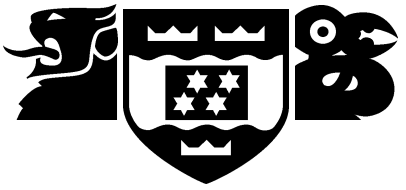
VICTORIA UNIVERSITY OF WELLINGTON
*Te Whare Wananga o te Upoko o te Ika a Maui*



Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

# Multiple Class Object Detection using Pixel Statistics in Neural Networks

Roy Chow

Supervisor: Dr Mengjie Zhang, Dr Peter Andreae

18 October, 2002

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

**Abstract**

In this project we investigate a domain independent approach to the multiple class object detection problem. We use a pixel statistics based approach with a neural network within a recognition system. The neural network used a square input field large enough to contain any single object of interest and was trained via the back propagation algorithm. Several important factors were discovered during the course of the project that contributed to the success of object detection based on pixel statistics and the solutions found were incorporated into the recognition system when testing the image database. The recognition system was tested with a database containing low resolution images ranging from the simple to the very difficult. All databases contained images of New Zealand 5 and 10 cent coins except one, which contained artificially generated images. We used low resolution images for reasons of efficiency, but we found this affected the recognition system when it was required to classify more difficult objects, such as the heads and tails side of the coins. The recognition system had most difficulty with detecting the tail side of coins, as the reduced resolution of the images eliminated much valuable information about the intricate tail design, such as the numerical digits.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

If the reader views figure 1.1 below it should be a trivial task to differentiate the 5 cents from the 10 cents, and the heads from the tails. Not only that, the reader would also find it trivial to state the locations of the 5 cent coin with tails topmost.


Figure 1.1: Image for object detection

The reader, by differentiating between the 5 cents and 10 cents, as well as the respective heads and tails of each coin, has performed **object classification**. By stating the location of 5 cent coin with tails topmost, the reader has performed **object localisation** for the 5 cent coin with the tails topmost.

Performing the above tasks for the 5 cent coin with heads topmost, followed by the 10 cent coin with heads/tails topmost on figure 1.1, the reader has performed **multiple class object detection** - where both object classification and object localisation are performed for each class. A class refers to instances of an object of interest, such as the 5 cent tails. 5 cent heads constitute another class, as learning how to recognise a 5 cent tail would not allow one to recognise a 5 cent head (even though one knows *how* to recognise a 5 cent).

Multiple class object detection is trivial for a human, but it is much more difficult for a computer to perform. In this project, we use neural networks (which are structures within a computer attempting to mimic the function of the human brain) to perform multiple class

object detection. Neural networks have attracted much attention in recent years as a promising method of solving detection problems.

Applications have included finding human faces in a photograph, tracking moving targets in real time video [4] and medical applications such as detecting hemorrhages in retina images[5].

Two main kinds of approaches have been used in neural networks for object classification - raw pixel based and feature based. The raw pixel approach inputs the raw pixel data into the network in which objects of interest are located into the recognition system. This approach is domain independent and avoids the hand crafting of features, but suffers from long training times as well as becoming very inefficient as the size of the image increases. In the feature based input approach, desired features (usually domain independent) are extracted from a larger image and fed into the neural network as input. However, selecting and extracting good features is usually very time consuming and programs for feature extraction and selection often have to be hand crafted.

In this project, we investigate a domain independent approach to the multiple class object detection problem by using a small number of pixel statistics as input. Pixel statistics are low level, domain independent features such as the mean and standard deviation of an image.

## 1.1 Goals

The goal of the project is to develop a domain independent approach to multiple class object detection using neural networks with pixel statistics. The recognition system developed will be applied to five databases containing images ranging from the simple to the very difficult. Specifically, we need to investigate the following:

- What are the effects of different statistics in object detection?
- What is the effect of a moment feature?
- What is the effect of regions in object detection, and role do they play?
- What happens to the performance as the difficulty of the problem increases?
- How does the recognition system perform on low resolution images in object detection?

An important subtask that arose during the project concerned the training data. Since the construction of the training and testing examples are of paramount importance in object detection problems, we had to develop a set of guidelines for constructing multiple training examples.

# Chapter 2

# Literature Survey

## 2.1 Neural Networks

### 2.1.1 Neural Networks and the Human Brain

In 1911, Ramon Y Cajal introduced the idea of a neuron as one of the fundamental constituent units of the human brain. Though the neuron has a relatively slow rate of operation, the brain compensates for this by having a staggering amount of them. The neurons interconnect, forming an extremely efficient structure. These neurons can be organised in many ways to perform computations such as pattern recognition, perception and motor control.

In humans, the most intense phase of "wiring" (ie forming the interconnections between neurons) occurs in the first two years of life. However, the wiring phase does not stop there but continues throughout life, albeit at a slower rate. This wiring corresponds to what we call "experience."

Thus in its most general form an artificial neural network is a structure within a machine which attempts to mimic the function of the human brain, by employing a large number of neurons with their associated interconnections. The strength of these interconnections are modeled using number values known as "weights". The values of the weights correspond to experience in human terms.

Learning in neural networks occurs via a process in which the aforementioned synaptic weights are updated in a systematic fashion so as to obtain a desired design objective. Neural networks may also be viewed as directed graphs, which motivates a mathematical definition of neural networks. More details can be found in Haykin [3].

### 2.1.2 Representation of the Neuron

A neuron is the fundamental unit of a neural network and essential to its successful operation. Figure 2.1 shows a model of an artificial neuron.

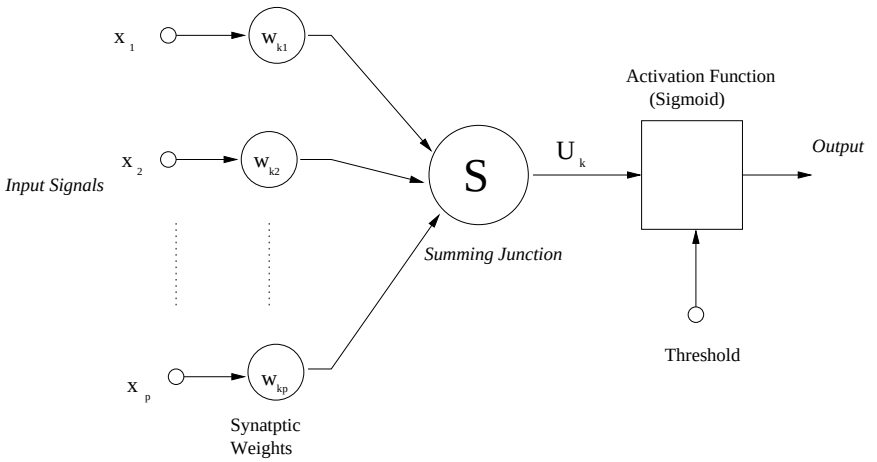



Figure 2.1: An artificial neuron

The components are as follows:

1. A set of connecting links (sometimes called synapses) with each one characterised by a unique weight. A signal $x_j$ at the input end of synapse $j$ connected to neuron $k$ is multiplied by the synaptic weight $w_{kj}$ . To remember how to read this, one can remember the synaptic weight $w_{kj}$ is said as "the synaptic weight of output neuron $k$ given input neuron $j$."

2. An adder for summing the input signals, producing an output $U_k$ which is given by the following formula:

$$U_k = \sum_{j=1}^{p} \; w_{kj} \; x_j$$

3. An activation function, which computes the output value of the neuron. For example if the activation is a sigmoid function (described below) the output values of the neuron are restricted to lie within the range [0,1].

### 2.1.3 Types of Activation Function

Various activation functions can be used to define a neuron's output. In this project we use an activation function known as the sigmoid function which is defined as:

$$\varphi(v) = \frac{1}{1 + e^{-av}}$$

where $a$ is the slope parameter of the sigmoid function. It is a strictly increasing function.

### 2.1.4 Neural Network Architectures

**Overview** Generally single or multiple layers are found in any neural network. The single layer case consists of an input layer accepting incoming input, followed immediately by an output layer.

Multilayered networks also consist of an input layer, but are followed by one or more hidden layers before the output layer is reached. We present an overview of the main architectures below:

1. **Single-Layer Feedforward Networks**
   Consisting only of an input and an output layer of nodes, the single layer type of network is a feedforward network since the input layer can project onto the outer layer of neurons, without the reverse occurring. The single layer refers to the outer layer of neurons and no hidden layers are present under this arrangement.

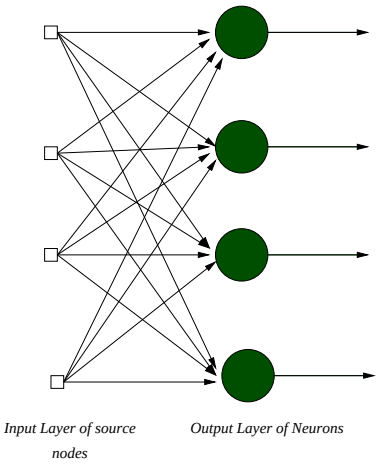



Figure 2.2: A single-layer feedforward network

2. **Multi-Layer Feedforward Networks**
   The distinguishing feature of these networks is the presence of one or more *hidden layers* which contain computation nodes called *hidden neurons* or *hidden units*. The purpose of having hidden units is to extract additional information, which is valuable especially if the input size is large. The network shown on page 6 is called *fully connected* in that every node in each layer of the network is connected to every other node in the next layer up. The other configuration when some synaptic connections are omitted is known as a *partially connected* network.

3. There are other network architecture including **Recurrent Networks** and **Lattice Structures**, but as they are not used in this project we refer the reader to Haykin [3] for further details.

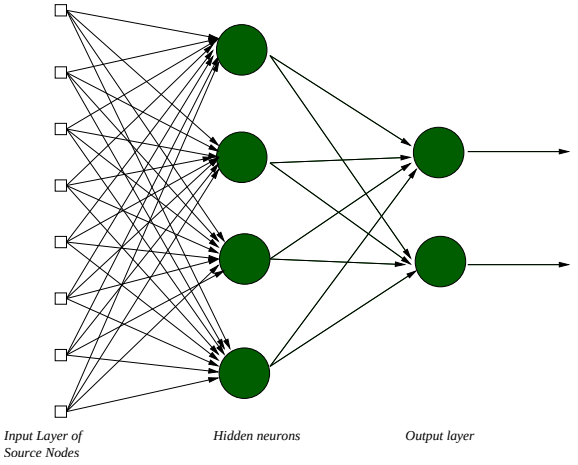



Figure 2.3: A multi-layer feedforward network

### 2.1.5 Learning in Neural Networks

**Definition** Haykin [3] defines learning in the context of neural networks as follows:

> *"Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place."*

In this project the *"free parameters"* referred to are the weights in the neural network, and the *"stimulation by the environment"* are the inputs and target (desired) outputs.
Upon receiving new input, we update our weights via a learning algorithm called **Back Propagation**.

### 2.1.6 The Back Propagation Algorithm

#### The Two Passes of Computation

There are two passes of computation which occur in the algorithm. The first pass is known as the *forward pass*, where the weights remain unchanged whilst propagating the input forward until it reaches the outer layer. In training, the input in this project is taken from a training set, which is a collection of input patterns or features from which the rules may be extracted, and in testing the input is taken from a test set which is a collection of unseen input patterns.

The second pass is known as the *backward pass* (or back propagation). Starting from the outer layer, the error signal $\delta$ (ie the difference between the target or desired output and the actual output) is passed backwards, layer by layer, whilst recursively computing $\delta$ for each neuron, each then computing its own error value from the layer above.

**The Algorithm**

1. *Initialisation.*
   Set all weights and biases of the network to small, uniformly distributed, random numbers.

2. *Presentation of training examples.*
   Present the network with training examples which it will be expected to learn in one cycle of the algorithm. Steps 3 and 4 below are to be performed for every training example.

3. *Forward pass.*
   Propagate the input until it reaches the output layer.

4. *Backward pass.*
   Compute the difference between the actual output and the desired output; the resulting difference $\delta$ is then used in conjunction with the output node to calculate a change in weight. To calculate the errors of the hidden nodes, for which no target outputs exist, the $\delta$ of the layers above are used instead. In this way all errors are propagated and the weights adjusted.

5. *Iteration*
   Steps 2-4 are iterated until a termination criterion is met.

### 2.1.7 Termination of Training

Criteria must exist so that the network knows when to terminate the training process. We list a selection below [5].

**Epoch/Cycle Control Strategy** Training will terminate once the number of epochs reaches a user defined number. An epoch is one complete presentation of the entire training set.

**Error Control Strategy** We use the mean squared error (MSE) as a stopping criterion.The training stops once the MSE is lower than a user defined number.

**Proportion Control Strategy** When the proportion of the number of patterns correctly classified among the number of the total training set reaches a pre defined percentage, the training will be terminated.

**Early Stopping Strategy** This is used to avoid overtraining, in which the network learns the training set too well and cannot apply it's knowledge to any other situation apart from the one it has been trained on.

**User Control Strategy** The user manually stops the training if he/she deems further training is unnecessary.

We use the proportion control strategy in this project.

## 2.2 The Object Detection Problem

The task of object localisation involves locating the exact position of the centre of an object of interest, and the task of object classification is to classify smaller objects within a larger image into their correct class. The amalgamation of both tasks is object detection, which is what we investigate in this project.

In a single class detection problem, only one class of interest exists within an image. This does not include the background, which is a class in its own right. The aim in this case would be to detect the location of the centres of all objects of this particular class within the larger image. Since only one class exists, once the object is located it is trivial to correctly classify it.

This project addresses the multiple class detection problem which extends the single class detection problem to incorporate more than one class (excluding the background). For example, both squares, circles and triangles may be present within an image - then the system must correctly classify and locate the object. We discuss the various approaches to the object detection problem in the next section.

### 2.2.1 Approaches to Object Detection Using Neural Networks

Various approaches using neural networks have been investigated. The following approaches have achieved reasonable success:

- **Raw pixel input**
  This simply inputs the raw pixel data of the image in which the objects of interest are located into the recognition system. This approach avoids the hand crafting of features, but suffers from long training times as well as becoming very inefficient as the size of the image increases. This approach usually needs learning and adaptive techniques to learn features for the detection task.

- **Feature based input**
  Desired features (usually domain independent) are extracted from a larger image and fed into the neural network as input. The drawback of this method is that an investigation of good features and time consuming hand-crafting of the appropriate feature extraction programs are needed. Features can be brightness, colour, or anything that can potentially distinguish the object of interest from others. The features are extracted and used as inputs into the recognition system.

### 2.2.2 Neural Networks for Object Detection

The following section is a brief review into the research done using neural networks for object detection in recent years. The following account is only a selection. More details of each can be found in the references.

#### Raw Pixel Input

By using raw pixel input, intensity values of both luminance and chrominance components of an image were used to detect face locations in videophone images [2]. This was performed by exploiting the distribution property of the facial components. The process comprised five stages over eight different types of facial images with the detection of the facial region in all testing images. A 100% detection rate was achieved.

The next example using the raw pixel approach involved a highly successful attempt at bacterial growth detection [5]. As raw pixel input was used, the network architecture was quite large, but an excellent result of 99.32% correct classification was achieved.

#### The Feature Based Approach

This approach was used to extract moving targets from a real-time video stream, classifying them into predefined categories according to image-based properties and then robustly tracking them [4]. Moving targets were detected using the pixelwise difference between consecutive image frames.

A classification metric was applied to these targets to classify them into three categories: human, vehicle or background clutter. Once classified, targets were tracked by a combination of temporal differencing and template matching. The resulting system robustly identified targets of interest and rejected background clutter.

Moving targets were detected using the pixelwise difference between consecutive image frames. This example illustrates the main idea of the feature based approach: to extract desired features from the larger image and feed them into the neural network.

The approach which we investigate is called pixel statistics, which Chapter 3 describes in detail.

## 2.3 Evaluating the Recognition System

### 2.3.1 Evaluation Criterion

We use the following two measures for evaluating the performance of the our recognition system.

The detection rate is the number of objects correctly reported by a detection system as a percentage of the total number of actual objects in the database and the false alarm rate refers to non-objects which are reported as true objects and included in the total number of actual objects by the network.

- Detection Rate (DR) for class $i$ - This is defined as:

$$DR_i = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} N_{true}(i,j)}{\sum_{j=1}^{n}\sum_{i=1}^{m} N_{known}(i,j)} \times 100\% \tag{2.1}$$

  where:

  - $DR_i$ is the detection rate for class $i$
  - $n$ is the total number of images in the image database
  - $m$ is the total number of images in the image database
  - $N_{known}(i,j)$ is the number of actual known objects for the $i$th class in the $j$th image in database

- False Alarm Rate (FAR) - This is defined as:

$$FAR_i = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} N_{reported}(i,j) - \sum_{j=1}^{n}\sum_{i=1}^{m} N_{true}(i,j)}{\sum_{j=1}^{n}\sum_{i=1}^{m} N_{known}(i,j)} \times 100\% \qquad (2.2)$$

where:

- $FAR_i$ is the false alarm rate for class $i$
- $n$ is the total number of images in the image database
- $m$ is the total number of images in the image database
- $N_{known}(i,j)$ is the number of actual known objects for the $i$th class in the $j$th image in database
- $N_{true}(i,j)$ is the number of objects correctly reported by the detection system
- $N_{reported}(i,j)$ is the number of objects reported by a system for class $i$th class in image $j$

### 2.3.2 Thresholds

Once the neural network is trained, we then execute the image sweeping procedure using the trained network (see chapter 3, section 3.2). The sweeping procedure produces a set of conjectured locations where an object of a particular class could exist. This set is then compared with a set containing the true locations of the class's centres.

Within a user defined tolerance (generally a certain deviation of a number of pixels about the centre of the actual object), the system can decide if the input field whose centre is at the conjectured location is an instance of an object or not an instance of an object. To decide if the conjectured location is an instance of an object the system takes the conjectured location to be a conjecture of the centre of the actual object. The threshold is used to decide if the conjectured location should be accepted or rejected as an object.

If the threshold is set to a very high value, then a recognition system will assert an object is present only when the recognition system is very confident that an object exists. This implies that assuming as the recognition system is reasonable, a high threshold will generally mean a very low false alarm rate, *but* many of the real objects will be missed(ie, a low detection rate). The converse is true when the detection rate is set to a low value.

The ideal threshold will give a 100% detection rate and a 0% false alarm rate, but this is only possible if the recognition system is able to clearly distinguish the between the classes. Since such performance is seldom possible, the threshold must be tuned.

In this project, the threshold values are in the range $(0.0, 1.0)$. Varying the threshold gives various values of detection rate and the corresponding false alarm rate, which can then be used to plot a Receiver Operating Characteristic (ROC) curve showing the performance of the recognition system on the image.

### 2.3.3 Receiver Operating Characteristic (ROC) Curves

ROC Curves show the tradeoff between the false alarm rate and the detection rate. The curves can only be plotted when the recognition system is evaluating the quality of a uniform real valued measure that gets used via a recognition system with a continuous controllable parameter (in this case the threshold). Systems with binary outputs would not have ROC

curves, for example.

Although there are two types of ROC curves, the **Standard** ROC curve and the **Extended** ROC curve, we shall only discuss the extended version here as it is the only one relevant to the criteria we will be using for performance evaluation, namely the detection rate and false alarm rate.

In extended ROC curves, the false alarm rate forms the $x - axis$ and the detection rate forms the $y - axis$. A sample extended ROC Curve is shown in figure 2.4. As seen in figure 2.5, higher curves represent greater detection capacity. The closer to the $x - axis$ the curve is, the poorer the recognition system is. There is an ideal case, which corresponds to the top left hand corner, 100% detection rate and 0% false alarm rate is achieved.

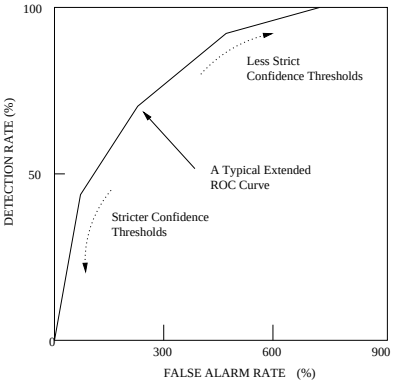



Figure 2.4: A typical extended ROC curve

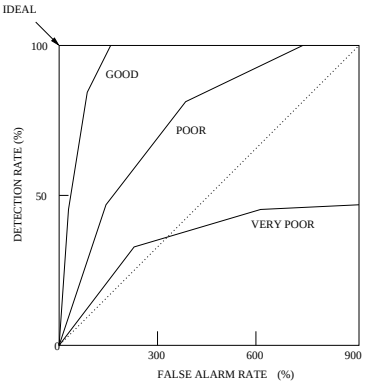



Figure 2.5: Quality of detectors with ROC Curves

# Chapter 3

# Pixel Statistics Based Approach to Object Detection Using Neural Networks

## 3.1 Introduction

We use pixel statistics as an alternative to using raw pixel or pixel based input in this project. Pixel statistics are low level, domain independent features computed from the pixel intensities of each image. The term pixel statistics refer to the statistical properties of the pixels. Standard statistical measures such as the mean and standard deviation can be found for any region of pixels about an image, which can then be used as input to a neural network for object detection.

The obvious advantage to this approach is unlike the raw pixel approach we will have far fewer inputs, and unlike the feature based approach, we avoid the hand crafting of features. Both methods were described in section 2.2.1.

## 3.2 Experimental Methodology

### 3.2.1 The Recognition System Used in Object Detection

In this section we describe the recognition system used for object detection in this project. The recognition system comprises six main stages, with each stage dependent on the previous stage. Figure 3.1 illustrates the main steps.

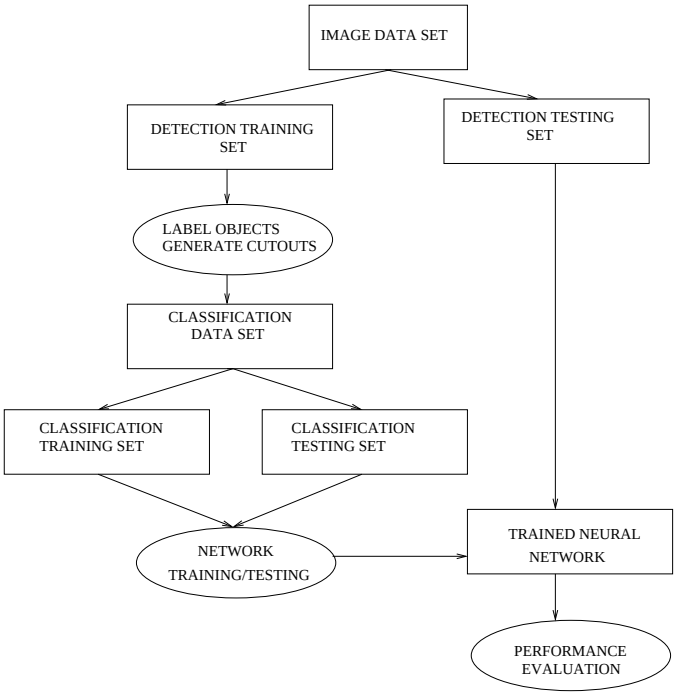



Figure 3.1: The recognition system used in object detection

### 3.2.2 Overview of the Approach

#### Creating Cutouts and Pattern files

Each cutout is a subimage of the main image, representing one instance of a class of interest. A cutout of a square is shown in figure 3.2.

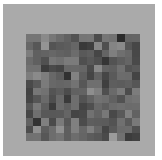


Figure 3.2: Cutout of a square

From the cutouts, we create a pattern file. The pattern file is used during the training and testing phases of the recognition system as input to the neural network, as well as providing the class which the set of features belong to. The general format of a pattern file contains a feature vector, whose components represent one pixel statistic feature of the cutout such as the mean or standard deviation, and the output vector representation of the class the feature vector belongs to.

The general format is shown below.

```
num_training_examples num_testing_examples
feature1 feature2 ....... featuren Class
feature1 feature2 ....... featuren Class
feature1 feature2 ....... featuren Class
feature1 feature2 ....... featuren Class
feature1 feature2 ....... featuren Class
feature1 feature2 ....... featuren Class

etc ...
```

A specific example of a pattern file used in this project is given below. This example contains feature vectors with six components corresponding to six pixel statistics, and the output representation for the neural network. Three classes are present in this example, with an output node for each class. A '1' signifies that the feature vector represents the class represented by that particular output node.

```
       Feature vector    Output representation
 -------------|-------------   ----|----
|                             | |         |
0.44 0.24 0.95 0.00 0.43 0.47 1.0 0.0 0.0
0.65 0.19 1.00 0.05 0.65 0.50 0.0 1.0 1.0
0.55 0.16 0.96 0.09 0.55 0.40 0.0 0.0 1.0
0.43 0.24 0.94 0.01 0.42 0.43 1.0 0.0 0.0
0.61 0.15 0.97 0.13 0.62 0.44 0.0 1.0 0.0

etc...
```

### Neural Network Training and Testing

The network is trained using the backward propagation algorithm as described in section 2.1.6. Training terminates using the proportion control strategy, which was described in section 2.1.7. When training is completed, the trained network is applied to the test set. The test set are the feature vectors in the pattern file which were not used in the training of the network.

Assuming performance on the test set is reasonable then we can proceed on to the sweeping stage (By "reasonable" we mean that an accuracy rate of about 70% or more is obtained. We generally do not require (or desire) 100% accuracy in the test set as this can lead to overtraining (see section 5.4 in chapter 5).

### Object Detection : Network Sweeping

The trained neural network is used to detect the classes and locations of the objects of interest in a large, unseen image. Both classification and localisation procedures are performed. The network is used as a template matcher and applied in a moving window fashion over the large image to detect the objects of interest. The template is swept across the image (from right to left) and down, pixel by pixel in every possible direction. This procedure is illustrated in figure 3.3.

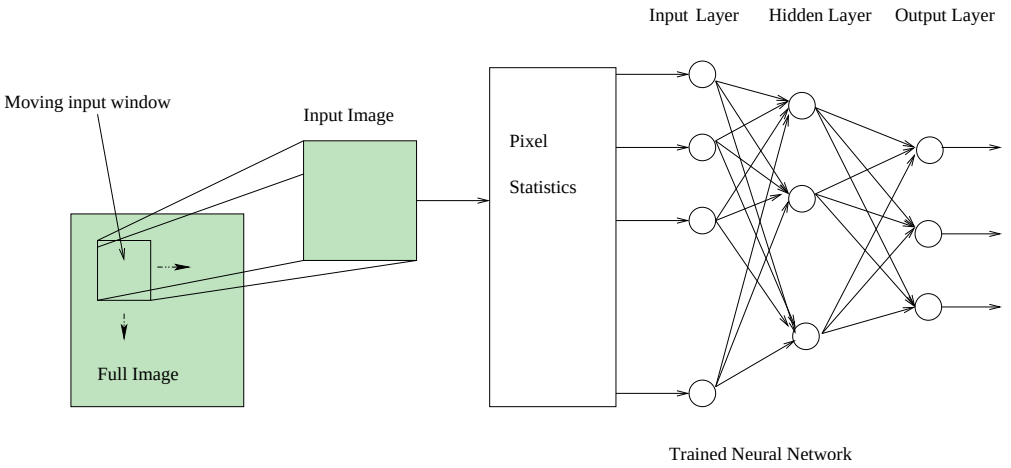
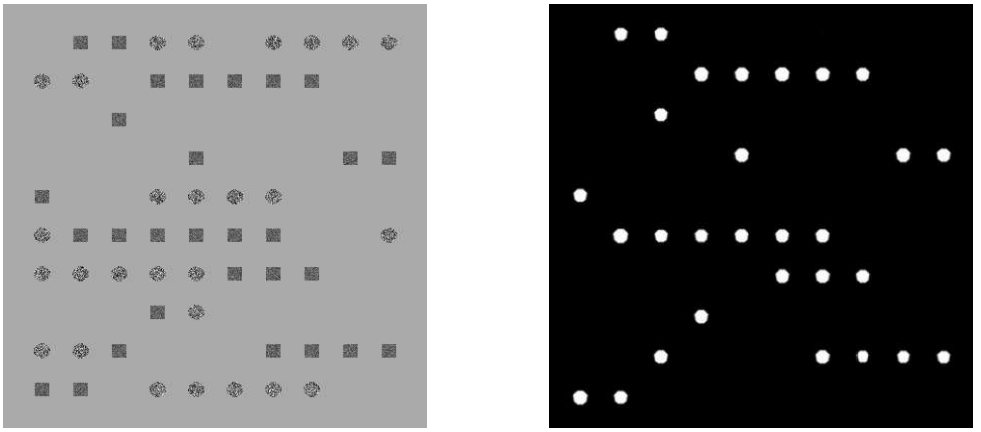



Figure 3.3: Network sweeping using a trained network with pixel statistics inputs

Upon completion of the sweeping process, object sweeping maps for each class, including the background are produced. A sample object sweeping map is shown in figure 3.4.



Original image Object sweeping map showing detected squares

Figure 3.4: A sample object sweeping map

The meaning of the object sweeping map produced is as follows. The recognition system attempts to identify the centre of the object belonging to a certain class for each object sweeping map by marking the area which it conjectures to be the centre by a white dot. Areas which it does not believe to be the object are marked with black. Should the system be unsure, shades of grey are used.

Shades of grey closer to white indicates the system has greater confidence that this particular region is likely to be the centre of an object than a shade of grey which is closer to black. Ideally, a perfect recognition system would would produce object sweeping maps of each class with white dots in the exact centre of the object in the correct class, and black everywhere else, but this ideal is almost impossible to achieve.

### The Centre Finding Program

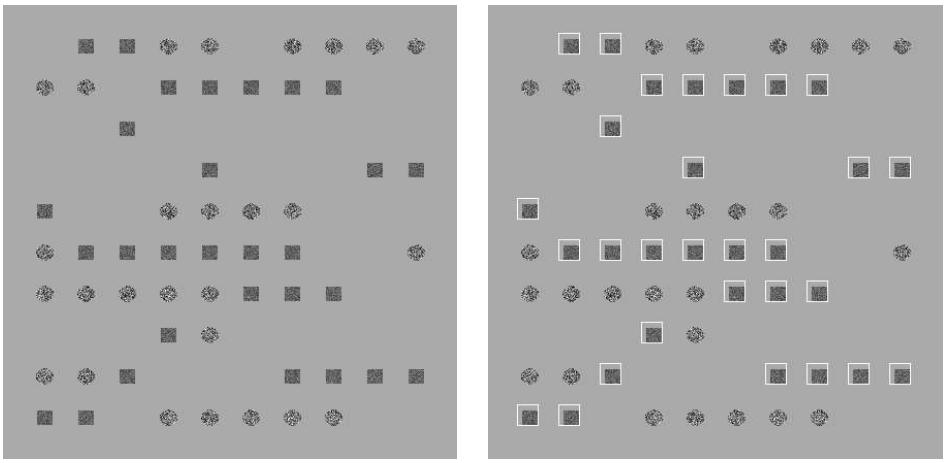
A centre finding algorithm [5] is used to find the centres of the detected objects whilst executing network sweeping. For each class of interest, the centre finding algorithm is used to find the centres of the objects in the class based on the corresponding object sweeping map.

A tolerance measure is used to indicate how much deviation from the true centre is permitted. This is known as the tolerance boundary. The greater the tolerance measure, the more lenient the recognition system will be in permitting a conjectured location to be considered a detected object. The smaller the tolerance measure, the more strict the recognition system will be in permitting a conjectured location to be considered a detected object, since the conjectured location must be very close to the actual location of the object. The former increases both detection rate and false alarm rates, whilst the latter reduces both detection rate and false alarm rates.

### Image Cuts

Image cuts are images of the original image swept during the sweeping procedure augmented with white square frames around the detected objects. Should more than one object be detected in any location, more frames will appear. Each image cut corresponds to one class, and frames over objects not in the class indicate the presence of false alarms.

In the ideal case where perfect detection is achieved, only one frame around each object in the desired class would be present. Figure 3.5 shows the image cut for the square class, where perfect detection is achieved.



(a) Original image (b) Class 1: Image cut of squares

Figure 3.5: Image Cuts

## 3.3 Issues Which Affect the Properties of Pixel Statistics

We list some criterion which will affect the statistical properties of the pixels:

- **Resolution** refers to the smallest size of elements within the image such that the features can still be distinguished. The features in this and the following cases are the parts of an object such as the digits of the tail side of a coin. When resolution is reduced, we may lose this information completely, and hence it will not contribute to the pixel statistics which will be used in detection. The recognition system cannot distinguish objects if the distinction requires that it must first have some representation of that particular feature in terms of a statistic which it contributes to. In this project we reduce the resolution of the coin images so that the largest coin can fit into a $20 \times 20$ input field. The main reason for this is efficiency during network sweeping (See section 3.2.2.
- **Invariance** refers to features with rotational or scale invariance (Note that scale is not investigated in this project). Pixel statistics are generally invariant, with the exception of the moment, which we maintain rotational invariance by taking it about the centre of the input field.
- **Redundancy** refers to repeated information. Redundancy in training examples is welcome; however we should also notice if there is any redundancy in features as we want to keep the number of features low. This is also an efficiency issue.
- **Lighting Invariance** refers to features invariant to a lighting source. These are potentially much easier to detect. Lighting invariance refers to both the intensity (brightness) of the lighting source as well as its direction.

## 3.4 Possible Properties

We use the following pixel statistics in this project:

**Maximum and Minimum** These refer to the maximum and minimum pixel value found in the region of interest.

**Mean** Defined as:

$$mean = \mu = \frac{\sum_{i=1}^{n} f(x_i)}{n} \tag{3.1}$$

where $n$ is the number of pixels in the selected region and $f(x_i)$ is the value of the pixel at location $x_i$. The mean is a most useful property in object detection as it is position invariant. This means an object need not be perfectly centred in an input field to be detected by the mean.

**Standard Deviation** Defined as:

$$sd = \sigma = \sqrt[2]{\frac{\sum_{i=1}^{n} (f(x_i) - \mu)^2}{n}} \tag{3.2}$$

where $n$ is the number of pixels in the selected region and $f(x_i)$ is the value of the pixel at location $x_i$. Like the counterpart the mean, standard deviation is also a position invariant property.

**Median** This is simply the middle value of the pixel intensities in the region of interest.

**Moments** The purpose of moments is to show the distribution among the locations in the small region of interest, a feature not shown in any of the above properties. The first order moment is defined as:

$$moment1 = \frac{\sum_{i=1}^{n} f(x_i) \cdot x_i}{n} \tag{3.3}$$

where $n$ is the number of pixels in the selected region and $f(x_i)$ is the value of the pixel at location $x_i$.

An alternative definition of the first moment is:

$$moment1 = \frac{\sum_{i=1}^{n} f(x_i, y_i) \cdot d(x_i, y_i)}{n} \tag{3.4}$$

where $d(x_i, y_i)$ is the distance between the pixel $(x_i, y_i)$ and the top left corner $d(x_i, y_i) = \sqrt[2]{x_i^2 + y_i^2}$ and $f(x_i, y_i)$ is the pixel value. $n$ in this case is the size of the square region, which is $n \times n$. The second order moments can be defined as:

$$moment2 = \frac{\sum_{i=1}^{n} (f(x_i) \cdot x_i - moment1)^2}{n} \tag{3.5}$$

or alternatively

$$moment2 = \frac{\sum_{i=1}^{n} (f(x_i, y_i) \cdot d(x_i, y_i) - moment1)^2}{n} \tag{3.6}$$

Moments are clearly not position invariant. A whole object in the corner of an input field yields a different moment from one taken from the centre. Moments can also be taken from different positions, such as any corner of an input field, distance from the x or y axis of an input field, or the centre of an input field. Consideration of the properties of the object of interest determines where the moment will be taken such that it most aids detection.

## 3.5 The Necessity of Regions in Object Detection

Input statistics need not be taken from the entire input field. This applies both to the calculation of the pixel statistics features for the cutouts as well as during sweeping, as outlined in section 3.2. As the image complexity increases, greater attention to detail is required, to reduce the possibility of false alarms as much as possible.

With the exception of the moment, pixel statistics give no localisation information. The mean, for example, does not care where in the input field the pixels are placed, and identical means could represent very different pixel distributions across an input field. The larger the input field, the greater this problem becomes, since the mean effectively ignores all localisation information.

Regions are a means to solve this problem. The entire input field is partitioned into small regions, and pixel statistics are calculated for each small region. This reduces the magnitude of the localisation problem, as the statistics for each small region will be different. Now during the sweeping phase, the network has more information to work with for each region. More inputs (information) about the input field during sweeping is now available to the network as if there are $m$ statistics per region with $n$ regions, then there is now a

total of $m \times n$ inputs to the neural network as opposed to the $m$ statistics for the entire region.

This means that there are more conditions (the inputs) which must be fulfilled before we can accurately classify an object, as every true object's individual regions each contain properties unique to that object, of which all must be fulfilled before a conjecture of an object's class can be made.

In this project, we use five regions, as shown in figure 3.6.

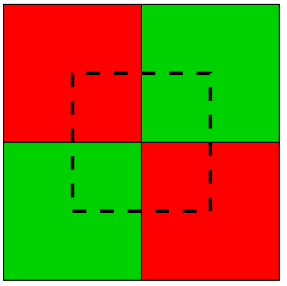


Figure 3.6: Five regions (including the centre one) in one input field

All definitions of the statistical properties remain the same, including the moment. The following example illustrates the need to be careful when dealing with moments within regions.

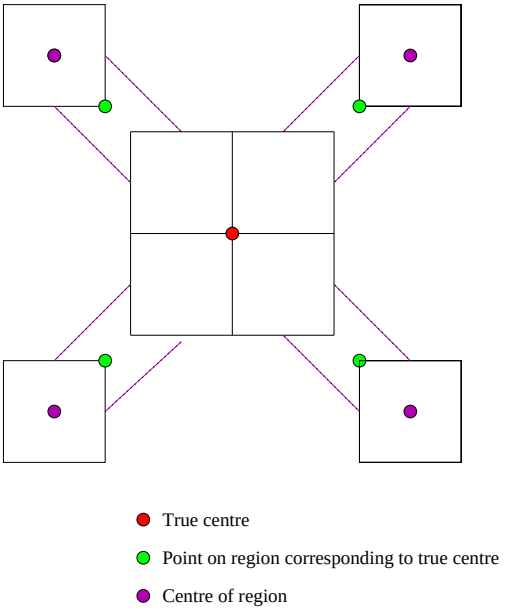



Figure 3.7: Regions and corresponding centres

**Example**

We refer to figure 3.7 on page 19. Suppose we are taking the moment from the centre. For each region, we could take the moment for that region from the region's corner corresponding to the true centre of the entire input field, or we take it from the centre of the region. It is better to take the moment from the centre of the region, since if we took it from the corner corresponding to the true centre of the entire input field, each region's moment is merely a part of the whole moment taken from the centre - thus giving us no new information.

As regions are simply smaller segments of an input field, there is no reason why they cannot be selected at random. In section 3.5.1, we present more systematic ways regions can be extracted from an input field.

### 3.5.1 Possible Regions of Statistics Extraction

**Entire Region** Statistics are calculated using *all* pixels within the current input field. This basic approach disregards all detail within the input field. It suffices only for very simple and uncluttered images.

**Rows and Columns** Take pixels of alternate rows or columns as illustrated in figure 3.8, 3.9 below. The pixel statistics are computed for alternate rows or alternate columns.

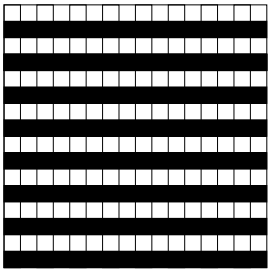


Figure 3.8: Rows and columns

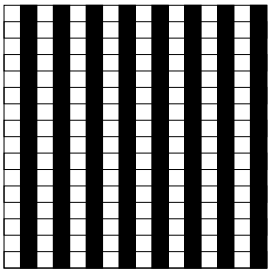


Figure 3.9: Rows and columns - another view

**Concentric Regions** A more complex variation on the above is to take concentric regions of a given shape. Such a shape could be a circle (which would give us a series of concentric rings), squares or triangles. Figure 3.10 shows an example of concentric rings. Note that only pixels in alternate circles are used to calculate the pixel statistics, and the gap between radii of two neighbouring circles is can be any number of pixels, provided that the next concentric circle has enough space to also extract statistics.

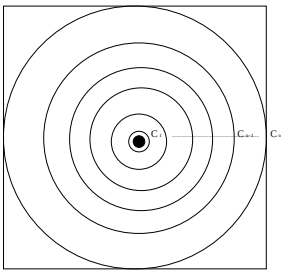


Figure 3.10: Concentric rings

A further variation on the concentric rings approach is to limit the number of rings allowed, and/or vary the gap between neighbouring rings. Depending on the objects of interest and their positions in the image, such an approach could prove useful.

**Central Regions** As illustrated in figure 3.11, we can take central regions of varying dimensions. This approach excludes other irrelevant data on the outskirts of the image, which could well be just noise.

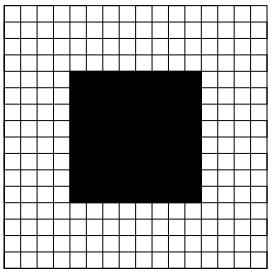


Figure 3.11: A centroid region

**Central Regions Interspersed with Concentric Regions** This is obtained by combining the above two approaches. The parameters to be determined are:

1. The size of the central region
2. The size of the smallest concentric region
3. The size of the largest concentric region

4. The number of concentric regions to allow

An example with a centroid region of 9 pixels with one square concentric region of size 25 pixels ($5 \times 5$) and largest AND smallest concentric regions being 18 pixels is shown in figure 3.12.

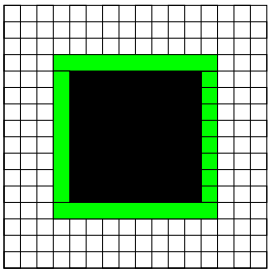
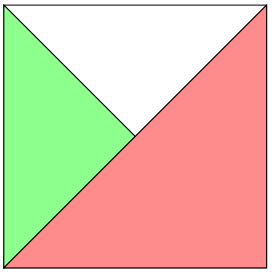


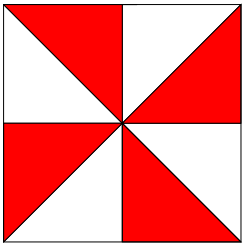
Figure 3.12: Central regions interspersed with concentric region

**Diagonal Regions** Excluding the rows and columns and entire region approach, all the above methods risk excluding information which lie around the edges and the corners. Diagonal regions offer a potential solution.

**Triangular and Random Regions** Another possible solution to the problem mentioned above is to use triangular (See figures 3.13 (a) and (b)) or even random regions (not shown).



(a) An example of three regions



(b) An example of a triangular region

Figure 3.13: Triangular regions

# Chapter 4

# The Image Database

## 4.1 Introduction

This chapter previews the database of images to be used in this project for object detection. We describe the properties of each database, and the challenge it presents to the network. The databases are ordered according to difficulty. Database one represents the simplest database, whilst database five represents the most difficult.

The sample images of each database we present in this chapter are the original, high resolution images. With the exception of the first database, these high resolution images are **not** the images used during testing of our recognition system. We use a lower resolution image of the original for reasons of efficiency as a larger image takes longer to sweep. However the lower resolution makes our task much more difficult as we have less features to work with.

## 4.2 Description of the Databases

| D/base No | Degree of Difficulty | Object Properties | Number of Classes | Rotation | Background |
|---|---|---|---|---|---|
| 1 | Trivial | Object outline | 2 | NO | Uniform |
| 2 | Simple | Single side of coin | 2 | NO | Relatively Uniform |
| 3 | Medium | Both sides of coin | 2 | NO | Relatively Uniform |
| 4 | Difficult | Both sides of coin | 2 | YES | Relatively Uniform |
| 5 | Very difficult | Both sides of coin | 2 | YES | Noisy |

Table 4.1: Databases at a glance

We will investigate the pixel statistics approach on five detection problems of increasing difficulty. All the coin databases use New Zealand coins.

The trivial database is used as a confirmation of the correct operation of the neural network using artificially generated images containing only squares and circles. Only one test image and one training image was generated. The other images in the database contain eight true objects.

There are four 5 cent coins and four 10 cent coins in the second database. For databases three to five, the number in each class can vary. This fact should be borne in mind when testing commences using both heads and tails in database three, but it does not impact

greatly on the actual results. A brief description of each problem follows, with a sample image from each database:

1. **Trivial Database**
   Here the background is uniform and only elementary geometric shapes are used. Furthermore, no more than three classes will be present. Each shape represents one class, and the background also counts as one class. The trivial database's primary purpose is to verify the correct operation of the recognition system. We expect to achieve perfect results (ie 100% detection rate and 0% false alarm rate) for this database - if we do not this indicates that the recognition system is faulty.

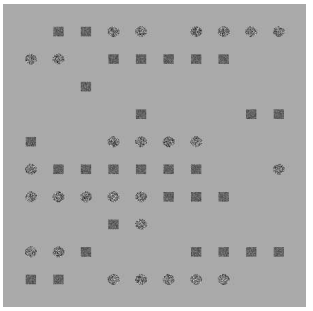


Figure 4.1: Sample image from database one

2. **Simple Coin Database**
   We place each coin with an upwards orientation at random locations on a surface with a relatively uniform background. Three classes are present in this database (*5 cent heads, 10 cent heads, background*). The purpose of this database is to test the ability of the neural network to be trained to recognise more advanced features.

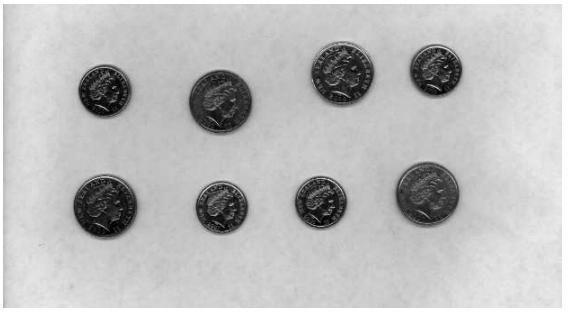


Figure 4.2: Sample image from database two

3. **Medium Difficulty Coin Database**
   The background is still relatively uniform, but this time both sides are used. Five classes are present in this database (*5 cent heads, 5 cent tails,10 cent heads, 10 cent tails, background*). This tests if the neural network is still able to recognise different views of the same object. Rotation is not permitted.

Figure 4.3: Sample image from database three

4. **Difficult Coin Database** Again, the background is kept relatively uniform, but rotation is permitted. Again both sides can be placed topmost at random. This presents much more of a challenge to the neural network, as it must be able to cope with correctly orienting the image upon discovering the coin to the image which it is acquainted with through training. There are five classes present in this database (*5 cent heads, 5 cent tails,10 cent heads, 10 cent tails, background*).

Figure 4.4: Sample image from database four

5. **Very Difficult Coin Database** The coins are rotated at random angles, with either side topmost. The background is noisy. This incorporates all the problems stated in the above four tests into one which the neural network must cope with. Obviously it is not possible to train all possible orientations and environments into the network so this is a true test of the neural network's ability to learn and detect the object in a realistic environment. There are five classes present in this database (*5 cent heads, 5 cent tails,10 cent heads, 10 cent tails, background*).


Figure 4.5: Sample image from database five

## 4.3 Number of True Objects Used in Database Testing

This section presents details of the image databases. It constitutes a reference for later chapters. The number of true objects represent the number of actual objects used during the training and testing phase, which are subsequently duplicated. In database one, there are 45 distinct examples of each class (ie squares and circles). In database two, all images have 4 distinct examples of 5 cents and 4 distinct examples of 10 cents.

For databases three to five we display this information in table 4.2.

| | Database 3 | Database 4 | Database 5 |
|---|---|---|---|
| Class1 (5c heads) | 8 | 6 | 6 |
| Class2 (5c tails) | 4 | 6 | 6 |
| Class3 (10c heads) | 5 | 5 | 3 |
| Class4 (10c tails) | 7 | 7 | 9 |

Table 4.2: Number of true objects used for training and testing per database

**Remark** An interesting question is whether the uneven number of distinct examples affects the results, and the answer appears to be that it does not. This was verified in an experiment using a greater number of distinct examples per class for each database. The result turned out to be much the same.

We should remember that this perhaps is not so much an issue in this project as it might otherwise be, since we are working largely in an artificial environment because the coins are scanned. Real world problems such as lighting intensity, direction and a coin's position on a surface are not taken into account, as well as accompanying problems on perspective. Such considerations would necessitate the development of a large number of distinct training examples.

The most important thing during training and testing is that there are **sufficient** examples of each class. The neural network does not care whether they are distinct or not; such subtleties are usually irrelevant as the network allows a certain margin of error during detection. However, we must have enough examples so that the network gets sufficient training.

# Chapter 5

# Results

## 5.1 Introduction

This chapter presents results for all five databases. We begin with results (section 5.2 to section 5.5) which have consequences for all databases. These initial results are also points to be carefully considered before embarking on the testing of all other databases, as neglect of these results leads to incorrect results in section 5.6.

We display the neural network architecture used during testing in the following format:

<number_of_input_nodes> – <number_of_hidden_nodes> – <number_of_output_nodes>

So <5> – <2> – <2> would mean the neural net contains five input nodes, two hidden nodes and two output nodes. The network in the current chapter was trained and tested as described in chapter 3, section 3.2 on experimental methodology. The parameters which were constant through all tests were:

- The learning rate was 0.2.
- No momentum was used.

## 5.2 The Effect of Insufficient Training Examples

This section shows the effects of an insufficient training set of examples. We are applying the system to database one, but the consequences for all other databases are the same. The set of statistics used in this section for database one were:

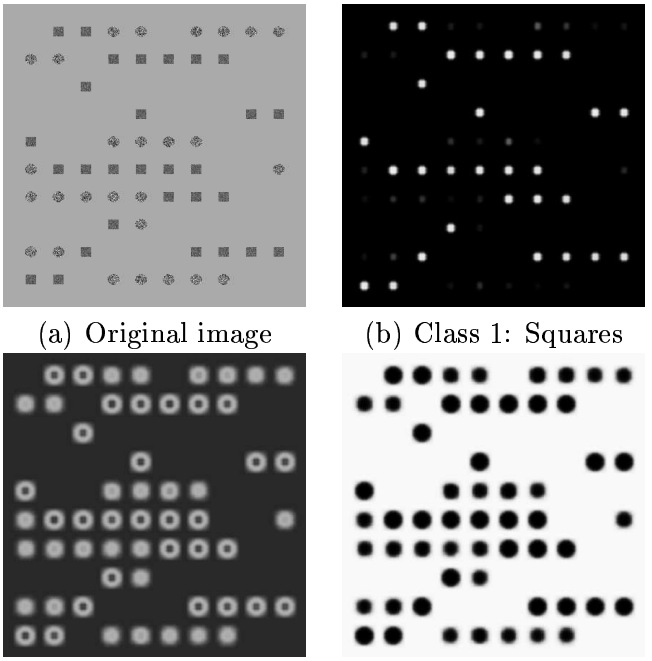
- Mean
- Standard deviation
- Maximum
- Minimum
- First moment taken from the top left hand corner of the input field

The neural network architecture used in this section was:

<5> – <4> – <3>

### 5.2.1 The Problem

A test run of the original image in database one produced the results shown in figure 5.1.



(a) Original image (b) Class 1: Squares

(c) Class 2: Circles (d) Class 3: Background

Figure 5.1: Database One : The problem

The following discussion refers to figure 5.1. Performance in class 1, figure 5.1 (b) was excellent. All squares were correctly detected as shown by the presence of white regions present at or around the true location of the squares. This can be seen if one compares the object sweeping map of figure 5.1 (b) with the original image, figure 5.1 (a). Grey regions are also present in the location of the circles, but as we mentioned in chapter 3, section 3.2, the recognition system is not certain that objects detected at these locations are instances of squares. However, these grey regions are quite dark, and the appropriate threshold can be set so that they are ignored, producing a 100% detection rate and 0% false alarm rate for this class.

Class two (figure 5.1 (c)) highlights the problem. Where a circle is present in the original image, the recognition system detects it, but not very precisely as indicated by the grey region. (A precise location would mean a white region should be present around the actual location of the object in the original image). It also clearly gives a higher rating to circles rather than squares - since where the circles are located in figure 5.1 (a), there is no black region in the middle of the object.

By doing this, the system indicates that a square in the original image is less likely to be a circle - yet it is not entirely confident. (If it were then locations containing squares should be black). A close up of the region of interest under discussion is shown in figure 5.2.
We discuss why the recognition system did this, and how this was solved in section 5.2.2.

The background was detected accurately, as shown in figure 5.1 (d). Here, black regions were present in the actual locations of all objects (both squares and circles), and the whole

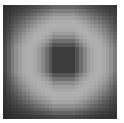


Figure 5.2: Object sweeping map of detected circles

background area was white, which indicated the region where the recognition system were not instances of either object (squares and circles).

### 5.2.2 Analysis

To explain figure 5.1 (c), we examine the actions of the sweeper. It sweeps across from left to right, and then down through the original image. Consider the following scenario, when the sweeper's input field is partially covering the square:

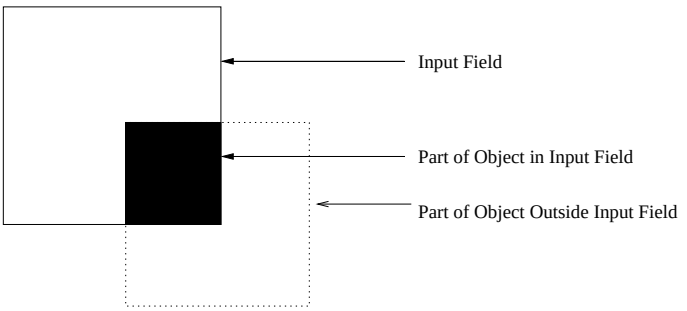



Figure 5.3: Part of a square object within an input field (Not to scale)

At this stage, the pixel statistics for the partial square are almost the same as that for the circle. This is true especially of position invariant statistics such as the mean. The moment too is involved, but if one views the original (figure 5.1 (a)) it is clear that the pixel intensity of the squares is less than that of the circle (ie the square is darker compared to the circle). Even if less of the square was present within the input field, the moment must have been similar as we took the moment from the top left hand corner of the input field.

The leads the recognition system to erroneously assume that the circle could be present at this location. As the statistics are not exactly those for the circle, it is close enough for the system to make a weak conjecture. This weak conjecture results in the grey regions.

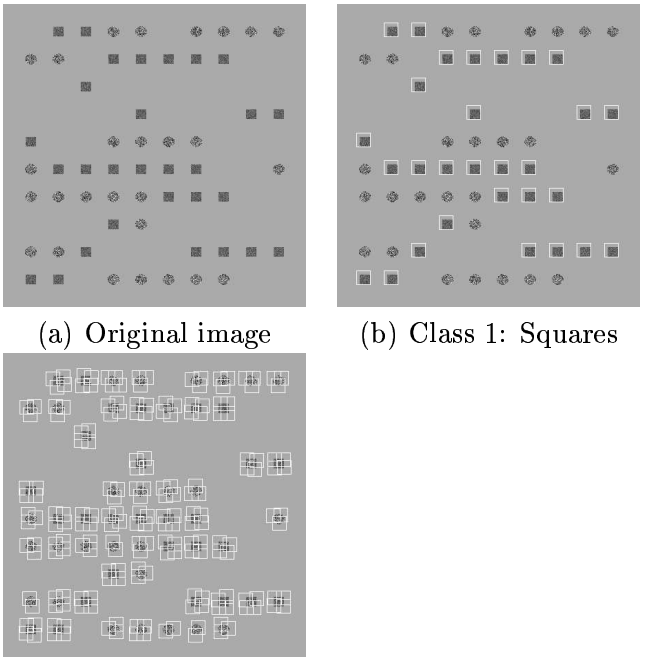
If the pixel intensity of the circles were similar to that of the squares and the circles were approximately the same size as the squares (in figure 5.1 (a) they are actually smaller in size), then such a scenario may not have arisen, as the moments for a partial square in the input field would then not be similar to the circle.

The centralised black dot arises as the recognition system only realises that the square is not a circle when it is perfectly centred and fully covering the square. The statistics at this

point would indicate this, due to the lighter intensity of the circle and its smaller size.

It may help to view the image cuts of the detected objects which correspond to these object sweeping maps (described in section 3.2). Although image cuts enable us to quickly ascertain the performance of the recognition system, they do not provide any additional information to the object sweeping maps, so we will omit them in later results. We include them in this initial example so that the reader may be able to compare them to the object sweeping maps. Also we hope showing the image cuts will make the meaning of the object sweeping map clearer in terms of describing the recognition systems behaviour during the sweeping.

The image cuts corresponding to the object sweeping maps in figure 5.1 are shown in figure 5.4.



(a) Original image (b) Class 1: Squares

(c) Class 2: Circles

Figure 5.4: Image cuts of detected objects corresponding to object sweeping maps in fig 5.1

The squares were detected perfectly, as can be seen from figure 5.4 (b). Only one white frame per square (the desired object) and none over the circles (which would imply the existence of false positives) are present. Note that the white frames are not perfectly centred; the reason for this is because the recognition system sweeps from left to right and top to bottom. The location of the white frame on the image cut is centred at the position where the recognition system first detected the object. This is because a square which is slightly offset from the central position has similar statistics to that of a perfectly centralised square (though not exact, because of the moment).

The image cut in figure 5.4 (c) reflects that the circles were not detected accurately. Multiple frames surrounding every object indicates that the system is uncertain about the circle's actual position. The white frames over the squares are false alarms, ie squares mistaken for the circles.

### 5.2.3 Solution

The above problem occurs because the image incorrectly recognises partial squares as circles. Constructing good training and test examples is important in all machine learning, and poor training and testing examples lead to incorrect results. Just as we must teach the neural network what to look for, it is equally, if not more important, to tell it what *not* to look for. These include examples of partially visible objects in the input field, which should not be considered as classes of objects.

The author developed a program for enhancing the training and testing sets with additional examples of images containing partial intrusions of the objects of all classes as in figure 5.3. We will not describe the implementation or pseudocode of the program, but we outline the considerations which must be addressed in order to construct such a program.

We also describe a set of considerations needed to construct examples containing the background only, with no partial or whole instances of objects. This is important for database five, whose background is noisy. The neural network must be trained on as many distinct examples of the background as possible.

In both constructions above, we assume that the actual $(x, y)$ coordinates of the actual object's centre location, the image from which the example must be created (these examples are the cutouts as described in section 3.2) and the input field size $w$ are given to us. To construct the solution for the partial intrusion examples, we must also know the tolerance about the centre location of each actual object.

The solution output given by these programs constructed using the solution below is a set of $(x, y)$ co-ordinate pairs, corresponding to the centre of the proposed example to be cut out from the image. So given the proposed $(x, y)$ pair and a square input field size of dimension $w$, the actual training example which is cutout has x values ranging from $(x - w/2, x + w/2)$ and y values ranging from $(y - w/2, y + w/2)$. Figure 5.5 illustrates this.

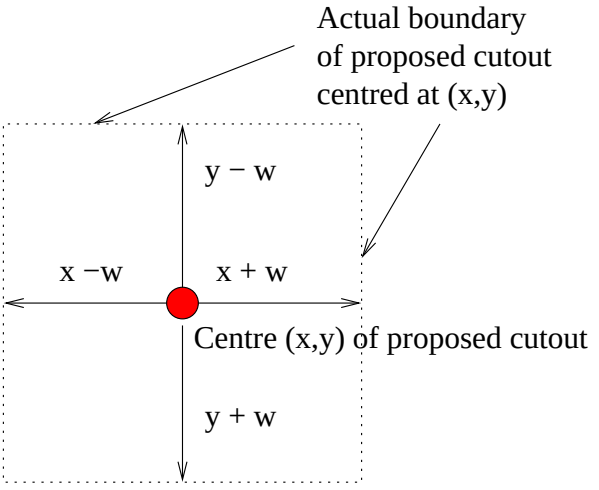



Figure 5.5: The actual cutout represented by proposed (x,y) coordinates

We must also ensure that if the dimensions of the image from which the cutouts of the proposed $(x, y)$ are to be cut that the cutouts fall within the dimensions of the image.

Given the x dimension is $max\_x\_size$ and the y dimension is $max\_y\_size$ for an image, for every pair of $(x, y)$ co-ordinates proposed as a centre of an example, we must check that:

$x - w/2 > 0$
$x + w/2 < max\_x\_size$
y$-w/2 > 0$
$y + w/2 < max\_y\_size$

The above conditions ensure that all the resultant examples cutout are the within the dimensions of the image.

### Creating Multiple Examples Containing Partial Intrusions of Objects

For each $(x, y)$ coordinate of the true objects, we calculate the set of proposed $(x, y)$ coordinates as follows.

1. Define the $x$ and $y$ boundaries which $(x, y)$ centres can be proposed. In this case they are $(x - w, x + w)$ for x, and $(y - w, y + w)$ for y, excluding the endpoints as any input field cutout centred on the edge will miss the object completely.

2. Mark off the region about the centre using the tolerance boundary. Examples taken within this boundary have an undesirable effect - it will cause the system great confusion, since the system initially believes it can assert an object is present within the given threshold, yet the example claims this is an instance of a partial intrusion of an object and thus is not a class.

If we reduce the tolerance so that we can take more examples closer to the object centre, the system must then be extremely precise in order to detect the object since only conjectured $(x, y)$ values within the boundary will be accepted as a detected instance of an object.

From figure 5.6 on page 33, the region of allowable $(x, y)$ coordinates are all those outside the shaded central region and inside the outermost boundary.

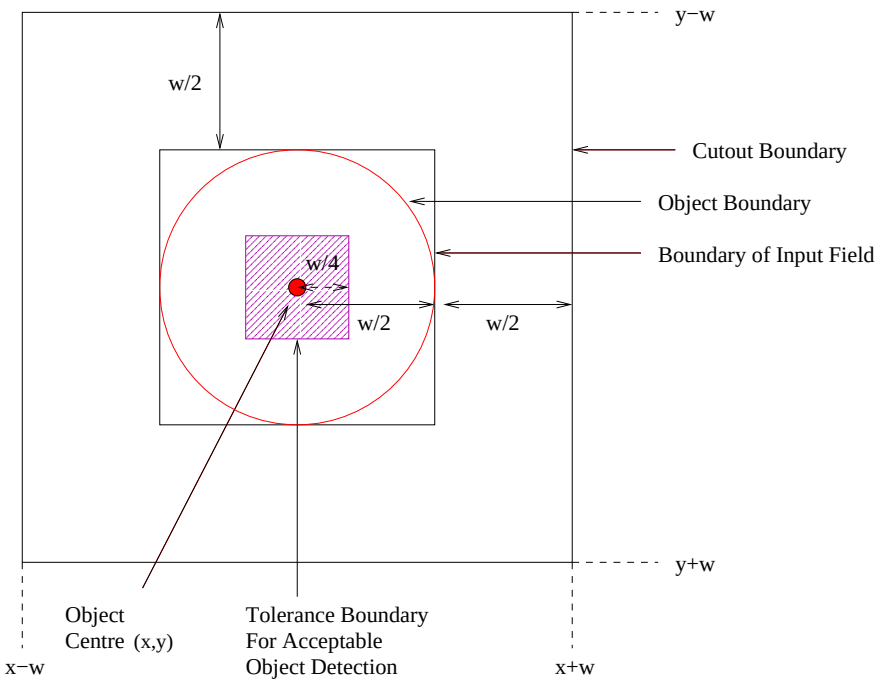



Figure 5.6: Region for creating multiple examples of instances of partially intruding objects

### Creating Multiple Plain Background Examples

The solution to the problem of constructing a set of $(x, y)$ coordinates for cutting out plain background only is slightly different. Rather than define boundaries from which all $(x, y)$ coordinates can be proposed, we need to test all $(x, y)$ coordinates inside the image.

The key point is as follows. Suppose we propose coordinates $(x, y)$. Recall figure 5.5 on page 31. The proposed $(x, y)$ coordinates actually represent a cutout which ranges $w/2$ pixels on either side of the proposed $(x, y)$ centre. To be a plain background, we must ensure that the resultant cutout from the proposed $(x, y)$ coordinates include no objects.

We are given the $(x, y)$ coordinates of the actual objects in the image. Using the same idea as for cutouts (figure 5.5 on page 31), we calculate the $x$ and $y$ boundaries for all objects which would be their boundaries were they all to be inside an input field of the given size $w$.

For the proposed $(x, y)$ coordinates we do the same, and calculate its $x$ and $y$ boundaries for this object which would be its boundaries if it, too, were inside an input field of the given size $w$. Then we check if the boundaries of our proposed $(x, y)$ coordinates are inside *any* of the boundaries for the actual objects. If they were, this would mean the two input fields overlap, and hence this implies that the resultant example from the proposed $(x, y)$ coordinates would contain part of the actual object within the input field.

The author terms this overlapping as "encroachment" (as the input field of our proposed $(x, y)$ coordinates encroach within the boundaries of the input field of the actual object). There are four possible cases in which encroachment may occur as shown in figure 5.7.

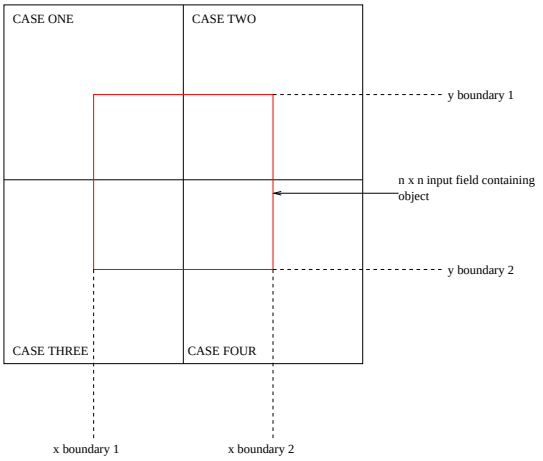
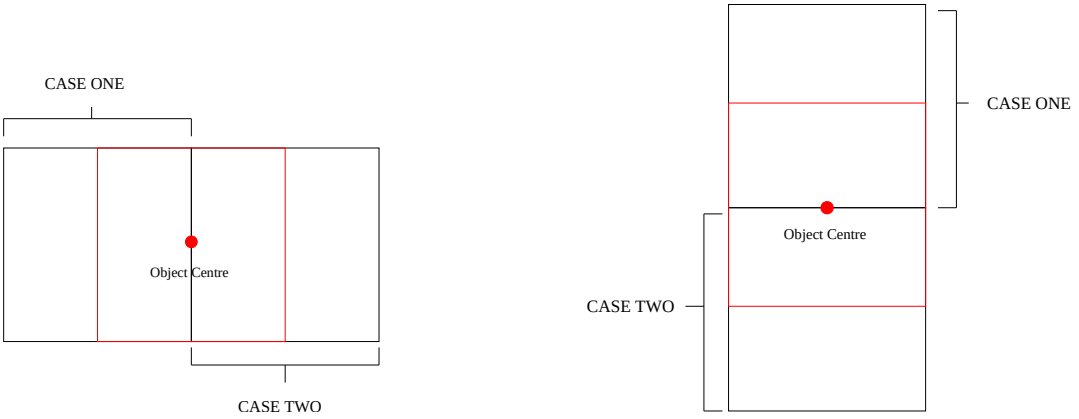



Figure 5.7: Four cases of encroachment on an object from proposed co-ordinates

There are two special cases where the $x$ and $y$ boundaries of some actual object happens to coincide with either or both the $x$ or $y$ boundaries of the proposed $(x, y)$ coordinates. In this case it is sufficient to check whether or not the other boundary encroaches or not, as shown in the figure 5.8.



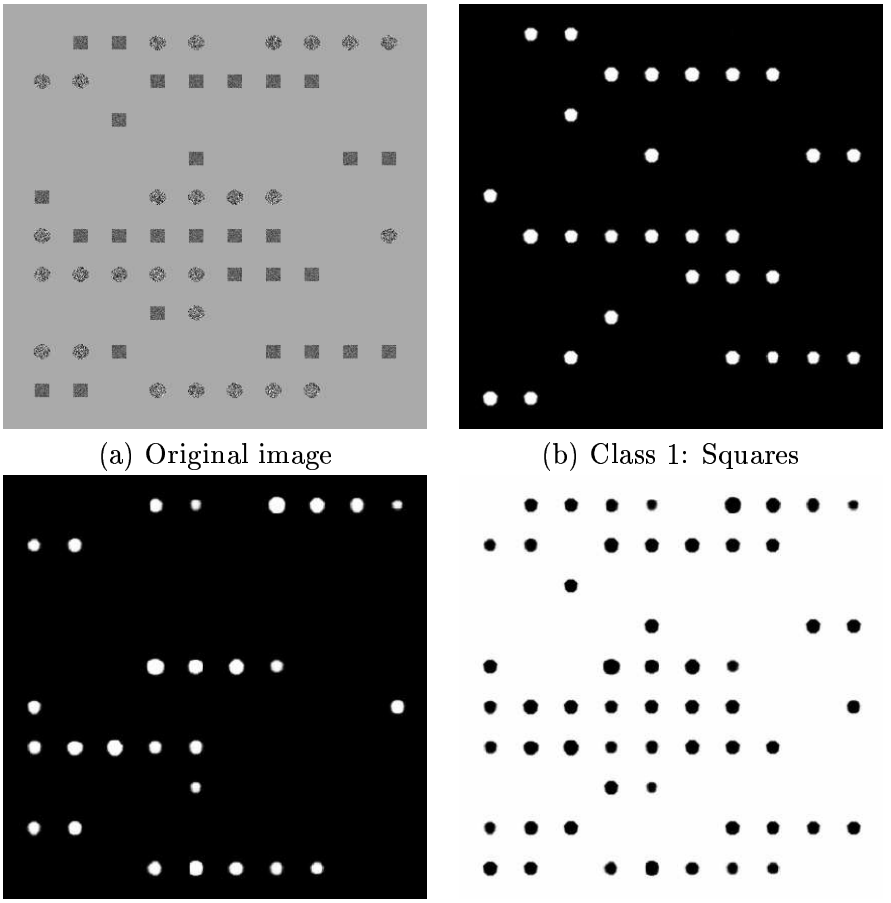

(a) Y-boundary coincides with object boundary (b) X-boundary coincides with object boundary

Figure 5.8: Special encroachment cases

Thus only proposed $(x, y)$ coordinates which do not encroach at all are included in the solution set of allowable $(x, y)$ coordinates.

### Results of Database One with Additional Examples

The effect of the additional training and testing examples on database one are shown in figure 5.9. This corrects the poor detection of the circles in figure 5.1 on page 28.



(a) Original image

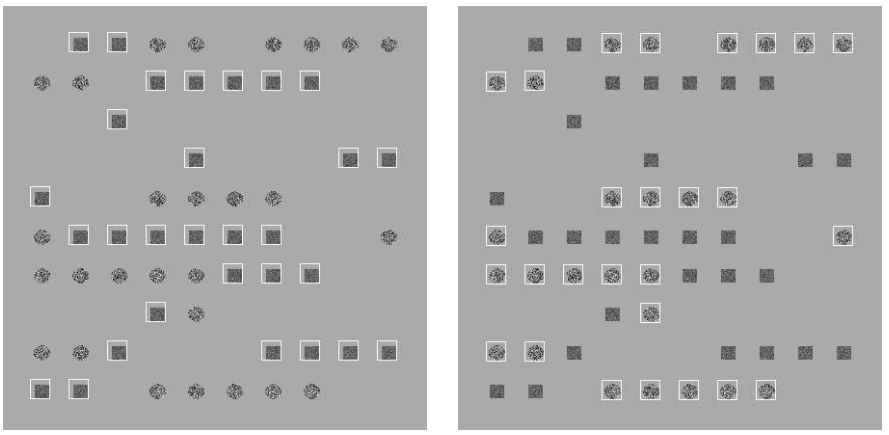
(b) Class 1: Squares

(c) Class 2: Circles

(d) Class 3: Background

Figure 5.9: Database One : Perfect object sweeping maps

The resulting image cuts corresponding to the perfect object sweeping maps in figure 5.9 are shown in figure 5.10. As these results are perfect, an ROC Curve would add no additional information, so it is not needed. In our testing of each database, we use the solution to this problem to ensure adequate numbers of training and testing examples are present.



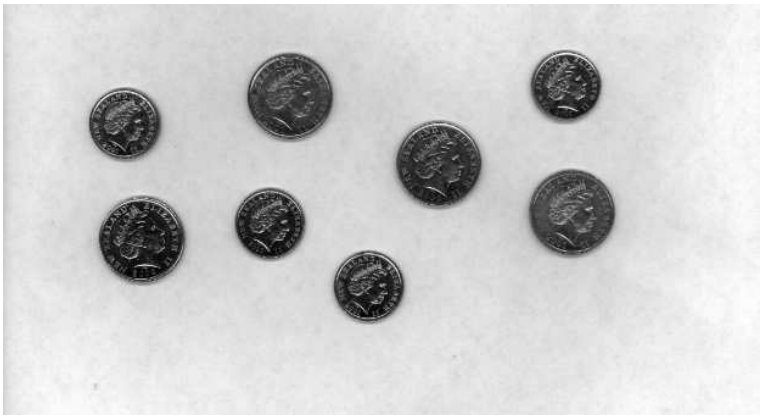
(a) Class 1 Cut (b) Class 2 Cut

Figure 5.10: Database One : Perfect cuts

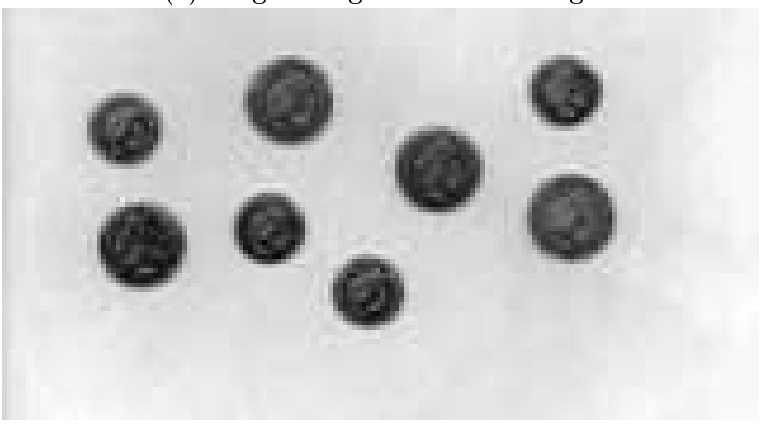
## 5.3 The Effect of a Position Dependent Statistic

The reader will recall from chapter 3 that statistics such as the mean and standard deviation are position invariant, which means as long as the same pixels are still there, they will produce the same mean/standard deviation irrespective of where they are positioned inside the input field.

This is not so for moments, whose values are position dependent. Where one takes a moment from will produce a different result. Selection of an appropriate position to take the moment from is an important consideration in this project. In this section, the original high and low resolution images are shown in figure 5.11.



(a) Original high resolution image



(b) Original low resolution image

Figure 5.11: Original images

The set of statistics used in this section were:

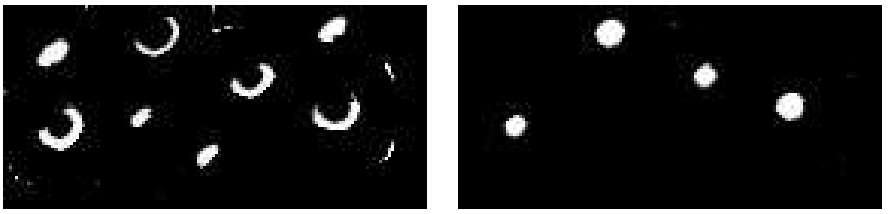
- Mean
- Standard deviation
- Maximum
- Minimum
- First moment taken from the centre of the input field

The neural network architecture used in this section was:

$$<5> - <4> - <3>$$

### 5.3.1 The Problem

A test run of the original low resolution image in database two produced the results shown in figure 5.12.



(a) Class 1: 5 cent coins (b) Class 2: 10 cent coins

Figure 5.12: Database Two : The problem

The object sweeping map for the 10 cent coins (figure 5.12 (b)) contain no error, but this is not so for the object sweeping maps of the 5 cent coins, (figure 5.12 (a)) where white crescent shaped regions are present on the object sweeping map where a 10 cent normally resides. Moreover, these false alarms appear consistently throughout, and are white, which suggests that the system is confident a 5 cent coin resides in this position. We presume there must be some underlying reason causing the recognition system to make these incorrect conjectures.

### 5.3.2 Analysis

We go back to the statistics used as input to discover what is going on. Of particular interest is the moment. Moments are position dependent statistics, and correct selection of where to take the moment from is an important consideration in this project.

Currently the moment is taken from the top left hand corner, ie the formula used is:

$$moment1 = \frac{\sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2} \cdot f(x_i)}{n} \tag{5.1}$$

where $f(x_i)$ is the pixel value at the given $x_i$ coordinate. As with the circles, what occurs is again only part of the coin appears in the input field. Assume for for the sake of simplicity that a 5 cent coin is exactly half the area of a 10 cent coin.

Now suppose that we have the scenario:

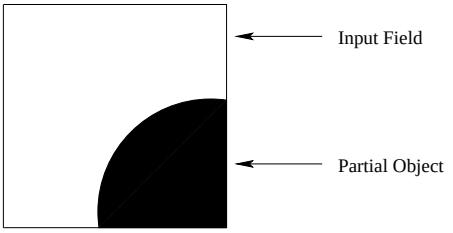



Figure 5.13: Partial coin object in input field (Not to scale)

In this instance, let the input object be the 10 cent coin. Assume **exactly** half the area of the 10 cent coin is shown. Now the problem is evident; the moment in this instance will turn out to be exactly (or very similar to) the value of a perfectly centred 5 cent coin. All other statistics will also be the same, hence the centre of this partial 10 cent coin will be deemed erroneously by the network to be a 5 cents.

There is also another interesting observation to make in this image (figure 5.12 (a)). Notice that the white region signifying the detected 5 cent coin is ellipsoidal in shape. The explanation underlying this phenomena is again to do with moments. In this instance the moment was taken from the top left hand corner. The 5 cent coin is also smaller in size to its 10 cent counterpart. As seen in figure 5.14 the small size of the coin enables it to occupy various positions within the input field, with it's centre equidistant from the top left hand corner. The mean, standard deviation, maximum and minimum of all the pixels are clearly identical in all cases; and because the centre is equidistant, the moment is too, leading to multiple conjectures.

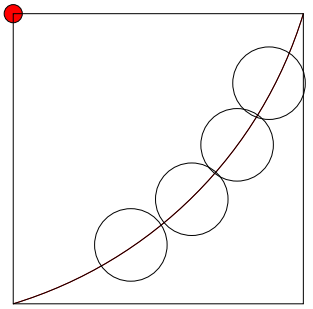


Figure 5.14: Why multiple 5 cents are detected when moment is taken from top left hand corner (red circle) - Not to scale

### 5.3.3 Solution

To alleviate the problem we make some modification to the moment so that such partial 10 cent coins can be distinguished from the 5 cents. To do this, we take the moment from the

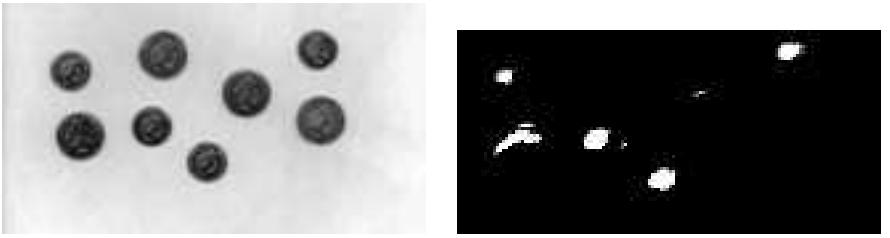
centre of the input field.

Recall that all cutouts of coins are perfectly centred in our training examples, and this perfect example is what we want the network to learn to recognise. Taking the moment from the centre avoids the case of the partially visible coin since a coin must now be perfectly centred in the input field to produce the required moment and hence the crescent shapes in the above image (figure 5.12 (a)) should disappear.

The formula is a simple modification of the above:

$$moment1 = \frac{\sum_{i=1}^{n} \sqrt{x_i^2 - (input\_field\_size/2) + y_i^2 - (input\_field\_size/2)} \cdot f(x_i)}{n} \quad (5.2)$$

As shown in figure 5.15, this eliminated the crescent regions so from this point onwards only this corrected moment (Equation 5.2) will be used.

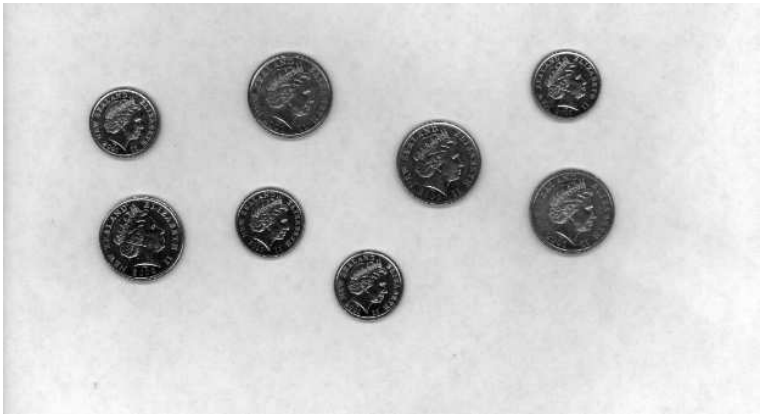


(a) Original low resolution image (b) Class 1: 5 cent coins

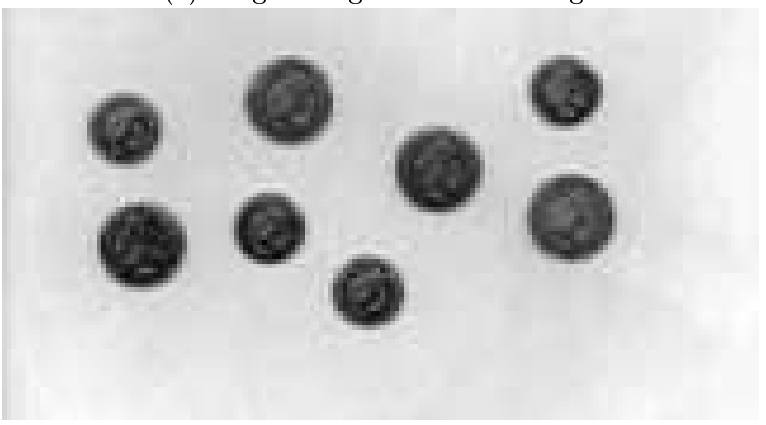
Figure 5.15: Object sweeping map with 5 cent coins using corrected moment

## 5.4 Recognising Overtraining

The original high and low resolution images used in this section are shown in figure 5.16, an instance of database two.



(a) Original high resolution image



(b) Original low resolution image

Figure 5.16: Original images

The set of statistics used in this section were:

- Mean
- Standard deviation
- Maximum
- Minimum
- First moment taken from the centre of the input field

The neural network architecture used this section was:

$$<25> - <4> - <3>$$

Overtraining is when a network is trained so well that it develops a solution so specific it works perfectly on the training and testing examples but performs very poorly on an unseen

image. An overtrained network generally produces very chaotic object sweeping maps on an unseen image, as seen in the object sweeping map for the 5 cent coins (figure 5.17). The chaotic image is a result of the overly constrained model produced by the network during training which prevents the recognition system from applying it to an unseen image.

Figure 5.17: Object sweeping map of 5 cent coins using an overtrained network

Avoiding this scenario usually involves the following steps:

- To prevent overtraining, adjust the stopping criterion. In this project, this usually means reducing the proportion of correctly classified training examples in order to allow some margin of error.
- To detect overtraining, check if the error rate on the training data is much lower than that of the test data - this implies that the current network fits the training data too well.

The simple images in in the trivial database in chapter 4 are an exception. It is very likely perfect results will be attained in both training and testing, but this does not imply overtraining in this case due to the simplicity of the images.

To achieve the overtraining result in figure 5.17, we set the proportion correct before termination of training to be 99.8% of all training examples, and we trained the network 40 times (ie the same network was retrained 40 times). 1249 epochs were initially required to reach termination on the first training.

Table 5.1 shows the evaluation results on the overtrained network:

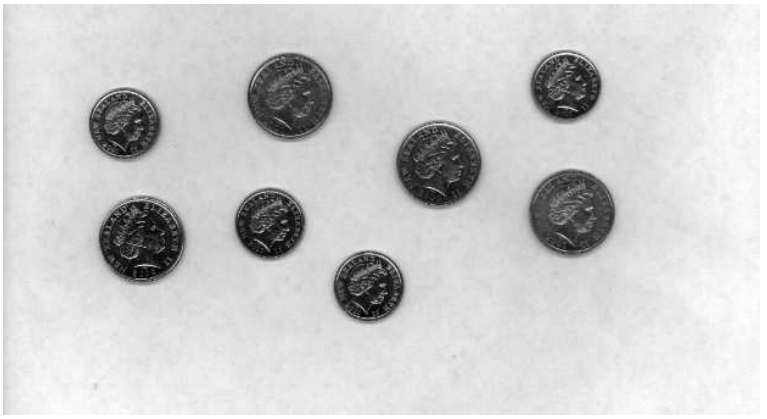
| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 350 | 300 | 250 | 225 | 200 | 150 | 125 | 100 |
| 2 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 275 | 250 | 250 | 250 | 200 | 175 | 75 | 50 |

Table 5.1: Test Results : Overtrained database two

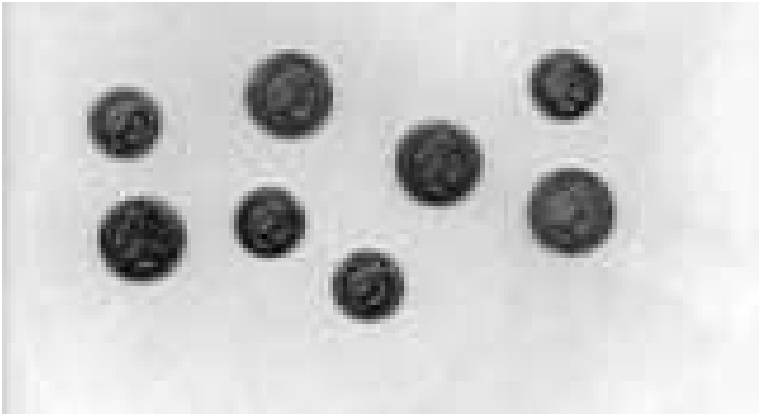
**Remark** From the table above, we note that as we increase the threshold value for for both classes, the detection rate remains unchanged at 100%, whilst the false alarm rate continues to fall. We should, however, resist stating the false alarm rate for both classes will continue to fall whilst maintaining a 100% rate, as we do not know whether the detection rate reaches 0% (or any percentage less than 100%) before the false alarm rate for either class reaches 0%.

## 5.5 Recognising Undertraining

The original high and low resolution images used in this section are shown in figure 5.18, an instance of database two.



(a) Original high resolution image



(b) Original low resolution image

Figure 5.18: Original images

The set of statistics used in this section were:

- Mean
- Standard deviation
- Maximum
- Minimum
- First moment taken from the centre of the input field

The neural network architecture used this section was:

$$<25> - <4> - <3>$$

Undertraining occurs when the network has not been trained enough, with the resulting solution being too general and simplistic to be of any use. The consequences of such a simplistic model are as follows. When applying the generated model to an unseen image (figure 5.18 (b)), we see in figure 5.19 the system found it difficult to accurately detect the classes of interest, as the simplistic model fits many possible objects (indicated by the grey regions), but none of them well (indicated by the lack of white regions).

Figure 5.19 also shows by the centralised black dots in the positions corresponding to the original 10 cent coins in figure 5.18 the model only recognises an instance of a 10 cent coin is not an object when the input field is perfectly centred over the coin. If the model used were any simpler, the black dot may not have appeared at all, which indicates the recognition system is only able to recognise that a coin is present, but completely unable to differentiate between the classes.

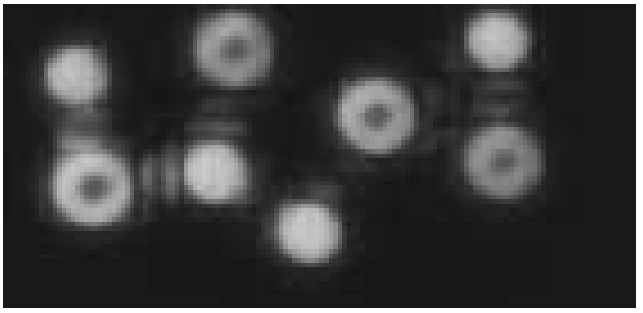


Figure 5.19: Object sweeping map of 5 cent coins using an undertrained network

To obtain undertraining, we set the proportion correct before termination of training to be 70.8% of all training examples, and we trained the network 2 times. Only 3 epochs were initially required to reach termination on the first training.

Table 5.2 shows the results for the undertrained network:

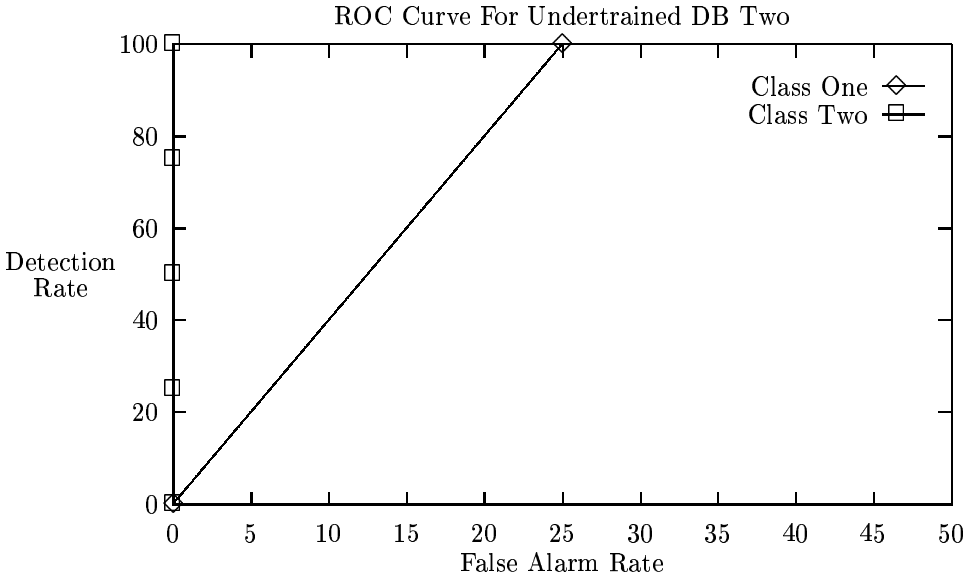
| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 100 | 100 | 100 | 100 | 100 | 0 | 0 |
| | FAR (%) | 275 | 225 | 200 | 100 | 25 | 0 | 0 |
| 2 | DR (%) | 100 | 75 | 75 | 50 | 25 | 25 | 0 |
| | FAR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5.2: Test Results : Undertrained database two

**Remark** We obtained a perfect result for the second class using a threshold of 0.3. No threshold for the first class produced a perfect result, although thresholds for class 1 in the interval $[0.7, 0.8]$ could be investigated.

Were this to be a more complex image with an accompanying noisy background, results could be disastrous, as the recognition system will not have any clue as to the whereabouts

of the object location, since the model available to the network is not specific enough. Note that undertraining can also be caused by insufficient training examples.




## 5.6 Results for Individual Databases

In this section we present results for all the coin databases. We incorporated all solutions found in the previous sections of this chapter when testing the recognition system on databases two to five.

The set of statistics used in this section were:

- Mean
- Standard deviation
- Maximum
- Minimum
- First moment taken from the centre of the input field

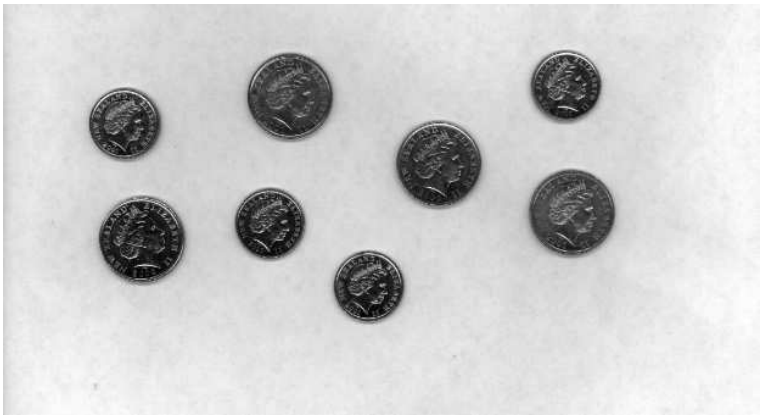
The neural network architecture used in database two was:

$$<25> - <4> - <3>$$

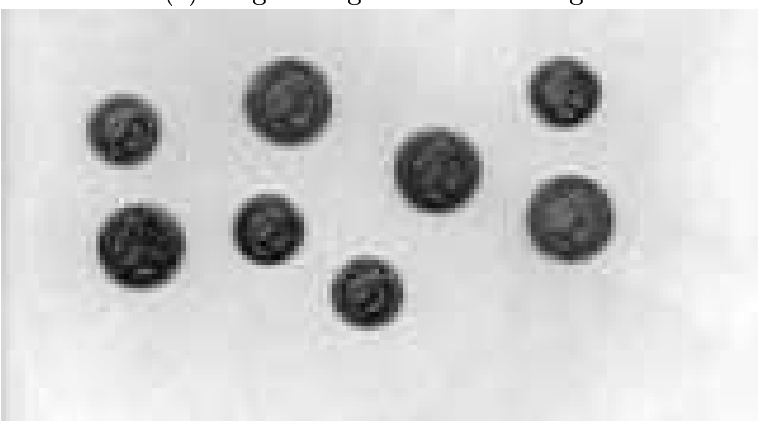
The neural network architecture used databases three to five was:

$$<25> - <4> - <5>$$

### 5.6.1 Database Two



(a) Original high resolution image



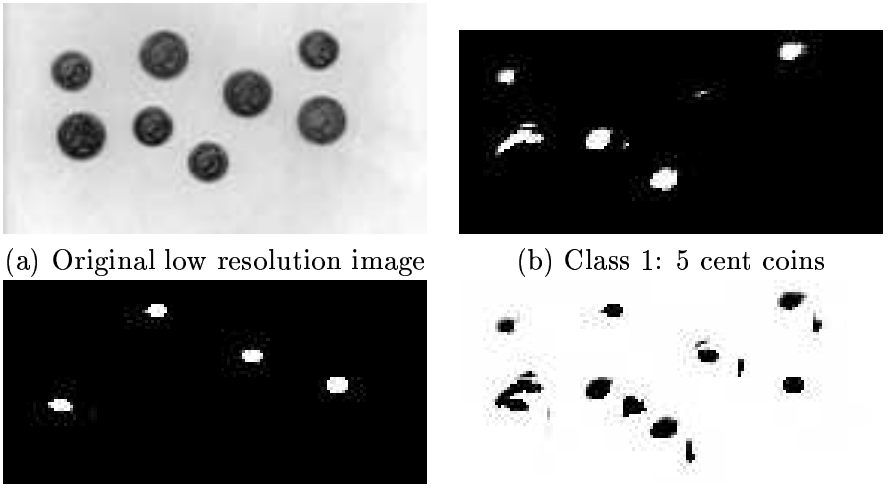
(b) Original low resolution image

Figure 5.20: Database Two : Original images

#### Overview of the Problem

In general, we did not expect database two to present a great challenge to the neural network, as it's task was simply to differentiate between the size of the two black circles corresponding to the coins and was not required to distinguish between any features, such as heads and tails.

**Discussion and Analysis**



(a) Original low resolution image (b) Class 1: 5 cent coins

(c) Class 2: 10 cent coins (d) Class 3: Background

Figure 5.21: Database Two : Test results

In figure 5.21 (b) the 5 cent coins have all been correctly detected. A false alarm is present in figure 5.21 (b), as indicated by the white patch on the left hand side corresponding to the position of the leftmost 10 cent coin with heads topmost in figure 5.20 (a).

Repeated testing using different parameters for the neural network such as altering the proportion of correct examples before termination failed to eliminate the false alarm in figure 5.21 (b). As database two was expected to be a simple problem, we did not believe that the false alarm was an error in the recognition system, but rather the recognition system producing a false alarm due to some property of pixel data in the input field.

We investigated the range of pixel values in the leftmost 10 cent coin with heads topmost in figure 5.20 (a) (call this **coin A**) compared with the range of pixel values of all other 10 cent coins also in figure 5.20 (a)(call this **coins B**) . We found that the pixel values in **coin A** ranged from [56,106] and pixel values in **coins B** ranged from [76,104]. As the 5 cent coin is smaller in diameter than the 10 cent coin, it takes up less space in the input field, both in the cutouts and during sweeping.

The position of the false alarm in figure 5.21 (b) suggests that due to the large difference in pixel intensities between **coin A** and **coins B**, and the close proximity of the false alarm to the 5 cent coin in the top left hand corner, the pixel statistics when the input field contains both the edge of the 5 cent coin nearest the top left hand corner and also part of the 10 cent below it in figure 5.21 (a) are very similar to that of a perfectly 5 cent coin in the input field. The scenarios we are referring to are shown in figure 5.22.

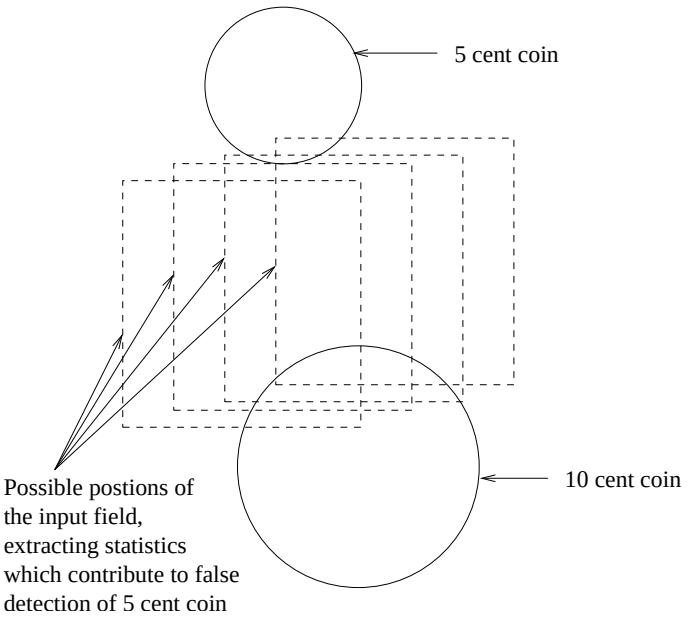



Figure 5.22: The input field over two coins

We can see from figure 5.22 that to produce the white false alarm region (this would be the conjectured centre of the location of the 5 cents) the input field must be in the positions shown. Notice that parts of the 5 cent coin can also appear in the input field, and the resulting contributions from parts of both coins present in the input field would have been sufficient to produce statistics similar to that of a perfectly 5 cent coin within the input field for position invariant statistics.

It appears the moment, even when taken from the centre has failed to detect that the input field is not an instance of an object. The moment certainly should not be the same as that of the 5 cents, as the majority of the input field is background. The background is much lighter than the coins, but somehow its influence is either diminished or overridden by the other statistics.

The background was also incorrectly detected in figure 5.21 (d). The background should completely ignore the parts where a coin is present (which should be black circles corresponding to the size of the coin objects) and mark the rest of the background with a white region. In this case, the background has been overtrained. A recognition with a correctly trained neural network should produce a object sweeping map for the background as shown in figure 5.23

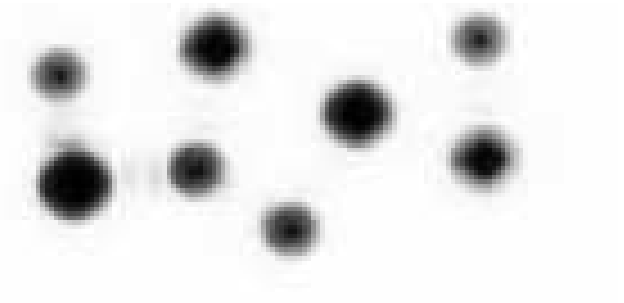


Figure 5.23: Object sweeping map of background using an undertrained network

Table 5.3 shows the evaluation results for figure 5.21.

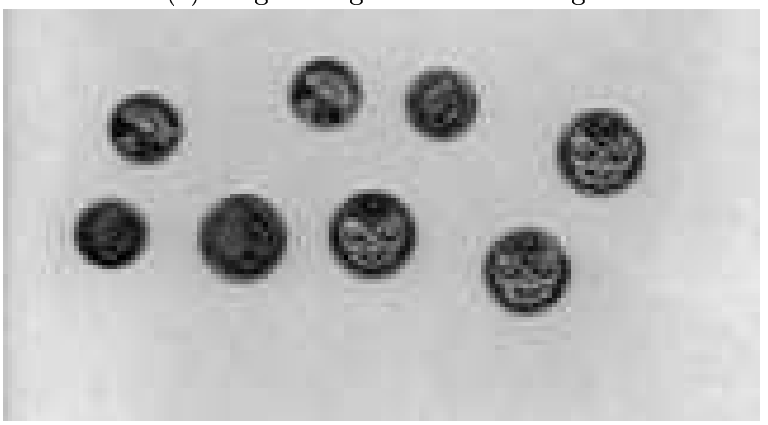
| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 50 | 50 | 50 | 50 | 50 | 25 | 25 | 25 |
| 2 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5.3: Test Results : Database two

**Remark** An excellent result was obtained for the second class (10 cent coins) at all thresholds, though we were unable to reduce the false alarm rate for class 1 below 25% due to the white region in figure 5.21 (b).

### 5.6.2 Database Three

(a) Original high resolution image



(b) Original low resolution image

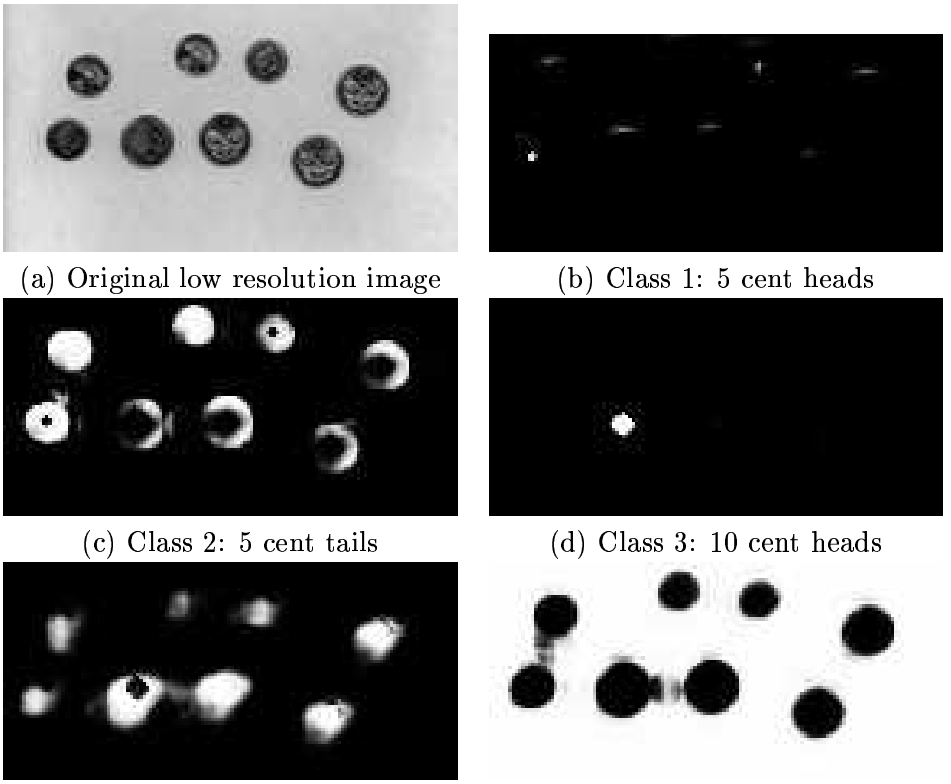
Figure 5.24: Database Three : Original images

#### Overview of the Problem

This database requires the recognition system to differentiate heads and tails as well as identifying which coin the heads and tails belong to. Due to the reduced resolution there will be fewer pixels on the coin surface to contribute to the pixel statistics features which aid distinction between heads and tails.

Even the human eye may be unable to distinguish between the heads and the tails, but it may be possible that the system may recognise this distinction by virtue of being able to differentiate between the small differences in pixel values via the statistics.

**Discussion and Analysis**



(a) Original low resolution image (b) Class 1: 5 cent heads

(c) Class 2: 5 cent tails (d) Class 3: 10 cent heads

(e) Class 4: 10 cent tails (f) Background

Figure 5.25: Database Three: Test results

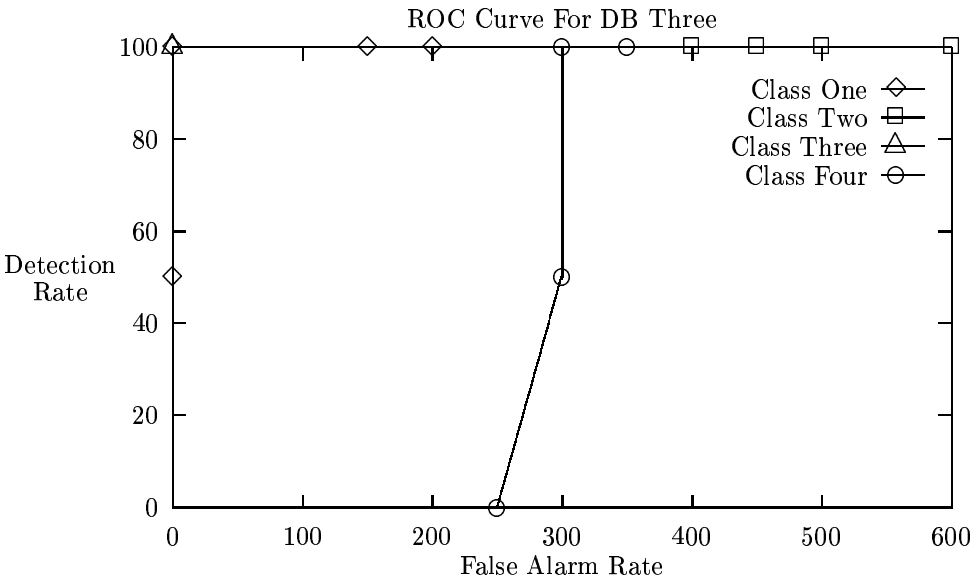
The recognition system achieves its best results on the 10 cent heads as shown in figure 5.25 (d). The 5 cent heads were also well detected. This is shown by the pinprick size of the white dot indicating detection in figure 5.25 (b), although some false alarms were present, as shown by the white streaks in the object sweeping map. However this is easily remedied by selecting an appropriate threshold.

The head was simple to detect for both classes. The low resolution does not affect the head much, as long as the basic shape is preserved. The recognition system then only has to identify the coin, and the resulting head shape which remains in the low resolution image.

The tails for both classes performed poorly. The information which contains the intricate design of the tail is lost when we reduce the resolution, as seen in figure 5.25 (a). An example is the numerical digit. It is a valuable feature in the original image, but reduces to almost nothing when resolution is lowered.

This clearly refutes the original conjecture by the author that the system would be able to detect the features which were not visible to the human eye. The reason is that by reducing the resolution, we are also eliminating much valuable information required in detection.

There were no problems with the background, which was plain and expected to be easy to detect. The proportion of training examples correct before training was set at 99.5%, and




the neural network was trained 40 times.

We display the evaluation results for the current database in table 5.4

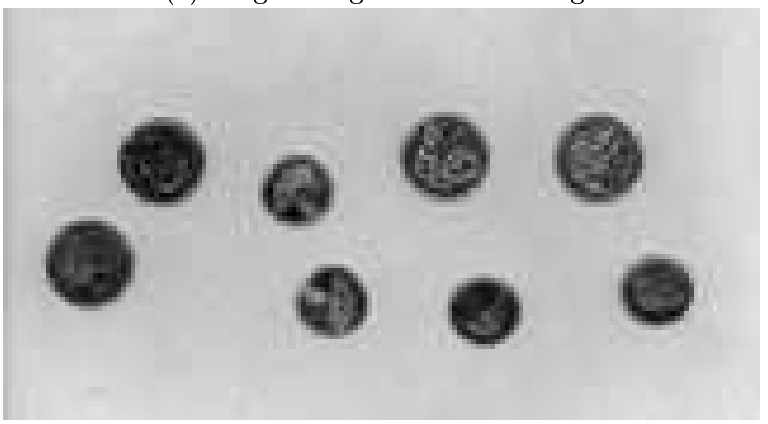
| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 50 |
| | FAR (%) | 200 | 150 | 0 | 0 | 0 | 0 | 0 |
| 2 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 600 | 500 | 500 | 500 | 500 | 450 | 400 |
| 3 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | DR (%) | 100 | 100 | 100 | 100 | 50 | 50 | 0 |
| | FAR (%) | 350 | 350 | 300 | 300 | 300 | 300 | 250 |

Table 5.4: Test Results : Database three

**Remark** A perfect result can be achieved for class 1 for any threshold greater than 0.5, and for class 3 using all thresholds. Both class 2 and class 4 failed to achieve perfect detection. The accompanying ROC Curve for the data above is shown.

### 5.6.3 Database Four

(a) Original high resolution image



(b) Original low resolution image

Figure 5.26: Database Four : Original images

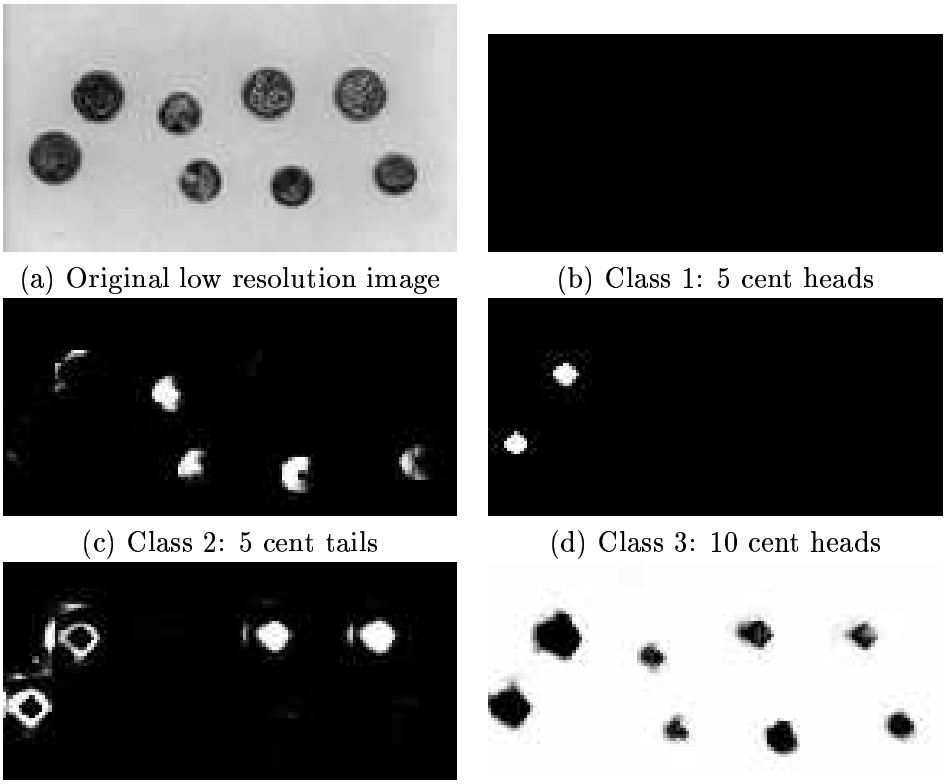
#### Overview of the Problem

This database introduces the extra problem of orientation. The limitations regarding low resolution mentioned in section 5.6.2 are still relevant, but barring that we assume that this database should not prove a great challenge to the network, as rotation is really just a re-ordering of the pixels.

Rotation does not affect position invariant statistics such as the mean and standard deviation, as the pixel simply moves to a different region or a different position within the same region (depending on degree of rotation). As we take the moment from the centre of the input field, rotation will not affect the moment as the pixels in question are still the same distance from the centre after rotation, albeit in a different position. Of course this would not the case if the moment were to be taken from the top left hand corner of the input field.

The author conjectured that similar performance to database three should be achieved for the above reasons.

**Discussion and Analysis**



(a) Original low resolution image (b) Class 1: 5 cent heads

(c) Class 2: 5 cent tails (d) Class 3: 10 cent heads

(e) Class 4: 10 cent tails (f) Background

Figure 5.27: Database Four: Test results

As with the third database, the system achieves its best results on the 10 cent heads, figure 5.27 (d) but the system fails to identify the 5 cent heads at all (figure 5.27 (b)). Subsequent test runs yielded the same result for the 5 cent heads, and a check of the incorrectly classified training examples showed that the incorrect classifications were across all classes.

A possible reason could be that the training examples were markedly different with regard to illumination, hence providing different pixel statistics properties compared to the pixel statistics of the 5 cent coin in the swept image in figure 5.26.

The recognition system had difficulty with tails of both classes (figure 5.27 (c) and (e)), although not as bad as in database three. This is verified by table 5.5, where the false alarm rates for the tail classes are lower. Excluding the first class, performance compared to database three is similar. We should note that the failure to produce a similar result to database three is most likely due to the fact that neural networks rarely produce the same results even on exactly the same data set.

The background in this database exhibits signs of overtraining, as the coin edges are no longer correctly defined. If the background detection was perfect, then all coin objects should be black and the remainder white, as mentioned in section 5.6.1.

The proportion of training examples correct before training was set at 99.79%, and the

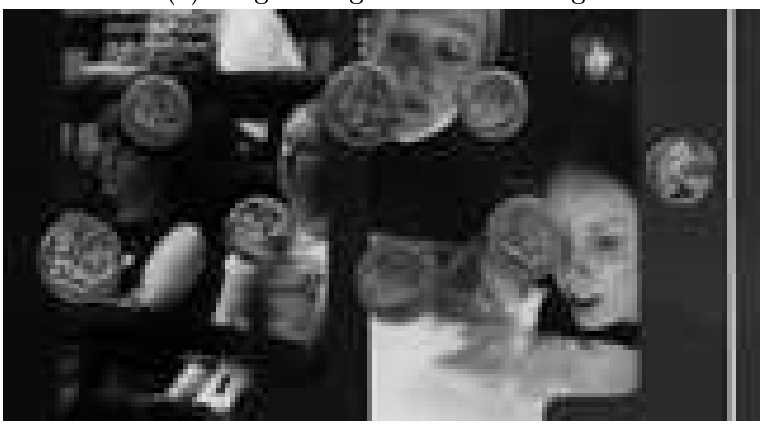
neural network was trained 15 times.

| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | FAR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 350 | 350 | 350 | 300 | 300 | 250 | 250 |
| 3 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 650 | 600 | 500 | 200 | 200 | 150 | 100 |

Table 5.5: Test Results : Database four

### 5.6.4 Database Five


(a) Original high resolution image



(b) Original low resolution image

Figure 5.28: Database Five : Original images

#### Overview of the Problem

In this chapter we take our recognition system to its limits. Not only must it distinguish between the coins, the respective heads and tails with various orientations, it must now do all the above tasks in a noisy background.

We investigate if the pixel statistics features we input are sufficient to train the network to extract only relevant data. We could have a better chance of solving the problems of the previous databases by using higher resolution images (allowing a greater number of features to train and test the network), but high resolution images are unlikely to aid object detection as we do not have the a simple and uncluttered background.

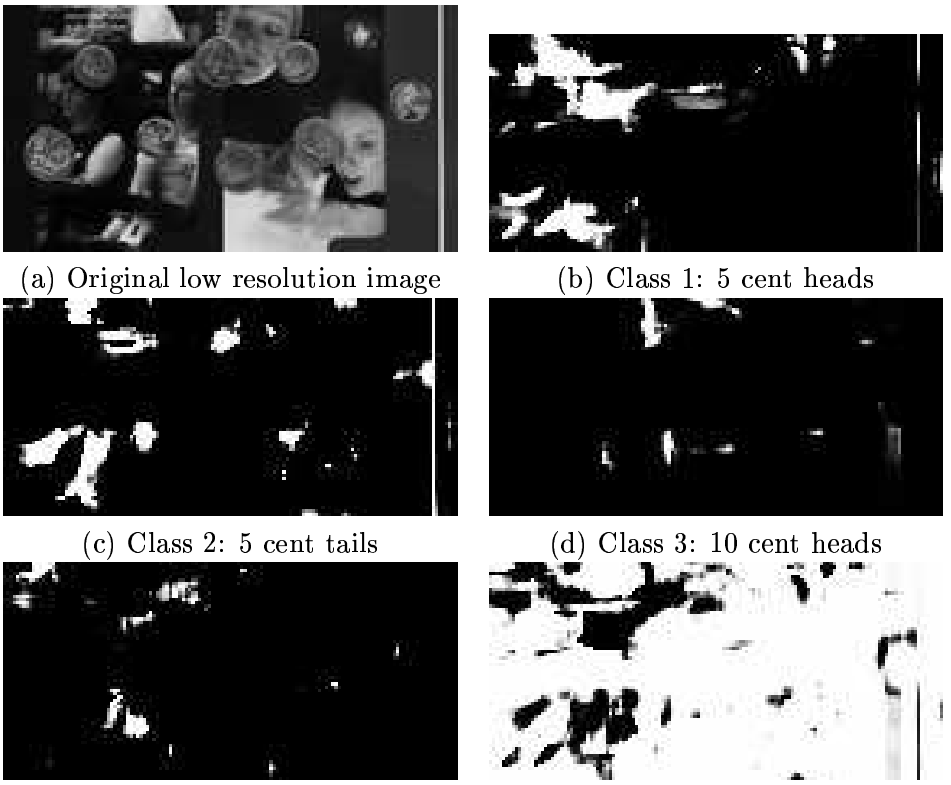
We can represent our objects of interest as as pixel statistics features, but it will not be easy for the recognition system to apply the learned solution to an unseen image.

The difficulty is to train the neural network to ignore the background, but no number of training examples could possibly represent this class except for the most artificial of examples. The solution to the problem is thus not learnable, as it cannot represented in its

entirety, and hence it cannot be learned in its entirety.

As an example, even if we keep the noisy background constant across all images in the database, the example of the object of interest trained on is unlikely to be in the same place on the noisy background as the same object in the unseen image during sweeping.

**Discussion and Analysis**



(a) Original low resolution image (b) Class 1: 5 cent heads

(c) Class 2: 5 cent tails (d) Class 3: 10 cent heads

(e) Class 4: 10 cent tails (f) Background

Figure 5.29: Database Five: Test results

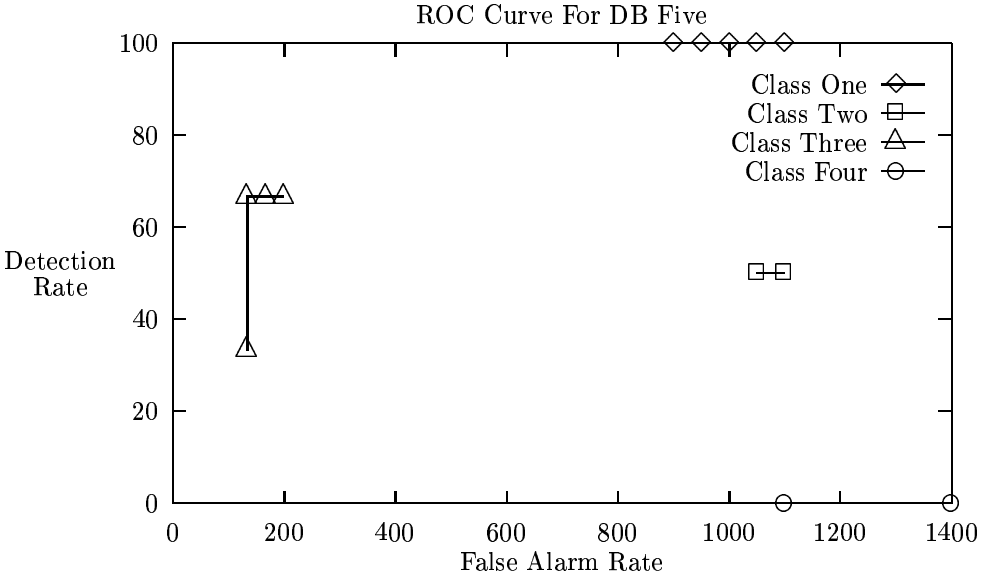
Not much meaningful analysis can be made of figure 5.29. The background noise has made the task very difficult for the system. Also remember that the noise in the training and testing examples are most definitely not the noise in this example.

The proportion of training examples correct before training was set at 99.5%, and the neural network was trained 40 times.

The poor result in this last database was not surprising. The reader may wonder whether using a higher resolution may aid detection or not. The main problem is this: Even if we use the same noisy background (as in this section) for all images, the statistical properties of the examples trained and tested on are very unlikely to be even remotely similar to those in the sweeping, as the pixels in each region are different, depending on where the coin is placed. The smaller the coin, the greater effect the noise will have, since the coin fills up less of the input field, thus allowing the noisy background to contribute more to the pixel input in the

| Class | Threshold | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|
| 1 | DR (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | FAR (%) | 1100 | 1050 | 1050 | 1000 | 1000 | 950 | 900 |
| 2 | DR (%) | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| | FAR (%) | 1100 | 1100 | 1100 | 1100 | 1100 | 1100 | 1050 |
| 3 | DR (%) | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 33.33 |
| | FAR (%) | 200 | 200 | 200 | 166.67 | 166.67 | 133.33 | 133.33 |
| 4 | DR (%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | FAR (%) | 1400 | 1100 | 1100 | 1100 | 1100 | 1100 | 1100 |

Table 5.6: Test Results : Database five




input field, hence producing different pixel statistics.

Thus the author conjectures that even with abundant features at a very high resolution, the system will still perform very poorly for this reason. The 5 cent coin still fills in the same portion of the input field, with the rest of the input field containing the noisy background.

# Chapter 6

# Conclusions

We found that it was of paramount importance in object detection to construct a large set of training examples. Failure to do so led to poor and inaccurate results. It was also important that not only were a sufficiently large number of training examples available, but enough of the right kind; the results we obtained attest to this.

Initial attempts at training the neural network to perform object detection on the artificially constructed images using only entire objects inside the input field for training and testing produced poor results. This concerned the author as perfect results on these artificial images are used as verification of the correctness of the recognition system. To solve the problem of insufficient training examples, the author specified a set of guidelines for constructing multiple examples consisting of partial objects within the input field. A program was written using these guidelines and perfect performance was obtained on the artificial images, thus verifying the correctness of the recognition system. The guidelines and the program were then used throughout the remaining databases to ensure that any future poor results would be the result of other causes and could not be attributed to a lack of training and testing examples.

We discovered several important factors that contributed to the success of object detection based on pixel statistics in general. Position invariant statistics such as the mean and standard deviation ignore the distribution of the pixel values across an input field. This has its advantages in cases such as rotation of an object of interest about a central point, as the pixels within the object simply change position within the input field. The value of the resulting position invariant statistic would still be the same value as the object in its original position within the input field. The disadvantage with a position invariant statistic occurs in more complex objects, where the position of the pixels within the input field are of greater importance in object detection to distinguish objects from non objects.

There was also a need for position dependent statistics such as the moment. Great care was required when selecting a point within the input field from which the moment was to be taken. We found that taking the moment from the centre was the most effective, as this also solves the problem of rotation. The pixels of a rotated object are still the same distance about the central point of the input field after rotation, but the value is unchanged. This would not the case if the moment was taken from the top left hand corner of the input field, for example.

Another factor was that the neural network in the recognition system was neither over or undertrained. We showed how to recognise both over and undertraining, and the effects that both had on object detection. The overly constrained model produced by overtraining

the neural network prevents the recognition system from correctly applying the model to an unseen image. The chaotic object sweeping maps are an indication that a model is too specific.

Undertraining the network produces models that are too simplistic to be of any use. When the generated model was applied by the recognition system to an unseen image, the system found it difficult to accurately detect the classes of interest. The resultant object sweeping maps show that almost every object is conjectured as an instance of a class. This results as the simplistic model fits many possible objects, but none of them well.

Regions were required for object detection in all coin databases but were not required in the first database consisting of artificially generated images, as they were so simple. Regions are parts of an input field, and statistics are taken for each region and then used for training into the neural network. Regions address the problem of pixel localisation, neglected by the position invariant statistics. This is because if an object changes its position within an input field, the corresponding invariant pixel statistics for each region will be different to when the object was perfectly centred within the input field.

With all these solutions incorporated into the recognition system, we applied the recognition system to the five databases. The results are summarised in table 6.1.

| Database | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Class 1 DR (%) | 100 | 100 | 100 | 0 | 100 |
| Class 1 FAR (%) | 0 | 25 | 0 | 0 | 900 |
| Class 2 DR (%) | 100 | 100 | 100 | 100 | 50 |
| Class 2 FAR (%) | 0 | 0 | 400 | 250 | 1050 |
| Class 3 DR (%) | N/A | N/A | 100 | 100 | 33.33 |
| Class 3 FAR (%) | N/A | N/A | 0 | 0 | 133.33 |
| Class 4 DR (%) | N/A | N/A | 100 | 100 | 0 |
| Class 4 FAR (%) | N/A | N/A | 300 | 100 | 1100 |

Table 6.1: Best results from each database

The perfect results of database one was expected. As expected for the coin databases (databases two to five) performance deteriorated as the difficulty of each problem posed by each database increased. This was also in line with expectations, as in all databases with the exception of the first, experiments were conducted using low resolution images.

From table 6.1, we see that databases three to five had the greatest false alarm rates. These databases had the additional task of distinguishing between heads and tails of the respective 5 and 10 cent coins. Low resolution made this task difficult, as the system had to learn to recognise the head and tail with fewer features. We found that the heads generally performed better than the tails, whose design is much more intricate. Features such as the numerical digit are lost when resolution was reduced. Time constraints prevented the author from investigating this, but future work in this area could consider running the same tests on the original, high resolution images, which would contain all the original features.

The perfect detection rates for databases two to four were due in part to the uniform background present in these databases. This property was not present in database five, which had a noisy and cluttered background.

# Chapter 7

# Acknowledgements

I would like to take this opportunity to thank both my supervisors, Dr Mengjie Zhang and Dr Peter Andreae at Victoria University of Wellington, New Zealand, for their continued support, encouragement and faith in me throughout the duration of this project which allowed me to bring the project to its successful completion.

# Bibliography

[1] R Beale and T Jackson. *Neural Computing.* Institute of Physics Publishing, 1994.

[2] Douglas Chai and King N Ngan. Automatic face location for videophone images. *IEEE TENCON'96*, pages 137–140, 1996.

[3] Simon Haykin. *Neural Networks; A comprehensive foundation.* MacMillan College Publishing Company Inc, 1994.

[4] Alan J. Lipton Hironobu Fujiyoshi Raju S. Patil. Moving target classification and tracking from real-time video. *Workshop Applications of Computer Vision*, 1998.

[5] Mengjie Zhang. A domain independent approach to 2D object detection based on the neural and genetic paradigms, 2000.