VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences

Computer Science

A Domain Independent Approach to Multiclass 2D Object Detection Using Neural Networks and Genetic Algorithms

Mengjie Zhang and Victor Ciesielski

Technical Report CS-TR-02/02
February 2002

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

A Domain Independent Approach to Multiclass 2D Object Detection Using Neural Networks and Genetic Algorithms

Mengjie Zhang and Victor Ciesielski

Technical Report CS-TR-02/02
February 2002

Abstract

This paper describes a domain independent approach to multiple class, translation and rotation invariant 2D object detection problems without any preprocessing, segmentation and specific feature extraction. The approach is based on learning/adaptive methods – neural networks and genetic algorithms. Rather than using specific image features, raw image pixel values are used as inputs to the learning/adaptive systems.

Five object detection methods have been developed and tested on three databases which represent detection problems of increasing difficulty. 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. The centred weight initialisation algorithm improved the detection performance over the basic approach on all three databases. In addition, refinement of weights with a genetic algorithm significantly improved detection performance on the three databases.

Keywords Network training; network testing; network sweeping; evolutionary process; object recognition; target recognition; target detection.

Author Information

The first author is a lecturer in computer science, Victoria University of Wellington, New Zealand. The second author is an associate professor of artificial intelligence, RMIT University, Melbourne, Australia.

A Domain Independent Approach to Multiclass 2D Object Detection Using Neural Networks and Genetic Algorithms

Mengjie Zhang and Victor Ciesielski
mengjie@mcs.vuw.ac.nz, vc@cs.rmit.edu.au

Abstract

This paper describes a domain independent approach to multiple class, translation and rotation invariant 2D object detection problems without any preprocessing, segmentation and specific feature extraction. The approach is based on learning/adaptive methods – neural networks and genetic algorithms. Rather than using specific image features, raw image pixel values are used as inputs to the learning/adaptive systems.

Five 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, (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. 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 a 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. The centred weight initialisation algorithm improved the detection performance over the basic approach on all three databases. In addition, refinement of weights with a genetic algorithm significantly improved detection performance on the three databases.

Keywords Network training; network testing; network sweeping; evolutionary process; object recognition; target recognition; target detection.

1 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 tumors 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 objects 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 [33, 79] 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 [33, 62, 64], this paper aims 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 [9, 15, 29, 86] 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 paper focuses on the development of a domain independent method without preprocessing and segmentation for multiple class object detection.

In recent years, neural networks and genetic algorithms have attracted attention as very promising methods of solving automatic target recognition and detection problems [16, 36, 66, 69, 75]. 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. To investigate the potential ability in this area, this paper applies neural networks and genetic algorithms to multiple class object detection problems.

In terms of the input patterns of the neural networks for object detection and recognition, two main approaches have previously 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 [19, 62, 83]. These features are usually different and specific for different problem domains. In pixel based approaches [20, 39, 66], the pixel values 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. To avoid these disadvantages, the approach described in this paper focuses on the use of raw pixel data.

1.1 Goals

The overall goal of this paper is to investigate a learning/adaptive, 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 trained and refined by back propagation algorithms and genetic algorithms. Rather than using specific image features, raw image pixel values are used as inputs to neural networks. Specifically, the following hypotheses will be explored on a sequence of detection problems of increasing difficulty to determine the strengths and limitations of the different methods introduced in this paper.

1. Can a basic pixel based neural network approach with the back propagation learning algorithm be developed and applied to multiple class object detection problems?

2. Can a centred weight initialisation algorithm be developed to improve the network training and the detection performance over the basic approach?

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 back propagation algorithm in the basic method?

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?

1.2 Structure

In the remainder of this paper, we first review relevant literature and describe the image databases used in the experiments. We then describe the basic approach and analyse the results. After the centred weight initialisation method and its improved results are presented, the network internal behaviour is revealed through analysing the network weights. Next, we describe a further improvement, training and refining networks using genetic algorithms, and the final results. Finally, we conclude the paper and summarise the contributions.

2 Background

2.1 Multiclass Object Detection

2.1.1 Object Detection vs Object Classification

The term *object classification* here refers to the task of discriminating between images of different kinds of objects. Each image contains only one of the objects of interest. The term *object detection* here refers to the detection of small objects in large pictures. This includes both object classification, as described above, and *object localisation*, which gives the positions of all objects of interest in the large pictures.

2.1.2 Multiclass Object Detection

In automatic object detection systems, in most cases, all the objects of interest are considered as one class of interest[53, 66]. These systems generally focus on finding the locations of the objects of interest and all other locations are considered *non-object* or the *background*. In contrast, the *multiple class* detection problem refers to the case where there is more than one class of objects and both their classes and locations must be determined. In general, multiclass object detection problems are much harder than single class detection problems, and multiclass detection using a single trained program (such as a neural network) is an even more difficult problem.

2.1.3 Performance Evaluation

Performance in object detection is measured by detection rate and false alarm rate. The detection rate is the number of objects correctly reported as a percentage of the total number of real objects and false alarm rate is the number of objects incorrectly reported as a percentage of the total number of real objects. For example, a detection system looking for grey squares in Figure 2 (a) may report that there are 25. If 9 of these are correct the detection rate will be $(9/18) * 100 = 50\%$. The false alarm rate will be $(16/18) * 100 = 88.9\%$.

It is important to note that detecting objects in pictures with very cluttered backgrounds is an extremely difficult problem and that false detection rates of 200-2,000% (that is the detection system suggests that there are 20 times as many objects as there really are) are common [64, 66]. Also note that most research which has been done in this area so far only presents the results of the classification stage and assumes that all other stages have been properly done. However, the results presented in this paper are the performance for the whole detection problem (both the localisation and classification). Thus, it is unfair to simply compare the results in this paper 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 achieved in this paper with that only for one class detection problems.

2.2 Training Set Size in Neural Networks

Probably Approximately Correct (PAC) Learning [5, 37, 40] addresses the relationship between the number of training examples and the accuracy with which a classifier can be learnt.

According to the PAC results for neural networks [5, 78], a large number of training examples are required to successfully train a neural network. However, the PAC results give a theoretical upper bound and in practice many networks have been successfully trained with far fewer training examples [48, 78]. In particular, Anthony [5] claimed that smaller training set than predicted by PAC learning should suffice. In [20, 50, 66], a much smaller set of examples were used to successfully train corresponding networks which had a very good generalisation.

2.3 Related work

2.3.1 Neural Networks for Object Classification and Detection

Since the late 1980s, the use of neural networks in object classification and detection has been investigated in a variety of application domains. These domains include military applications, 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 [65], self organising maps [44], higher order networks [34, 58] and ART networks [17, 18].

A brief review of neural networks for object classification is summarised in Table 1 (page 6) according to the kinds of the networks, application domains/problems, the first authors and the year of publication. It is noted that most of these systems are feature based, where specific image features are used to inputs to neural networks.

A breif review of neural networks for object detection is presented in Table 2 (page 7). Typically, they belong to the *one-class object detection* problems, 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.

In this paper, we will investigate pixel based neural network approaches for multiclass object detection problems.

2.3.2 Genetic Algorithms for Evolving Neural Networks

In addition to the back propagation algorithm, genetic algorithms can be also used to train (Evolve) neural networks. Related work in this area can be grouped into three approaches.

Evolving weights in fixed networks [45, 57, 81]: 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.

Kind of Network	Applications/Problems	Authors	Year	Source
Feed Forward Networks	Classification of mammograms	Verma	1998	[77]
	Target recognition with sonar	Barshan et al.	2000	[10]
		Ayrulu et al.	2001	[6]
	Classification of missiles, Planes and helicopters	Howard et al.	1998	[38]
	Maneuver target recognition	Wong et al.	1998	[85]
	Classification of tanks and Trucks on laser radar images	Troxel et al.	1988	[73]
	Underwater target classification	Azimi-Sadjadi et al.	2000	[7]
	Mine and mine-like Target detection	Miao et al.	1998	[56]
	Handwritten character recognition	Verma	1998	[76]
	Agricultural product recognition	Winter et al.	1996	[83, 82]
	Multispectral remote sensing Data classification	Lee et al.	1997	[51]
	Natural object classification	Singh et al.	2000	[67]
	Helicopter Classification	Stahl et al.	2000	[70]
Shared Weight Neural Networks	Handwritten optical character recognition	Soulie et al.	1993	[68]
	Zip code recognition	LeCun et al.	1989	[47, 50]
	Digit recognition	de Redder et al.	1996	[23, 24]
	Lung nodule detection			
	Microcalcification classification	Lo et al.	1995	[55]
Auto-associative Memory Networks	Face recognition,	Abdi Valentin et al.	1988 1994	[1] [74] [75]
ART Networks	Vehicle recognition	Bernardon et al.	1995	[12]
	Tank recognition	Fogler et al.	1992	[61]
Serf-Organising Maps	Handwritten word recognition	Wessels et al.	2000	[80]
		Dehghan et al.	2000	[25]
Probability Neural Networks	Cloud classification	Tian et al.	1998	[72]
	Radar target 'detection'	Kim et al.	1992	[42, 43]
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	[32]
High Order Neural Networks	2D and 3D helicopter (F18) recognition	Spirkovska et al.	1994	[69]
	Apple sorting (classification)	Hecht-Nielsen	1992	[36]
	Recognition of bars, triangles and squares	Cross et al.	1995	[22]
Hybrid/Multiple Neural Networks	River identification(classification)	Liu et al.	1998	[54]
	Classification of "hot" and "cold" objects	Casasent et al.	1995	[19]
	Handwritten character recognition	LeCun et al.	2001	[49]
	Review	Egmont-Petersen et al.	2001	[27]

Table 1: Neural networks for object classification

Kind of Network	Applications/Problems	Authors	Year	Source
Feed Forward Neural Networks	Target detection in thermal infrared images	Shirvaikar et al.	1995	[66]
Shared Weight Neural Networks	Vehicle detection	Khabou et al.	2000	[41]
		Won et al.	2000	[84]
Probabilitistic Decision-Based Neural Networks	Face detection and recognition	Lin	2000	[52]
Other/ Hybrid/Multiple Neural Networks	Aircraft detection and recognition	Waxman et al.	1995	[79]
	Vehicle detection	Bosch et al.	1998	[13]
	Face detection	Feraud et al.	2001	[30]
		El-Bakry et al.	2000	[28]
	Triangle detection	Ahmad et al.	1990	[2]
	Detection/extraction of weak targets for Radars	Roth	1989	[63]
	Review of target detection	Rogers et al.	1995	[60]
	Review of target recognition	Roth	1990	[64]

Table 2: Object detection based on neural networks.

Evolving network architectures [59, 71]: 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. Honavar [8] present a nice overview and classification of the research in the area of evolutionary design of neural network architectures.

Evolving other factors in neural networks: This includes evolving learning parameters [11], learning rules [31], delta values [21, 46] and input features [14].

The major advantage of these approaches is that the "local minima" problem in which gradient descent algorithms often result can be avoided. The networks evolved by genetic algorithms presented in the literature are relatively small and the process runs quite well. For the object detection problems investigated in this paper, however, the networks are very large due to the use of pixels as inputs. We will investigate whether genetic algorithms can be used to train and refine large networks for multiclass object detection problems in this paper.

3 Image Data

3.1 Image Data Sets

To avoid confusion, we define a number of terms related to the image data.

A set of entire images in a database constitutes an *image data set* for a particular problem domain. In this paper, it is randomly split into two parts: a *detection training set*, which is used to learn a detector, and a *detection test set*, which is used for measuring object detection performance. *Cutouts* refer to the object examples which are cut out from a detection training set. These cutouts form a *classification data set*, which is randomly split into two parts: a *classification training set* used for network training, and a *classification test set* for network testing in object classification. An *input field* refers to a square in the large images. This is used as a moving window for the network sweeping (detection) process. The size of the input field is the same as that of the cutouts for network training. The relationships between the various data sets are shown in figure 1.

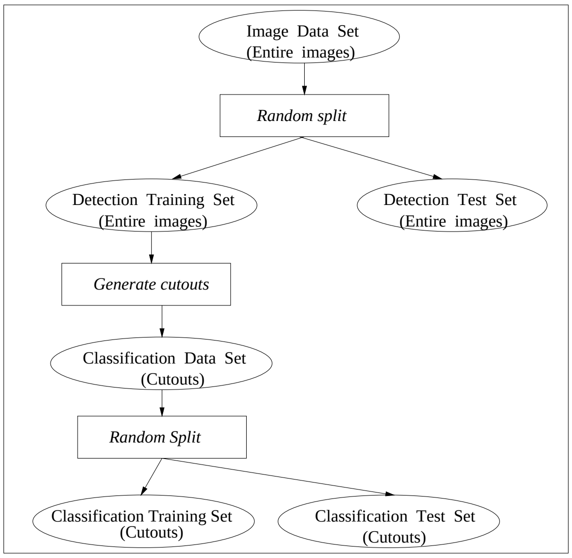


Figure 1: Relationships between classification and detection data sets.

3.2 Three Databases

To investigate the strengths and limitations of the learning/adaptive approaches described in this paper, three different grey scale databases were used in the experiments. The pictures were selected to provide detection problems of increasing difficulty. Example pictures and key characteristics are given in Figure 2.

3.2.1 Easy Pictures

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

There are three classes of 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

Number of images: 10	Number of images: 20	Number of images: 20
Number of classes: 4	Number of classes: 5	Number of classes: 5
Input field size: 14×14	Input field size: 24×24	Input field size: 16×16
Number of objects: 240	Number of objects: 200	Number of objects: 164
Picture size 700x700	Picture size 640x480	Picture size 1024x1024
Easy (circles and squares)	Medium difficulty (coins)	Very difficult (retinas)

Figure 2: Object Detection Problems of Increasing Difficulty

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.

3.2.2 Coin Pictures

The second database (the coin pictures) 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. All the objects in each class have a similar size. They are located at arbitrary positions and with different rotations.

Each of the pictures contains four object classes of interest, that is, the head side of 5 cent coins (class *head*005), the head side of 20 cent coins (class *head*020), the tail side of 5 cent coins (class *tail*005) and the tail side of 20 cent coins (class *tail*020). The background (class *other*) is relatively uniform – not totally uniform because of the different lighting conditions and camera positions.

3.2.3 Retina Pictures

The retina pictures (database 3) were taken by a professional photographer with special apparatus at a clinic. 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*). To give a clear view of representative samples of the target objects in the retina pictures, one sample piece of these pictures is presented in figure 3. In this figures, haemorrhage and micro-aneurism examples are labeled using white surrounding squares.

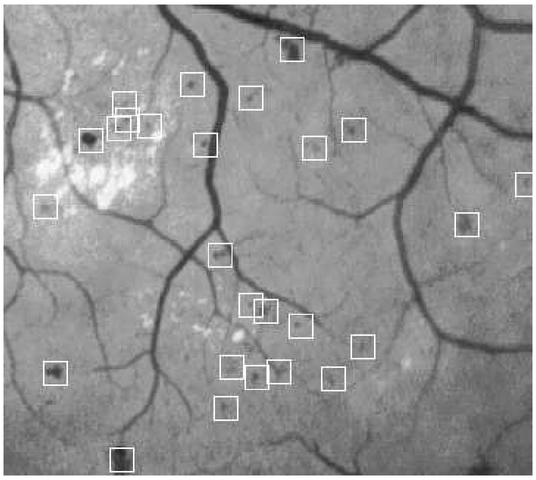


Figure 3: An enlarged view of one piece of the retina pictures

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*) with different shapes and the retina "edges" (class *edge*). The backgrounds (class *other*) are varied, some parts are quite black, some parts are very bright, and some parts are highly cluttered.

4 The Basic Approach

4.1 Overview

An overview of the basic approach is presented in figure 4. It consists of seven 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 entire images into two sets: a detection training set and a detection test set.

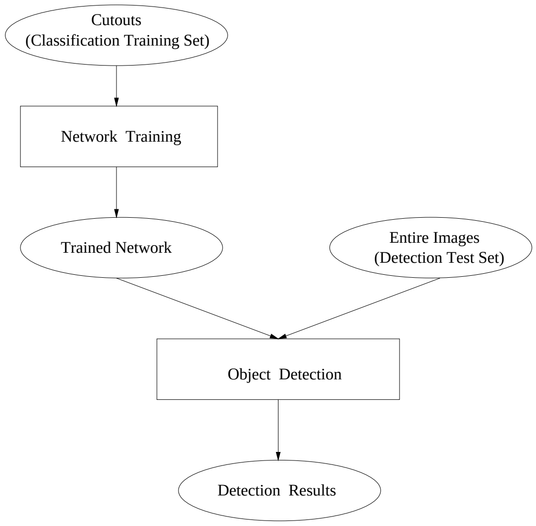


Figure 4: An overview of the basic approach.

2. Determine an appropriate size (n) of a square which will cover all objects of interest and form the input field of the network.

3. Generate a classification data set by cutting out squares of size $n \times n$ from the detection training set. Randomly split these cutouts into a classification training set and a classification test set.

4. 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 and the classification is the output. The number of hidden nodes is empirically determined.

5. Train the network by the backward error propagation algorithm on the classification training data. The trained network is then tested on the classification test set to measure the object classification performance.

6. Use the trained network as a moving window template [9] 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.

7. Evaluate the object detection performance of the network by comparing the classes and locations detected with the known classes and locations in the detection test set and calculating the detection rate and the false alarm rate.

In the remainder of this section, we mainly describe the network training, testing and sweeping procedure and present the object classification and object detection results.

4.2 Object Classification

4.2.1 Network Training

We use the backward error propagation algorithm [65] with the variation of online learning and the fan-in factor [23, 47] to train the networks. In the *online learning* procedure, also called the *stochastic gradient procedure*, weight changes are applied to the network after each training pattern. 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.

The training is terminated when the classification accuracy in the classification training set reaches a pre-defined percentage. When training is terminated, the trained network weights and biases are saved for the use in network testing or subsequent resumption of training.

4.2.2 Network Testing

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.

4.2.3 Classification Criterion

During network training and testing, the classification is regarded as correct if the output node with the largest activation value corresponds to the desired class of a pattern.

4.3 Object Detection

After network training is successfully done, the trained network is used to detect the classes and locations of the objects of interest in the detection test set, which is not used in any form for network training/testing. Classification and localisation are performed by the procedures – *network sweeping*, *finding object centres* and *object matching*.

4.3.1 Network Sweeping

During network sweeping, 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.

After the sweeping is finished, an *object sweeping map* for each object class of interest will be produced. An object sweeping map corresponds to a PGM image. Sample object sweeping maps for *class1*, *class2* and *class3* together with the original image for the easy detection problem are shown in figure 5. During the sweeping process, if there is no match between a square in a detecting picture 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.

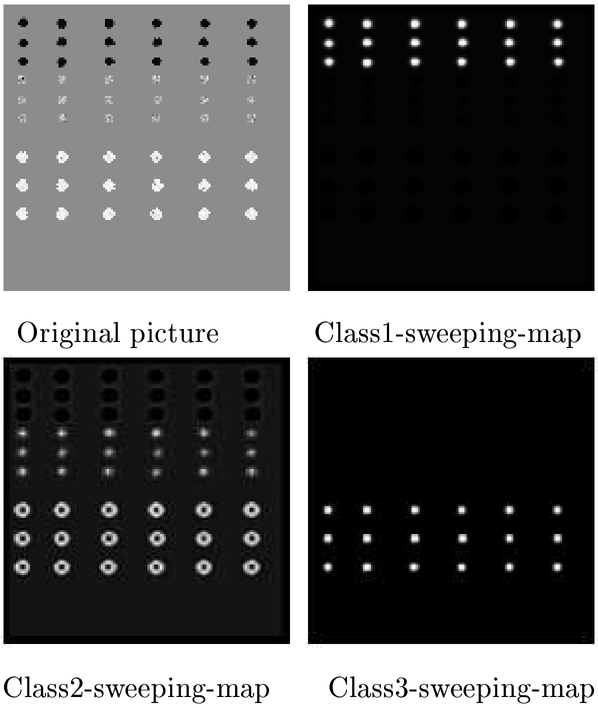


Figure 5: Sample object sweeping maps in object detection.

4.3.2 Finding Object Centres

In this paper, the centres of the objects are used to represent the objects themselves. We developed a *centre-finding algorithm* to find the centres of all objects detected by the trained network. For each object class of interest, 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 6.

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 the detected object at this position since the activation for this class is the biggest one.

For each object sweeping map:

Step 1 Set a *threshold* for the class (see section 4.3.5).

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 labeled centre (x, y) to zero.

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

Figure 6: Centre-finding algorithm

4.3.3 Object Matching

The number of objects detected by the trained network is reported by the procedure of finding object centers. Object matching compares all the object centres reported by the detector with all the desired known object centres and reports the number of objects correctly detected. Here, we allow a tolerance of 4 pixels in the x and y directions. For example, if the coordinates of a known object are *(21, 19)* and the coordinates of a detected object are *(22, 21)*, we still consider them 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.

4.3.4 Object Detection Performance Measurement

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 2.1.2 (page 4). It is important to note that the detection performance here refers to the performance achieved by a trained network on the detection test set.

4.3.5 Choice of Thresholds

During the object detection process, various thresholds result in different 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. Thus there is a trade-off between the detection rate and the corresponding false alarm rate.

To measure the detection performance of a learned neural network, the detection rates and false alarm rates under different thresholds need to be calculated. For an object sweeping map of an object class of interest, the thresholds were selected according to the following procedure:

1. Initialise a threshold (T) to 0.7, apply the centre-finding algorithm and object matching process and calculate the detection rate (DR) and the corresponding false alarm rate (FAR).

2. If the DR is less than 100%, decrease the T to a new value and calculate the new DR and FAR. Repeat this to obtain all the possible DRs and FARs until a new DR reaches 100% or a new T is less than or equal to 0.40.

3. From the point of step 1, if the FAR is not zero, increase T in order to obtain a new point (a DR with its corresponding FAR). Repeat this procedure until either the FAR is zero, or DR is zero, or the T is greater than or equal to 0.999.

Note that the thresholds applied in step 2 and 3 may need to be adjusted. For example, in step 2, the current threshold should be increased if a possible point is lost between the detection rate with the current threshold and that with the previous threshold.

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 3.

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

4.4 Results

4.4.1 Object Classification Results

Object classification involves network training and testing on the cutouts. The performance is measured by the number of epochs used for network training, mean squared error (MSE) and the classification accuracy.

It is important to note that the results for object classification are not the final detection results. These results are used to establish that the classification networks can be successfully trained with some sets of parameters and to determine what is the best classification on cutouts to use as a reference point for object detection.

Expt. No.	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	175	5.2	100	5.2	100
2	182	5.1	100	5.1	100
3	196	5.0	100	5.1	100
4	171	6.1	100	6.1	100
5	197	5.0	100	5.1	100
6	210	5.0	100	5.1	100
7	203	5.0	100	5.1	100
8	214	4.9	100	5.0	100
9	212	5.0	100	5.1	100
10	210	5.0	100	5.1	100
11	234	4.8	100	4.9	100
12	184	5.1	100	5.2	100
13	222	4.9	100	5.0	100
14	199	5.1	100	5.2	100
15	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: 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.)

In the easy pictures, the number of input nodes is defined as 196, corresponding to the size of the input field 14×14. There are four classes (three classes of interest plus the background) and we defined the number of output nodes as 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 and 150, and the experiment indicated that 3-10 hidden nodes gave similar results and the performance was best for 4 hidden nodes. The network training and testing results with the network (196-4-4) are presented in table 4. The learning rate (η) used here was 0.5, the momentum (α) was zero. Training was terminated when all the training patterns in the classification training set were correctly classified ($percent$ = 100%). The last line of the table shows that 15 runs terminated at an average of 199.40 epochs with an average standard deviation of 18.09 and the mean squared error on the classification training set was $(5.09{\pm}0.30){\times}10^{-2}$ and 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%.

A similar series of experiments was performed for the coin pictures. We empirically determined the best network architecture – 576-3-5 and the learning parameters – a learning rate of 0.5 without momentum. The results of 15 runs are shown in table 5.

Expt. No.	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	197	3.1	100	3.1	100
2	214	2.3	100	2.3	100
3	175	2.9	100	2.9	100
4	258	2.5	100	2.5	100
5	190	2.5	100	2.5	100
6	238	2.5	100	2.5	100
7	197	2.3	100	2.3	100
8	242	1.9	100	1.9	100
9	259	2.2	100	2.2	100
10	219	3.5	100	3.5	100
11	343	1.9	100	1.9	100
12	231	2.2	100	2.2	100
13	175	3.1	100	3.1	100
14	175	2.7	100	2.7	100
15	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 5: Results of network training and testing for object classification in the coin pictures using the basic approach. (Network architecture: 576-3-5; $\eta = 0.5$; $\alpha = 0$; Stop criterion: $Percent =$ 100%; Training set size = 100; Test set size = 100; Repetitions = 15.)

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 best network architecture and the learning parameters were empirically determined and the results are presented in table 6. Note that 10 runs of experiments were performed for the retina pictures due to long computation times.

4.4.2 Object Detection Results

This section describes the detection performance of the basic approach on the three detection problems.

The best results for the three classes in the easy pictures are presented in table 7. The basic approach always achieved a 100% detection rate and a corresponding zero false alarm rate for *class1* (black circles) and *class3* (white circles). However, this is not the case for *class2* (grey squares).

Expt. No.	Epochs	Training MSE ($\times 10^{-2}$)	Training Accuracy (%)	Test MSE ($\times 10^{-2}$)	Test Accuracy (%)
1	431	6.9	75.4	7.7	72.1
2	345	7.2	75.4	9.7	68.9
3	645	6.0	75.4	7.7	73.8
4	412	6.5	75.4	8.0	68.9
5	408	7.3	75.4	9.3	73.8
6	373	7.1	75.4	9.3	68.9
7	516	5.9	75.4	7.6	73.8
8	415	7.2	75.4	8.1	75.4
9	445	6.7	75.4	7.7	73.8
10	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 6: Results of network training and testing for object classification in the retina pictures using the basic approach. (Network architecture: 256-4-5; $\eta = 1.5$; $\alpha = 0$; Stop criterion: $Percent$ = 65%; Training set size = 100; Test set size = 61; Repetitions = 10.)

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

Table 7: Object detection results for the easy pictures using the basic approach. (Network architecture: 196-4-4; Repetitions = 15.)

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

Table 8: Object detection results for *class2* in the easy pictures using the basic approach. (Network architecture: 196-4-4; Repetitions = 15.)

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 8.

The best detection results for the coin pictures are described in table 9. 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 classes *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

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

Table 9: Object detection results for the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

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

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

Table 10: Object detection results for class *head020* in the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

Tables 10 and 11 show the details of the average detection performance for classes *head020* and *tail020* in the coin pictures.

Coin Pictures	Object Classes				
(Basic Method)	tail020				
Detection Rate(%)	100	93.75	87.50	81.25	75.0
False Alarm Rate(%)	37.5	25.0	18.8	12.5	0

Table 11: Object detection results for class *tail020* in the coin pictures using the basic approach. (Network architecture: 576-3-5; Repetitions = 15.)

Tables 12 and 13 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 (a detection rate of 100%) but with a false alarm rate of 10104%.

4.5 Summary and Discussion

The goal of this section was to investigate a domain independent approach to multiclass object detection problems. This goal was achieved by using raw image pixel data as inputs to neural networks, and accordingly feature extraction was avoided. The experimental results showed that

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	
False Alarm Rate(%)	956.1	873.0	682.6	534.8	426.1	373.9	187.0	65.2	

Table 12: 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
False Alarm Rate(%)	10104	9800	9332	8680	7440	7196	6309	5327	4980	3694

Table 13: Object detection results for class *micro* in the retina pictures using the basic approach. (Network architecture: 256-4-5; Repetitions = 10.)

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.

However, 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 paper 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?

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?

3. In this approach, the network was trained based on the cutouts of the objects 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?

5 Centred Weight Initialisation

5.1 Main Considerations

In this section, we introduce a centred weight initialisation method, which is expected to be able to improve both object classification and object detection performance. The main considerations are:

- In general neural approaches to pattern classification, the inputs are usually independent. Accordingly, the weights are initialised randomly, which usually results in reasonable network training speed and 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 pixel based neural approaches to object detection, corresponding weights to image pixels could be initialised in a different way.

- In the back propagation algorithm used in the basic approach, the network weights are initialised to small random values. In this method, we expect that the central pixels in a square are more important than those on the perimeter and initialise network weights accordingly.

5.2 Centred Weight Initialisation Method

In the basic method, three layer feed forward neural networks are used and the weights in a network are initialised with small random floating point numbers between -0.5 and +0.5. This would result in a pattern of weights between the input layer and a hidden node, an *input-hidden weight matrix*, as shown in the Hinton diagram in figure 7 (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 7 (a) corresponds to $pixel(i, j)$ in the input field.

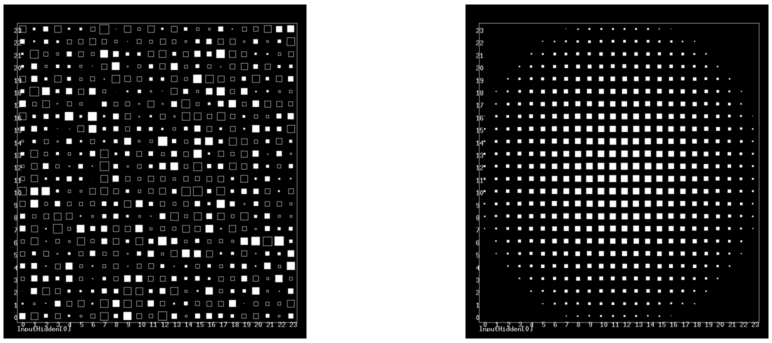
Figure 7 (b) shows a matrix with the centred initial weights. The centred weight initialisation method is described as follows:

1. Set the parameter *max_weight* for the central weight. Obtain the gap between the two neighbouring weights (*weight_gap*) according to equation 1.

$$weight\_gap = 2 * max\_weight \; / \; size\_square \tag{1}$$

   For each *input-hidden weight matrix*, initialise all the weights according to steps 2 to 4:

2. Calculate the magnitude of the four central weights corresponding to the four central pixels of the object according to equation 2. Because the width and the height (both equal to



(a) Random Initialisation (b) Centred Initialisation

Figure 7: Sample initial *input-hidden* weights. (a) Random initial weights; (b) Centred initial weights.

$size\_square$) of a training square that we use are even number of pixels, there are four central pixels in the training square.

$$cen\_weight = max\_weight + \epsilon \tag{2}$$

ϵ is a small random number generated by a normal distribution [4] as in equation 3:

$$\epsilon = normalised\_gaussian(\mu, \sigma) \tag{3}$$

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

3. For each weight (*weight(i, j)*) other than the central ones, calculate the distance of the corresponding pixel (*pixel(i, j)*) from the nearest central pixel in the object according to equation 4:

$$distance(i, j) = \sqrt{(i - x)^2 + (j - y)^2} \tag{4}$$

Here, *(x, y)* denotes the coordinates of the nearest central pixel and *(i, j)* is the position of the pixel in the training square.

4. Calculate the magnitude of each weight according to equation 5:

$$weight(i, j) = max\_weight - distance(i, j) * weight\_gap + \epsilon \tag{5}$$

If $weight(i,\ j) < 0$ then $weight(i, j) = 0$. In this way, all corresponding weights in the different input-hidden weight matrices are slightly different due to the small random number ϵ.

5. For initialisation of the hidden-output weights and all the biases, the standard random weight initialisation method is used.

The algorithm ensures that weights are largest at the centre and decrease uniformly to the perimeter as shown in figure 7 (b). While the weights are not truly random they provide a satisfactory starting point for training the network with the backward error propagation algorithm.

5.3 Results

5.3.1 Object Classification Results

Expt No.	Initial Weights format	Max-Wei	Wei-Gap	Epochs ($\mu \pm \sigma$)	Training MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)	Test MSE ($\mu \pm \sigma$) ($\times 10^{-2}$)
1	random			199.40±18.09	5.09±0.30	5.17±0.27
2	centred	0.420	0.060	430.80±29.78	4.81±0.10	4.85±0.11
3	centred	0.350	0.050	263.00±24.56	5.01±0.07	5.03±0.05
4	centred	0.280	0.040	219.47±14.20	4.47±0.12	4.55±0.11
5	centred	0.210	0.030	143.80±23.07	4.63±0.17	4.70±0.15
6	centred	0.140	0.020	109.87±10.64	5.04±0.11	5.08±0.12
7	centred	0.070	0.010	128.87±9.42	5.29±0.10	5.35±0.07
8	centred	0.035	0.005	139.33±9.03	5.37±0.12	5.45±0.11
9	centred	0.014	0.002	153.40±11.76	5.11±0.06	5.19±0.04

Table 14: 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; $\eta = 0.5$; $\alpha = 0$; Stop criterion: *Percent* = 100%; Training set size = 60; Test set size = 180; Repetitions = 15.)

Object classification results for the easy pictures are shown in table 14. Line 1 of the table shows that 15 runs with random initial weights terminated at an average of 199.40 epochs with an average standard deviation of 18.09 and the mean squared error on the test set was 0.0517 with a standard deviation of 0.0027. The same data for the centred weight initialisation with different values of *max_weight* are given in the other lines of the table. It is evident that for 0.07< *max_weight* <0.28 the centred weight initialisation method is superior.

A similar series of experiments were performed for the other databases. In both cases there was a range of values for *max_weight* which gave improved training times and lower test mean squared error. These results are summarised in table 15. On the coins database, for example, a network of 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.

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 subsection compares the detection performance of the two weight initialisation methods.

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

5.3.2 Object Detection Results

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

Table 16: Object detection results for the easy pictures using the centred initial weights. (Network architecture: 196-4-4; $Max\_weight = 0.14$; Repetitions = 15.)

The best results for the easy pictures are presented in table 16. Similarly to the results for the basic approach with random initial weights (table 7), detecting objects in *class1 (black circles)* and *class3 (white circles)* is relatively straight forward, which achieved a 100% detection rate without any false alarms. However, this is not the case for detecting *class2 (grey squares)*, which resulted 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 17.

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

Table 17: 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 8), these results are much better. At a detection rate of 100%, the false alarm rate is 46.4%, which is much lower than 91.2% achieved by the random initial weights. 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 level of detection rate.

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

Experiments with the coin pictures gave a similar pattern to the easy pictures. The overall results are presented in table 18. Compared with the corresponding results achieved using the basic approach (table 9), 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* resulted 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 random initial weights. The detailed results for class *head020* with the centred weights are presented in table 19.

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

Table 19: 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	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 20: 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.)

The results for classes *haem* and *micro* in the retina pictures are presented in table 20 and table 21. Compared with those for the easy pictures and the coin pictures, these results are disappointing.

However, they are much better than the corresponding results (tables 12 and 13) obtained by the basic method.

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
False Alarm Rate(%)	2903	2581	2232	1947	1940	1438	560	460	420	311

Table 21: 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 detection results are summarised in figure 8 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.

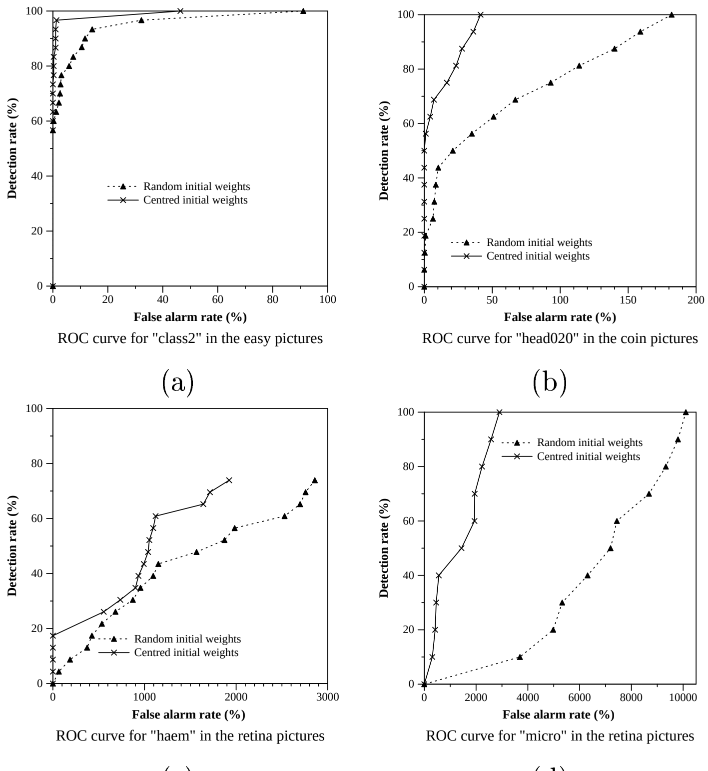


Figure 8: Comparison of results (in ROC curve) for object detection in the three databases between the centred weight initialisation and the random weight initialisation.

For the easy pictures and coin pictures, network thresholds can be chosen which give good detection and false alarm rates (figure 8 (a) and (b)). This is not the case for the more difficult retina pictures (figure 8 (c) and (d)).

5.4 Analysis of Weights

This section investigates the network internal behaviour through visual analysis of the weights in trained networks. For presentation convenience, we use the two trained networks for the coin pictures.

Figure 9 shows a network architecture of 576-3-5 used in both the random and the centred weight initialisation methods for the coin pictures. The weight groups between the input nodes and the 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 10 and 11 correspond to weight groups (a), (b), (c) and (d) in this figure.

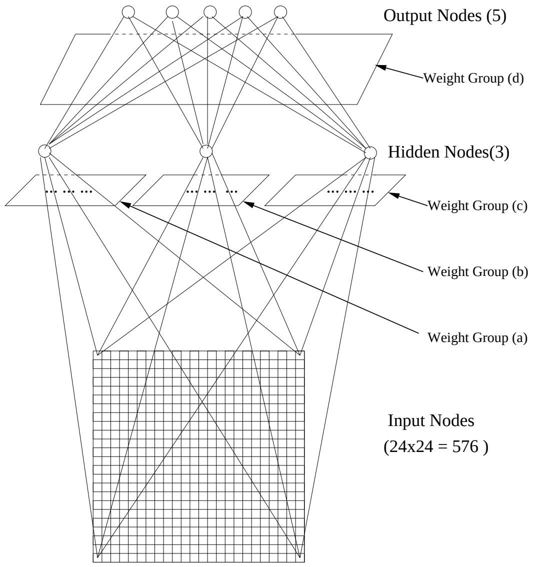
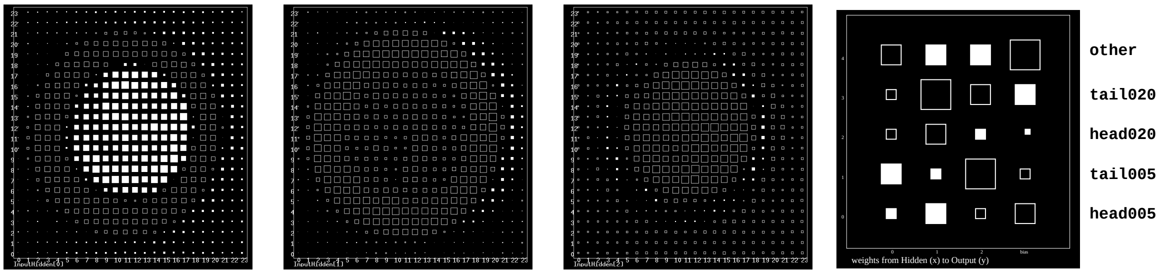


Figure 9: Network architecture with four weight groups for object detection in the coin pictures.

Figure 10 shows the weights from a trained 576-3-5 network which was successfully applied to the coin pictures. The network with these weights was trained with random initial weights. 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. Part (d) shows the weights from the hidden layer to the five output nodes 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, the second, the third hidden node (associated with weight matrices (a), (b), (c)) to the five output nodes and the biases of the five output nodes.

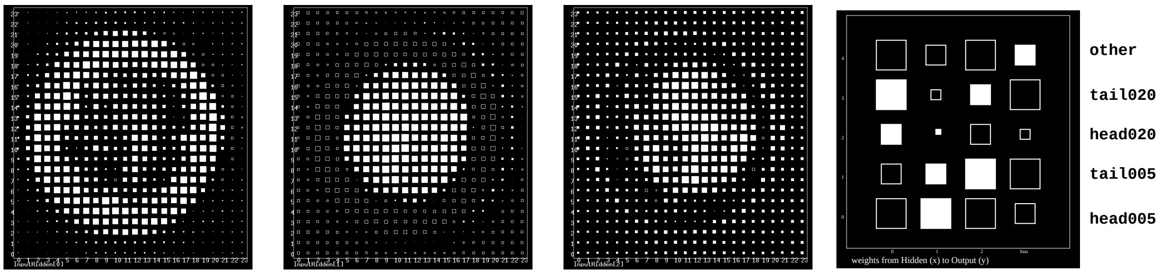
Inspection of the first column of figure 10 (d) reveals that weight matrix (a) has a positive influence on *head005* and a very strong positive influence on *tail*005. The same matrix has a negative effect on the other classes. Inspection of the second column reveals weight matrix (b) has



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

Figure 10: Weights in a trained network for object detection in the coin pictures based on random initial weights.

a positive influence 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. Weight matrix (c) 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 10 (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 background.



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

Figure 11: Weights in a trained network for object detection in the coin pictures based on the centred initial weights.

Figure 11 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 10 but in this case they are much better defined. According to this figure, weight matrix (a) has a strong positive effect on the 20 cent coins and a strong negative influence on the 5 cent coins; weight matrix (b) has a significant positive effect on the 5 cent coins and a slightly positive or a negative influence on the 20 cent coins; matrix (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 coins at the same time. This indicates that weight matrices (a) and (b) could distinguish the 20 cent coins from the 5 cent coins and matrix (c) separated all the tail sides from the head sides of these coins. In addition, all the three weight matrices have a very strong negative influence 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. It appears that the centred weights initialisation method has resulted in learning which is focused on features necessary to discriminate the classes.

5.5 Summary and Discussion

The goal of this section was to investigate a 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 were compared on three detection problems of increasing difficulty. On the easy (circles and squares), medium difficulty (coins) and very difficult (retinas) problems, it was possible to find centred initial weights which resulted in fewer training epochs and lower test mean squared error. More importantly, the centred weight method produced networks which were much better, in terms of detection rate and false alarm rate, at the task of detecting multiclass objects of interest in large pictures. The amount of improvement did not appear to be related to the difficulty of the problem.

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

Visualisation of the weights in trained networks resulting from both initialisation methods revealed that trained networks from both approaches contained feature detectors which "made sense" for the domain, but learning in networks with centred initial weights was more focused on features which discriminated between the classes.

The results also show that there were still false alarms for some classes. The false alarm rates were still high in the retina pictures. The next section will investigate whether detection performance can be improved using genetic algorithms to train and refine networks.

6 Genetic Algorithms for Network Training and Network Refinement

This section introduces a two phase approach, which uses genetic algorithms to train and refine the networks for object classification and detection. An overview of the two phase approach is shown in figure 12.

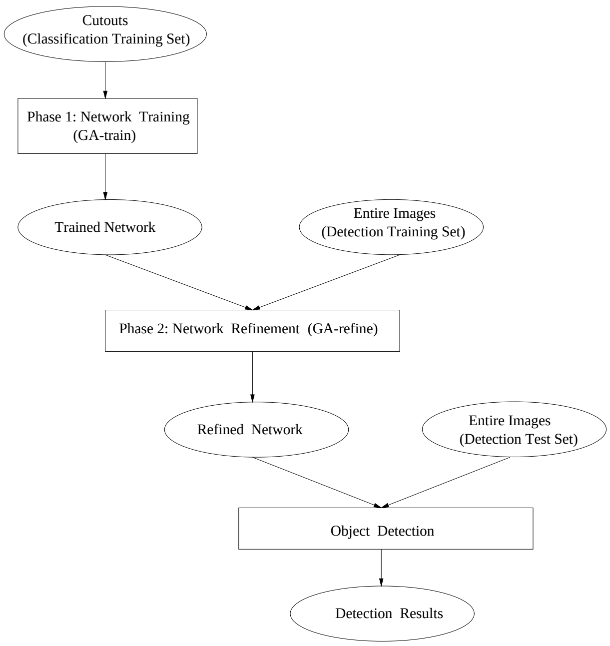


Figure 12: An overview of the two phase approach.

In the first phase, the network is trained on the cutouts for object classification. Rather than using the backward error propagation algorithm as in the basic approach, 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 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 ($GA - train$); (2) Network training by the first genetic algorithm followed by network refinement by the second genetic algorithm ($GA - train + GA - refine$); (3) Network training by the backward error propagation algorithm in the basic approach followed by network refinement by the second genetic algorithm ($BP - train + GA - refine$).

6.1 Genetic Algorithm for Network Training – GA-train

The first phase of this approach requires a network to be trained on the cutouts in the classification data set. We use the *2DELTA-GANN*[1] [21, 46] system as the first genetic algorithm. For presentation convenience, unless otherwise specified, a bias in the network is treated as just another weight in this section.

6.1.1 Gene Structure and Weight Updating Algorithm

A gene in the *GA-train* 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. 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. 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).

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 13. 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.

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.

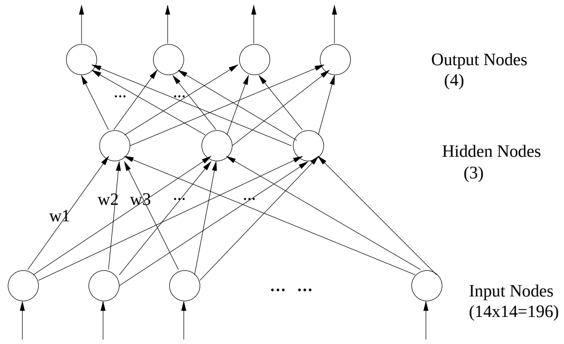
6.1.2 Sample Network with Chromosomes

Figure 14 shows a network architecture of 196-3-4 for 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 labeled as *w1, w2, ...* As mentioned earlier, the network architecture and the actual weights are not encoded in the genetic algorithm but saved outside.

[1]This algorithm was only tested on very small networks, that is, the *xor* and the *424 encoder* networks. In this paper, we investigate whether it can be applied to our large networks.

```
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 13: Weight updating mechanism in the genetic algorithm.



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

Supposing the original values of *w1* and *w2* are 0.34 and -0.49, we explain how the mechanism of updating weights is applied and the new weights are calculated.

For $w1$ in *network1*, $x1 = 1$, $x2 = 1$, $x3 = 0$, $delta1 = 0.13$ 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 $w2$, 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 $w1$ in *network2*, since $x1 = 0$, $delta1$ and $delta2$ will remain unchanged and the new value of $w1$ becomes $w1 = w1 + delta1 = 0.34 + 0.42 = 0.76$.

6.1.3 Fitness Function and Genetic Operators

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.

The commonly used biased roulette wheel mechanism [35] is used for parent selection.

The crossover operator is based on parameterised uniform crossover [26]. 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 15 shows an example of the crossover operator in the genetic algorithm.

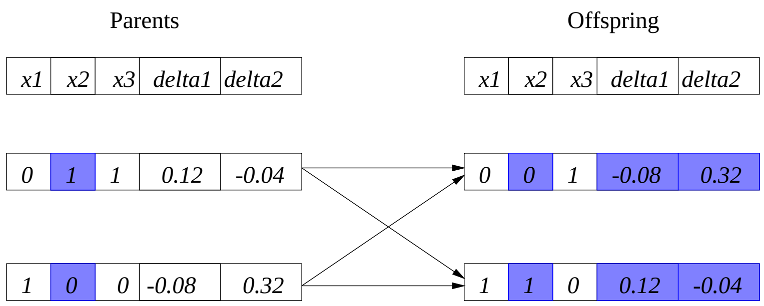


Figure 15: Crossover in the *GA-train* algorithm.

The mutation operator is based on the standard single bit mutation. If any bit on a gene is mutated, the $delta1$ and $delta2$ values for that gene will be replaced with randomly generated values. Figure 16 shows an example of the mutation operator in the genetic algorithm.

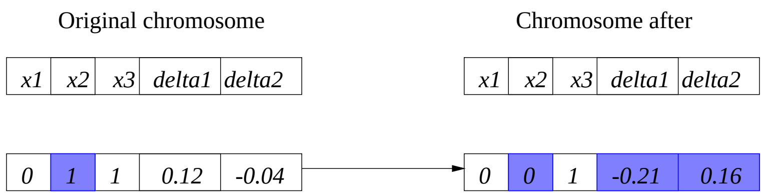


Figure 16: Mutation in the *GA-train* algorithm.

6.1.4 The Entire Genetic Algorithm

The entire genetic algorithm is as follows:

Step 1: Initialise the network weights to random values in (-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 set. 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 Genetic Algorithm for Network Refinement – GA-refine

The second phase of the approach involves improving the trained networks found in the first phase. This is done by another genetic algorithm.

6.2.1 Characteristics

The *GA-refine* algorithm uses the same encoding of chromosomes and genetic operators as the first genetic algorithm, *GA-train*. The main differences between them are as follows:

- 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 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 on the entire images is used instead of the mean squared error on the cutouts. The details will be described in the next subsection.
- Different parameters and termination criteria are used. The details will be presented in section 6.3.

6.2.2 Fitness Function

In network refinement, the fitness of a chromosome is calculated as follows:

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 to locate the object of interest. This is identical to the corresponding detection procedure used in the basic approach except that this is now performed as part of the training procedure.
3. Compare the detected object centres with known locations of the desired objects, obtain the number of objects correctly and incorrectly detected by the network, and determine the overall detection rate and false alarm rate of this network.
4. Compute the fitness according to equation 6.

$$fitness(FAR, DR) = A * FAR/(FAR + DR) + B * (1 - DR) \tag{6}$$

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[2].

Under this design, 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 Results

6.3.1 Object Classification Results

This subsection presents the object classification performance of the *GA-train* method on the cutouts of the three databases. The number of *evaluations* and the test mean squared error are compared with those obtained by the *BP-train* algorithm in the basic approach.

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

[2]Theorectically, A and B could be replaced by a single parameter since they have only one degree of freedom. However, the two cases have different effects for stopping the evolutionary process.

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.

The main parameters used in the *GA-train* algorithm for the three problems is shown in table 22.

Parameters	Easy pictures	Coin pictures	Retina pictures
Population size	200	200	300
Crossover rate	95%	95%	90%
Mutation rate	5%	5%	10%
Generations	100	100	150
Delta1 range	± 0.06	± 0.01	± 0.004
Delta2 range	± 0.02	± 0.004	± 0.002
Net Architecture	196-4-4	576-3-5	256-4-5

Table 22: The main parameters used for the network training in the *GA-train* algorithm for the three detection problems.

Table 23 shows a comparison of the network training and testing results for the two algorithms for the easy pictures. The number of evaluations needed by the *GA-train* algorithm is 5146.6 ± 794.4, which is much bigger than the 398.8 ± 36.18 for the *BP-train* algorithm 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	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 23: 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.)

A similar series of experiments were performed for the coin and the retina pictures. The results are shown in table 24 and table 25.

As can be seen from these tables, on all cases for training the relatively large networks with

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 24: 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.)

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 25: 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 inputs of the image pixel data described here, the *GA-train* algorithm needed more evaluations 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.

6.3.2 Object Detection Results

This section describes a series of comparisons of the object detection results of the three methods introduced in this section with the basic approach.

Table 26 shows the main parameters used by the *GA-refine* algorithm in the *GA-train+GA-refine* method and 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 *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 26 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.

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

Table 27 shows a comparison of the results for the easy pictures. As in the basic approach, the detection rate for the three classes obtained using the three methods introduced in this section 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. 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.

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 28.

According to table 28, 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 17. As can be seen from this figure, at all levels of detection

Easy Pictures		Object Classes		
		Class1	Class2	Class3
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	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 27: Comparison of object detection results for the easy pictures using the four detection methods. (Network architecture: 196-4-4; Repetitions = 15.)

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
	GA-train	357	19	333	778
	BP-train + GA-refine	0	0	0	0
	GA-train + GA-refine	125	0	37.5	215

Table 28: Comparison of object detection results for the coin pictures using the four detection methods. (Network architecture: 576-3-5; Repetitions = 15.)

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*.

Table 29 shows a comparison of the best detection rate and the corresponding false alarm rate achieved using the four detection methods on the retina pictures. These results were not as good as those for the easy and the coin pictures. 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 18 describes the results in ROC curves for classes *haem* and *micro*, which show the same pattern as the other two databases except that the *GA-train* method produced fewer false alarms than the *BP-train* method at most detection rates for detecting class *micro*. However, it could not achieve 100% detection rate.

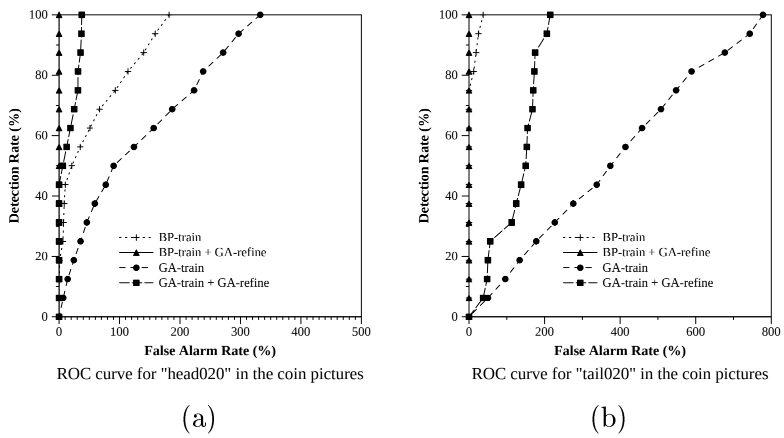


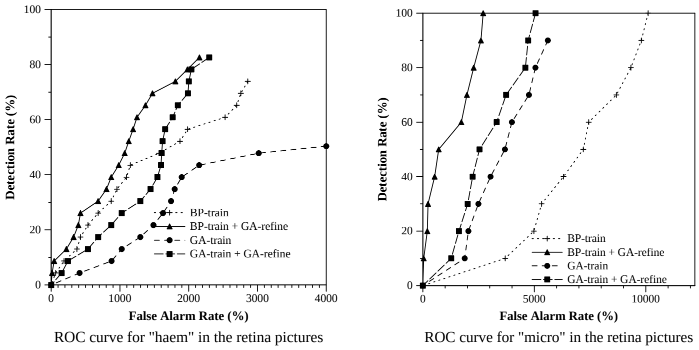
Figure 17: Comparison of the results for class *head020* and class *tail020* in the coin pictures using the four detection methods.

Retina Pictures	Object Classes	
	haem	micro
Detection Results	Best DR (%)/FAR(%)	Best DR(%)/FAR(%)
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 29: 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.)

6.4 Summary

This section investigated the use of genetic algorithms for network training and network refinement for the object classification and detection problems. The three methods introduced in the two phase approach and the basic approach 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 or better test performance on the cutouts of any of the three databases, and without the network refinement it did not give better detection performance in most cases, compared with the basic approach. The methods which incorporated the refinement genetic algorithm always resulted in better detection performance than the corresponding methods without the refinement in all three databases. In other words, the second genetic algorithm, *GA-refine*, improved the detection performance of networks trained by either the backward error propagation algorithm or the first genetic algorithm. Of the four methods, the integration of the backward error propagation



(a) (b)

Figure 18: Comparison of the results for detecting class *haem* and class *micro* in the retina pictures using the four detection methods.

algorithm and the refinement genetic algorithm, the *BP-train + GA-refine* method, always produced the best detection performance.

7 Conclusions

The goal of this paper was to investigate a learning/adaptive, domain independent approach to multiple class, translation and 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 method that gave the best detection performance was the refinement genetic algorithm based on the network trained by the backward error propagation algorithm, *BP-train+GA-refine*. This method 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 paper, while not satisfactory, were consistent with results obtained by alternative methods on similar problems.

We expect that the methods described in this paper could be successfully used for any databases similar to the easy and the coin pictures, that is, detecting multiple class, translation and rotation invariant objects with size less than or equal to 24×24 pixels against a relatively uniform background.

The methods described here were not particularly effective in detecting haemorrhages in retina pictures, which suggests that they could not be used to detect objects with different sizes which are in the same class.

The size of a 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.

This paper has the major contributions as follows:

- This work has shown how object detection systems can be developed with image pixel inputs and how traditional specific feature extraction and selection can be avoided. This was done by the direct use of image pixels as inputs to neural networks in which the features relevant to a particular domain can be automatically learnt through a learning/adaptive process.

- This work has shown how to solve multiple class object detection problems by a single learnt network. 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 neural network for both object classification and object localisation in multiple class object detection problems.

- 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.

The genetic algorithm for network refinement involve long training times. We are investigating ways of shortening the training time. This could be greatly improved with parallel hardware.

To achieve the goal of domain independence, we use the raw pixel values directly as inputs to neural networks. However this led to large networks to be defined and trained, and accordingly longer training time is needed. In the future, it might be very interesting to investigate another type of network inputs, say, *pixel statistics* or pixel level, domain independent image features such as the means and standard deviations of some local regions or edges in the image cutouts.

Acknowledgement

We would like to thank Dr. James Thom and Dr. Zhi-Qiang Liu for a number of useful discussions on image processing and retrieval, and Chris Kamusinski for providing and labeling the retina pictures.

References

[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] Subutai Ahmad and Stephen 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] Subutai Ahmad and Volker Tresp. Some solutions to the missing feature problem in vision. In Cowan J.D. Hanson S.J. and Giles 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] M. Anthony and N. Biggs. *PAC learning and neural networks*, pages 694–697. In The Handbook of Brain Theory and Neural Networks, 1995.

[6] Birsel Ayrulu and Billur Barshan. Neural networks for improved target differentiation and localization with sonar. *Neural Networks*, 14:355–373, 2001.

[7] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck. Underwater target classification using wavelet packets and neural networks. *IEEE Transactions on Neural Networks*, 11(3):784–794, May 2000.

[8] K. Balakrishnan and Vasant 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.

[9] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Englewood Cliffs, N.J: Prentice-Hall, Inc., 1982.

[10] Billur Barshan, Birsel Ayrulu, and Simukai W. Utete. Neural network-based target differentiation using sonar for robotics applications. *IEEE Transactions on Robotics and Automation*, 16(4):435–442, Augest 2000.

[11] Richard K. Belew, John McInerney, and Nicol 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.

[12] Ann Bernardon and James 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.

[13] Holger Bosch, Ruggero Milanese, and Abderrahim 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.

[14] Frank Z. Brill, Donald E. Brown, and Worthy N. Martin. Fast genetic selection of features for neural network classifiers. *IEEE Transactions on Neural Networks*, 3(2):324–328, March 1992.

[15] R. Brunelli and T. Poggio. Face recognition through geometrical features. In S. Margherita Ligure, editor, *Proceedings of ECCV '92*, pages 792–800, 1992.

[16] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on PAMI*, 15(10):1042–1052, 1993.

[17] 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.

[18] G. A. Carpenter and S. Grossberg. Stable self-organisation of pattern recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[19] David P. Casasent and Leonard M. Neiberg. Classifier and shift-invariant automatic target recognition neural networks. *Neural Networks*, 8(7/8):1117–1129, 1995.

[20] 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.

[21] Victor Ciesielski and Jeff 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, April 1998.

[22] Nicola Cross and Roland Wilson. Neural networks for object recognition. Technical report, Department of Computer Science, University of Warwick, Coventry, October 1995.

[23] Dick 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.

[24] Dick de Ridder, Aarnoud Hoekstra, and Robert 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.

[25] M. Dehghan, K. Faez, and M. Ahmadi. A hybrid handwritten word recognition using self-organizing feature map, discrete hmm, and evolutionary programming. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume*, Como, Italy, July 2000.

[26] K.A. DeJong and W.M. Spears. On the virtues of parameterised uniform crossover. In Richard K. Belew and Lashon B Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, July 1991. Morgan Kaufman.

[27] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks - a review. *Pattern Recognition*, 2001. Preprint, to appear in Pattern Recognition, 2001.

[28] H. M. El-Bakry, M. A. Abo-Elsoud, and M. S. Kamel. Fast modular neural nets for human face detection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume III*, Como, Italy, July 2000.

[29] Olivier Faugeras. *Three-Dimensional Computer Vision – A Geometric Viewpoint*. The MIT Press, 1993. ISBN 0-262-06158-9.

[30] Raphael Feraud, Olivier J. Bernier, Jean-Emmanuel Viallet, and Michael Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, Janurary 2001.

[31] J.F. Fontanari and R. Meier. Evolving a learning algorithm for the binary perceptron. *Network*, 2:353–359, November 1991.

[32] Kunihiko Fukushima, Eiji Kimira, and Hayaru 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.

[33] Paul D. Gader, Joseph R. Miramonti, Yonggwan Won, and Patrick Coffield. Segmentation free shared weight neural networks for automatic vehicle detection. *Neural Networks*, 8(9):1457–1473, 1995.

[34] G. L. Giles and T. Maxwell. Learning, invariances, and generalisation in high-order neural networks. *Applied Optics*, 26(23):4972–4978, 1987.

[35] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine learning*. Addison Wesley, Reading, Ma, 1989.

[36] Robert 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.

[37] S. Holden and M. Niranjan. On the practical applicability of VC dimension bounds. *Neural Computation*, 7(6):1265–1288, 1995.

[38] Ayanna Howard, Curtis Padgett, and Carl Christian 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.

[39] Jack S. N. Jean and Jin Wang. Weight smoothing to improve network generalisation. *IEEE Transactions on neural networks*, 5(5):752–763, September 1994.

[40] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, Cambridge, Massavhusetts, London, England, 1994.

[41] M. Khabou, P. D. Gader, and J. M. Keller. Ladar target detection using morphological shared-weight neural networks. *Machine Vision and Applications*, 11(6):300–305, May 2000.

[42] Moon W. Kim and Mohammed 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.

[43] Moon W. Kim and Mohammed 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.

[44] Teuvo Kohonen. *Self-organization and Associative Memory.* Springer, Berlin Heidelberg New York, 3rd edition, 1988.

[45] Peter 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.

[46] Rajendra Krishnan and Victor 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.

[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard 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.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[50] 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.

[51] Chulhee Lee and David A. Landgrebe. Decision boundary feature extraction for neural networks. *IEEE Transactions on Neural Networks*, 8(1):75–83, 1997.

[52] Shang-Hung Lin. An introduction to face recognition technology. *Special Issue on Multimedia Technologies and Informing Science, Part II, Information Science, the International Journal of an Emergin Discipline*, 3(1):1–7, 2000.

[53] Shang-Hung Lin, Sun-Yuan Kung, and Long-Ji lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1):114–132, Jan 1997.

[54] Xiuwen Liu, Deliang Wang, and J. Raul 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.

[55] Shih-Chung B. Lo, Heang-Ping Chan, Jyh-Shyan Lin, Huai Li, Matthew T. Freedman, and Seong K. Mun. Artificial convolution neural network for medical image pattern recognition. *Neural Networks*, 8(7/8):1201–1214, 1995.

[56] Xi Miao, Mahmood R. Azimi-Sadjadi, Abinash C. Dubey, and Ned 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.

[57] 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.

[58] 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.

[59] 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.

[60] Steven K. Rogers, John M. Colombi, Curtis E. Martin, James C. Gainey, Ken H. Fielding, Tom J. Burns, Dennis W. Ruck, Matthew Kabrisky, and Mark Ocley. Neural networks for automatic target recognition. *Neural Networks*, 8(7/8):1153–1184, 1995.

[61] 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.

[62] 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.

[63] Michael 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.

[64] Michael W. Roth. Survey of neural network technology for automatic target recognition. *IEEE Transactions on neural networks*, 1(1):28–43, March 1990.

[65] 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.

[66] Mukul V. Shirvaikar and Mohan M. Trivedi. A network filter to detect small targets in high clutter backgrounds. *IEEE Transactions on Neural Networks*, 6(1):252–257, Jan 1995.

[67] Sameer Singh, Markos Markou, and John Haddon. Natural object classification using artificial neural networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume III*, Como, Italy, July 2000.

[68] F. Fogelman 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.

[69] Lilly Spirkovska and Max B. Reid. Higher-order neural networks applied to 2D and 3D object recognition. *Machine Learning*, 15(2):169–199, 1994.

[70] C. Stahl, DaimlerChrysler Aerospace, and P. Schoppmann. Advanced automatic target recognition for police helicopter missions. In F. A. Sadjadi, editor, *Proceedings of SPIE Volume 4050, Automatic Target Recognition X*, April 2000. [4050-30].

[71] 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.

[72] Bin Tian, Mahmood R. Azimi-Sadjadi, Thomas H. Vonder Haar, and Donald 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.

[73] 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.

[74] 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.

[75] 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.

[76] Brijesh 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.

[77] Brijesh 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.

[78] P. D. Wasserman. *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York, 1993. Neural Engineering, Chapter 11. ISBN: 0-442-00461-3.

[79] Allen M. Waxman, Michael C. Seibert, Alan Gove, David A. Fay, Ann Marie Bernandon, Carol Lazott, William R. Steele, and Robert K. Cunningham. Neural processing of targets in visible, multispectral ir and sar imagery. *Neural Networks*, 8(7/8):1029–1051, 1995.

[80] T. Wessels and C. W. Omlin. A hybrid system for signature verification. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume V*, Como, Italy, July 2000.

[81] 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.

[82] 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.

[83] Philip Winter, Shahab Sokhansanj, Hugh C.Wood, and William 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.

[84] Yonggwan Won, Jiseung Nam, and Bae-Ho Lee. Image pattern recognition in natural environment using morphological feature extraction. In Francese J. Ferri, Jose M. Iñesta, Adnan Amin, and Pavel Pudil, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR 2000 and SPR 2000, Lecture Notes in Computer Science, Vol. 1876*, pages 806–815. Springer, August/September 2000.

[85] Yee Chin Wong and Malur 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.

[86] 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.