative influence on the tail side of 5 cent coins. If we regard the nodes of the hidden layers as representing feature detectors learnt by the network, then figures 5.4 (a)-(c) are a visual representation of these features. Visually these features 'make sense' as there are regions corresponding to the 5 cent coins, the annulus remaining when a 5 cent coin is 'removed' from the centre of a 20 cent coin and to the backgrounds.



(a) (b) (c) (d)

Figure 5.5: Weights in a trained network for object detection in the coin pictures based on the centred initial weights.

Figure 5.5 shows a similar set of weight diagrams for a 576-3-5 network which has been initialised with the centred weight method. The same features are evident as in figure 5.4 but in this case they are much better defined. According to this figure, the first weight matrix (part (a)) has a strong positive effect on the 20 cent coins and a strong negative influence on the 5 cent coins; the second (part (b)) has a significant positive effect on the 5 cent coins and a slightly positive or a negative influence on the 20 cent coins; the third (part (c)) has a very strong positive effect on the tail side of both the 5 cent and the 20 cent coins and a very strong negative influence on the head side of both the 5 cent and the 20 cent coins at the same time. Accordingly, the first two weight matrices in figure 5.5 can distinguish the 20 cent coins from the 5 cent coins. The third one has a very strong ability to separate all the tail sides from the head sides of these coins. In addition, all the three weight matrices have a very strong negative effect on the background, which suggests that they are able to discriminate the objects of interest from the background. More interestingly, the biases also play a strong role in the discrimination between the objects and the background by giving a positive influence on the background, as shown in column four of figure 5.5 (d). It appears that the centred weights initialisation method has resulted in learning which is focused on features necessary to discriminate the classes.

In this section, the weight analysis reveals a network internal behaviour which suggests that the approach with centred initial weights can improve the detection performance achieved with the random initial weights.

5.5 Summary and Discussion

5.5.1 Next Step

The centred weight initialisation method described in this chapter does improve the network training and test performance on the classification of the cutouts, and the object detection performance on entire pictures in the detection test set of all the three databases. However, there are still false alarms for some classes (*class2* in the easy pictures, *head020* in the coin pictures, *haem* and *micro* in the retina pictures). The false alarm rates are still quite high in the retina pictures. The next chapter will investigate whether detection performance can be improved by using genetic algorithms to train and refine the networks.

5.5.2 Summary

The goal of the work described in this chapter was to investigate a new method of setting initial weights in pixel based neural networks for object detection problems. Our results show that, for the three detection problems investigated, centred weight initialisation is superior to standard random initialisation.

The methods are compared on three detection problems of increasing difficulty. On the easy (circles and squares), medium difficulty (coins) and very difficult (retinas) problems, it is possible to find centred initial weights which can result in fewer training epochs and lower test mean squared error. More importantly, the centred weight method can produce networks which are much better, in terms of detection rate and false alarm rate, at the task of detecting multiple class objects of interest in large pictures. The amount of improvement does not appear to be related to the degree of difficulty of the problem. Overall, detection performance on the easy and medium difficulty problems is very good, but the performance on the difficult retina problem still poor.

The central weight initialisation has the disadvantage of requiring an empirical search for a good initialisation value but this is more than offset by the increase in detection accuracy.

Visualisation of the weights in trained networks resulting from both initialisation methods reveals that trained networks from both approaches contained feature detectors which ‘made sense’ for the domain, but learning in networks with centred initial weights is more focused on features which discriminate between the classes.

Chapter 6

Genetic Algorithms for Network Training and Network Refinement in Object Detection

Chapter 5 extended the basic approach by the use of centred weight initialisation for network training using the backward error propagation algorithm. This chapter introduces a two phase approach, another extension to the basic approach, which uses genetic algorithms to train and refine the networks defined in chapter 4 for object classification and detection. The algorithms, methods, corresponding results and a comparison with the basic approach for network training and refinement are described in this chapter.

6.1 Introduction

This section gives an overview of the two phase approach to the use of pixel based neural networks for multiple class object detection problems, as shown in figure 6.1. This is an extension of the process shown in figure 4.2 in chapter 4 (page 90) which does not include phase two. The terminology used in this figure is defined in section 4.1.1 (page 88).

As shown in this figure, in the first phase, the network is trained on the cutouts in the classification data set for object classification. Rather than using the backward error propagation algorithm as in the basic approach in chapter 4, a genetic algorithm is used. In the second phase, the weights of the trained network are refined using a second genetic algorithm. The first genetic algorithm uses a fitness function which maximises classification accuracy on the cutouts in the

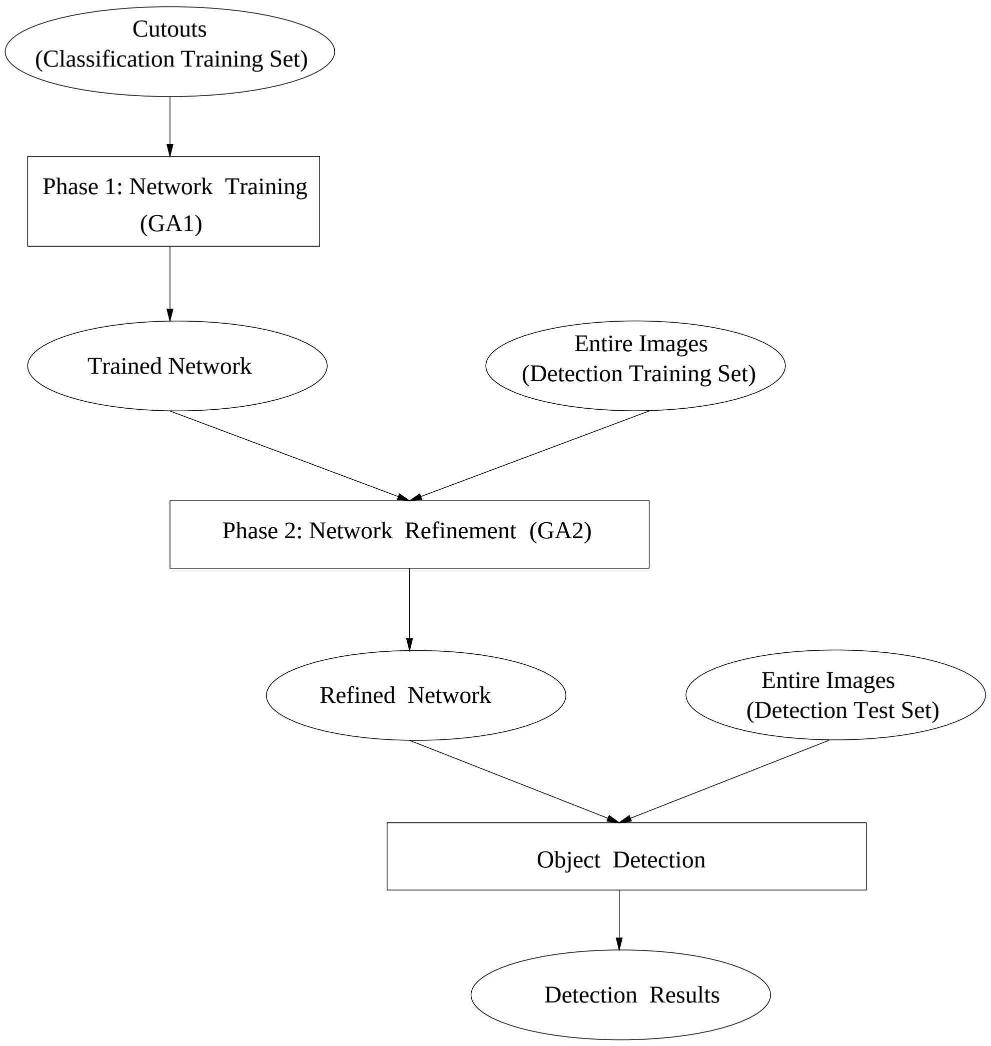


Figure 6.1: An overview of the two phase approach.

classification training set. The second genetic algorithm uses a fitness function which maximises detection performance on the entire images in the detection training set.

This two phase approach results in three new object detection methods:

1. Network training by the first genetic algorithm ($GA1$).
2. Network training by the first genetic algorithm ($GA1$) followed by network refinement by the second genetic algorithm ($GA2$).
3. Network training by the backward error propagation algorithm (BP) in the basic approach followed by network refinement by the second genetic algorithm ($GA2$).

6.1.1 Definitions

To avoid confusion, we define the following terminology:

Definition 6.1 *BP-train algorithm* and *BP-train method*
The *BP-train* algorithm refers to the backward error propagation algorithm used in the basic approach presented in chapter 4. The *BP-train* method refers to the basic approach, as described in chapter 4.

Definition 6.2 *GA-train algorithm* and *GA-train method*
The *GA-train* algorithm refers to the genetic algorithm applied in phase one of the two phase approach. The *GA-train* method refers to the method of directly applying the network trained by the *GA-train* algorithm to the entire images in the detection test set using the sweeping procedure described in chapter 4. The fitness function for this genetic algorithm is based on the mean squared error on the cutouts in the classification training set.

The *GA-train* method uses the same procedure as the basic approach, except that the *BP-train* algorithm is replaced by the *GA-train* algorithm.

Definition 6.3 *GA-refine algorithm*
The *GA-refine* algorithm refers to the genetic algorithm used in phase two of the approach. It begins with networks that have been initially trained with the *GA-train* algorithm or the *BP-train* algorithm and attempts to improve the detection performance by using a fitness function based on the detection rate and false alarm rate on the large pictures in the detection training set.

Definition 6.4 *GA-train+GA-refine method*
The *GA-train + GA-refine* method refers to the object detection method in which the *GA-train* algorithm is used for network training on the cutouts in the classification data set in phase one and the *GA-refine* algorithm is used for network refinement on the entire images in the detection training set in phase two.

Definition 6.5 *BP-train+GA-refine method*
The *BP-train + GA-refine* method is the same as the *GA-train + GA-refine* method except that the *BP-train* algorithm is used for network training on the cutouts in the classification data set. In other words, the network trained in the basic approach (chapter 4) is refined by the second genetic algorithm.

Definition 6.6 *Network Training* and *Network Refinement*
These two terms represent the two learning procedures in the two phases. The network learning procedure performed on the cutouts by the *GA-train* algorithm in phase one or by the *BP-train* algorithm in the basic approach (chapter 4) is called *network training.* Network training is directly associated with the object classification procedure in this thesis. The learning procedure performed on the entire images in phase two by the *GA-refine* algorithm is called *network refinement.* Thus, to some extent, network refinement is further or incremental learning of the trained networks.

Definition 6.7 *Trained Network, Evolved Network* and *Refined Network*
In this approach, we call the networks trained on the cutouts in the classification data set *the trained networks.* The networks obtained during the evolutionary process by the second genetic algorithm are called *the evolved networks.* The network with the best fitness produced at the end of the second phase is called *the refined network* and will be the resulting object detector.

6.1.2 Flow Diagram of the Approach

According to the definitions described above, a flow diagram of this approach is shown in figure 6.2.

6.1.3 Chapter Goals

The overall goal of this chapter is to investigate whether the use of genetic algorithms can result in improved performance over the basic approach. Specifically, we investigate the following research questions:

1. Will the first genetic algorithm, *GA-train*, lead to faster network training and better test performance on the cutouts for object classification than the backward error propagation algorithm, *BP-train*?

2. Will the first genetic algorithm, or the *GA-train* method, result in better detection performance than the basic approach, the *BP-train* method?

3. Will the second genetic algorithm, *GA-refine*, improve the object detection performance of the *GA-train* method or the *BP-train* method?

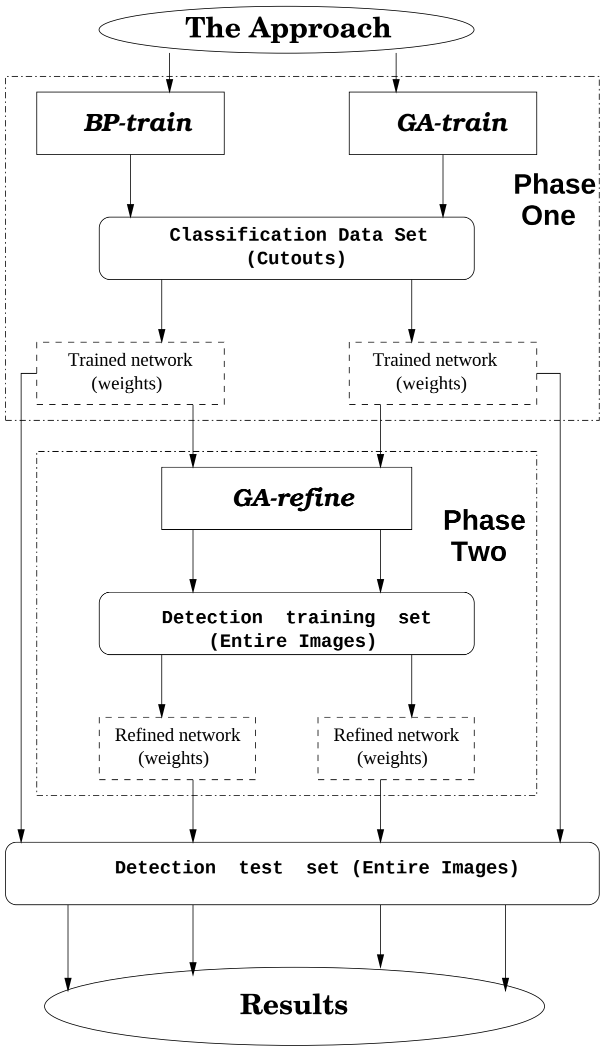

The Approach
BP-train
GA-train
Phase One
Classification Data Set (Cutouts)
Trained network (weights)
Trained network (weights)
GA-refine
Phase Two
Detection training set (Entire Images)
Refined network (weights)
Refined network (weights)
Detection test set (Entire Images)
Results

Figure 6.2: The flow diagram of the two-phase approach associated with four methods: *BP-train, GA-train, BP-train + GA-refine* and *GA-train + GA-refine*.

4. Which of the four methods, that is, *BP-train*, *GA-train*, *GA-train+GA-refine* and *BP-train+GA-refine*, gives the best detection performance?

In the remainder of this chapter, we first describe network training with the *GA-train* algorithm, followed by network refinement with the *GA-refine* algorithm. A series of experimental results are given afterwards. This chapter ends with a summary and a discussion.

6.2 Genetic Algorithm for Network Training – GA-train

The first phase of this approach requires a network to be trained on the cutouts of the multiple class objects in the classification data set. We use the *2DELTA-GANN* [34, 107, 108, 153] system as the first genetic algorithm. This has been used for training small neural networks such as the *xor* network and the *4-2-4 encoder* network with reasonable success. In this approach, we extend this algorithm to training the relatively large networks we are using for object classification and object detection. As mentioned earlier, we call this algorithm *GA-train* to distinguish it from the one in phase two. The goal is to investigate whether the *GA-train* algorithm can lead to better classification performance on the cutouts than the backward error propagation algorithm in the basic approach and whether the *GA-train* method can result in better performance for detecting small objects in large pictures.

For presentation convenience, unless otherwise specified, a bias in the network is treated as just another weight in this chapter.

6.2.1 Gene Structure

A gene in the *2Delta-GANN* algorithm corresponds to a single weight in the network and is a composite structure:

- There are three rule bits, called $x1, x2$ and $x3$.
- There are two floating point values called $delta1$ and $delta2$.

With this gene structure, the number of rule bits in each chromosome, which corresponds to a single network, will be three times as many as the number of weights in the network. In addition, the number of the floating point values which represent the *delta*s associated with each chromosome is twice as many as the number of the weights in the network. For example, if the network has 100 weights, then there will be 100 genes which contain 300 (100×3) rule bits and 200 (100×2) floating points values. Each chromosome in this case has 300 rule bits. In other words, the length of a chromosome is 300.

Note that the weights of the network are not a part of the chromosome, whose genes represent a method of changing the weights of the network in some way (see weight updating rule in the next subsection). The network is not defined in the genetic algorithm in any way – the network is pre-defined and is always a multilayer feed forward network in the experiments reported in this thesis.

6.2.2 Weight Updating Mechanism

The $x1, x2$ and $x3$ values specify a heuristic rule to apply to $delta2$. $Delta1$ will then be modified by $delta2$, and this will in turn be applied to the weight associated with the gene. The interpretation of the rule bits and the mechanism of updating weights is presented in figure 6.3. For example if $x1, x2$ and $x3$ are all 1 then $delta2$ is doubled, the new value of $delta2$ is then added to $delta1$ and the new weight is calculated by adding the new value of $delta1$ to the original weight.

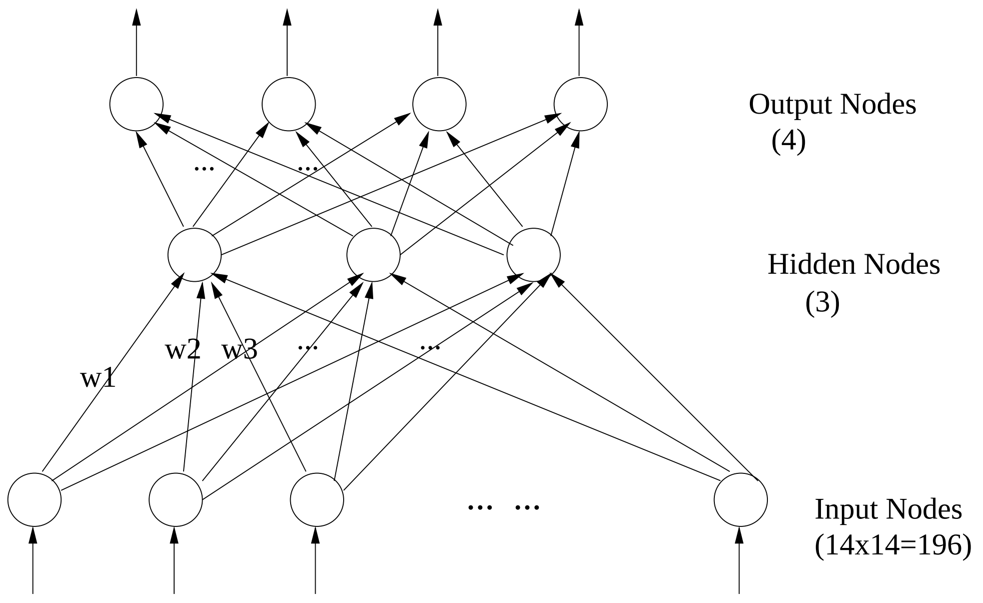
```
begin
    if x1 = 1 then
        if x2 = 1 and x3 = 1 then
            delta2 := delta2 * 2;
        else
            delta2 := delta2 / 2;
        endif
        delta1 := delta1 + delta2;
    endif
    weight := weight + delta1;
end
```

Figure 6.3: Weight updating mechanism in the genetic algorithm.

The changing mechanism allows for a weight to be changed by a $delta1$ value which in turn is modified by the $delta2$ value. The rate of change of the weight and $delta1$ (the "gradient") is changeable due to the $delta2$ value. This is used to provide a heuristic "second order" changing mechanism to the weight modification rule.

6.2.3 Sample Network with Chromosomes

Figure 6.4 shows the network architecture (196-3-4) for the detection problems in the easy pictures with sample chromosomes and genes for weight changes. In this figure, there are two chromosomes corresponding to two evolved networks. The weights are labelled as *w1, w2, ...*. As mentioned earlier, the network architecture and the actual weights are not encoded in the genetic algorithm but saved outside. To explain the calculation of the weights during the evolutionary



Weights	w1	w2	...	w600	...
Genes	x1 x2 x3 delta1 delta2	x1 x2 x3 delta1 delta2	...	x1 x2 x3 delta1 delta2	...
Network1	1 1 0 0.13 0.02	1 1 1 0.35 0.05	...	1 0 1 -0.48 0.21	...
Network2	0 0 1 0.42 - 0.12	1 0 0 -0.25 0.06	...	0 1 0 0.27 -0.10	...

Figure 6.4: Sample genes and chromosomes associated with a network for the simple object detection problem in the easy picture in the genetic algorithm.

process, we suppose the original values of *w1* and *w2* are 0.34 and -0.49 before the mechanism of updating weights is applied.

For *network1*, the three rule bits and the two *delta*s in the gene for $w1$ are: $x1 = 1$, $x2 = 1$, $x3 = 0$, $delta1 = 0.13$ and $delta2 = 0.02$. According to the weight changing mechanism, the value of the $delta2$ should be halved and the new value of $delta2$ becomes $0.02/2 = 0.01$. Then this value is added to $delta1$ and the new value of $delta1$ becomes $0.13 + 0.01 = 0.14$. The new value of $w1$ is accordingly updated to $0.34 + 0.14 = 0.48$. Similarly, since $x1 = x2 = x3 = 1$ in the gene of $w2$, after the weight changing mechanism is applied, the new values for $delta2$, $delta1$ and the actual weight for $w2$ are: $delta2 = delta2*2 = 0.05*2 = 0.10$, $delta1 = delta1+delta2 = 0.35 + 0.10 = 0.45$ and $w2 = w2 + delta1 = (-0.49) + 0.45 = -0.04$.

For *network2* in figure 6.4, for $w1$, since $x1 = 0$, after the weight updating mechanism is applied, $delta1$ and $delta2$ are unchanged and the new value of $w1$ becomes $w1 = w1 + delta1 =$

$0.34 + 0.42 = 0.76$. For $w2$, since $x1 = 1, x2 = x3 = 0$, the new values of $delta2$, $delta1$ and the actual weight $w2$ become $delta2 = delta2/2 = 0.06/2 = 0.03$, $delta1 = delta1 + delta2 = (-0.25) + 0.03 = -0.22$ and $w2 = w2 + delta1 = (-0.49) + (-0.22) = -0.71$.

6.2.4 Fitness Function

The fitness of a chromosome is obtained by realising a network from the weights and *delta* values encoded in the chromosome, passing all cases in the classification training data through this network and calculating the mean squared error (equation 2.12, page 31).

6.2.5 Genetic Operators

The commonly used biased roulette wheel mechanism (section 2.4.1, page 37) is used for parent selection.

The crossover operator is based on parameterised uniform crossover (section 2.4.1, figure 2.9, page 38). Crossover is applied only to the *rule* bits of the chromosome, not the *deltas*. If there is an exchange of any of the rule bits during crossover, the $delta1$ and $delta2$ values are exchanged between the parent chromosomes. Figure 6.5 shows an example of the crossover operator in the genetic algorithm.

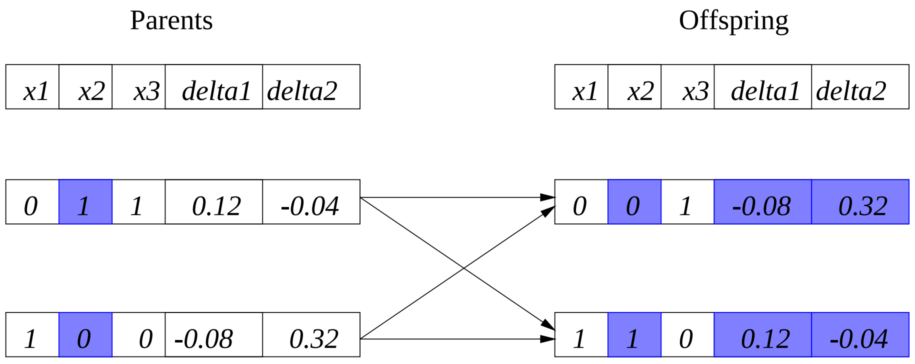


Figure 6.5: Crossover in the *GA-train* algorithm.

The mutation operator is based on the standard single bit mutation (section 2.4.1, figure 2.8, page 38). Mutation is also applied only to the rule bits of the chromosome, not the *delta*s. If any bit on a gene is mutated, the $delta1$ and $delta2$ values for that gene will be replaced with randomly generated values. Figure 6.6 shows an example of the mutation operator in the genetic algorithm.

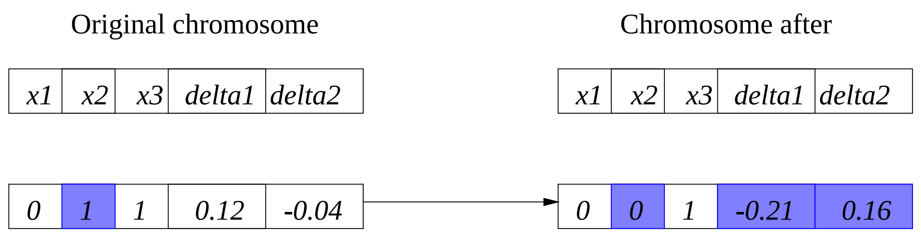


Figure 6.6: Mutation in the *GA-train* algorithm.

6.2.6 Description of the Entire GA-train Algorithm

The entire algorithm is described as follows:

Step 1: Initialise the network weights to random values in the range (-0.5, 0.5);

Step 2: Create a population with randomly assigned rule bits and *delta* values;

For the evolutionary process, repeat step 3 to step 8:

Step 3: Evaluate each individual of the population according to the fitness function.

Step 4: If the best performer is an improvement on the previous best, then apply the delta values of the best performer to the network weights to calculate the new weights;

Step 5: Measure the classification accuracy on the classification training data. If the best one can solve the problem or some other termination criteria have been reached, then save the best network, report the result and stop the evolutionary process;

Step 6: Rank and select members of the population;

Step 7: Perform crossover;

Step 8: Perform mutation;

6.2.7 Characteristics of GA-train

The method described in this section is to have the genetic algorithm evolve the changes, or the values of the two *delta*s, to the weights of the network being trained. This is done by modifying the weights according to the gene structure, the combination of the rules and the weight updating mechanism.

Rather than have the genetic algorithm evolve the actual weights and biases themselves, only the way in which the rules are to be applied and the *delta* values are evolved. Accordingly, the chromosomes being modified by the genetic algorithm do not represent the weights or biases of the network, only the method by which the rules are to be applied and the *delta* values are represented.

6.3 Genetic Algorithm for Network Refinement – GA-refine

Directly applying trained networks obtained in the first phase to detecting large pictures containing the small objects of interest often results in some false alarms, particularly when the large pictures have a highly cluttered background, as shown in chapter 4 and chapter 5. To decrease the false alarm rate, the second phase is developed for refining the networks trained in phase one. This is done by another genetic algorithm *GA-refine*.

6.3.1 Overview of the GA-refine Algorithm

Figure 6.7 shows a flow diagram of the *GA-refine* algorithm. After reading the parameters set/defined by the user, this genetic algorithm uses the entire images in the detection training set and the centres of the desired objects in the multiple classes of interest to perform the evolutionary process. As mentioned earlier, to distinguish it from the network training on the cutouts in the first phase, further training of the networks on the entire images in the second phase is called refining the networks, or *network refinement*. The *GA-refine* algorithm uses the same gene structure as the *GA-train* algorithm and begins with the population associated with the trained network produced by phase one. The fitness of an individual is computed by generating a network from the saved weights and encoded weight changes and applying it, in a moving window fashion, over these large pictures to detect the objects of interest. Through the evolutionary process, the best individual (network) in the population will continue to become fitter. The genetic operators used here – selection, crossover and mutation – are all the same as those in the *GA-train* algorithm in the first phase. However, the fitness function used here is different from that in the *GA-train* algorithm. In the remainder of this section, we will described the fitness function used in the *GA-refine* algorithm and compare this algorithm with the *GA-train* algorithm.

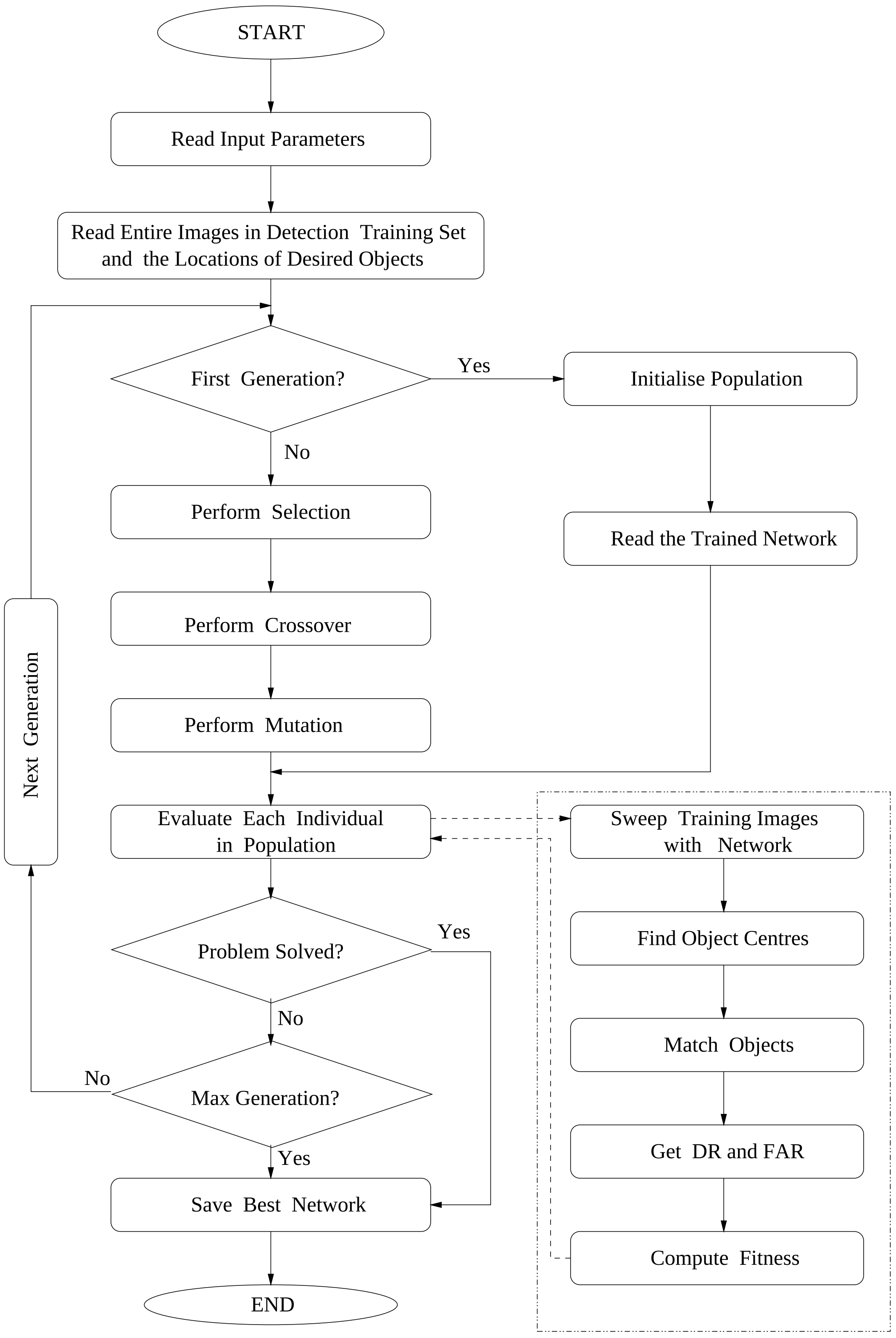


Figure 6.7: Flow diagram of the *GA-refine* algorithm.

6.3.2 Fitness Function

In network refinement, the *GA-refine* algorithm directly uses the detection rate and false alarm rate performance to calculate the fitness of a network. As shown in figure 6.7, the fitness of a chromosome (associated with a network) is computed according to the following six steps:

1. Realise the network from the weights and the weight changes encoded in the chromosome.

2. Apply the network as a moving $n \times n$ window template across the entire images in the detection training set and obtain the object sweeping maps for the object classes of interest. We use the neural network to scan all these pictures pixel by pixel. This is identical to the corresponding detection procedure used in the basic approach (section 4.6.1, page 98) except that this is now performed as part of the training procedure.

3. Find the centres of all of the objects detected by the *centre-finding algorithm* as presented in chapter 4 (section 4.6.2, page 100).

4. Compare the detected object centres with known locations of the desired objects and obtain the number of objects correctly and incorrectly detected by the network according to the *object matching* method described in chapter 4 (section 4.7.1, page 101).

5. Determine the overall detection rate and false alarm rate of this network according to equation 2.3 (page 16) and equation 2.5 (page 16).

6. Compute the fitness according to equation 6.1.

$$fitness(FAR, DR) = A \times FAR/(FAR + DR) + B \times (1 - DR) \tag{6.1}$$

   where DR and FAR represent the detection rate and the false alarm rate of the network, and A and B are constants which reflect the relative importance of the false alarm rate and the detection rate.

Under this design, it is clear that the smaller the fitness, the better the detection performance. The best case is zero fitness when the detection rates for all classes are 100% and the false alarm rates are zero, that is, all the objects of interest are correctly detected by the network without any false alarms.

6.3.3 Characteristics of GA-refine

The differences between the first genetic algorithm *GA-train* and the second genetic algorithm *GA-refine* are:

- The weights of the network are initialised with the trained weights obtained in the first phase instead of being randomly generated.
- The entire images in the detection training set are used directly for the network learning process rather than the cutouts in the classification training set.
- Only the object classes of interest are considered. The non-interesting object classes such as the background are not considered.
- A new fitness function based on the detection rate and false alarm rate for the entire images is used instead of the mean squared error on the cutouts.
- Different parameters and termination criteria are used. The details will be presented in the next section.

6.4 Results

This section presents two groups of results. The first the results of network training and testing for object classification on the cutouts of the three databases using the first genetic algorithm, *GA-train*, described in this chapter compared with the backward error propagation algorithm, *BP-train*, used in the basic approach described in chapter 4. The second is the results of object detection on the entire images in the detection test set achieved by the three detection methods introduced in this chapter compared with the basic approach in chapter 4.

6.4.1 Object Classification Results

To investigate the object classification performance of the *GA-train* algorithm on large networks with raw image pixel data, we applied this algorithm to the cutouts in the classification data sets for the three image databases. Again, this only involves network training and testing but not network sweeping. The number of evaluations and the test mean squared error (*MSE*) are compared with those obtained by the *BP-train* algorithm in the basic approach. The classification accuracy is also presented to keep the consistency with the previous chapters. Before presenting the results, we define the term, *evaluations* as follows.

Definition 6.8 *Number of Network Evaluations*

For the purpose of comparison with the *BP-train* algorithm, a single *evaluation* of an individual network in the population of the *GA-train* algorithm is considered to be computationally equivalent to a single forward or backward pass of the *BP-train* algorithm. A generation requires evaluation of each member of the population for the entire classification training set, so in terms of the *BP-train* algorithm a generation is typically computationally equivalent to a number of epochs. The number of epochs that a generation is equivalent to varies with the number of individuals in the population.

For example, for the *GA-train* algorithm with a population of 100 individuals (networks) which takes 20 generations to train, each network will have been evaluated 20 times for any training pattern, so there will be $20 \times 100 = 2000$ network evaluations per training pattern. For the *BP-train* algorithm, a training period of 500 epochs will have resulted in $500 \times 2 = 1000$ network evaluations per training pattern. This takes into consideration the forward and backward passes of the backward propagation technique. For ease of comparison, the number of trials for the *GA-train* algorithm and the number of epochs for the BP-train algorithm are converted into the number of network evaluations per training pattern.

Easy Pictures

The parameter values used by the *GA-train* algorithm for the simple object classification in the easy pictures are described in table 6.1. These parameters were empirically determined through experiments in order to obtain good results.

Population Size	200	Max Generations	100
Crossover Rate	95%	Delta1 Range	±0.06
Mutation Rate	5%	Delta2 Range	±0.02

Table 6.1: Main parameter values used in the evolutionary process for object classification in the easy pictures.

Table 6.2 shows the network training and testing results for the easy pictures of the *GA-train* algorithm for the same network architecture and training termination criterion as table 4.1 (page 103): the network architecture is 196-4-4, termination is when all the patterns in the classification training set are correctly classified, or *percent* = 100%, and the experiment is repeated 15 times. As shown in table 6.1, there is another termination condition for *GA-train*, that is, the training will also be terminated when the number of generation reaches 100. There are two main reasons: one is to avoid overtraining and the other is that if *GA-train* has not converged at this generation one should change the parameters and retrain the network.

Expt. No.	Generations	Training MSE ($\times 10^{-2}$)	Training Accuracy(%)	Test MSE ($\times 10^{-2}$)	Test Accuracy(%)
1	28	5.7	100	5.8	100
2	21	5.9	100	5.9	100
3	30	6.5	100	6.4	100
4	31	6.7	100	6.8	100
5	22	6.1	100	6.2	100
6	25	7.1	100	7.2	100
7	19	5.2	100	5.2	100
8	32	5.6	100	5.7	100
9	26	4.9	100	5.0	100
10	24	5.7	100	5.8	100
11	28	6.2	100	6.3	100
12	23	4.8	100	5.0	100
13	30	5.5	100	5.7	100
14	22	6.0	100	6.1	100
15	25	6.2	100	6.2	100
$\mu \pm \sigma$	25.733±3.972	5.87±0.64	100±0	5.95±0.62	100±0

Table 6.2: Results of network training and testing for object classification in the easy pictures using the *GA-train* algorithm. (Network architecture: 196-4-4; Stop criterion: *Percent* = 100%; Training set size = 60; Test set size = 180; Repetitions = 15.)

As shown in figure 6.2, the number of generations required for network training is 25.733± 3.972, the training mean squared error is $(5.87 \pm 0.64) \times 10^{-2}$ and the test mean squared error is $(5.95 \pm 0.62)\ \times 10^{-2}$. As with networks trained using the *BP-train* algorithm in the basic approach, networks trained using the *GA-train* algorithm also correctly classify all the patterns in the classification test set. To compare the results here with those for the basic approach, as mentioned earlier, we convert the number of generations taken by the *GA-train* algorithm here and the number of epochs used by the *BP-train* algorithm in the basic approach into the number

Training Algorithms	Evaluations ($\mu \pm \sigma$)	Training MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Training Accuracy (%) ($\mu \pm \sigma$)	Test MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Test Accuracy (%) ($\mu \pm \sigma$)
GA-train	5146.6±794.4	5.87±0.64	100±0	5.95±0.62	100±0
BP-train	398.8±36.18	5.09±0.30	100±0	5.17±0.27	100±0

Table 6.3: Comparison of the results of network training and testing for object classification in the easy pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network architecture: 196-4-4; Stop criterion: *Percent* = 100%; Training set size = 60; Test set size = 180; Repetitions = 15.)

of network evaluations, and present the comparison in table 6.3.

According to table 6.3, the number of evaluations needed by the *GA-train* algorithm is 5146.6±794.4, which is much bigger than that 398.8±36.18 for the basic approach in both mean and standard deviation. The test mean squared error is $(5.95 \pm 0.62) \times 10^{-2}$, which is also slightly higher than that $(5.17 \pm 0.27) \times 10^{-2}$ achieved by the basic approach.

Training Algorithms	Evaluations ($\mu \pm \sigma$)	Training MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Training Accuracy (%) ($\mu \pm \sigma$)	Test MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Test Accuracy (%) ($\mu \pm \sigma$)
GA-train	4365.8±537.2	5.95±0.54	95.0±0	5.96±0.52	92.34±1.30
BP-train	354.4±17.36	5.73±0.10	95.0±0	5.75±0.10	93.58±1.06

Table 6.4: Comparison of the results of network training and testing for object classification in the easy pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network architecture: 196-5-4; Stop criterion: *Percent* = 95%; Training set size = 60; Test set size = 180; Repetitions = 15.)

Table 6.4 shows the results of the *GA-train* algorithm for the same network and termination criterion as table 4.2 – the *BP-train* algorithm in the basic approach (page 104). This table shows a similar pattern to table 6.3. The number of evaluations required by the *GA-train* algorithm is 4365.8±537.2, which is much larger than that 354.4±17.36 used by the *BP-train* algorithm. The test mean squared error is $(5.96 \pm 0.52) \times 10^{-2}$, which is higher than that $(5.75 \pm 0.10) \times 10^{-2}$ for the *BP-train* algorithm. The classification accuracy is 92.34%, which is also slightly lower than that (93.58%) for the *BP-train* algorithm.

In summary, the results show that the *GA-train* algorithm can be used to train the large networks in the easy pictures, however it needed more evaluations and resulted in a higher mean squared error on the classification test set than the *BP-train* algorithm in the basic approach.

Coin Pictures

The parameter values for regular object classification in the coin pictures used by the genetic algorithm are described in table 6.5.

Population Size	200	Max Generations	100
Crossover Rate	95%	Delta1 Range	± 0.01
Mutation Rate	5%	Delta2 Range	± 0.004

Table 6.5: Main parameter values used in the evolutionary process for object classification in the coin pictures.

Training Algorithms	Evaluations $(\mu \pm \sigma)$	Training MSE $(\times 10^{-2})$ $(\mu \pm \sigma)$	Training Accuracy (%) $(\mu \pm \sigma)$	Test MSE $(\times 10^{-2})$ $(\mu \pm \sigma)$	Test Accuracy (%) $(\mu \pm \sigma)$
GA-train	9379.3±1031.1	3.50±0.65	100±0	3.53±0.67	100±0
BP-train	469.2±131.88	2.60±0.47	100±0	2.60±0.46	100±0

Table 6.6: Comparison of the results of network training and testing on object classification in the coin pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network Architecture: 576-3-5; Stop criterion: *Percent* = 100%; Training set size = 100; Test set size = 100; Repetitions = 15.)

Table 6.6 shows the results of the *GA-train* algorithm for the same network and termination criterion as table 4.3 – the *BP-train* algorithm in the basic approach (page 105). Using the *GA-train* algorithm, the number of evaluations needed for network training is 9379.3±1031.1, which is about 20 times longer than that (469.2±131.88) for the *BP-train* algorithm in the basic approach. In addition, the test mean squared error is $(3.53 \pm 0.67) \times 10^{-2}$, which is also higher than for the *BP-train* algorithm in the basic approach. As in the basic approach, the *GA-train* algorithm also led to 100% classification accuracy on the classification test data for the coin pictures.

Table 6.7 describes the results of the *GA-train* algorithm on the coin classification for the same network and termination criterion as table 4.4 – the *BP-train* algorithm in the basic approach (page 106). Table 6.7 shows a very similar pattern to table 6.6, which suggests the *GA-train* algorithm can be used to train the networks on the cutouts of the coin pictures, however the training process takes longer and the trained network produces higher mean squared error on the same test data.

Training Algorithms	Evaluations ($\mu \pm \sigma$)	Training MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Training Accuracy (%) ($\mu \pm \sigma$)	Test MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Test Accuracy (%) ($\mu \pm \sigma$)
GA-train	3325.3±432.6	3.97±0.55	95.0±0	3.98±0.56	92.36±1.87
BP-train	174.26±10.16	3.33±0.39	95.0±0	3.17±0.36	94.53±1.25

Table 6.7: Comparison of the results of network training and testing for object classification in the coin pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network Architecture: 576-5-5; Stop criterion: *percent* = 95%; Training set size = 100; Test set size = 100; Repetitions = 15.)

Retina Pictures

The parameter values for very complex object classification in the retina pictures used by the genetic algorithm are described in table 6.8.

Population Size	300	Max Generations	150
Crossover Rate	90%	Delta1 Range	±0.004
Mutation Rate	10%	Delta2 Range	±0.002

Table 6.8: Main parameter values used in the evolutionary process for object classification in the retina pictures.

Table 6.9 shows the object classification results of the *GA-train* algorithm on the retina pictures for the same network and training termination criterion as table 4.5 – the *BP-train* algorithm in the basic approach (page 107). Compared with network training for the easy and

Training Algorithms	Evaluations ($\mu \pm \sigma$)	Training MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Training Accuracy (%) ($\mu \pm \sigma$)	Test MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Test Accuracy (%) ($\mu \pm \sigma$)
GA-train	31350±4021.6	11.5±1.63	65.0±0	13.31±1.53	63.41±3.57
BP-train	401.6±53.34	9.44±0.22	65.0±0	9.12±0.18	65.05±2.82

Table 6.9: Comparison of the results of network training and testing for object classification in the retina pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network Architecture: 256-4-5; Stop criterion: *Percent* = 65%; Training set size = 100; Test set size = 61; Repetitions = 10.)

the coin databases, the *GA-train* algorithm took much a longer time to reach the termination point. The number of network evaluations required for network training is 31350±4021.6, which is about 75 times more than that (401.6±53.34) for the basic approach. The *GA-train* also resulted in a higher test mean squared error than the *BP-train* algorithm.

Table 6.10 shows the object classification results of the *GA-train* algorithm on the retina pictures for the same network and training termination criterion as table 4.6 – the *BP-train* algorithm in the basic approach (page 108).

Training Algorithms	Evaluations ($\mu \pm \sigma$)	Training MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Training Accuracy (%) ($\mu \pm \sigma$)	Test MSE ($\times 10^{-2}$) ($\mu \pm \sigma$)	Test Accuracy (%) ($\mu \pm \sigma$)
GA-train	42341.5±7538.3	9.81±0.93	75.4±0	13.30±1.72	62.30±7.45
BP-train	951.6±265.52	6.70±0.52	75.4±0	8.34±0.79	71.83±2.63

Table 6.10: Comparison of the results of network training and testing for object classification in the retina pictures using the *GA-train* algorithm and the *BP-train* algorithm. (Network Architecture: 256-5-5; Stop criterion: *Percent* = 75%; Training set size = 100; Test set size = 61; Repetitions = 10.)

Table 6.10 shows a similar pattern to table 6.9. The *GA-train* algorithm took very a long time to train the network to the termination criterion, that is, when 75% of the training patterns were correctly classified. Furthermore, the network trained by this algorithm resulted in a much higher test mean squared error than the basic approach. Comparing the test mean squared error with the training mean squared error, it is found that the overtraining problem occurred in both training procedures. This was conformed by checking the classification accuracy. The network trained by the *BP-train* algorithm in the basic approach produced an average classification accuracy of 71.83% on the classification test set, while the *GA-train* algorithm resulted in 62.30% classification accuracy.

This subsection described a series of comparisons of the network training and testing results for object classification using the first genetic algorithm in this approach with the backward error propagation algorithm in the basic approach on the cutouts of the three image databases. On all cases for training the relatively large networks with the inputs of the image pixel data described here, the *GA-train* algorithm needed more evaluations to train the networks and the trained network resulted in a higher test mean squared error than the *BP-train* algorithm in the basic approach. While the *GA-train* algorithm is worse on object classification, the real goal is object detection so we persevere.

6.4.2 Object Detection Results

This section describes a series of comparisons of the object detection results of the three methods introduced in this chapter, that is, the *GA-train* method, the *BP-train+GA-refine* method and the *GA-train+GA-refine* method, with the *BP-train* method or the basic approach described in chapter 4.

For the *GA-train* method, each of the 15 trained networks shown in table 6.2 was applied to the detection test set of the easy pictures, each of the 15 trained networks shown in table 6.6 was applied to the entire images in the detection test set of the coin pictures, and each of the 10 trained networks shown in table 6.9 was applied to the detection test set of the retina pictures. The averages are presented in tables 6.12, 6.13 and 6.14 and figures 6.8 and 6.9.

For the *GA-train+GA-refine* method, each of the 15 trained networks learnt by the *GA-train* algorithm in table 6.2 was used in the network refinement procedure to obtain 15 refined networks. Each of the 15 refined networks was then applied to the the entire images in the detection test set of the easy pictures. Similarly, each of the 15 refined networks based on the 15 trained networks shown in table 6.6 was applied to the entire images in the detection test set of the coin pictures. Each of the 10 refined networks based on the 10 trained networks shown in table 6.9 was applied to the detection test set of the retina pictures. The averages are presented in tables 6.12, 6.13 and 6.14 and figures 6.8 and 6.9.

For the *BP-train+GA-refine* method, the 15, 15 and 10 trained networks shown in tables

4.1, 4.3 and 4.5 for the easy, the coin and the retina pictures were used in the second genetic algorithm for network refinement. Then each of the 15, 15 and 10 refined networks was applied to the entire images in the detection test set of the easy, the coin and the retina pictures to obtain the object detection results and the averages are presented in tables 6.12, 6.13 and 6.14 and figures 6.8 and 6.9.

The choosing of thresholds was done in the same way as for the basic approach, as shown in figure 4.8 (page 110).

Main Parameters for Network Refinement and Termination Strategy

Algorithms	Parameters	Easy Pictures	Coin Pictures	Retina Pictures
GA-refine Based on GA-train	Net Architecture	196-3-4	576-3-5	256-4-5
	Population size	200	200	500
	Crossover rate	95%	95%	90%
	Mutation rate	5%	5%	10%
	Max Generations	50	50	50
	Delta1 range	±0.08	±0.04	±0.02
	Delta2 range	±0.05	±0.02	±0.005
	A	5	2	3
	B	2	3	2
GA-refine Based on BP-train	Net Architecture	196-3-4	576-3-5	256-4-5
	Population size	200	200	500
	Crossover rate	95%	95%	90%
	Mutation rate	5%	5%	10%
	Max Generations	50	50	50
	Delta1 range	±0.06	±0.008	±0.02
	Delta2 range	±0.02	±0.002	±0.005
	A	2	1	3
	B	1	1	2

Table 6.11: The main parameters for network refinement used by the *GA-refine* algorithm based on the *GA-train* algorithm and the *BP-train* algorithm.

Table 6.11 shows the main parameters used by the *GA-refine* algorithm based on the *GA-train* algorithm in the *GA-train+GA-refine* method and the *BP-train* algorithm in the *BP-train+GA-refine* method for the three databases.

For example, for the easy pictures, *GA-refine* in both *GA-train+GA-refine* and *BP-train+GA-refine* used the same network architecture (196-3-4), population size (200), crossover rate (95%), mutation rate (5%) and the maximum number of generations (50). The ranges of *delta1* and *delta2* used in the *GA-refine* algorithm for the *GA-train+GA-refine* method were ±0.08 and ±0.05 and the constants A and B were 5 and 2. For the *GA-refine* algorithm in the *BP-train+GA-refine* method, the ranges of *delta1* and *delta2* were ±0.06 and ±0.02 and the fitness parameters A and B were 2 and 1.

The parameters described in figure 6.11 were carefully selected through empirical search to achieve good results. For both cases, the evolutionary process for network refinement was terminated when either the problem was solved or the number of generations reached 50.

Easy Pictures

Easy Pictures		Object Classes		
		Class1	Class2	Class3
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	BP-train (the basic approach)	0	91.2	0
	GA-train	80	839	273
	BP-train + GA-refine	0	0	0
	GA-train + GA-refine	0	663	144

Table 6.12: Comparison of object detection results for the easy pictures using the four detection methods. (Network architecture: 196-4-4; Repetitions = 15.)

Table 6.12 shows a comparison of the results for the easy pictures. As in the basic approach (the *BP-train* method), the detection rate for the three classes obtained using the three methods introduced in this chapter reached 100%. Under this detection rate, the best corresponding false alarm rates achieved for each class, however, were quite different. Take *class2* detection as an example, *BP-train+GA-refine* did not produce any false alarms, which was much better than the basic approach (the *BP-train* method) that produced a false alarm rate of 91.2%. The *GA-train* method and the *GA-train+GA-refine* method resulted in 839% and 663% false alarm rates respectively, which suggests that the *GA-train+GA-refine* method is superior to the *GA-train* method for the easy pictures. As can be seen from table 6.12, the *GA-train* method led to more false alarms than the basic approach, however the methods with the *GA-refine* algorithm resulted in fewer false alarms for all classes than the corresponding methods without the refinement. The *BP-train+GA-refine* method gave the best detection performance, that is, all the objects of interest in each class were correctly detected without any false alarms produced.

Coin Pictures

Experiments with the coin images gave similar results to the easy pictures. The best false alarm rates for all the four classes *head005, tail005, head020* and *tail020* under the detection rate of 100% obtained by the four methods are shown in table 6.13. Of all the four methods, *BP-train+GA-refine* gave the best results, where the detection rate for all the four object classes reached 100% without any false alarms. The methods with the *GA-refine* algorithm always resulted in fewer false alarms than the corresponding basic methods without the refinement. The *GA-train* method produced more false alarms than the basic approach.

Coin Pictures		Object Classes			
		head005	tail005	head020	tail020
Best Detection Rate (%)		100	100	100	100
False Alarm Rate (%)	BP-train (the basic approach)	0	0	182	37.5
	GA-train	357	19	333	778
	BP-train + GA-refine	0	0	0	0
	GA-train + GA-refine	125	0	37.5	215

Table 6.13: Comparison of object detection results for the coin pictures using the four detection methods. (Network architecture: 576-3-5; Repetitions = 15.)

According to table 6.13, detecting heads and tails of 5 cent coins turns out to be relatively straight forward, while detecting the heads and tails of 20 cent coins is a difficult problem. To give a clearer view of the comparison of the four methods, we present the extended ROC curves of detecting the heads and tails of 20 cent coins in figure 6.8. As can be seen from this figure, at all levels of detection rate, the *BP-train+GA-refine* did not produce any false alarms, the *GA-train+GA-refine* method always resulted in fewer false alarms than the *GA-train* method and *GA-train* always resulted in more false alarms than *BP-train.*

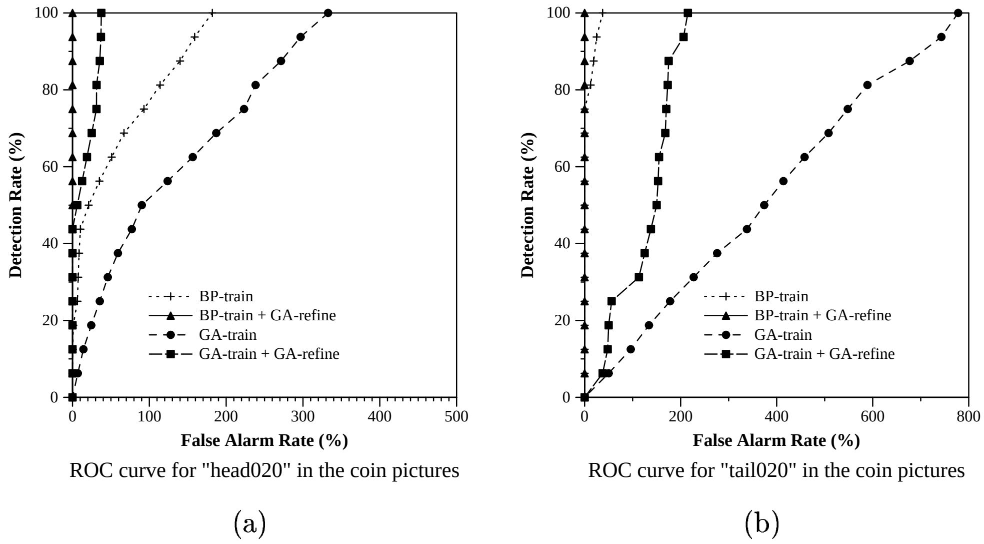


Figure 6.8: Comparison of the results for class *head020* and class *tail020* in the coin pictures using the four detection methods.

Retina Pictures

Retina Pictures	Object Classes	
	haem	micro
Detection Results	Best DR (%)/FAR(%)	Best DR(%)/FAR(%)
BP-train(the basic approach)	73.91 /2859	100 / 10104
GA-train	50.37 /4000	90 / 5606
BP-train + GA-refine	82.61 / 2156	100 / 2706
GA-train + GA-refine	82.61 / 2298	100 / 5055

Table 6.14: Comparison of object detection results for the retina pictures using the four detection methods. (Network architecture: 256-4-5; Repetitions = 10. DR: Detection Rate. FAR: False Alarm Rate.)

Table 6.14 shows a comparison of the best detection rate and the corresponding false alarm rate achieved using the four detection methods. These results were not as good as those for the easy and the coin pictures. As can be seen from this table, none of these methods resulted in 100% detection rate for detecting haemorrhages. However, the two methods incorporating the refinement genetic algorithm resulted in 82.61% detection rate at which fewer false alarms were produced than the other two methods without the refinement. *BP-train, GA-train+GA-refine* and *BP-train+GA-refine* correctly detected all the micro aneurisms in the retina pictures, however resulted in 10104%, 5055% and 2706% false alarm rates. The *GA-train* only achieved a detection rate of 90%.

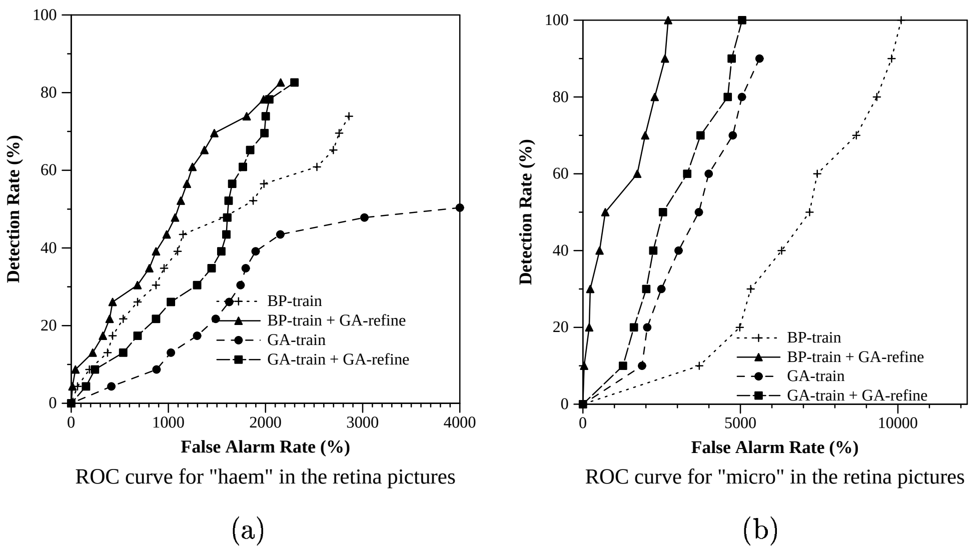


Figure 6.9: Comparison of the results for detecting class *haem* and class *micro* in the retina pictures using the four detection methods.

Figure 6.9 shows the ROC curves of detecting class *haem* and class *micro* in the retina pictures. These results show the same pattern as the other two databases. In all cases, the methods with the refinement genetic algorithm, *GA-refine*, were clearly superior to the corresponding methods without the refinement – the false alarm rate at all levels of detection rate was lower, and the *BP-train+GA-refine* method gave the best results among the four methods. Unlike the results of detecting other objects, the *GA-train* method produced fewer false alarms than the *BP-train* method (the basic approach) at most detection rates for detecting class *micro* in the retina pictures, but it could not result in 100% detection rate. The overall detection performance on the retina pictures, however, is still not satisfactory.

6.5 Summary and Discussion

6.5.1 Discussion

This approach has some disadvantages:

- The training times for the network refinement are quite long. Some of the runs took longer than 48 hours on a SPARC station.
- Overall, the method of the backward error propagation algorithm with refinement by genetic algorithms (*BP-train+GA-refine*) produces pixel based networks that work well on objects on a relatively uniform background. However it does not work very well for detecting very irregular objects against highly cluttered backgrounds in the retina pictures, even if it has greatly improved the detection performance achieved by the basic approach.
- The best method for object detection described in this chapter is based on two phases. This needs the further computational effort of network refinement.

6.5.2 Next Step

As discussed earlier, genetic algorithms can be applied to the further training/refinement of the networks on the full images in the detection training set and improve the object detection performance. This suggests the evolutionary process can produce better solutions for the detection problems. This raises the question of whether some other use of the evolutionary process can be used to improve the detection performance, particularly on the retina pictures. In chapter 7, we investigate another evolutionary process, genetic programming, for the three detection problems.

6.5.3 Summary

The goal of the work described in this chapter was to investigate the use of genetic algorithms for network training and network refinement for the object classification and detection problems. A two phase approach was introduced here. In phase one, the networks were trained by a genetic algorithm with the fitness of mean squared error on cutouts of the classification data set. In phase two, the trained networks produced in phase one were refined by applying a second genetic algorithm. The fitness function in this case was based on a linear combination of the false alarm rate and detection rate on the entire images in the detection training set. The trained networks produced by the backward error propagation algorithm in the basic approach were also used with the network refinement procedure. The two phase approach described here produced three new detection methods, the *GA-train* method, the *BP-train + GA-refine* method and the *GA-train + GA-refine* method. These three methods and the basic approach (the *BP-train* method) were compared on three detection problems of increasing difficulty. The experimental results showed that:

- The first genetic algorithm, *GA-train*, did not lead to faster training and better test performance on the cutouts of any of the three databases than *BP-train*.
- The genetic algorithm without the network refinement, the *GA-train* detection method, did not give better detection performance in most cases compared with the basic approach.
- The methods which incorporated the refine genetic algorithm always resulted in better detection performance than the corresponding methods without the refinement in all three detection problems. In other words, the second genetic algorithm, *GA-refine*, can be used to improve the detection performance of networks trained by either the *BP-train* algorithm or the *GA-train* algorithm.
- Of the four methods, the hybrid method of the backward error propagation algorithm in the basic approach and the refinement genetic algorithm, the *BP-train + GA-refine* method, always produced the best detection performance.

Chapter 7

Genetic Programming for Multiple Class Object Detection

7.1 Introduction

Chapter 4 described the detection results on the three databases achieved by the basic approach in which the networks were trained on the cutouts by the backward error propagation algorithm. In chapter 6, it was shown that the detection performance could be improved by an extension of the work – applying a genetic algorithm based refinement process to the entire images. In this chapter, we investigate a further extension of these methods. We consider the replacement of the neural networks with programs that will be evolved using genetic programming. As in the refinement step described in the previous chapter the fitness will be based on detection performance on the entire training images. We call this method the *genetic programming based approach.*

7.1.1 Overview of the Approach

Figure 7.1 shows an overview of the genetic programming based approach. As in the basic approach, this approach also has a learning process and a testing procedure. In the learning process the evolved programs are applied, in a moving window fashion, to the entire images in the detection training set to locate the objects of interest. This is very similar to the network sweeping procedure in the basic approach where the neural networks were applied to the entire images in the detection test set. This learning process is also similar to the network refinement procedure by the genetic algorithm except that general computer programs are evolved rather than neural networks. The best evolved programs are applied to the entire images in the detection test set to measure object detection performance.

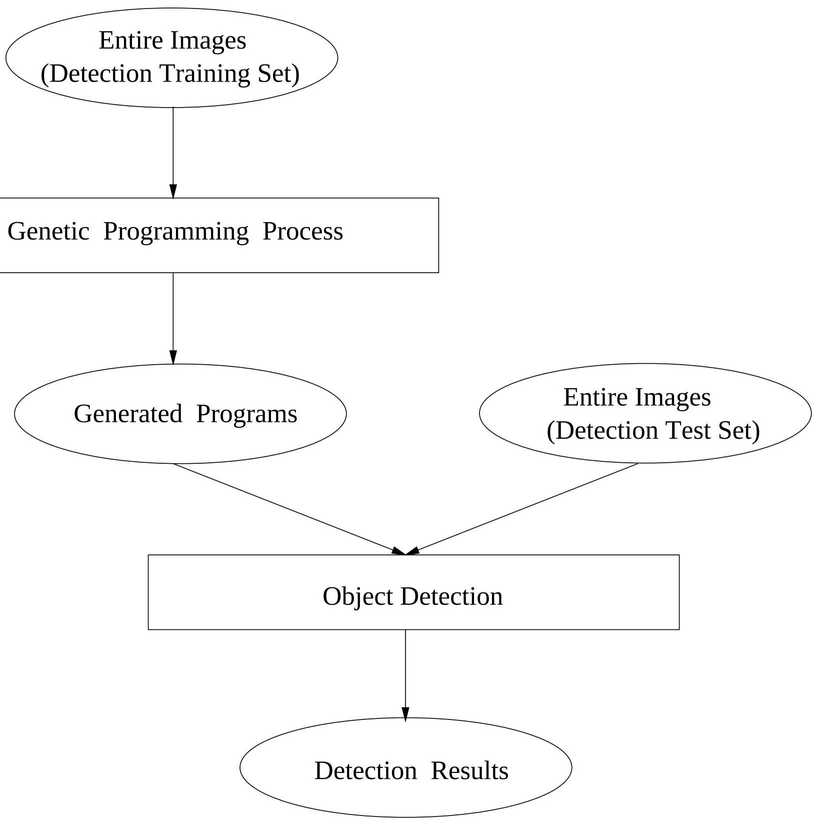


Figure 7.1: An overview of the genetic programming based approach for multiple class object detection.

7.1.2 Chapter Goals

In keeping with the goal of domain independent object detection, a number of *pixel level, domain independent features*, or *pixel statistics* are used as terminals and a set of *standard arithmetic operators* are used as functions. The genetic programming evolutionary process is expected to automatically select just those relevant to a specific domain. The overall goal of the genetic programming based approach is to determine whether programs evolved by genetic programming can do a good enough job of detecting the objects of interest in the three databases used in this thesis (chapter 3). Specifically we are interested in:

- What pixel based, domain independent image features would make useful terminals.

- Whether the four standard arithmetic operators $(+, -, *, /)$ will be sufficient for the function set.
- How the fitness function can be constructed given that there are several classes of interest.
- How the performance will vary with increasing difficulty of image detection problems.
- Whether the performance will be better than the basic approach on the same detection problems.

7.1.3 Structure of the Chapter

In the remainder of this chapter, we will describe the different components of this approach, the key issues relating to the terminal set, the function set and the fitness function. After giving the parameters used in this approach, we present and analyse the experimental results and compare them with the basic approach. This chapter ends with a summary and a discussion.

7.2 Genetic Programming Adapted to Object Detection

As in the neural network sweeping procedure, the programs use a square input field which is large enough to contain each of objects of interest and are applied, in moving window fashion, over the large pictures in the detection training set to locate the objects of interest. The terminals are computed from the square input field and form the inputs of the programs during the evolutionary process. The output of an evolved program is a floating point value, which is used for object classification during the sweeping procedure.

The flow diagram presented in figure 7.2 shows the learning procedure of genetic programming adapted to multiple class object detection. Before the learning process starts, a series of pre-defined input parameters, the entire images in the detection training set, and the pre-determined terminal set and function set are loaded. At the beginning of the learning process, a number of individual programs in the population are randomly created according to the terminals, functions and the program maximum depth. Each of the programs is then applied to the sweeping procedure to locate the objects of interest. The evolved programs are then evaluated by the fitness function. If the best of the programs reaches the termination criterion, then the programs are saved and the learning process is terminated. Otherwise, the evolutionary process will continue by applying the genetic operators of selection, reproduction, crossover and mutation to produce new programs in the population for the next generation. In general, the best program in the population will continue becoming fitter through the evolutionary learning process.

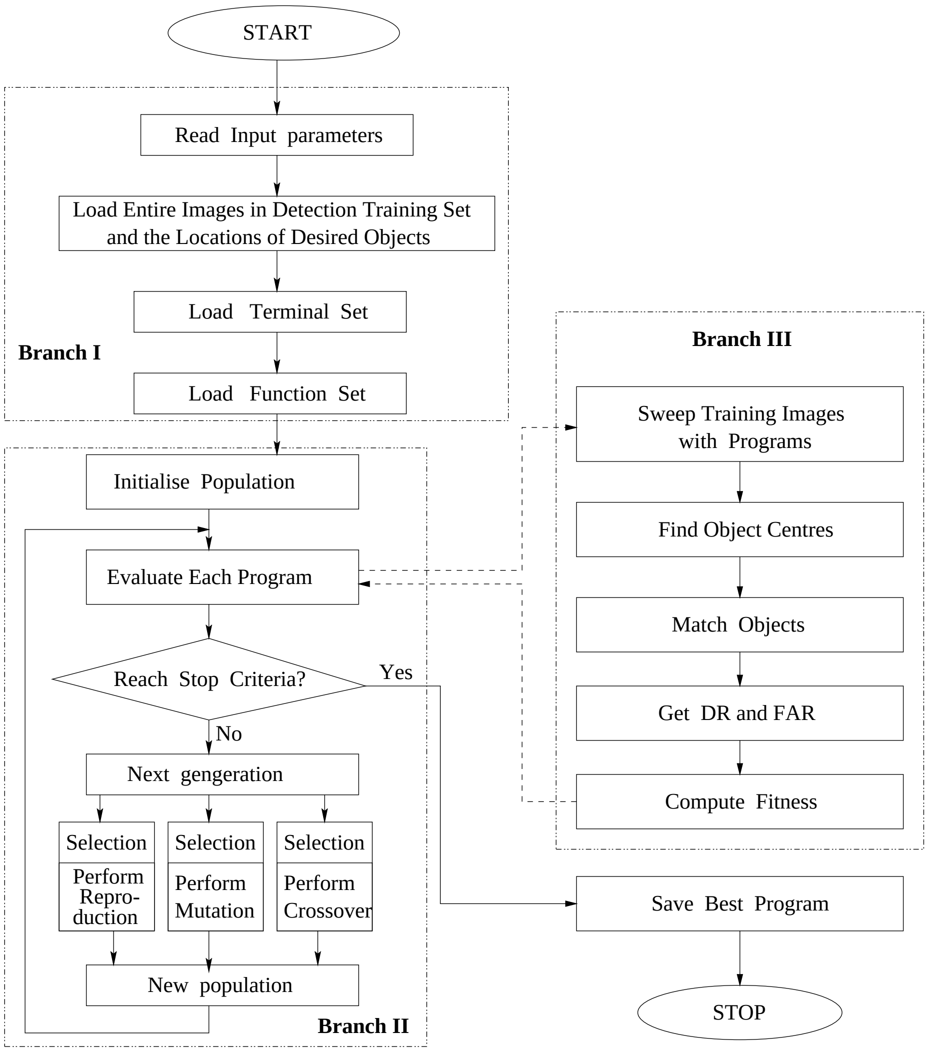


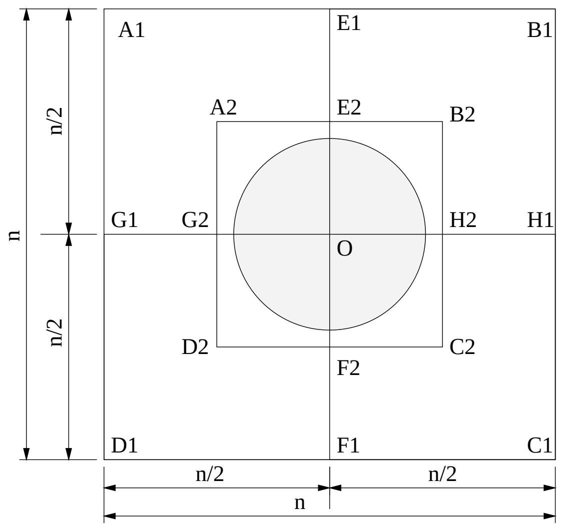
Figure 7.2: Learning procedure of genetic programming adapted to multiple class object detection.

For presentation convenience, we separate this approach into three main branches – *Branch I, Branch II* and *Branch III*, as shown in figure 7.2. The first branch loads a set of input parameters, the terminal set, function set and the entire training images for the evolutionary process. The terminal set and the function set are pre-determined. The second branch is the actual genetic programming process, which was described in section 2.4.2 (page 42). *Branch III* describes the key steps in the development of the fitness function for multiple class object detection problems. *Branch I* and *Branch III* vary with different tasks, while *Branch II* is relatively task independent and can be constructed from the existing genetic programming tools. In this approach, we used the genetic programming shell written by Wilson [205].

The following tasks are necessary for using genetic programming for multiple class object detection problems:

1. Determine the terminal set;
2. Determine the function set;
3. Determine the method of measuring fitness;
4. Determine the input parameters and the termination criterion.

7.3 The Terminal Set



Squares:
A1-B1-C1-D1-A1, A2-B2-C2-D2-A2, A1-E1-O-G1-A1, E1-B1-H1-O-E1, G1-O-F1-D1-G1, O-H1-C1-F1-O

Rows and Columns:
G1-H1, E1-F1, G2-H2, E2-F2

Sizes of the Squares and lines:
Suppose: A1-B1 = A1-D1 = n, then:
A1-E1 = E1-B1 = n/2
A1-G1 = G1-D1 = n/2
E1-O = O-F1 = n/2
G1-O = O-H1 = n/2
A2-B2 = G2-H2 = A2-D2 = E2-F2:
User Defined (Default: n/2)

Figure 7.3: The input field and the image regions and lines for feature selection in constructing terminals.

For object detection problems, terminals correspond to image features. To pursue the goal of domain independent object detection, we use twenty general, pixel level "features" (pixel statistics) rather than domain specific features. These features are obtained from the input field as shown in figure 7.3. The method of determining the size of the square input field can be found in section 4.3.2 (page 94). In figure 7.3 (left), the grey filled circle denotes an object of interest, the square A1-B1-C1-D1-A1 represents the square input field. The other five squares represent local regions from which features will be computed. The four central lines (rows and columns) are also used for a similar purpose. These six squares and four lines and their sizes are presented in figure 7.3 (right). The mean and standard deviation of the pixels comprising each of these regions are used as two separate features. There are 6 regions giving 12 features, *F1 to F12*. Also we use pixels along the main axes (four lines) of the input field, giving features *F13 to F20*. These 20 features are shown in table 7.1. Compared with domain specific features, these mean and standard deviation features do not need the hand-crafting of feature extraction programs; they still belong to the pixel level and represent the least step away from pure pixel based object detection.

Features		Regions and Axes of interest
mean	standard deviation	
F1	F2	big square A1-B1-C1-D1-A1
F3	F4	small central square A2-B2-C2-D2-A2
F5	F6	upper left square A1-E1-O-G1-A1
F7	F8	upper right square E1-B1-H1-O-E1
F9	F10	lower left square G1-O-F1-D1-G1
F11	F12	lower right square O-H1-C1-F1-O
F13	F14	central row of the big square G1-H1
F15	F16	central column of the big square E1-F1
F17	F18	central row of the small square G2-H2
F19	F20	central column of the small square E2-F2

Table 7.1: Twenty domain independent, pixel level features based on image pixel intensities.

In addition to these features we have a terminal which generates a random number in the range [0,255]. This corresponds to the number of grey levels in the images.

These features have the following characteristics:

- They are symmetrical and contain some information of object translation and rotation invariance.
- Local region features are included. This assists the finding of object centres in the sweeping procedure – if the evolved program is considered as a moving window template, the match between the template and the sub-image forming the input field will be better when the moving template is close to the centre of an object.

- They are general, domain independent and easy to extract. These features belong to the pixel level and can be part of a domain independent pre-existing feature library of terminals from which the genetic programming evolutionary process is expected to automatically learn and select only those relevant to a particular domain. This is quite different from the traditional image processing and computer vision approaches where the specific features for a particular domain are often needed.

7.4 The Function Set

We use the function set: F = $\{+, -, *, /\}$ which represents four arithmetic operations that form the second order nodes (i.e. 2 arguments). The +, -, and * operators have their usual meanings while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. These functions are standard arithmetic functions which are consistent with the goal of development of domain independent approaches.

A generated program consisting of the four functions and a number of terminals is shown in figure 7.4. This program performed particularly well for the coin pictures.

```
(+ (- (+ (+ (/ f16 f14) f5) (+ (/ (/ f11 (* f14 f20)) f11) (- f12
f14))) (- (* ( - (* (* (* f9 f11) f1) f10) (* f9 f17)) (/ f5 f18))
(- (+ (+ f17 (* (+ f11 f12) f20)) (* (- (+ f2 145.765) (/ f6 f11))
(- 133.082 f17))) (/ f11 (* f14 f20))))) (* (- (* (- (- f6 f5) (*
f3 f6)) (/ (+ (+ f1 145.765) (* f16 f10)) f18)) f12) (+ (+ f17 (*
(+ f17 f12) f20)) (* (+ f14 f12) (- (+ f1 f12) f17)))))
```

Figure 7.4: A generated program for the coin detection problem

7.5 The Fitness Function

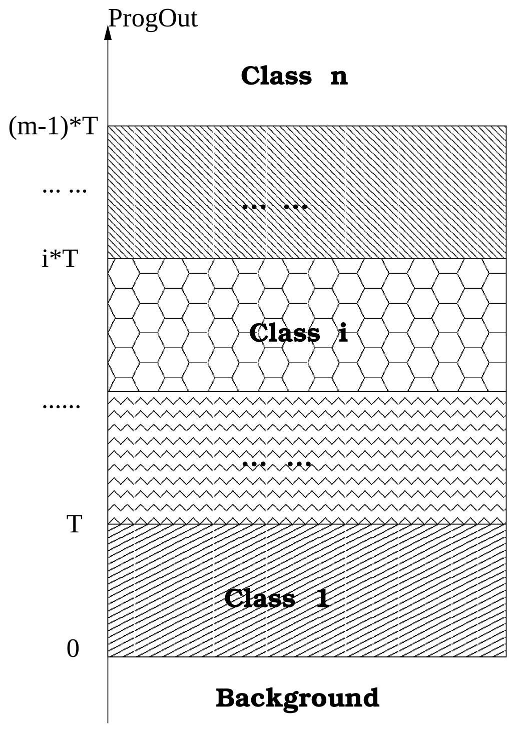
As presented in *Branch III*, figure 7.2 (page 174), the fitness of a program in the population is calculated by using its detection rate and false alarm rate on the entire images in the detection training set.

7.5.1 Object Classification Strategy

The output of any program is a floating point number which must indicate which of the objects of interest is currently in its input field. This is achieved by the *program classification map*, as shown in figure 7.5 where m is the number of object classes of interest, *ProgOut* is the output value of the evolved program and T is a constant defined by the user.

```
begin
    if (ProgOut $\in$ $(-\infty, 0)$) then
        the object is classified as background
    else if (ProgOut $\in$ $[0, T)$)
        the object is classified as class 1
    ... ... ... ...
    else if (ProgOut $\in$ $[(i-1)\times T, i\times T)$)
        the object is classified as class i
    ... ... ... ...
    else if (ProgOut $\in$ $[(m-1)\times T, +\infty)$)
        the object is classified as class n
    endif
end
```

(a)



(b)

Figure 7.5: Mapping of program output to an object classification.

7.5.2 Fitness Function

The fitness of a program is obtained as follows:

1. Apply the program as a moving $n\times n$ (n is the size of the input field) window template to each of the entire images in the detection training set and obtain the output value of the program at each possible pixel position. Label each pixel position with the 'detected' object according to the object classification strategy described in figure 7.5. Call this data structure a detection map. This step is similar to the network sweeping procedure. A detection map is similar to an object sweeping map described in section 4.6.1 (page 98),

except that an object in a detection map is associated with a floating point program output varying with the fitness value rather than a neural activation value within $(0, 1)$.

2. Find the centres of *objects of interest only.* The objects in non-interesting classes including the backgrounds are ignored. This is done as follows:

   - Scan the detection map for an object of interest. When one is found mark this point as the centre of the object and continue the scan $n/2$ pixels later in both horizontal and vertical directions.

3. Match these detected objects with the known locations of each of the desired true objects and their classes. A match is considered to occur if the detected object is within TOLERANCE pixels of its known true location. Here, TOLERANCE is a constant parameter defined by the user.

4. Calculate the detection rate DR and the false alarm rate FAR of the evolved program according to equation 2.3 (page 16) and 2.5 (page 16).

5. Compute the fitness of the program as shown in equation 7.1.

$$fitness(FAR, DR) = A \times FAR + B \times (1 - DR) \tag{7.1}$$

   where A and B are constants which reflect the relative importance of false alarm rate versus detection rate. It is noted that this fitness function is slightly different from the one for the refinement genetic algorithm (page 155). It was determined according to the empirical search during experiments. The evolutionary process converges faster using this fitness function than using equation 6.1 for the refinement genetic algorithm (page 155).

With this design, it is clear that the smaller the fitness, the better the performance. Zero fitness is the ideal case, which corresponds to the situation in which all of the objects of interest in each class are correctly found by the evolved program without any false alarms.

7.6 Main Parameters and Termination Criterion

7.6.1 Main Parameters

The values for the various system parameters used in the experiments are shown in table 7.2. POPULATION_SIZE is the number of individuals in the population, ELITISM_PCNT gives the

Parameters	Easy Pictures	Coin Pictures	Retina Pictures
POPULATION_SIZE	100	500	500
ELITISM_PCNT	10%	1%	2%
CROSS_RATE	65%	74%	73%
MUTATION_RATE	25%	25%	25%
CROSS_CHANCE_TERM	15%	15%	15%
CROSS_CHANCE_FUNC	85%	85%	85%
INITIAL_MAX_DEPTH	5	5	5
MAX_DEPTH	8	12	20
MAX_GENERATIONS	100	200	250
T	100	100	100
A	50	50	50
B	1000	1000	3000
TOLERANCE (pixels)	2	2	2
INPUT_FIELD_SIZE	14×14	24×24	16×16

Table 7.2: Parameters used for GP training for the three databases.

percentage of the best individuals in the current population that are copied unchanged to the next generation, CROSS_RATE is the percentage of individuals in the next generation that are to be produced by crossover, MUTATION_RATE is the percentage of individuals in the next generation that are to be produced by mutation (thus ELITISM_PCNT + CROSS_RATE + MUTATION_RATE = 100%), CROSS_CHANCE_TERM is the probability that in a crossover operation two terminals will be swapped, CROSS_CHANCE_FUNC is the probability that in a crossover operation random subtrees will be swapped (thus CROSS_CHANCE_TERM + CROSS_CHANCE_ FUNC = 100%), INITIAL_MAX_DEPTH is the maximum depth of the randomly generated programs in the initial population, MAX_DEPTH is the maximum depth permitted for programs resulting from crossover and mutation operations, MAX_GENERATIONS gives the stopping condition, T, A, B and TOLERANCE are as defined in the previous section, INPUT_FIELD_SIZE is the size of the input field.

For detecting circles and squares in the easy pictures, we set 100 programs in the population, 10% of the programs (the number is *100×10%=10*) are used for reproduction (elitism), 65% of the individuals (the number is *100×65%=65*) for crossover, and 25% (*100×25%=25*) for mutation. Of the programs for crossover, 15% (*65×15%=10*) are used to swap terminals and 85% (*65×85%=55*) to swap subtrees. The programs are randomly initialised with a maximum depth of 5 at the beginning of the evolutionary process and the depth can be increased to 8 during the process. We also use 100, 50, 1000 and 4 as the constant parameters T, A, B and TOLERANCE, which are used for the program classification and the calculation of the fitness function during the evolutionary process. The maximum generation permitted for the evolutionary process is 100 for this detection problem. The size of the input field is the same as the basic approach, that is, 14×14.

7.6.2 Termination Criteria

In this approach, the learning and evolutionary process will be terminated when one of the following conditions is met:

- The detection problem has been solved, that is, all objects in each class of interest have been correctly detected with no false alarms. In this case, the fitness of the best individual program becomes zero.
- The number of generations reaches the pre-defined number, MAX_GENERATIONS.

7.7 Results

This section presents the results of a series of the experiments on the three databases described in chapter 3. The results are compared with those obtained using the basic neural network approach for object detection (chapter 4).

It is important to note that:

- The results presented here were obtained by applying the best generated programs to the entire images in the detection test set;
- No object classification results are presented here, since the genetic programming evolutionary process was applied directly to the entire images in the detection training set, rather than cutouts;
- The constant parameter T (figure 7.5, page 178) is, in a sense, equivalent to the threshold in the basic approach (figure 4.8, page 110) in that the detection performance can be changed by changing T at the test stage. However, the parameter T has to be pre-defined in the

training stage and the genetic process will search for the best program for a particular detection problem. Thus one generated program corresponds one value of T. It is not possible to set different values for T during the test stage. Accordingly, we only present the results achieved by the best programs for each of the three detection problems and compare them with the best results achieved by the basic approach for the same problem.

- Each of best generated programs in 10 runs was applied to the entire images in the detection test set of the easy and the coin pictures. Each of the five best generated programs in 5 runs was applied to the entire test images of the retina pictures due to the high computational cost. The averages are presented in this section.

7.7.1 Easy Pictures

Table 7.3 shows a comparison of the best results between the two methods. For class *class1 (black circles)* and class *class3 (grey circles)* both methods achieved a 100% detection rate with no false alarms. For class *class2 (grey squares)* the genetic programming based method also achieved 100% detection rate with 0% false alarms. However the basic approach had a false alarm rate of 91.2% at a detection rate of 100%. The average training time of the genetic programming method was less than two minutes.

Easy Pictures		Object Classes		
		Class1	Class2	Class3
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	The Basic Approach	0	91.2	0
	Genetic programming	0	0	0

Table 7.3: Comparison of the object detection results for the easy pictures: Genetic programming based approach versus the basic neural network approach. (Input field size = 14×14; Repetitions = 10.)

7.7.2 Coin Pictures

Experiments with coin pictures gave similar results to the easy pictures. These are shown in table 7.4. Detecting the heads and tails of 5 cents (class *head005, tail005*) appears to be relatively straight forward. The basic neural network approach had a 100% detection rate without any false alarms. Detecting heads and tails of 20 cent coins (class *head020, tail020*) is more difficult and the basic neural network approach resulted in many false alarms. The genetic programming based method gave the ideal results, that is, all the objects of interest were correctly detected without any false alarms for all the four object classes. However, the genetic programming method took a long time for training. The average training time was 28 hours on an eight processor ULTRA-SPARC4.

Coin Pictures		Object Classes			
		head005	tail005	head020	tail020
Best Detection Rate(%)		100	100	100	100
False Alarm Rate (%)	The basic approach	0	0	182	37.5
	Genetic programming	0	0	0	0

Table 7.4: Comparison of the object detection results for the coin pictures: Genetic programming based approach versus the basic neural network approach. (Input field size = 24×24; Repetitions = 10.)

7.7.3 Retina Pictures

The results for the retina pictures are summarised in table 7.5. Compared with the results for the previous image databases, these results are not satisfactory. However, the false alarm rate is greatly improved over the basic neural network method. The training time was quite long here (110 hours on an eight processor ULTRA-SPARC4).

Retina Pictures		Object Classes	
		haem	micro
Best Detection Rate(%)		73.91	100
False Alarm Rate (%)	The basic approach	2859	10104
	Genetic programming	1357	588

Table 7.5: Comparison of the object detection results for the retina pictures: Genetic programming based approach versus the basic neural network approach. (Input field size = 16×16; Repetitions = 5.)

The results over the three databases show similar patterns: the genetic programming based method can be used for the multiple class object detection problems and always gave a lower false alarm rate for the same detection rate.

The comparison we have shown in tables 7.3, 7.4 and 7.5 is not entirely fair since the basic approach used pixel values directly while the genetic programming approach used pixel statistics. We address this issue further in section 7.8.3, page 190.

7.8 Summary and Discussion

7.8.1 Analysis of Results on Retina Pictures

The genetic programming based approach achieved the ideal results on the easy pictures and the coin pictures, but resulted in some false alarms on the retina pictures, particularly for the detection of objects in class *haem* in which the false alarm rate was very high.

We found three main reasons why the results on the retina pictures are not as good as those on the easy and the coin pictures. Firstly, in the easy and coin pictures the background is relatively uniform, whereas in the retina pictures it is highly cluttered. Secondly, in the retina pictures, there are only two classes of interest, that is, class *micro* (micro-aneurisms) and class *haem* (haemorrhages), but there are also several other classes such as veins and other anatomical features. Thus the objects of non-interest are classified as the background. It appears that this makes the background too complex to be considered as a single class. Thirdly, in the easy and coin pictures all of the objects in a class have similar sizes whereas the sizes of the objects in each class in the retina pictures are quite different. The sizes of class *micro* vary from 3×3 to 5×5 pixels and the sizes of class *haem* vary from 7×7 to 14×14 pixels. Thus the sizes of class *micro* are relatively similar, but this is not the case for class *haem*. This might be the main reason that the results for detecting class *micro* are much better than those for class *haem*. The results also suggest that the current set of functions and terminals cannot be successfully used for detecting objects in the same class with a large variation in size. In other words, this genetic programming based approach can be successfully used for translation and rotation invariant detection problems, but not size invariant problems.

7.8.2 Analysis of the Evolved Programs

This subsection gives an analysis of the best generated programs for the three databases.

Easy Pictures

Figure 7.6 shows four sample good evolved programs for detecting the simple objects in the easy pictures. All of these programs achieved the ideal results: all the circles and squares were correctly detected with no false alarms.

```
(/ (+ (- (- (/ (* (* f3 f14) f15) (* f6 f14)) f19) (* (* (* (* f7 f10)
f17) f16) f18)) (+ (* (/ f5 f5) f14) (/ f3 f5))) (+ (* (/ f3 f5) (/ (/
f5 f6) (/ f11 f15))) (/ (/ f19 f6) (/ f5 f15))))

(- (/ f18 (- (+ f3 f15) (* f3 f5))) (* (* f4 f16) (/ (- (* f18 (+ f5
f18)) (+ (+ f7 f14) (- f10 f19))) (* f4 f16))))

(/ (/ (* (/ (- f4 (/ f5 (/ f18 f3))) (/ f18 f10)) f7) (+ (/ (- 227.013
f12) f7) (+ (/ (- (* f5 f8) f2) (* f3 f2)) (* f3 f2)))) (+ (+ (* f18
f3) (/ (/ f5 (/ f18 f1)) f7)) (+ (/ f18 (/ f18 f3)) f18)))

(* (/ (* (+ f16 f7) (* f15 f4)) (+ (- f19 (/ (* f13 f5) (/ f18 f10)))
f11)) (/ f9 (* f9 f4)))
```

Figure 7.6: Sample generated programs for simple object detection in the easy pictures.

The frequency of terminals used in the four sample generated programs for the easy pictures is shown in table 7.6. In the first evolved program, F5 was used 6 times, f3, f6, f14 and f15 were used 3 times, f19 was used twice, f7, f10, f11, f16, f17 and f18 were used once only while other features, f1, f2, f4, f8, f9, f12, f13 and f20, were not used at all. In the second program, f3, f4, f5, f16 and f18 appeared two or more times and f7, f10, f14, f15 and f19 were used once only but other features were not used. The third and the fourth generated programs gave a similar pattern to the first and the second programs: the twenty features play different roles and some features were used while other not. However, all these programs resulted in the best detection performance. This suggests that there is some redundancy in the twenty features and the four standard arithmetic operators are sufficient for detecting simple objects in the easy pictures.

No. Programs	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11
1	0	0	3	0	6	3	1	0	0	1	1
2	0	0	2	2	2	0	1	0	0	1	0
3	1	3	5	1	3	0	3	1	0	1	0
4	0	0	0	2	1	0	1	0	2	1	1

No. Programs	f12	f13	f14	f15	f16	f17	f18	f19	f20	Rand
1	0	0	3	3	1	1	1	2	0	0
2	0	0	1	1	2	0	3	1	0	0
3	1	0	0	0	0	0	7	0	0	1
4	0	1	0	1	1	0	1	1	0	0

Table 7.6: Frequency of terminals used in the four sample generated programs for the easy pictures.

Coin Pictures

In addition to the program shown in figure 7.4 (page 177), we present another generated program in figure 7.7, which performed perfectly for detecting regular objects in the coin pictures.

```
(+ (/ (/ (+ (- (- (* f10 f12) f9) f2) (- (* f12 (/ (- (* f10 f12) f9)
87.251)) (- f2 (+ (/ (- (* f12 f12) (- f17 f2)) f1) (/ 87.251 (/ f19
f5)))))) (+ (- (* f11 (- (/ f9 f16) (/ (+ (- f17 f2) (- (* f12 f12)
f11)) (* f15 (/ (/ f16 f15) f8))))) f15) f8)) (- f17 f2)) (+ (+ (/ (/
f9 (- f13 f15)) f1) (/ 87.251 (/ f19 f5)) (- (* f10 f12) f9)))
```

Figure 7.7: A sample generated program for regular object detection in the coin pictures.

Compared with those for the easy pictures, these programs are more complex, which can be interpreted as the detection problems in the coin pictures are more difficult than those in the easy pictures. Again, different terminals gave different effects in these programs. Random numbers were also used in these generated programs.

Retina Pictures

One of the five generated programs by the evolutionary process for detecting very complex objects in the retina pictures is presented in figure 7.8.

```
(* (* (- (/ f6 (+ (* (/ (* f2 (/ (* f6 (+ f1 (- f10 f15))) (- (- f18
f17) (- f19 87.0518)))) (+ 17.0792 (+ f9 f14))) (/ (+ f19 (* (+ (+ f11
(- (* (- (- f15 f18) (+ 40.5811 f16)) (- (* f13 (+ (/ 57.6382 f16)
f13)) (- f9 f6))) (/ (* f3 f1) f1))) (* (- (* (- (/ (+ (+ f18 (+ (/ (/
f14 f6) (+ f6 f1)) 89.7037)) (* f10 f12)) f2) f9) (+ (+ f16 14.7513)
f9)) f18) (/ (/ f13 f1) (* (+ f6 f12) f9)))) (+ f16 f8))) (+ (- (- (+
(/ f10 (* f9 f6)) f13) f10) f18) (+ (* (- (+ f1 f2) (+ f17 f8)) f5) (*
(* f20 f16) f10))))) (* (+ (- (* (+ f11 (+ (* f14 f3) (/ f15 (/ (+ (*
f2 14.5251) (* (* (/ (* f18 (/ (* f2 f13) f15)) f1) (/ (/ f11 f13) (/
f7 f5))) (+ (+ f18 (* f2 f13)) (/ f8 f12)))) f17)))) f11) f16) (* (-
f1 (+ f3 f8)) f5)) (/ (+ (- f7 f20) f18) f20)))) (* (* (* (* f2 f13)
f2) (/ (* f4 (/ (* f2 f13) f15)) (* f18 f12))) (* f14 f2))) (+ (+ (-
(+ (- f19 f3) f2) f7) (- (+ f8 f17) f18)) (/ (+ f15 60.1046) (* (* f1
(/ (/ f12 (- (+ (/ (/ f12 f13) (/ f15 f5)) f17) f18)) (/ f7 f5)))
f8)))) (* (/ (* f10 (/ (* f2 f13) f15)) f18) (* (* (* (* f2 f2) (/ (/
(/ f18 (+ f1 f2)) f13) (/ (/ (- f15 96.16) (* f4 14.5251)) f5))) f4)
(/ (/ f12 f13) (/ f1 (+ (/ f10 f1) f4))))))
```

Figure 7.8: A sample generated program for very difficult detection problems in the retina pictures

As can be seen from figure 7.8, this program is much more complex in both the length and the number of image features than those generated for the easy pictures and the coin pictures. As discussed earlier, this program resulted in a better performance than the neural networks, but still gave a quite high number of false alarms. This is due mainly to the difficulty of the detection problem in the retina pictures particularly the size variation and the highly cluttered background. All the 20 features and 8 random floating point numbers were used in this program, but the frequencies of these terminals occurring in the program were quite different. This suggests that the 20 features might not be sufficient for such difficult detection problems, and different features have different effects in the evolved programs.

Summary

Table 7.7 summarises the relationship between the 10, 10 and 5 generated programs for the easy, the coin and the retina pictures and the 20 features. For example, f1 was used in three of ten generated programs for the easy pictures, six of the ten generated programs for the coin pictures and all the five generated programs in the retina pictures.

Features/Terminals	Number of programs using features		
	Easy Pictures	Coin Pictures	Retina Pictures
f1	3	6	5
f2	4	8	5
f3	9	7	5
f4	6	9	5
f5	10	8	5
f6	5	5	5
f7	8	4	5
f8	3	6	5
f9	2	5	5
f10	7	3	5
f11	5	6	5
f12	4	8	5
f13	2	3	5
f14	5	6	5
f15	6	5	5
f16	6	4	5
f17	1	3	5
f18	9	8	5
f19	6	1	5
f20	0	2	5
Rand	2	6	5

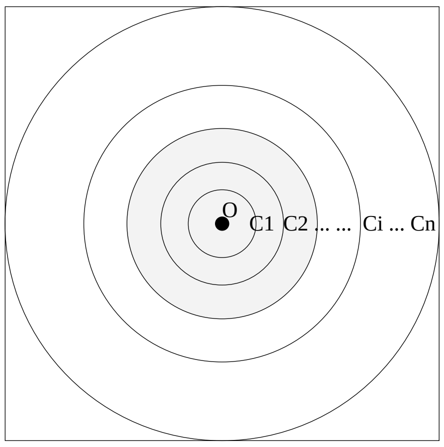
Table 7.7: Summary of the frequency of terminals used in the generated programs.

In general, the programs generated by the evolutionary process for the three detection problems are difficult to interpret. However, they are similar in the following points:

- The twenty features have different influences in a generated program. A feature has different influences in different generated programs.
- The genetic programming evolutionary process can automatically select the features (terminals) that are relevant to a particular domain.
- The twenty features and four functions are sufficient for detecting simple objects and regular objects against a relatively uniform background in the easy and the coin pictures, but might not be enough for detecting complex, irregular objects against a highly cluttered background in the retina pictures.

7.8.3 Further Experiments

Experiments with an Alternative Terminal Set



(a)

Features		local boundaries
mean	standard deviation	
F1	F2	central pixel
F3	F4	circle boundary C1
F5	F6	Circle boundary C2
...	...	... ...
F(2i+1)	F(2i+2)	Circle boundary Ci
...	...	... ...
F(2n+1)	F(2n+2)	Circle boundary Cn

(b)

Figure 7.9: The input field and the image boundaries for feature extraction in constructing terminals.

We experimented with an alternative set of terminals based on a circular set of features, as shown in figure 7.9. The features were computed based on a series of concentric circles centred in the input field. This terminal set focused on boundaries rather than regions. The number of features in this case depends on the size of the input field. For instance, if the input field is 19×19 pixels, then the number of central circles will be $19/2 + 1 = 10$ (the central pixel is considered as a circle with a zero radius); accordingly, there would be 20 features. The gap between the radii of two neighbouring circles is one pixel.

As with the square feature set, this terminal set gives ideal results for the easy pictures. In the coin and retina pictures, however, the results based on this terminal set are slightly worse than those obtained with the square feature set. This suggests that the local region features are better for these detection problems than these boundary features.

Experiments with an Alternative Function Set

We also experimented with a different function set. We hypothesised that convergence might be quicker if the function values were close to the range (-1,1). We used *dabs, sin* and *exp*, (absolute value, sine and exponent to base e) in addition to the four arithmetic operators (*+, -, ** and */*). The detection results are similar to those based on the function set of only the four arithmetic operators. Convergence was slightly faster for training the coin and retina pictures, but slightly slower for the easy pictures. This suggests that *dabs, sin* and *exp* may be useful for more difficult problems, however considerably more experimentation is required.

Experiments with "Pixel" Inputs

We also investigated this approach with "pixels" as terminals. To decrease the computation cost, we considered a 2×2 square as a single "pixel". In this way, there are 49 (7×7), 144 (12×12), and 64 (8×8) terminals, which correspond to the sizes of the input fields of 14×14, 24×24, and 16×16, for the easy, the coin, and the retina pictures respectively. For the easy pictures, the learning took about 70 hours on an 8 processor ULTRA-SPARC4 machine to reach perfect detection performance and 78 generations were needed. The population size used was 500, the maximum depth of the program was 30, the maximum initial depth 10, the maximum number of generations 100. For the coin pictures and the retina pictures, the situation was worse. Since a large number of terminals were used, the maximum depth of the program trees was increased to 50 for the coin pictures and 60 for the retina pictures. The population size for both databases used was 1000 with a maximum number of generations of 100. The evolutionary process took three weeks to complete 50 generations for the coin pictures and four weeks to complete 50 generations for the retina pictures. The best detection results were 90% false alarm rate at a 100% detection rate for the coin pictures, and about 850% false alarm rate at a detection rate of 100% for micro aneurisms in the retina pictures.

These results indicate that using pixel statistics is better than using pixels directly. This in turn suggests that it might be necessary to investigate whether the neural network results will be improved by pixel statistics in the future.[1]

7.8.4 Limitations

The genetic programming based approach has the following limitations:

- The training times for the coin problem and the retina problem are quite long. Some of the runs took longer than 48 hours on a 8 processor ULTRA-SPARC4.
- The method is not particularly effective in detecting objects with different sizes which are in the same class, for example the haemorrhages in the retina pictures. This might be related to insufficient image features being used. More features which contain size invariant information need to be investigated.
- The classification strategy employed in the generated programs is not easy to determine.
- Some experimentation is required to find good values of the various parameters for each different problem.
- A threshold T (figure 7.5) was used to give fixed size ranges for determining the class of the object from the output of the program. It is not clear how T should properly be chosen.

7.8.5 Summary

The goal of this chapter was to extend the basic approach to a genetic programming based approach for detecting small objects of multiple classes in large pictures and to investigate the improvements in detection performance of the method over the basic approach described in chapter 4. In the basic approach, the neural networks were trained by the backward error propagation algorithm on cutouts in the classification data set constructed from the detection training set, and tested on the entire images in the detection test set. In this approach, the programs were directly trained by the genetic programming process on the entire images in the detection training set and the locations of the known objects of interest in each class. Twenty domain independent, pixel level image features were selected to form a general, pre-existing feature library to be the terminal set. The four standard arithmetic operators were used as the function set. A program classification map was developed for classifying the detected object

[1]Subsequent work by Rai [150] which used the 20 pixel statistics as inputs to neural networks showed that pixel statistics gave significantly larger error rates.

in each pixel position by the evolved program. The fitness function was based on a linear combination of the detection rate and false alarm rate of the evolved program. After the learning process was finished, the best individual obtained was directly applied to the entire images in the detection test set. In all cases on the easy and medium difficulty detection problems in the easy and the coin pictures the genetic programming based approach produced ideal results, that is, all the objects of interest in every class were correctly detected without any false alarms. For *micro* and *haem* detection in the very difficult retina pictures, the genetic programming method resulted in much better performance than the basic neural network approach. These results suggest that the genetic programming based approach with pixel statistics is viable as a domain independent, learning/adaptive approach for multiple class object detection problems.

Chapter 8

Conclusions

In this chapter, we first summarise the object detection results achieved using the methods described in this thesis. We then examine the hypotheses and present the conclusions. Finally, some related future work is suggested.

8.1 Summary of the Detection Results

In this thesis, six object detection methods were introduced and described. They are the basic approach (*BP-train*), the central weight initialisation method, the *GA-train* method, the *BP-train+GA-refine* method, the *GA-train+GA-refine* method and the genetic programming based method. The best detection results, that is, the best detection rate and the corresponding false alarm rate, for the easy, the coin and the retina pictures achieved by the six detection methods are summarised in this section.

Easy Pictures

Table 8.1 compares the best results achieved by the six detection methods for the three object classes of interest in the easy pictures: *class1* (black circles), *class2* (grey squares) and *class3* (white circles).

As shown in table 8.1, all the six detection methods described in this thesis successfully detected all the objects of the three classes (detection rate = 100%). For detecting *class1*, the *GA-train* method resulted in 80% false alarm rate at a detection rate of 100%, while all other methods did not produce any false alarms. Detecting *class3* is similar: most of the six methods gave ideal results while the *GA-train* method and the *GA-train+GA-refine* method led to 273% and 144% false alarm rates at a detection rate of 100%. However, it is clear that the *GA-*

Easy Pictures		Object Classes		
		Class1	Class2	Class3
The Best Detection Rate(%)		100	100	100
Corresponding False Alarm Rate (%)	The Basic Approach (BP-train)	0	91.2	0
	Centred Weight Initialisation	0	46.4	0
	GA-train	80	839	273
	GA-train + GA-refine	0	663	144
	BP-train + GA-refine	0	0	0
	Genetic Programming	0	0	0

Table 8.1: Comparison of the best detection results achieved using the six detection methods for the easy pictures.

train+GA-refine method is superior to the *GA-train* method. Detecting *class2* is a relatively difficult problem. Only the genetic programming based method and the *BP-train+GA-refine* method achieved ideal performance and other methods produced different numbers of false alarms. The *GA-train* method resulted in a large number of false alarms. The false alarm rate was 839%. The method with the network refinement *GA-train+GA-refine* was better, the false alarm rate was 663%. The basic approach resulted in a 91.2% false alarm rate at a detection rate of 100%, while the centred weight initialisation improved the detection performance by decreasing the false alarm rate to 46.4%. For all the three classes in the easy pictures, the *BP-train+GA-refine* method and the genetic programming based method always led to ideal performance, that is, all the objects of interest were correctly detected with no false alarms.

Coin Pictures

Table 8.2 shows a comparison of the results for the coin pictures. The best detection rate and the corresponding false alarm rate for the four object classes of interest, that is, class *head005* (head side of 5 cent coins), class *tail005* (tail side of 5 cent coins), class *head020* (head side of 20 cent coins) and class *tail020* (tail side of 20 cent coins) are presented in this table.

As shown in table 8.2, detecting heads and tails of 5 cent coins was relatively easy. The basic approach (*BP-train*), centred weight initialisation method, the *BP-train+GA-refine* method and the genetic programming based method always resulted in ideal results where all the objects

Coin Pictures		Object Classes			
		head005	tail005	head020	tail020
The Best Detection Rate(%)		100	100	100	100
Corresponding False Alarm Rate (%)	The Basic Approach (BP-train)	0	0	182	37.5
	Centred Weights Initialisation	0	0	41.4	0
	GA-train	357	19	333	778
	GA-train+GA-refine	125	0	37.5	215
	BP-train+GA-refine	0	0	0	0
	Genetic Programming	0	0	0	0

Table 8.2: Comparison of the best detection results achieved using the six detection methods for the coin pictures.

of interest were correctly detected with no false alarms. For detecting class *head005* objects, the *GA-train* method produced 357% false alarm rate while the *GA-train+GA-refine* method resulted in a better performance, that is, 125% false alarm rate. For class *tail005*, the *GA-train* method led to a 19% false alarm rate at the detection rate of 100%, while the *GA-train+GA-refine* achieved the ideal results.

Detecting heads and tails of 20 cents was more difficult. However, the results show a similar pattern to those for heads and tails of 5 cents:

- The basic approach, could successfully detect these regular objects against a relatively uniform background, but it produced a number of false alarms for *head020* and *tail020*.
- The centred weight initialisation produced a lower false alarm rate than the basic approach for all classes in this database.
- The *GA-train* method resulted in more false alarms for any class in this database than the basic approach.
- The genetic algorithm for network refinement, *GA-refine*, always improved the detection performance achieved by the *GA-train* method or the basic approach for all the four classes in this database.

- The genetic programming based approach and the *BP-train+GA-refine* method always resulted in ideal detection performance for all the object classes of interest in this database.

Retina Pictures

Table 8.3 compares the best results for detecting very irregular, complex objects against a highly cluttered background in the retina pictures. In this database, there are only two classes of interest: *micro* (micro aneurisms) and *haem* (haemorrhages); however there are also some other classes of non-interest such as *veins* and other eye anatomy.

Retina Pictures	Object Classes	
	micro	haem
Detection Results	Best DR (%)/FAR(%)	Best DR(%)/FAR(%)
BP-train(the basic approach)	100 / 10104	73.91 /2859
Centred Weight Initialisation	100 / 2903	73.91 /1924
GA-train	90 / 5606	50.37 /4000
BP-train+GA-refine	100 / 2706	82.61 / 2156
GA-train+GA-refine	100 / 5055	82.61 / 2298
Genetic Programming	100 / 588	73.91 /1357

Table 8.3: Comparison of the best object detection results achieved using the six detection methods for the retina pictures. (DR: Detection Rate. FAR: False Alarm Rate)

For detecting class *micro* objects, even if most of the six methods successfully detected all the objects of interest (the *GA-train* method is an exception, which only detected 90% of these objects), they also resulted in a large number of false alarms. The basic approach, for example, produced 100 times more false alarms than the desired objects in the database (false alarm rate was 10104%) at a detection rate of 100%. The genetic programming based approach improved the performance by 17 times as the false alarm rate was decreased to 588% at the same detection rate.

Detecting class *haem* objects was even more difficult, where none of the six methods successfully detected all the objects of this class. The best detection rate is 82.61%, which was achieved by the two phase methods with the refinement genetic algorithm. As analysed earlier, this is due mainly to the large variation in the size of the objects in this class (section 7.8.1, page 184).

These results also show that:

- The basic approach produced a large number of false positives for these classes, which suggests that it can not be successfully applied to very difficult detection problems with very complex, irregular objects against a highly cluttered background.
- The centred weight initialisation method always improved the detection performance for all classes in this database.
- The genetic algorithm used for network training without network refinement did not detect all the objects in class *micro* (the detection rate was about 90%), but resulted in less false alarms than the basic approach.
- The methods with the refinement genetic algorithm always produced a lower false alarm rate for each class than the corresponding methods without the refinement.
- The genetic programming approach greatly improved the false alarm rate for detecting class *micro* at a detection rate of 100%.
- None of the six methods detected all the objects in class *haem* in this database, which suggests that these methods could not be successfully applied to the detection problems with a large variation in size of the objects for the same class.

8.2 Conclusions

The overall goal of this thesis was to investigate a learning/adaptive, domain independent approach to multiple class, translation and limited rotation invariant object detection problems. This goal has been achieved for the easy and medium difficulty detection problems, that is, simple object detection against a uniform background in the easy pictures and regular, medium complexity object detection against a relatively uniform background in the coin pictures. The methods that gave the best detection performance for the easy and the medium difficulty detection problems were the genetic programming based method and the refinement genetic algorithm based on the network trained by the backward error propagation algorithm, *BP-train+GA-refine*. Both of them achieved ideal performance on these detection problems. For the very difficult detection problems, that is, very irregular and complex object detection against a highly cluttered background in the retina pictures, the results obtained in this thesis, while not satisfactory, were consistent with results obtained by alternative methods on similar problems. The method that gave the best results on the very difficult, translation and limited rotational invariance detection problems in the retina pictures was the genetic programming based method.

We expect that the methods described in this thesis could be successfully used for any databases similar to the easy and the coin pictures, that is, detecting multiple class objects with translation invariance and limited rotational invariance and with the size less than or equal to 24×24 pixels against a relatively uniform background.

Specifically, the theoretical research and experimental results have led to the following conclusions.

The five hypotheses addressed in chapter 1 were examined in the course of the investigation:

1. *Hypothesis 1: Can the basic pixel based neural network approach be applied to multiple class object detection problems?*

   For the three databases described in this thesis, the basic approach was successfully applied to the easy and the coin databases but not for the retina pictures. The basic approach could successfully detect all the objects with translation and rotation variation in each class described in this thesis, however with a number of false alarms for some detection problems, particularly for detecting very irregular, complex objects against a highly cluttered background.

2. *Hypothesis 2: Can the centred weight initialisation algorithm be used to improve the network training and the detection performance over the basic approach?*

   On all of the detection problems in the easy, the coin and the retina databases, it was always possible to find centred initial weights which resulted in fewer training epochs and lower test mean squared error than the basic approach with random weight initialisation. More importantly, the centred weight initialisation method produced networks which were much better, in terms of detection performance, at the task of detecting multiple class objects of interest in large pictures. The amount of the improvement did not appear to be related to the difficulty of the problem. Overall, the detection performance for the easy and medium difficulty problems was good. On the very difficult detection problems the detection rate was good but the false alarm rate was still quite high.

   Visualisation of the weights in trained networks resulting from both initialisation methods revealed that the trained networks from both approaches contained feature detectors which "made sense" for the problem domain and that learning in networks with the centred initial weights was more focused on features which discriminated between classes.

3. *Hypothesis 3: Can a genetic algorithm with the fitness of mean squared error result in an improvement of network training and object detection performance over the backward error propagation algorithm?*

   The genetic algorithm with the fitness of mean squared error for network training, the *GA-train* algorithm, was successfully applied to training the networks for classifying the cutouts in the three databases described in this thesis. It did not, however, result in any improvement of network training and testing performance, either in terms of faster training speed or lower test error, over the backward error propagation algorithm for the same problems. As an independent detection method, it did not improve the detection performance over the basic approach on most of the object detection problems in the three databases described in this thesis.

   The networks used here were considerably larger than those previously used with the *GA-train* (formerly called *2Delta-GANN*) algorithm.

4. *Hypothesis 4: Can network refinement using a genetic algorithm, with the fitness based on the detection rate and the false alarm rate, improve the detection performance of the networks trained on the object cutouts?*

   For all the detection problems described in this thesis, the refinement genetic algorithm markedly improved the detection performance achieved by the networks trained on the cutouts by the corresponding training algorithm without the refinement. The *BP-train+GA-refine* method achieved better detection results than the *BP-train* method and the *GA-train+GA-refine* method performed better than the *GA-train* method.

5. *Hypothesis 5: Can genetic programming be applied to multiple class object detection and produce an improvement of detection performance over the basic approach on the same detection problems?*

   The genetic programming paradigm has been successfully applied to multiple class object detection problems and a genetic programming based approach has been developed for this purpose.

   On all detection problems described in this thesis, the genetic programming based approach markedly improved the detection performance over the basic neural network approach. This approach led to ideal performance for all detection problems in the easy and the coin pictures. In particular, it greatly improved the performance for the detection problems in the retina pictures.

One of the major differences between the genetic programming method and all neural network based detection methods described in this thesis is that these neural network based methods used raw pixel data as inputs, while the genetic programming based method used 20 pixel statistics, that is, 20 domain independent, pixel level features as inputs. The fact that genetic programming based method achieved the best results might be related to this difference. It is necessary to investigate whether better performance can be achieved if the 20 pixel statistics are applied to these neural network based methods. We suggest this as a future research direction.

Although they are not primarily related to the hypotheses, the following observations have been also resulted from this thesis.

1. Compared with all other detection methods described in this thesis, the genetic programming based approach achieved the best results for all the translation and rotation invariant object detection problems. For all the object detection problems described in this thesis except for class *haem* (in which the objects have a big variation in size), the genetic programming based approach was the best, followed by *BP-train+GA-refine* and centred weight initialisation, and then the *GA-train+GA-refine*, the basic method and the *GA-train* method.

2. The detection method *BP-train+GA-refine* achieved the ideal results for detecting the easy and coin pictures, that is, this method successfully detected all the objects of interest with no false alarms on all the detection problems in the easy and the coin pictures. It also resulted in better detection performance on the difficult detection problems in the retina pictures than all methods other than the genetic programming based approach, including the basic approach, centred weight initialisation method, the *GA-train* method and the *GA-train+GA-refine* method.

3. The methods described here were not particularly effective in detecting haemorrhages in retina pictures, which suggests that they can not be used to detect objects with different sizes which are in the same class.

4. In all the methods described in this thesis, it is necessary to find good values for the various parameters. This generally has to be done through empirical search for each detection problem.

5. For developing a detection system for a completely new image database, the genetic programming based method requires the least work of all the detection methods described in this thesis.

6. The genetic algorithm for network refinement and the genetic programming approach involve long training times. However, this could be greatly improved with parallel hardware. The centred weight initialisation method has the shortest training time.

7. The genetic programming approach, which generates computer programs after training, has a shorter detection time in general than neural network related methods, since there are sigmoid and other functions and all the pixel inputs in the feed forward pass while there are only simple arithmetic operators and fewer pixel statistics inputs in the genetic programming approach. Under the same network architecture, all neural network related methods have the same object detection time.

8. The size of training set used for network training is consistent with the literature, that is, networks can be successfully trained using fewer training examples than those suggested by the learning theory (section 2.3.5, page 34).

8.3 Future Work

Several potential research directions or topics were suggested during the investigation of this thesis.

1. In the basic approach and the two phase approaches raw image pixel data were used as inputs to neural networks while pixel statistics were used in genetic programming base approach. Investigation of using the twenty domain independent, pixel level features as inputs to the networks is an interesting direction and would reveal whether good performance of the genetic programming approach was due to these pixel statistics.

2. For the centred weight initialisation algorithm, it might be also interesting to investigate an alternative way of initialising the weights: set the big values for the weights corresponding the the edges of the input field and small value for the weight in the centre.

3. In the centred weight initialisation method, the range of the biggest centred weight needs to be properly set to obtain fast training speed and good detection performance. Ways or

rules for quickly setting the biggest value for the centred weight need to be investigated and whether or not a general starting point for different problems can be found.

4. In the genetic algorithm related detection methods – *GA-train*, *BP-train+GA-refine* and *GA-train+GA-refine*, the network weights are randomly initialised before the network training starts. The investigation of integrating the methods with the centred weight initialisation algorithm is another interesting direction.

5. In the genetic programming based approach, a threshold T was used to give fixed size ranges for determining the class of the objects from the output of an evolved program. It might be very interesting to investigate the ways of finding individual thresholds for different classes.

6. In the genetic programming based method, twenty local region features are used to form the general pre-existing feature library in the terminal set. This seems to be sufficient for the easy and medium difficulty detection problem. It would be interesting to try a larger pre-existing feature library as the terminals, especially for the difficult detection problems.

7. In the genetic algorithm for network refinement and the genetic programming based approach, the training times are quite long. Ways of shortening the training time need to be investigated.

8. It might be also interesting to investigate the improvement of the object detection performance by rotating the cutouts in the classification data set to produce more examples for network training.

Bibliography

[1] H. Abdi. A generalized approach for connectionist auto-associative memories: interpretation, implications and illustration for face processing. In J. Demongeot, T. Herve, V. Rialle, and C. Roche, editors, *Artificial Intelligence and Cognitive Sciences*, chapter 6, pages 149–165. Manchester University Press, 1988.

[2] S. Ahmad and S. Omohundro. A network for extracting the locations of point clusters using selective attention. In *the 12th Annual Conference of the Cognitive Science Society*, MIT, July 1990. Also in Technical Report #90-011.

[3] S. Ahmad and V. Tresp. Some solutions to the missing feature problem in vision. In C. J. Hanson S.J. and G. C.L., editors, *Advances in Neural Information Processing Systems*, chapter 5. Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[4] J. H. Ahrens and U. Dieter. Extensions of Forsythe's method for random sampling from the normal distribution. *Math. Comput.*, 27(124):927–937, Oct. 1973.

[5] J. T. Alander. An indexed bibliography of genetic algorithms and neural networks. Technical Report 94-1-NN, University of Vaasa, Department of Information Technology and Production Economics, September 1998. (ftp.uwasa.fi: cs/report94-1/gaNNbib.ps.Z).

[6] D. Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinnear, editor, *Advances in Genetic Programming*, pages 477–494. MIT Press, 1994.

[7] R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.

[8] N. Ansari and K. Li. Landmark-based shape recognition by a modified Hopfield neural network. *Pattern Rocognition*, 26(4):531–542, 1993.

[9] M. Anthony and N. Biggs. *PAC learning and neural networks*, pages 694–697. In The Handbook of Brain Theory and Neural Networks, 1995.

[10] T. Back, U. Hammel, and H. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, April 1997.

[11] J. Bala, K. D. Jong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognising visual concepts. *Evolutionary Computation*, 4(3):297–312, 1997.

[12] K. Balakrishnan and V. Honavar. Evolutionary design of neural architectures – a preliminary taxonomy and guide to literature. Technical Report CS TR 95-01, Department of Computer Science, Iowa State University, Ames,Ioma 50011-1040, USA, Jan 1995.

[13] D. H. Ballard and C. M. Brown. *Computer Vision.* Englewood Cliffs, N.J: Prentice-Hall, Inc., 1982.

[14] S. Bannour and M. R. Azimi-Sadjadi. Principal component extraction using recursive least squares learning method. *IEEE transactions on neural networks*, 6:457–469, March 1995.

[15] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications.* San Francisco, Calif. : Morgan Kaufmann Publishers; Heidelburg : Dpunkt-verlag, 1998. Subject: Genetic programming (Computer science); ISBN: 1-55860-510-X.

[16] Y. Bar-Shalom and X. R. Li. *Estimation and Tracking.* Artech House, 1993.

[17] P. L. Barlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *Journal of Evolutionary Economics*, 3:1–22, 1998.

[18] E. B. Baum and D. Haussler. What size net gives valid generalisation? *Neural Computation*, 1:151–160, 1989.

[19] R. K. Belew, J. McInerney, and N. N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. Technical Report CS 90-174, University of California, San Diego, 1990. Also in Artificial Life II, pages 511–547, Addison-Wesley, 1992.

[20] A. Bernardon and J. E. Carrick. A neural system for automatic target learning and recognition applied to bare and camouflaged SAR targets. *Neural Networks*, 8(7/8):1103–1108, 1995.

[21] H. Bosch, R. Milanese, and A. Labbi. Object segmentation by attention-included oscillations. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1167–1171, Anchorage, Alaska, 1998. 0-7803-4859-1/98, IEEE.

[22] F. Z. Brill, D. E. Brown, and W. N. Martin. Fast genetic selection of features for neural network classifiers. *IEEE Transactions on Neural Networks*, 3(2):324–328, March 1992.

[23] R. Brunelli and T. Poggio. Face recognition through geometrical features. In S. M. Ligure, editor, *Proceedings of ECCV '92*, pages 792–800, 1992.

[24] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on PAMI*, 15(10):1042–1052, 1993.

[25] L. L. Burton and H. Lai. Active sonar target imaging and classification system. In *Proceeedings of the SPIE International Symposium on Aerospace/Defence Sensing and Control*, pages 19–33, Orlando, FL, April 1997.

[26] T. Caelli and W. F. Bischof. *Machine Learning and Image Interpretation.* Plenum Press, New York and London, 1997. ISBN 0-306-45761-X.

[27] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organising neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[28] G. A. Carpenter and S. Grossberg. Stable self-organisation of pattern recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[29] D. P. Casasent and L. M. Neiberg. Classifier and shift-invariant automatic target recognition neural networks. *Neural Networks*, 8(7/8):1117–1129, 1995.

[30] K. Chellapilla and D. B. Fogel. Evolution, neural networks, games, and intelligence. *Proc. IEEE*, 87(9):1471–1498, 1999.

[31] R. Chellappa, K. Fukushima, A. K. Katsaggelos, S.-Y. Kung, Y. LeCun, N. M. Nasrabadi, and T. A. Poggio. Applications of artificial neural networks to image processing. *IEEE Transactions on image processing*, 7(8):1093–1096, Augest 1998.

[32] V. Cherkassky and F. Mulier. Vapnik-Chervonenkis (VC) learning theory and its applications. *IEEE Transactions on Neural Networks*, 10(5):985–987, Spetember 1999.

[33] Y. Chigawa, O. Hasegawa, and K. Nakayama. Handwritten digit recognition by using feature extracting neural network and modified self-organising feature map. In *Report of Technical Meeting on Neural Computing*, pages 15–22, Japan, Oct 1991. IEICE. Vol. NC91-50.

[34] V. Ciesielski and J. Riley. An evolutionary approach to training feed forward and recurrent neural networks. In L. C. Jain and R. K. Jain, editors, *Proceedings of the Second International Conference on Knowledge Based Intelligent Electronic Systems*, pages 596–602, Adelaide, Apr. 1998.

[35] V. Ciesielski and M. Zhang. Using genetic algorithms to improve the accuracy of object detection. In N. Zhong and L. Zhou, editors, *Knowledge Discovery and Data Mining – Research and Practical Experiences*, pages 19 – 24, Beijing, China,, April 1999. Tsinghua University Press. Proceedings of the third Pacific-Asia Knowledge Discovery and Data Mining Conference.

[36] V. Ciesielski and J. Zhu. A very reliable method for detecting bacterial growths using neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 62–67, Beijing, November 1992.

[37] V. Ciesielski, J. Zhu, J. Spicer, and C. Franklin. A comparison of image processing techniques and neural networks for an automated visual inspection problem. In *Proceedings of the 5th Joint Australian Conference on Artificial Intelligence*, pages 147–152, Hobart, Tasmania, 1992. World Scientific.

[38] A. G. Constantinides, S. Haykin, Y. H. Hu, J.-N. Hwang, S. Katagiri, S.-Y. Kung, and T. A. Poggio. Special issue on neural networks. *IEEE Transactions on signal processing*, 45(11), Nov 1997.

[39] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(4):297–301, 1965.

[40] D. Corne, H. L. Fang, and C. Mellish. Solving the modular exam scheduling problem with genetic algorithms. In *Proc. of 6th Int'l Conf. on Industrial and Engineering Applications*

of Artificial Intelligence and Expert Systems, pages 370–373. Gordon and Breach Science Publishers, 1993.

[41] N. Cross and R. Wilson. Neural networks for object recognition. Technical report, Department of Computer Science, University of Warwick, Coventry, October 1995.

[42] D. de Ridder. Shared weights neural networks in image analysis. Master's thesis, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands, Feb 1996.

[43] D. de Ridder, A. Hoekstra, and R. P. W. Duin. Feature extraction in shared weights neural networks. In *Proceedings of the Second Annual Conference of the Advanced School for Computing and imaging, ASCI*, pages 289–294, Delft, June 1996.

[44] K. DeJong and W. Spears. On the virtues of parameterised uniform crossover. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, July 1991. Morgan Kaufman.

[45] P. Del Moral and A. Guionnet. Large deviations for interacting particle systems: Applications to non linear filtering problems. *Stochastic Processes and their Applications*, 78:69–95, 1998.

[46] T. Dillon, P. Arabshahi, and R. J. Marks, II. Everyday applications of neural networks. *IEEE Transactions on Neural Networks*, 8(4):825–826, 1997. Special Issue on Everyday Applications of Neural Networks.

[47] P. S. Dunston, S. "Ranji", and L. E. Bernold. Neural network model for the automated control of springback in rebars. *IEEE Expert – Intelligent system and their applications*, 11(4):45–49, August 1996.

[48] O. Faugeras. *Three-Dimensional Computer Vision – A Geometric Viewpoint.* The MIT Press, 1993. ISBN 0-262-06158-9.

[49] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, IEEE Neural Network Council, Inc., New York, 1995.

[50] D. B. Fogel. Evolutionary computation: A new transactions. *IEEE Transactions on Evolutionary Computation*, 1(1):1–2, April 1997.

[51] D. B. Fogel. A message from the program committee chair. In *The fifth IEEE International Conference on Evolutionary Computation*, 1998. WCCI'98.

[52] J. Fontanari and R. Meier. Evolving a learning algorithm for the binary perceptron. *Network*, 2:353–359, November 1991.

[53] R. Friedberg. A learning machine, Part I. *IBM Journal of Research and Development*, 2:2–13, 1958.

[54] R. Friedberg, B. Dunham, and J. North. A learning machine, Part II. *IBM Journal of Research and Development*, 3:282–287, 1959.

[55] M. Fuchs. Crossover versus mutation: An empirical and theoretical case study. In J. R. Koza, W. Banzhaf, and et al., editors, *Genetic Programming 1998, Proceedings of the third annual conference*, pages 78–85, University of Wisconsin, Madison, July 1998. Morgan Kaufmann Publisher.

[56] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Publisher, New York, 1990.

[57] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

[58] K. Fukushima, E. Kimira, and H. Shouno. Neocognitron with improved bend-extractors. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1172–1175, Anchorage, Alaska, 1998. 0-7803-4859-1/98, IEEE.

[59] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[60] P. D. Gader, J. R. Miramonti, Y. Won, and P. Coffield. Segmentation free shared weight neural networks for automatic vehicle detection. *Neural Networks*, 8(9):1457–1473, 1995.

[61] M. N. Gibbs and D. J. C. MacKay. Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, November 2000.

[62] C. L. Giles, R. Sun, and J. M. Zurada. Neural networks and hybrid intelligent models: Foundations, theory, and applications. *IEEE Transactions on Neural Networks*, 9(5):721–723, 1998. Special Issue on Neural networks and hybrid intelligent models.

[63] G. L. Giles and T. Maxwell. Learning, invariances, and generalisation in high-order neural networks. *Applied Optics*, 26(23):4972–4978, 1987.

[64] D. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine learning.* Addison Wesley, Reading, Ma, 1989.

[65] A. L. Gorin and R. J. Mammone. Introduction to the special issue on neural networks for speech processing. *IEEE Transactions on Speech and Audio Processing*, 2(1):113–114, 1994.

[66] E. Gose, R. Johnsonbaugh, and S. Jost. *Pattern Recognition and Image Analysis.* Prentice Hall PTR, Upper Saddle River, NJ 07458, 1996. ISBN 0-13-236415-8.

[67] J. Y. Goulermas and P. Liatsis. Genetically fine-tuning the Hough transform feature space, for the detection of circular objects. *Image and Vision Computing*, 16:615–625, 1998.

[68] K. Harries and P. Smith. Exploring alternative operators and search strategies in genetic programming. In J. R. Koza, W. Banzhaf, and et al., editors, *Proceedings of the second international conference on Genetic Programming (GP-97)*, pages 147–155. Morgan Kaufmann Publisher, 1997.

[69] C. Harris. *An Investigation into the Application of Genetic Programming Techniques to Signal Analysis and Feature Detection.* PhD thesis, the University of London, London, UK, September 1997. Department of Computer Science, University College London.

[70] N. R. Harvey and S. Marshal. The use of genetic algorithm in morphological filter design. *Signal Processing: Image Communication*, 8:55–71, 1996.

[71] S. Hayes and V. Ciesielski. A neural network approach to positional and rotational invariant recognition of 3D objects. Technical report, Royal Melbourne Institute of Technology, Melbourne, Victoria, Australia, 1997.

[72] S. Haykin. *Adaptive Filter Theory.* Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

[73] R. Hecht-Nielsen. Neural networks and image analysis. In Carpenter and Grossbury, editors, *Neural networks for vision and image processing*, chapter 17, pages 449–460. MIT Press, 1992.

[74] M. Hilario. An overview of strategies for neurosymbolic integration. In *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, chapter 2. Kluwer Academic Publishers, 1997.

[75] M. Hilario, C. Pellegrini, and F. Alexandre. Modular integration of connectionist and symbolic processing in knowledge-based system. In *International Symposium on Integrating Knowledge and Neural Heuristics*, pages 123–132, Pensacola, Florida, May 1994.

[76] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989. Reprinted in J. Carbonell, editor, "Machine Learning: Paradigms and Methods", MIT Press, 1990. Also appears as Technical Report CMU-CS-87-115 (version 2), Carnegie Mellon University, Pittsburgh, PA, December 1987.

[77] S. Holden and M. Niranjan. On the practical applicability of VC dimension bounds. Technical Report CUED/F-INFENG/TR.155, Cambridge University Engineering Department, Cambridge CB2 1 PZ, Octobor 1994.

[78] S. Holden and M. Niranjan. On the practical applicability of VC dimension bounds. *Neural Computation*, 7(6):1265–1288, 1995.

[79] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor : University of Michigan Press; Cambridge, Mass. : MIT Press, 1975.

[80] J. J. Hopfield. Neural networks and physical system with emergent collective computational abilities. In *Proceedings of National academy of Sciences, USA79, vol. 2*, pages 554–558, 1982.

[81] A. Howard, C. Padgett, and C. C. Liebe. A multi-stage neural network for automatic target detection. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 231–236, Anchorage, Alaska, 1998. 0-7803-4859-1/98.

[82] D. Howard, S. C. Roberts, and R. Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.

[83] J.-S. Huang and H.-C. liu. Object recognition using genetic algorithms with a Hopfield's neural model. *Expert Systems with Applications*, 13(3):191–199, 1997.

[84] Q. Huang, M. R. Azimi-Sadjadi, B. Tian, and G. Dobeck. Underwater target classification using wavelet packets and neural networks. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 177–182, Anchorage, Alaska, 1998. IEEE. 0-7803-4859-1/98, IEEE.

[85] B. Igelnik, Y. H. Pao, S. R. LeClair, and C. Y. Shen. The ensemble approach to neural-network learning and generalisation. *IEEE Transactions on Neural Networks*, 10(1):19–30, January 1999.

[86] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision Graphics and Image Processing*, 44:87–116, 1988.

[87] B. Irie and S. Miyake. Capability of three-layered perceptrons. In *Proceedings of the IEEE 1988 International Conference on Neural Networks*, pages I(641–648), New York, 1988. IEEE. Vol. 1.

[88] S. Isaka. An empirical study of facial image feature extraction by genetic programming. In J. R. Koza, editor, *the Genetic Programming 1997 Conference*, pages 93–99. Stanford Bookstore, Stanford University, CA, USA, July 1997. Late Breaking Papers.

[89] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[90] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29(3):31–44, March 1996.

[91] J. S. N. Jean and J. Wang. Weight smoothing to improve network generalisation. *IEEE Transactions on neural networks*, 5(5):752–763, September 1994.

[92] J. L. Johnson, M. L. Padgett, and O. Omidvar. Overview of pulse coupled neural network special issue. *IEEE Transactions on Neural Networks*, 10(3):461–463, May 1999.

[93] G. G. Judge and et al. *Introduction to the Theory and Practice of Econometrics.* New York: Wiley, 2nd edition, 1988.

[94] G. G. Judge, W. E. Griffiths, R. C. Hill, and T. Lee. *The Theory and Practice of Econometrics.* New York: Wiley, 1980.

[95] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, Cambridge, Massavhusetts, London, England, 1994.

[96] M. W. Kim and M. Arozullah. Generalised probabilistic neural network based classifier. In *International Joint Conference on Neural Networks (IJCNN'92)*, pages III–648–653, Baltimore, Maryland, June 1992. IEEE.

[97] M. W. Kim and M. Arozullah. Neural network based optimum radar target detection in non-Gaussian noise. In *International Joint Conference on Neural Networks (IJCNN'92)*, pages III–654–659, Baltimore, Maryland, June 1992. IEEE.

[98] T. Kohonen. *Associative Memory: A System Theoretic Approach.* SpringerVerlag, Berlin, 1977.

[99] T. Kohonen. *Self-organization and Associative Memory.* Springer, Berlin Heidelberg New York, 3rd edition, 1988.

[100] P. G. Korning. Training neural networks by means of genetic algorithms working on very long chromosomes. *International Journal of Neural Systems*, 6(3):299–316, September 1995.

[101] J. R. Koza. *Genetic programming : on the programming of computers by means of natural selection.* Cambridge, Mass. : MIT Press, London, England, 1992.

[102] J. R. Koza. Simultaneous discovery of reusable detectors and subroutines using genetic programming. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 295–302, Morgan Kauffman, 1993.

[103] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, Mass. : MIT Press, London, England, 1994.

[104] J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, D. E. Goldberg, H. Iba, and R. Riolo. *Genetic Programming 1998 – Proceedings of the Third Annual Conference on Genetic Programming.* San Francisco, CA: Morgan Kaufmann, 1998.

[105] J. R. Koza, F. H. Bennet, III, D. Andre, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann, San Francisco, California, 1999.

[106] J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. Riolo. *Genetic Programming 1997 – Proceedings of the Second Annual Conference on Genetic Programming.* San Francisco, CA: Morgan Kaufmann, 1997. ISBN 1-55860-483-9.

[107] R. Krishnan. 2DELTA-GANN: A new method of training neural networks using genetic algorithms. Enhanced minor thesis, RMIT, Department of Computer Science, Melbourne, Apr. 1994.

[108] R. Krishnan and V. Ciesielski. 2DELTA-GANN: a new approach to using genetic algorithms to train neural networks. In A. C. Tsoi, editor, *Proceedings of the Fifth Australian Neural Networks Conference*, pages 38–41, University of Queensland, Brisbane, Feb 1994.

[109] S. Y. Kung and J. S. Taur. Decision-based neural networks with signal/image classification applications. *IEEE Transactions on Neural Networks*, 6:170–181, Jan 1995.

[110] W. B. Langdon. Scheduling planned maintenance of the national grid. In T. C. Fogarty, editor, *Evolutionary Computing*, pages 132–153. Springer-Verlag, 1995. Vol. 993, Lecture Notes in Computer Science.

[111] S. Lawrence, C. Giles, and A. Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical Report UMIACSTR -96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, April 1996.

[112] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A hybrid neural network approach. Technical Report UMIACS-TR-96-16 and CS-TR-3608, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, April (revised Augest) 1996.

[113] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural network approach. *IEEE transactions on Neural Networks*, 8(1):98–112, Jan 1997.

[114] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. H. W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[115] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. H. W. Hubbard, and L. D. Jackel. Handwritten zip code recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufmann, San Mateo, CA, 1990.

[116] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hibbard. Handwritten digit recognition: application of neural network

chips and automatic learning. *IEEE Communications Magazine*, pages 41–46, November 1989.

[117] C. Lee and D. A. Landgrebe. Decision boundary feature extraction for neural networks. *IEEE Transactions on Neural Networks*, 8(1):75–83, 1997.

[118] M. Li and A. Maruoka. *Lecture Notes in Artificial Intelligence 1316.* Springer-Verlag, 1997. Proceedings of the Eighth International Workshop on Algorithmic Learning Theory (ALT'97), Sendai, Japan.

[119] S.-H. Lin, S.-Y. Kung, and L.-J. lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1):114–132, Jan 1997.

[120] X. Liu, D. Wang, and J. R. Ramirez. Extracting hydrographic objects from satellite images using a two-layer neural network. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 897–902, Anchorage, Alaska, 1998. IEEE. 0-7803-4859-1/98.

[121] S.-C. B. Lo, H.-P. Chan, J.-S. Lin, H. Li, M. T. Freedman, and S. K. Mun. Artificial convolution neural network for medical image pattern recognition. *Neural Networks*, 8(7/8):1201–1214, 1995.

[122] B. J. Lucier, S. Mamillapalli, and J. Palsberg. Program optimisation for faster genetic programming. In *Genetic Programming – GP'98*, pages 202–207, Madison, Wisconsin, July 1998.

[123] L. B. Lusted. General problems in medical decision making – with comments on ROC analysis. *Seminars Nucl. Med*, 8:299–306, 1978.

[124] A. Mahalanobis, B. V. K. Kumar, and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 26:3633–3640, 1987.

[125] W. McCulloch and W. Pitts. A logical calculus of the ideal immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[126] C. E. Metz. Basic principle of ROC analysis. *Seminars Nucl. Med*, 8:283–298, 1978.

[127] C. E. Metz. ROC methodology in radiologic imaging. *Investigative Radiology*, 21(9):720–732, September 1986.

[128] X. Miao, M. R. Azimi-Sadjadi, A. C. Dubey, and N. H. Witherspoon. Detection of mines and minelike targets using principal component and neural network methods. *IEEE Transactions on Neural Networks*, 9(3):454–463, May 1998.

[129] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning, An Artificial Intelligence Approach.* Tioga Publishing Company, Palo Alto, California, 1983.

[130] R. Milanese, S. Gil, and T. Pun. Attentive mechanisms for dynamic and static scene analysis. *Optical Engineering*, 34(8):2428–2434, 1995.

[131] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, Mass., 1969.

[132] T. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[133] D. J. Montana and L. Davis. Training feedforward networks using genetic algorithms. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 762–767, San Mateo, CA, 1989. Morgan Kaufmann.

[134] B. Muller, J. Reinhardt, and M. T. Strickland. *Neural Networks: An Introduction.* Springer-Verlag, Berlin Heidelberg, Germany, 2nd edition, 1995.

[135] P. Murphy and D. Aha. UCI repository of machine learning datasets. Technical report, Department of Information and Computer Science, University of California, Irvine, 1994.

[136] K. Nakayama, Y. Chigawa, and O. Hasegawa. Handwritten alphabet and digit character recognition using feature extracting neural network and modified self-organising map. In *International Joint Conference on Neural Networks (IJCNN'92)*, pages IV–235–240, Baltimore, Maryland, June 1992. IEEE.

[137] N. M. Nasrabadi and L. Wei. Object recognition by a Hopfield neural network. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1623–1535, 1991.

[138] E. Niebur and C. Koch. Control of selective visual attention: Modeling the "where" pathway. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, pages 802–808, Japan, 1996. The MIT Press. Vol. 8.

[139] P. Nordin and W. Banzhaf. Programmatic compression of images and sound. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996:*

Proceedings of the First Annual Conference, pages 345–350, Stanford University, CA, USA, 1996. MIT Press.

[140] S. Nowlan. Maximum likelihood competitive learning. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 574–582. Morgan Kaufmann, San Mateo, CA, 1990.

[141] H. Ogata, Y. Akiyama, and M. Kanehisa. A genetic algorithm based molecular modelling technique for RNA stem-loop structures. *Nucleic Acid Research*, 23(3):419–426, 1995.

[142] M. A. E. Okure and M. A. Peshkin. Quantitative evaluation of neural networks for NDE applications using the ROC curve. In *the 21th Annual Review of Progress in Quantitative Nondestructive Evaluation*, Snowmass CO, 1994. http://apo.mech.nwu.edu/okure-lib/ROC.html.

[143] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principle, L. Gile, N. Margan, and E. Wilson, editors, *Neural Networks for signal processing VII, Proceedings of the 1997 IEEE Workshop*, pages 276–285, Amelia Island, FL, September 1997.

[144] D. A. Panagiotopoulos, R. W. Newcomb, and S. K. Singh. Planning with a functional neural-network architecture. *IEEE Transactions on Neural Networks*, 10(1):115–127, January 1999.

[145] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, NY, USA, 1993.

[146] W. Pitts and W. S. McCulloch. How we know universals: The perception of auditory and visual foems. *Bulletin of Mathematical Biophysics*, 9:127–147, 1947. University of Chicago Press, Chicago.

[147] V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben. *Evolutionary Programming VII: Proceedings of 7th International Conference, EP98.* San Diego, CA: Springer-verlag, 1998. ISBN 3-540-64891-7.

[148] M. A. Potter. A genetic cascade-correlation learning algorithm. In Schaffer and Whitley, editors, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 366–372. Morgan Kaufmann, July 1992.

[149] J. R. Quinlan. *C4.5: Programs for Machine Learning – Constructing Decision Trees*, chapter 2, pages 17–43. Morgan-Kaufmann, San Mateo, California, 1993. ISBN 1-55860-238-0.

[150] N. Rai. Pixel statistics in neural networks for object detection. Minor thesis, RMIT University, Department of Computer Science, 2001.

[151] H. S. Ranganath, D. E. Kerstetter, and S. R. F. Sims. Self partitioning neural networks for target recognition. *Neural Networks*, 8(9):1475–1486, 1995.

[152] H. S. Ranganath and G. Kuntimad. Object detection using pulse coupled neural networks. *IEEE Transactions on Neural Networks*, 10(3):615–620, May 1999.

[153] J. Riley. An evolutionary approach to training feed forward and recurrent neural networks. Minor thesis, RMIT University, Department of Computer Science, 1997.

[154] G. Robinson and P. McIlroy. Exploring some commercial applications of genetic programming. In T. C. Fogarty, editor, *Evolutionary Computation, Volume 993, Lecture Note in Computer Science*. Springer-Verlag, 1995.

[155] S. K. Rogers, J. M. Colombi, C. E. Martin, J. C. Gainey, K. H. Fielding, T. J. Burns, D. W. Ruck, M. Kabrisky, and M. Ocley. Neural networks for automatic target recognition. *Neural Networks*, 8(7/8):1153–1184, 1995.

[156] R. J. Rogler, M. W. Koch, M. M. Moya, L. D. Hostetler, and D. R. Hush. Feature discovery via neural networks for object recognition in SAR imagery. In *International Joint Conference on Neural Networks (IJCNN'92)*, pages IV–408–413, Baltimore, Maryland, June 1992. IEEE.

[157] H. L. Roitblat, W. W. L. Au, P. E. Nachtigall, R. Shizumura, and G. Moons. Sonar recognition of targets embedded in sediment. *Neural Networks*, 8(7/8):1263–1273, 1995.

[158] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, New York, 1962.

[159] M. W. Roth. Neural network technology and its applications. *The Johns Hopkins APL Tech. Dig.*, 9(3):242–253, 1988.

[160] M. W. Roth. Neural networks for extraction of weak targets in high clutter environments. *IEEE Transactions on System, Man, and Cybernetics*, 19(5):1210–1217, September-October 1989.

[161] M. W. Roth. Survey of neural network technology for automatic target recognition. *IEEE Transactions on neural networks*, 1(1):28–43, March 1990.

[162] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, Jan 1998.

[163] C. D. W. Ruck. Multisensor target detection and classification. In *Proceedings of the SPIE, 931*, April 1988.

[164] D. E. Rumelhart, G. E. Hinton, and J. L. McClenlland. A general framwork for parallel distributed processing. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed Processing, Explorations in the Microstruct ure of Cognition, Volume 1: Foundations*, chapter 2. The MIT Press, Cambridge, Massachusetts, London, England, 1986.

[165] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 8. The MIT Press, Cambridge, Massachusetts, London, England, 1986.

[166] D. E. Rumelhart, J. L. McClelland, and the PDP research group. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition.* The MIT Press, Cambridge, Massachusetts, London, England, 1986. Volume 1: Foundations.

[167] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, December 1994.

[168] J. D. Schaffer, D. Whitely, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *COGANN-92, International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37, Baltimore, June 1992. IEEE Computer Society Press.

[169] R. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of 17th International Conference on Machine Learning*, pages 839–846, Stanford University, USA, 2000. Morgan Kaufmann, San Francisco, CA. 29 June - 2 July.

[170] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 304–312, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[171] M. V. Shirvaikar and M. M. Trivedi. Design and evaluation of a multiple stage object detection approach. In *Proceedings of Applied Artificial Intelligence VII Conference*, pages 14–22, Orlando, FL, April 1990. SPIE.

[172] M. V. Shirvaikar and M. M. Trivedi. A network filter to detect small targets in high clutter backgrounds. *IEEE Transactions on Neural Networks*, 6(1):252–257, Jan 1995.

[173] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach.* Springer-Verlag, Heidelberg, 1997.

[174] M. Sipper, D. Mange, and A. Perez-uribe. *Evolable Systems 1998 – Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware.* Heidelberg, Germany: Springer-verlag, 1998. ISBN 3-540-64954-9.

[175] F. F. Soulie, E. Viennet, and B. Lamy. Multi-modular neural network architectures: applications in optical character and human face recognition. *International journal of pattern recognition and artificial intelligence*, 7(4):721–755, 1993.

[176] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[177] L. Spirkovska and M. B. Reid. Higher-order neural networks applied to 2D and 3D object recognition. *Machine Learning*, 15(2):169–199, 1994.

[178] R. Sun. On neural networks and symbolic processing. In R. Sun and L.Bookman, editors, *Computational Architectures Integrating Neural and Symbolic Processes*, chapter 1. Kluwere Academic Press, 1994.

[179] R. Sun. Introduction to connectionist symbolic integration. In R. Sun and F. Alexandre, editors, *Connectionist-Symbolic Integration*, chapter 1. Lawrence Erlbaum Associates, 1997.

[180] D. Swets and B. Punch. Genetic algorithms for object localisation in a complex scene, 1995. Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University. http://garage.cps.msu.edu/papers/GARAGe95-01-06.ps.

[181] D. L. Swets, B. Punch, and J. Weng. Genetic algorithms for object recognition in a complex scene. In *Proceedings of International Conference on Image Processing*, pages 595–598, Washington D. C., Oct 1995. Vol. II.

[182] O. Syed. Applying genetic algorithms to recurrent neural networks for learning network parameters and architecture. Master's thesis, Department of Electrical Engineering, Case Western Reserve University, Ohio, USA, 1995.

[183] W. A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[184] W. A. Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs.* PhD thesis, Faculty of the Graduate School, University of Southern California, Canoga Park, California, USA, April 1994.

[185] I. Taha and J. Ghosh. Three techniques for extracting rules from feedforward networks. In C. F. Dagli Akay and Ghosh, editors, *Intelligent Engineering Systems Through Artificial Neural Networks.* ASME Press, 1996.

[186] A. Teller and M. Veloso. A controlled experiment : Evolution for learning difficult image classification. In C. Pinto-Ferreira and N. J. Mamede, editors, *Proceedings of the 7th Portuguese Conference on Artificial Intelligence*, volume 990 of *LNAI*, pages 165–176, Berlin, 3–6 Oct. 1995. Springer Verlag.

[187] A. Teller and M. Veloso. PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[188] B. Tian, M. R. Azimi-Sadjadi, T. H. V. Haar, and D. Reinke. A temporal adaptive probability neural network for cloud classification from satellite images. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1732–1737, Anchorage, Alaska, 1998. IEEE. 0-7803-4859-1/98, IEEE.

[189] T. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3:561–573, 1990.

[190] L. C. W. Tong and et al. Multisensor data fusion of laser radar and forward looking infrared (FLIR) for target segmentation and enhancement. *Proceedings of the SPIE*, 782:10–18, Arpil 1987.

[191] G. Towell and J. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine learning*, 13(1):71–101, 1993.

[192] G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1/2):119–165, 1994.

[193] S. E. Troxel, S. K. Rogers, and M. Kabrisky. The use of neural networks in PSRI target recognition. In *IEEE International Conference on Neural Networks*, pages I–593 – 600, Sheraton Harbor Island, San Diego, California, July 1988.

[194] P. W. M. Tsang. A genetic algorithm for aligning object shapes. *Image and Vision Computing*, 15:819–831, 1997.

[195] D. Valentin and H. Abdi. Can a linear autoassociator recognise faces from new orientations? *Journal of the Optical Society of America, series A,*, 13:717–724, 1996.

[196] D. Valentin, H. Abdi, and O'Toole. Categorization and identification of human face images by neural networks: A review of linear auto-associator and principal component approaches. *Journal of Biological Systems*, 2(3):413–429, 1994.

[197] D. Valentin, H. Abdi, and A. J. O'Toole. Principal component and neural network analyses of face images: Explorations into the nature of information available for classifying faces by sex. In P. T. C. Dowling, F.S. Roberts, editor, *Progress in Mathematical Psychology.* Hillsdale: Lawrence Erlbaum, Hillsdale, Erlbaum, 1996. in press.

[198] D. Valentin, H. Abdi, A. J. O'Toole, and G. W. Cottrell. Connectionist models of face processing: A survey. *Pattern Recognition*, 27:1208–1230, 1994.

[199] B. Verma. A feature extraction technique in conjunction with neural network to classify cursive segmented handwritten characters. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 332–336, Anchorage, Alaska, 1998. IEEE. 0-7803-4859-1/98, IEEE.

[200] B. Verma. A neural network based technique to locate and classify microcalcifications in digital mammograms. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1790–1793, Anchorage, Alaska, 1998. 0-7803-4859-1/98, IEEE.

[201] P. D. Wasserman. *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York, 1993. Neural Engineering, Chapter 11. ISBN: 0-442-00461-3.

[202] A. M. Waxman, M. C. Seibert, A. Gove, D. A. Fay, A. M. Bernandon, C. Lazott, W. R. Steele, and R. K. Cunningham. Neural processing of targets in visible, multispectral ir and sar imagery. *Neural Networks*, 8(7/8):1029–1051, 1995.

[203] P. Werbos. *Beyond Regression: New tools for prediction and analysis in the behavioral science.* PhD thesis, Department of Applied Mathematics, Harvard University, Cambridge, Mass., 1974.

[204] D. Whitley and T. Hanson. Optimising neural networks using faster more accurate genetic search. In *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, pages 391–396. Morgan Kaufman, 1989.

[205] P. Wilson. Development of genetic programming strategies for use in the robocup domain. Technical report, Department of Computer Science, RMIT, 1998. Honours Thesis.

[206] J. F. Winkeler and B. S. Manjunath. Genetic programming for object detection. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 330–335, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[207] P. Winter, S. Sokhansanj, H. C.Wood, and W. Crerar. Quality assessment and grading of lentils using machine vision. In *Agricultural Institute of Canada Annual Conference*, Saskatoon, SK S7N 5A9, Canada, July 1996. Canadian Society of Agricultural Engineering. CASE paper No. 96-310.

[208] P. Winter, W. Yang, S. Sokhansanj, and H. Wood. Discrimination of hard-to-pop popcorn kernels by machine vision and neural network. In *ASAE/CSAE meeting*, Saskatoon, Canada, Sept. 1996. Paper No. MANSASK 96-107.

[209] Y. Won, P. D. Gader, and P. C. Coffield. Morphological shared-weight networks with applications to automatic target recognition. *IEEE Transactions on neural networks*, 8(5):1195–1203, September 1997. ISSN 1045-9227.

[210] Y. C. Wong and M. K. Sundareshan. Data fusion and tracking of complex target maneuvers with a simplex-trained neural network-based architecture. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1024–1029, Anchorage, Alaska, May 1998. 0-7803-4859-1/98.

[211] J. Yang and V. Honavar. Feature subset selection using A genetic algorithm. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 380–385, Stanford University, CA, USA, 1997. Morgan Kaufmann. Also in IEEE Intelligent System Special Issue: Feature Transformation and Subset Selection, 1998.

[212] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, September 1999.

[213] A. Yla-Jaaski and F. Ade. Grouping symmetrical structures for object segmentation and description. *Computer Vision and Image Understanding*, 63(3):399–417, May 1996.

[214] A. Yli-Jaaski and F. Ade. Grouping symmetrical structures for object segmentation and description. *Computer Vision and Image Understanding*, 63(3):399–417, May 1996.

[215] K. Yuen, J. Princen, J. Illingworth, and J. Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[216] A. Zell, G. Mamier, and et al. *SNNS User Manual.* University of Stuttgart, 1995. Version 4.1.

[217] M. Zhang. A pixel-based approach to recognising small objects in large pictures using neural networks. In *Proceedings of the Annual RMIT Computer Science Postgraduate Students' Conference*, pages 57–68, Melboune, Australia, December 1997. Department of computer science, RMIT. TR 97-51.

[218] M. Zhang and V. Ciesielski. Centred weight initialisation to improve the performance of network training speed and the performance of object detection. Technical Report TR 98-10, Department of Computer Science, RMIT University, Melbourne, Australia, May 1998.

[219] M. Zhang and V. Ciesielski. Using back propagation algorithm and genetic algorithms to train and refine neural networks for object detection. In *Proceedings of Annual RMIT Computer Science Postgraduate Students' Conference (CSPSC'98)*, pages 37–48, Melbourne, Australia, December 1998. Department of Computer Science, RMIT. Also in technical report TR 98-25.

[220] M. Zhang and V. Ciesielski. Centred weight initialisation in neural networks for object detection. In J. Edwards, editor, *Proceedings of the Twenty Second Australasian Computer Science Conference*, pages 39–50, Auckland, Feb 1999. http://goanna.cs.rmit.edu.au/~vc/papers/vc-conf.html.

[221] M. Zhang and V. Ciesielski. Genetic programming for multiple class object detection. In N. Foo, editor, *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99)*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag Berlin Heidelberg. Lecture Notes in Artificial Intelligence (LNAI Volume 1747).

[222] M. Zhang and V. Ciesielski. Genetic programming for multiple class object detection. In *Proceedings of Annual RMIT Computer Science Postgraduate Students' Conference (CSPSC'99)*, pages 70–78, Melbourne, Australia, December 1999. Department of Computer Science, RMIT. Also in technical report TR-99-10.

[223] M. Zhang and V. Ciesielski. Using back propagation algorithm and genetic algorithm to train and refine neural networks for object detection. In T. Bench-Capon, G. Soda, and A. M. Tjoa, editors, *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99)*, pages 626–635, Florence, Italy, August 1999. Springer-Verlag. Lecture Notes in Computer Science, (LNCS Volume 1677).

[224] J. M. Zurada. Challenges and changes. *IEEE Transactions on Neural Networks*, 10(1):1–2, 1999.

Index