A Pixel-Based Approach to Recognising Small Objects in Large Pictures Using Neural Networks

Mengjie Zhang *

Department of Computer Science
Royal Melbourne Institute of Technology
GPO Box 2476V, Melbourne Victoria 3001, Australia

mengjie@cs.rmit.edu.au

Abstract

The development of traditional object recognition systems normally involves a time consuming investigation of 'good' preprocessing and filtering methods, an extraction of appropriate image features and a creation of/search for a suitable classifier for feature vectors. The major goal of this approach is to avoid these problems and develop a different method by using raw pixel data of both small objects and background in large pictures as inputs to neural networks. This paper presents a series of algorithms and techniques for this approach, including the determination of neural network architecture, network training and testing, small object detection and finding the centres of the objects, and an evaluation of the method. Experiments are carried out on easy, medium and difficult pictures. Both precision and recall can reach 100% for the easy and medium pictures, but the best precision is below 10% for the difficult pictures though its recall is 100% under a certain threshold. These results suggest that this approach can be used to recognise small objects with translation and rotation invariance but not size invariance in large pictures with uniform background.

Keywords Small object recognition, pixel-based neural method, network training, network testing, object detection, finding the centers of the objects, uniform background, highly cluttered background

1 Introduction

The volume of pictures stored in computer systems is very large and rapidly growing larger. In order to meet requests for applications, such as early diagnosis of fatal diseases by using x-ray pictures and land resource planning by using satellite photos, finding small objects in large pictures becomes more and more important. However, retrieval of pictures containing desired objects can be very difficult. There is, therefore, a clear need for systems which perform this task.

The object recognition task using traditional image processing methods [1, 2, 4, 13] usually involves the following subtasks: *preprocessing*, *segmentation*, *feature extraction* and *classification*. The main goal of preprocessing is to remove noise or enhance edges. Segmentation aims to divide the image into coherent regions. Feature extraction is concerned with finding transformations to map patterns to lower-dimensional spaces for pattern representation and to enhance class separability. The output of feature extraction, which often is a vector of feature values, is then passed to the classification step. The classifier is built or selected for the specific domain. Using these vectors, the classifier determines which are the distinguishing features of each object class, such that new vectors are placed in the correct class within a predetermined error. It can be seen that recognising small objects using this method involves a time consuming investigation of reasonable preprocessing and filtering methods, an extraction of important image features and a creation of/search for a suitable classifier for feature vectors.

Research on finding pictures containing desired objects using neural networks has been reported in recent years[3, 6, 7, 8, 9, 10]. In terms of the input patterns of neural networks, most work has focused on using image features as the input vectors. Similar to the traditional image processing method mentioned above, in this method features of small objects are at first identified by manual analysis of object samples. For example, morphological features (such as area, length and perimeter) and colour features (such as the mean value of the green pixels and the maximum value of the red pixels) were used in the quality assessment of lentils[12]. Programs are then hand-crafted to extract the features that will be used in a neural network classifier. Obviously, this method still has the problems of classical image processing method

*My first supervisor is Dr. Vic Ciesielski and Dr. James Thom is my second supervisor

Proceedings of the 1997 Computer Science Postgraduate Students Conference, Royal Melbourne Institute of Technology, Melbourne, Australia, December 9–10, 1997.

mentioned above. Some systems have excellent performance but generally require very long development and tuning times.

1.1 Hypotheses

In order to avoid the above problems, this approach aims to eliminate image feature extraction and related stages by directly using the raw image pixel data as the inputs to neural networks. Here, we explore a sequence of problems of increasing difficulty to determine the strengths and limitations of the technique. So, we investigate the following hypotheses:

- Will the pixel-based method work for detecting simple objects on a uniform background in a large picture?
- Can it be used for regular object detection in large pictures with a uniform background?
- Is it possible to find complex objects in large pictures with highly cluttered background using this method?

The second part of this paper describes a series of techniques and related algorithms for this approach. Experiments, results and analyses are given in section three. Our conclusions are presented in section four, and we end with some open questions and future work directions.

2 Design of a pixel-based neural method of recognising small objects

It is well known that the area of neural networks (connectionism) has become one of the two main strands of research in artificial intelligence. This strand is based on building computational models of neural activity in the human brain. To date, researchers have developed many different types of neural networks such as multi-layer feed-forward networks, ART (Adaptive Resonance Theory) networks, SOM (Self-Organising Maps) and recurrent networks. They have been used in different application areas. In this approach, we use the multi-layer feed-forward neural networks to recognise small objects in large pictures.

This method consists of four stages: the determination of the neural network architecture, network training and testing, small object detection and finding the centers of the objects, and the evaluation of the method. The working procedure of this approach is illustrated in Figure 1.

2.1 Determination of the neural network architecture

The network architecture will be determined as the first stage of this method. In the feed-forward neural networks, there is only one output layer and one input layer in the network architecture. There may be, however, more than one hidden layers or no hidden layer at all. Thus, this task consists of defining the number of input nodes, the number of output nodes, the number of hidden layers and the number of hidden nodes in each layer. One example for this kind of neural network is shown in Figure 2.

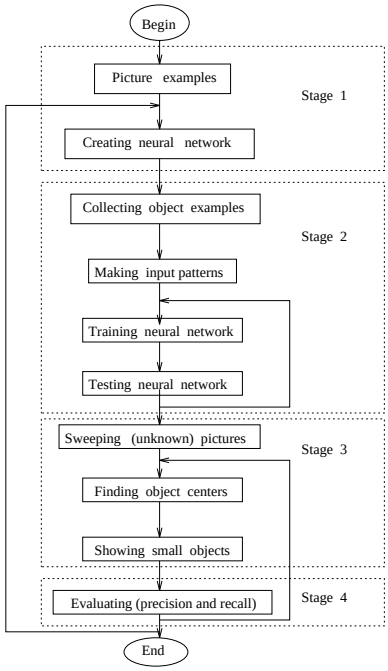


Figure 1: Flow chart of this approach

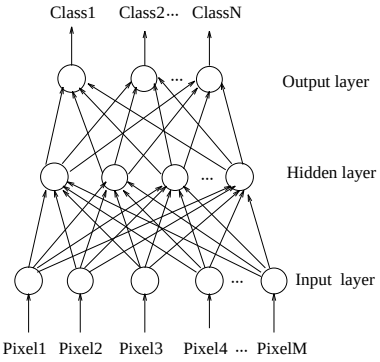


Figure 2: A sample neural network of this method

2.1.1 Definition of output nodes

In this pixel-based neural approach, the output nodes of the neural network are related to the classes of the small objects in a given application domain, as shown in Figure 2. As a general rule, the number of output nodes is defined as the number of classes of the small objects contained in the large pictures which usually include one background class called 'other', as shown in Eq. 1.

$$No.\ of\ output\ nodes = No.\ of\ classes\ of\ objects \quad (1)$$

In some cases (see section 3), more than one 'other' class is needed.

2.1.2 Definition of input nodes

In this approach, the raw pixel data are used as the inputs to the neural network, as shown in Figure 2. We use squares, each of which contains the small object and its background, as the training examples. Accordingly, the number of input nodes is mainly determined by the sizes of the small objects being recognised. Here, we propose a general formula, as presented in Eq. 2.

$$No.\ of\ input\ nodes\ =\ (size\ of\ object + n)^2 \quad (2)$$

where, 'size of object' means the biggest size of these objects to be recognised, and n stands for a certain size of the background surrounding the object. In this approach, for example, the typical value of the input nodes is $10^2 - 22^2$ (100-484 pixels).

2.1.3 Definition of hidden layers and hidden nodes

Unlike the input layer/nodes and output layer/nodes, the number of hidden layers and the number of hidden nodes in each hidden layer do not have any formal formulas. However, there is a general rule for the definition of the number of hidden layers and the number of hidden nodes in each hidden layer in this approach. The rule is:

$$Use\ as\ few\ as\ possible \quad (3)$$

There are two main reasons for this. One concerns the network training and object detecting time. The more hidden layers and/or hidden nodes are put into the network architecture, the longer the time that will be taken in network training and object detection. The other concerns the possibility of testing errors. Sometimes using more hidden layers can achieve results in high accuracy on the training set but a high error rate on the testing set. This is the problem of overfitting/overtraining. In Figure 2, there is only one hidden layer. Some problems related to network training will be discussed in the next section.

2.2 Neural Network training and testing

Network training is the second stage of this approach. It is very important because the whole learning procedure is included in this stage. Its main idea is as follows. After the creation of the network architecture and the collection of input patterns, the training algorithm should be selected or developed. When training the network using some of the input patterns (training set) and the selected algorithm, the weights inside the network are dynamically changed. When the total sum gap between the output value and desired value is smaller than a given critical error, all the weights and network architecture will be saved and the training will be stopped. Then the saved weights and the other collected input patterns (testing set) will be used for testing the network architecture. If the testing error percentage of incorrect testing patterns is reasonably low, then this network architecture and these weights will be used for small object detection in large pictures in next stage. Otherwise, it is necessary to change the network architecture and related parameters and re-train the network.

In this stage, the following tasks are required: collection of input patterns, selection of the training algorithm, determination of criteria for stopping the network training.

2.2.1 Collection of input patterns

In this approach, input patterns refer to a number of small object examples(containing some background) of every class in large pictures. All the raw pixel data of each small object example will be used as a input vector of the neural network. We divide these input patterns into two parts: *training set* and *testing set*. Generally speaking, the more classes of small objects in large pictures in a given problem domain, the more the input patterns that should be collected; the bigger the small objects, the more the input patterns that should be collected; and the more complicated background of the large picture, the more the input patterns that should be collected.

2.2.2 Selection of training algorithm

There are many existing training algorithms for feed-forward neural networks [14], such as back error propagation, back percolation and quickprop. Among them, back error propagation learning algorithm is the most common one for the feed-forward network training. In this approach, we chose the modified back error propagation algorithm in which the size of the weight change of a node is divided by the fan-in. The details of this modified algorithm can be found in [5].

2.2.3 Selection of termination strategy

There are three common strategies to stop the network training. The first one is the *error control* strategy, which uses the total sum squared error (TSS) or root mean square error(RMSE) of the training set as termination criteria. In other words, when TSS or RMSE is smaller than a given value, which is often a small floating number, the training is stopped. The second is the *epoch/cycle control* strategy, where the training will keep going until the training epochs/cycles reach a given number, which often is big integer. The last one is

the *user control* strategy, where the user/network trainer forces the training end when he/she thinks the training is OK or wants to check testing errors.

In this approach, we use the *error control* strategy as default. At the same time, the *user control* and *epoch control* strategies are also acceptable.

2.3 Object detection

When the network training and testing is finished, the network architecture and all the weights are saved and can be used for small object detection in large pictures.

Object detection aims to find the real positions of all small objects contained in a given large picture. This task includes three parts: network sweeping, finding the centers of the objects and showing the objects.

2.3.1 Network sweeping

The trained neural network acts as a template matcher in this stage. Then the template is swept across and down a given large picture containing some small objects to be found (in every possible location) pixel by pixel. After the sweeping is finished, a position file will be produced. The position file consists of three parts: the position (x, y) of each pixel, the maximum activation of the pixel produced by the template (neural network), and the class (corresponding to the maximum activation) of the pixel recognised/ classified by the trained neural network. At the same time, a PGM image for each class of the recognised objects is generated also, which includes all the activation values of this class produced by sweeping the template across and down the large picture. The examples of the sweeping output images are presented in section 3.

2.3.2 Finding the centers of the objects

Although every position/pixel in the large picture has been recognised to be a certain class in the network sweeping stage, the centers/coordinates of these objects in every class are not found. To do this, we propose two algorithms here.

A. Image-based algorithm

The first algorithm to find the centers of small objects is called *image-based algorithm*. It uses the PGM image of each object class generated by the network sweeping as the input, and finds the centers of the small objects of the given class as the output. This algorithm is described as follows:

For a given PGM image of one object class, do the following four steps:

S1: Give a *threshold*, and set to zero all pixels which are less than the threshold pixel value. For a PGM image, the value of every pixel is between 0 and 255. In this approach, the value of every pixel is equal to a integer value of its activation times 255. If a threshold is given as 0.8, then the threshold pixel value will be $0.8 \times 255 = 204$.

S2: Search for the maximum pixel value in this image. If the maximum value is not equal to zero, save the position (x, y).

S3: Set the value of all pixels in a given square, which has the same size as the template objects and has the center at the position(x, y), to zero.

S4: Repeat S2 and S3, until the values of all the pixels in this image are zero.

At the end of this algorithm, the centers of the small objects recognised by the neural network will be found in this given object class. By repeating this algorithm for each class, the object centers of other object classes can be found.

B. Location-based algorithm

We call the second algorithm *location-based algorithm*, which uses the only position file produced by the network sweeper as input, and finds the centers of the small objects of all the object classes in a given domain. The algorithm is:

For the position file,

S1: Find the range/scope of every object class.

S2: Give a *threshold*, and set to zero for the activation values of all locations whose activations are less than this threshold. Here, both the activation value and the threshold are floating values between 0 and 1.

For the first class, do steps S3, S4 and S5:

S3: Search for the maximum activation of one given object class. If the maximum activation is not zero, then save the corresponding location (x, y) to a given file of this given class.

S4: Set the activations to zero, if their corresponding locations are within the range of the training square size to position (x, y) above.

S5: Repeat S3 and S4, until all the activations in this given object class are zero.

S6: Go to next object class and repeat S3, S4 and S5, until all the classes have been parsed.

When this algorithm finishes, the object centers of all the classes to be recognised in the large picture will have been found.

The comparison of the two algorithms will be described in next section according to the experiment results.

2.3.3 Showing the objects

All objects in every class have been recognised and saved their centres to a corresponding centre file

in the stage of "finding the centres of the objects". However, users can not clearly see where these objects are in the original large picture. In this part, a new picture for each object class is produced, which shows all the centers of the small recognised objects of the class by using a given size squares surrounding them and the corresponding centre file. Through these pictures, users can intuitively know the positions of these objects recognised by the neural network. The algorithm is omitted here.

2.4 Evaluation

There are several existing criteria for gauging the performance of information retrieval and data mining systems. One widely accepted method is to use precision and recall tests. These tests have been used extensively in the field of information storage and retrieval and have proved to be an appropriate method for these kinds of systems[11]. The problem domain in this approach is one kind of picture/object retrieval or small object mining, so we choose them as the evaluation criteria.

Precision, or specificity as it is known in many pattern recognition tests, is the measure of how 'good' the information retrieved by a system is. In this approach, it means how many percent of the number of relevant small objects retrieved in the total number of retrieved objects by this method, as illustrated in Eq. 4.

$$precision = \frac{no.of.relevant.objects.retrieved}{total.no.of.small.objects.retrieved} \quad (4)$$

Recall, or sensitivity, is a measure of how much of the relevant information was retrieved by a system. Here, it means how many percent of the number of relevant small objects retrieved in the total number of relevant small objects in original database (of large pictures), as shown in Eq. 5.

$$recall = \frac{no.of.relevant.objects.retrieved}{total.no.of.relevent.objects.in.database} \quad (5)$$

Precision and recall tests are effective for evaluating performance in the data mining area, because it is important that the system can retrieve a high percentage of relevant information, and more importantly, maintains a high percentage of relevant information in the information retrieved.

In this approach, we used a tolerance of 4 pixels to measure the performance. For example, if the coordinates of a desired object are (21, 19) and the coordinates of an object retrieved are (22, 21), we still consider them as the same object though there is one and two pixels gap in x and y directions respectively.

3 Experiments and analysis of results

We perform three experiments on three different groups of large pictures containing different desired small objects with this method. The first one, which is an 'easy'-picture, contains only some small circles and squares. The second group of pictures include several heads and tails of different coins. The third group consists of a series of retina pictures, which contain many kinds of small objects such as haemorrhages and micro-aneurisms. The results and some analyses are given in this section.

3.1 Recognition of squares and circles

Here is an 'easy' picture. As can be seen from Figure 3, there are three kinds of small objects inside, class1 ('black' circles with 10 pixel size diameter), class2 ('gray' squares with the length of 8 pixels) and class3('white' circles whose diameters are 10 pixels). Each object was produced by a random number generator. Our goal is to recognise these small simple objects of every class in the uniform background picture.

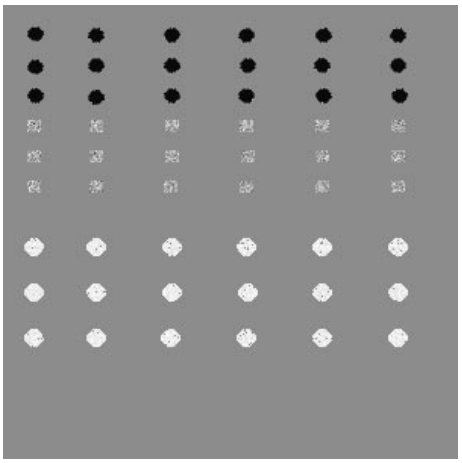


Figure 3: A sample 'easy' picture

After analysis of the problem domain, the neural network architecture to be used is shown in Table 1.

The number of output nodes is 4, corresponding to three classes of small objects mentioned before and one 'other' class(background). Because the sizes of the three classes of small objects are 10, 8 and 12 pixels respectively, we choose the maximum one, that is 12, plus a little background, for example 4 pixels, as the training object/square size, which is 16. Accordingly, the number of input nodes is $16^2 = 256$ (pixels). At the same time, we define only one hidden layer, which only has 8 hidden nodes. Other parameters such as lr, m, ce and r are for network training.

Parameters	meaning
lr 1.5	– learning rate
m 0.0	– momentum
ce 11.0	– critical error
r 0.5	– random range
3	– number of layers
256	– number of input nodes
8	– number of hidden nodes
4	– number of output nodes

Table 1: network architecture for square and circle recognition

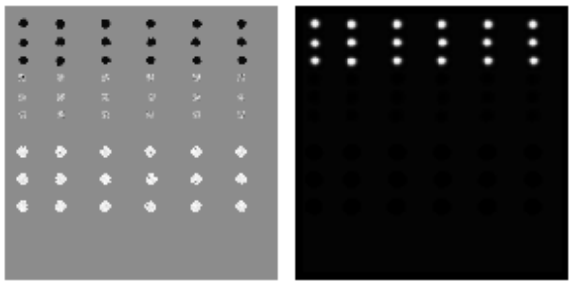
In order to train and test this neural network, 75 input patterns were cut out from the original picture. From these input patterns, 32 patterns were selected as the training set, and the other as the testing set. Table 2 shows the training and testing detail.

epochs	TSS	error in training set	error in testing set
100	10.972	0 / 32	10/43
150	9.075	0 / 32	10/43
200	7.391	0 / 32	5/43
282	1.483	0 / 32	0/43

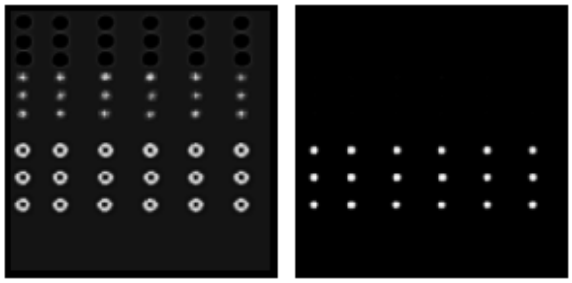
Table 2: network training and testing

According to this table, after 100 epochs of training, the TSS error was 10.972, which was less than the critical error 11.0, and the training was stopped. Here, the error in training set was 0/32 but the testing set error was 10/43, which was quite high. In this case, we changed the training parameter "critical error" to 9.1, 7.4, 1.5 and re-trained the network. After 282 epochs, the TSS error became 1.483, which is less than the critical error 1.5, the training was stopped and all the weights were saved. Because the error in the testing set was 0/43, the network architecture and the saved weights can be used in the object detection stage.

After the network sweeping, a position file and a PGM image for each object class are produced. These images together with the original picture are shown in Figure 4. During the sweeping, if there is no match between a square(supposed object) in the detecting picture and the template, then the neural network output is 0, which corresponds to 'black' in the sweep-output images; 'partial match' corresponds to gray on the center of the object, and 'best match' is close to 'white' in the output images. In this figure, the image for class 'others' is omitted.



Original picture Class1-sweep-output



Class2-sweep-output Class3-sweep-output

Figure 4: Output images in object detection

Using the image-based algorithm and the location-based algorithm, the coordinates of the centers of the small objects in each class were found. Table 3 shows the result of using the sweeping output image of class1 and the *image-based algorithm* with threshold of 85. Table 4 describes the result of using the position file and location-based algorithm under the same threshold.

Desired objects (x , y)	Retrieved objects (x , y)	gaps by pixels
21 , 19	21 , 19	0
21 , 41	21, 41	0
21 , 60	23 , 60	2
..., ...	... , ...	...
160 , 20	160 , 21	1
161 , 40	161 , 40	0
210 , 40	208, 40	2
... , ...	..., ...	...
Total no. of relevant objects in database= 18		
Total no. of retrieved objects = 18		
No. of relevant objects retrieved = 18		
Precision = 18 / 18 = 100.00%		
Recall =18 / 18 = 100.00%		

Table 3: Results using the image-based algorithm

Regarding the centers found by the two algorithms of object center finding, as shown in Table 3 and Table 4, the result based on the *location-based algorithm* is better than that using the *image-based algorithm*.

Compared with the location-based algorithm, the image-based algorithm has two main problems.

Desired objects (x , y)	Retrieved objects (x , y)	gaps by pixels
21 , 19	21 , 19	0
21 , 41	21, 41	0
21 , 60	21 , 60	0
... , ...	... , ...	...
160 , 20	160 , 20	0
161 , 40	161 , 40	0
210 , 40	210, 40	0
... , ...	... , ...	...
Total no. of relevant objects in database= 18		
Total no. of retrieved objects = 18		
No. of relevant objects retrieved = 18		
Precision = 18 / 18 = 100.00%		
Recall =18 / 18 = 100.00%		

Table 4: Results using the location-based algorithm

Firstly, different activations may lead to the same gray value (in a PGM image) of pixels. For example, suppose that there are two different activations 0.9038 and 0.9000. Their pixel gray values in a PGM image are $0.9038 \times 255 = 230.479 \approx 230$ and $0.9000 \times 255 = 229.50 \approx 230$ respectively. In other words, they have the same pixel value, that is, 230. This may make the system find 'wrong' object centers. Secondly, this algorithm needs to process all the images of all the object classes produced by the network sweeping. Each of these images has nearly the same size as the original large picture. Moreover, the more object classes there are in the original large picture, the longer the process will take.

Tables 5, 6 and 7 show the evaluation results of using different thresholds for class1, class2 and class3 respectively.

Threshold	0.80	0.90	0.95
Precision	18/18 (100%)	18/18 (100%)	18/18 (100%)
Recall	18/18 (100%)	18/18 (100%)	18/18 (100%)

Table 5: Results for class1 by using different thresholds

Threshold	0.40	0.45	0.49
Precision	14/61 (22.95%)	10/30 (33.33%)	7/7 (100%)
Recall	14/18 (77.78%)	10/18 (55.56%)	7/18 (38.8%)

Table 6: Results for class2

Threshold	0.80	0.99	0.995
Precision	18/18 (100%)	18/18 (100%)	18/18 (100%)
Recall	18/18 (100%)	18/18 (100%)	18/18 (100%)

Table 7: Results for class3

3.1.1 Improved network architecture and results

As can be seen from Table 5 and Table 7, the results for 'class1' and 'class3' are very good, where both precision and recall are 100%. According to Figure 4 and Table 6, the results for 'class2' are not good, where they can not achieve a reasonable performance at the same time. In particular, Figure 4 gives us some hints, which is that this trained neural network considers some of the surrounding objects of 'class3' as 'class2'. So, if we add one class, say 'other1', which stands for these surrounding objects of 'class3', hopefully, we may get better results. In this case, the number of output nodes will become 5, which are class1, class2, class3, other1 and other. So the network architecture will be changed to an improved one, which is presented in Table 8. The hypothesis is proved to be correct by the experiments, whose results are shown in Table 9, where both precision and recall are 100% if the threshold is chosen as 0.80.

Parameters	meaning
3	– number of layers
256	– number of input nodes
10	– number of hidden nodes
5	– number of output nodes

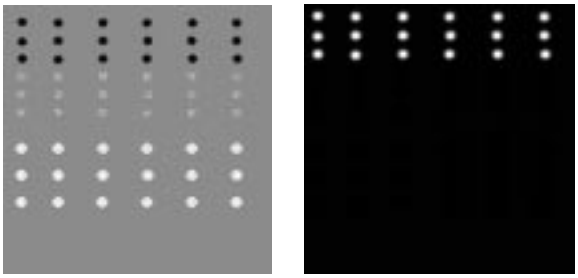
Table 8: Improved network architecture

Threshold	0.50	0.80	0.90
Precision	18/23 (78.23%)	18/18 (100%)	10/10% (100%)
Recall	18/18 (100%)	18/18 (100%)	10/18 (55.56%)

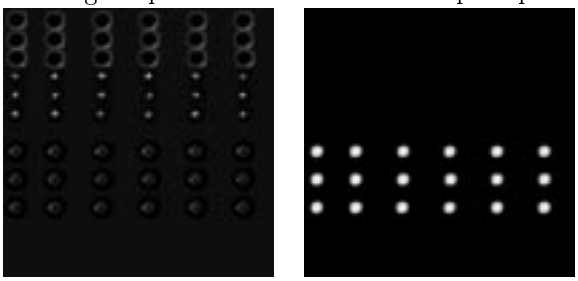
Table 9: Improved results for class2

The output images for class 'class1', 'class2', 'class3' produced in the object detection stage by using the improved neural network (with original picture) are illustrated in Figure 5.

Figure 6 presents the result pictures in the stage of "showing the objects", where all the small objects recognised by the improved neural network in



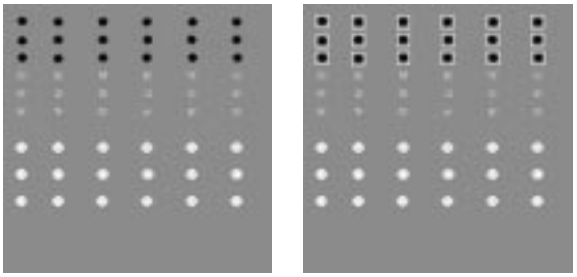
Original picture Class1-sweep-output



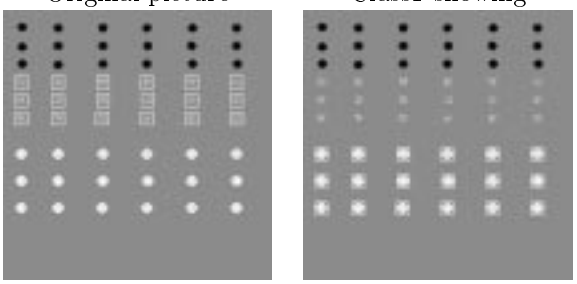
Class2-sweep-output Class3-sweep-output

Figure 5: Output images in object detection using improved neural network

each class are shown in the corresponding positions by using white surrounding squares. From this figure, we can clearly see where the small objects are.



Original picture Class1-showing



Class2-showing Class3-showing

Figure 6: Improved results of recognised objects – object showing

3.2 Recognition of coin tails and heads

In this part, experiments are presented for a medium difficulty problem, that of recognising tails and heads of different coins such as 5 cents, 10 cents, 20 cents, $1, $2, etc. We take a series of 640×480 pictures containing different sides, say tail and head, of different coins(i.e. 5c and 20c). Obviously, the small objects to be recognised here are regular. A sample coin picture containing both head side and tail side of 5 cent and 20 cent coins is shown in Figure 7.

Figure 7: Coin sample pictures

As can be seen from Figure 7, the coins in every class (i.e. 'tail005' – tail of 5c, 'head005' – head of 5c, 'tail020' – tail of 20c, 'head020' – head of 20c) are located in different positions, that is the translation problem, and are rotated in different angles, the rotation problem in image processing.

The neural network architecture is illustrated in Table 10. We take 215 input patterns, where 100 are used for training and 115 for testing. The network training and testing detail is presented in Table 11.

Parameters	meaning	description
3	no. of layers	
484	input nodes	training size = 22
10	hidden nodes	one hidden layer
5	output nodes	4 classes+'other'

Table 10: network architecture for coin recognition

We use the *user control* and *epoch control* strategies to stop the network training here. After 1000 epochs, the training was stopped, when the TSS error became 0.1, the error in training set was

epochs	TSS	error in training set	error in testing set
1000	0.100	0 / 100	25/115
1300	0.045	0 / 100	13/115
1400	0.033	0 / 100	12/115
1500	0.030	0 / 100	14/115

Table 11: network training for coin recognition

0/100, but the error in testing set was still 25/115, which was more than 20%. So we re-trained the network several times and stopped the training at 1300, 1400 and 1500 epochs, respectively. Because the error in the testing set was the lowest one (12/115) when the training was stopped at 1400 epochs, we used the network architecture and the saved weights of this training in the object detection stage.

As mentioned above, this trained neural network (template) is used to detect small objects of the four classes mentioned above in an 'unknown' large coin picture, which is not used to train or test the network. The results for the four classes are presented in Table 12, Table 13, Table 14 and Table 15 respectively.

Threshold	0.60	0.98	0.995
Precision	10/14 (71.43%)	10/10 (100%)	9/9 (100%)
Recall	10/10 (100%)	10/10 (100%)	9/10 (90.0%)

Table 12: Results for tail side of 5c

Threshold	0.60	0.99	0.995
Precision	5/24 (20.83%)	5/22 (22.73%)	4/17 (22.73%)
Recall	5/10 (50.00%)	5/10 (50.00%)	4/10 (40.00%)

Table 13: Results for head side of 5c

According to the results shown in these tables, two points should be mentioned here. Firstly, different thresholds should be selected for different classes of objects in order to get better performance. For example, for class 'head020', the best threshold is 0.60, where both precision and recall are 100%. Similarly, for class 'tail020', 0.98 may be a appropriate threshold. So how to get/select appropriate threshold automatically is necessary to be investigated. Secondly, unlike class 'tail005', 'tail020' and 'head020', the result of class 'head005' (table 13) is not good even if

Threshold	0.90	0.98	0.99
Precision	10/11 (90.91%)	10/10 (100.0%)	9/9 (100.0%)
Recall	10/10 (100.0%)	10/10 (100.0%)	9/10 (90.0%)

Table 14: Results for tail side of 20c

Threshold	0.85	0.60	0.50
Precision	9/9 (100.0%)	10/10 (100.0%)	10/12 (83.3%)
Recall	9/10 (90.00%)	10/10 (100.0%)	10/10 (100.0%)

Table 15: Results for head side of 20c

many different thresholds are tried. According to the experience in the 'easy'-picture, the better performance may be achieved if the network architecture is improved.

3.2.1 Improved network architecture and results for coin recognition

Based on the hypothesis above, we changed the network architecture by adding one class called 'other1', which stands for the class of objects surrounding 'head020'. Because the previous network architecture has a relatively high testing error, some hidden nodes are added to the architecture. Table 16 shows the improved neural network architecture. Table 17 presents the network training detail. The new performance for class 'head005' and 'head020' for the same large picture as that used by the previous network, are shown in Table 18 and Table 19.

Parameters	meaning	description
3	no. of layers	
484	input nodes	training size = 22
20	hidden nodes	one hidden layer
6	output nodes	4 classes+'other' + 'other1'

Table 16: Improved network architecture for coin recognition

Compared Table 18 with Table 13, it can be seen that the performance of class 'head005' coins under the new network architecture is much better than before. In Table 18, when threshold is 0.99, both precision and recall are 80%, while the best precision and recall under the previous network shown in Table 13 is 22.73% and 50% respectively.

epochs	TSS	error in training set	error in testing set
1000	53.359	3 / 120	37/137
2000	47.928	4 / 120	31/137
3000	43.220	3 / 120	27/137
4000	34.601	2 / 120	24/137
5000	23.629	2 / 120	21/137
6000	15.526	2 / 120	23/137

Table 17: Improved network training for coin recognition

Threshold	0.80	0.95	0.99	0.9971
Precision	9/19 47.37%	8/14 57.14%	8/10 80.0%	4/4 100.0%
Recall	9/10 90.00%	8/10 80.0%	8/10 80.0%	4/10 40.0%

Table 18: Results for head side of 5c under improved network architecture

Threshold	0.68	0.65	0.55	0.33
Precision	6/6 100.0%	8/8 100.0%	9/9 100.0%	10/10 100.0%
Recall	6/10 60.0%	8/10 80.00%	9/10 90.0%	10/10 100.0%

Table 19: Results for head side of 20c under improved network architecture

Thus, the network architecture is a key factor to get 'good' results.

Regarding the results of class 'head020' coins under the two network architectures shown in Table 15 and Table 19, the thresholds for achieving the best precision and recall, that is 100% for both here, are quite different. In table 15, the threshold for this is in 0.60 using the previous network architecture, whereas it is only 0.33 under new network shown in Table 19. If comparing Table 17 with Table 11, we can notice that the training TSS errors of the two networks have a big gap. The old network training TSS error is only 0.030 when the training stops, while it is 15.526 under the new network. This experiment suggests that the appropriate threshold is related to the training degree of the network architecture.

As can be seen from these result tables, different thresholds can cause different results. As far as the result of each class is concerned, it can be seen that the bigger the threshold is selected, the better precision may be achieved, the smaller recall might be obtained.

Furthermore, these result tables show that if the system is adjusted to increase recall, precision performance may be decreased, and visa versa. The development of this approach, which is a form of data mining in image databases, involves a trade-off between them.

3.3 Small object recognition in retina pictures

The third image database on which we performed experiments consists of a series of retina pictures, which contain many kinds of small objects such as haemorrhages, micro-aneurisms, veins, etc. The size of these pictures is 1024×1024 pixels. One example is illustrated in Figure 8, where all the haemorrhages and micro-aneurisms are shown using white surrounding squares.

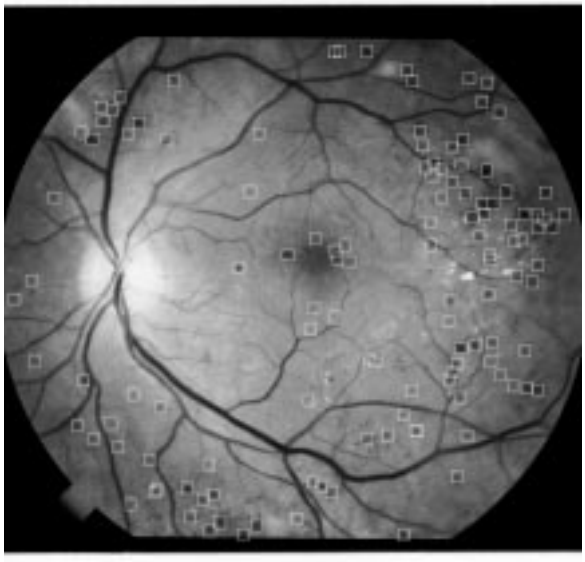


Figure 8: Retina sample pictures

In order to use this method to recognise small objects in these large pictures, we classify these small objects into 6 classes, which are presented in Table 20.

Class names	meaning
haem	haemorrhages
micro	micro-aneurisms
vein1	big veins
vein2	middle veins
vein3	small veins
others	background

Table 20: Small object classes in retina pictures

As can be seen from the sample picture, the sizes of the small objects in each class above are quite different and the background of these pictures is highly cluttered. This situation is totally different from that of 'easy-pictures' or 'coin pictures',

where some small objects that have the same or very similar size(s) are contained in each class, and the background is uniform or not very complicated.

The neural network architecture is created, as illustrated in table 21. We select 316 examples as input patterns, where 150 are used for training and 166 for testing. Table 22 shows the network training details.

Parameters	meaning	description
3	no. of layers	
256	input nodes	square size = 16
100	hidden nodes	one hidden layer
6	output nodes	object classes

Table 21: network architecture for small object recognition in retina pictures

TSS	error in training set	error in testing set
100.5	13 / 150	127/166
80.50	13 / 150	121/166
75.50	12 / 150	121/166
60.52	10 / 150	123/166

Table 22: network training and testing

After network sweeping, object coordinates finding and evaluating, the results of the first two classes, say 'haem' and 'micro', are presented in Table 23 and Table 24, respectively.

Threshold	0.50	0.60	0.85	0.90
Precision	5/660	5/622	3/494	3/458
	0.76%	0.80%	0.61%	0.66%
Recall	5/5	5/5	3/5	3/5
	100%	100%	60%	60%

Table 23: Results for class 'haem'

According to Table 23 and Table 24, the results, particularly the precision, are not good at all although the recalls are OK under some (different) thresholds (for different classes). We have tried to change the network architecture, for example, to change hidden nodes to 30, 50, 150, etc., but the similar results are obtained. That is, recall can be very good under a certain threshold, but precision is still below 10%. This experiment suggests that this method is not very successful if the sizes of small objects in each class are quite different and the background of the large picture is highly cluttered.

Threshold	0.76	0.80	0.85	0.90
Precision	8/202	8/168	7/118	4/67
	3.96%	4.76%	5.93%	5.97%
Recall	8/9	8/9	7/9	4/9
	88.9%	88.9%	77.8%	44.4%

Table 24: Results for class 'micro'

4 Conclusion

This paper presents a pixel-based approach for recognising small objects in large pictures by using multi-layer feed-forward neural networks. Four stages of this method are detailed: the creation of the neural network architecture, network training/testing, object detection/object coordinate finding and the evaluation of this method. After the experiments on three kinds of image data, we can draw the following conclusions.

Firstly, regarding the three hypotheses, the experiments suggest that this approach can recognise small simple and regular objects (which have the same or nearly the same size in each class) on a uniform background. The approach does not work very well for recognising complex objects (which have different sizes) on a highly cluttered background. In other words, this method can be used to perform translation and rotation invariant recognition of small regular objects on a uniform background.

Secondly, the neural network architecture plays an important role in this approach. Defining an appropriate network architecture is one of the key factors in getting reasonable results.

Thirdly, regarding each small object class, there can be different performance (precision and recall) if different thresholds are selected in the object detection stage. The results show that the bigger the threshold chosen the bigger the precision that can be obtained, but the recall will be smaller.

Moreover, different object classes need different thresholds to get good results under the same network training.

Furthermore, the results suggest that appropriate thresholds are related to the training degree of the neural network. In other words, the appropriate threshold for each class may be changed when the network training stops at different TSS errors (or epochs), that is, the smaller the TSS error when training stops, the bigger the appropriate threshold that is needed for each class.

Finally, this approach is not a totally automatic method of recognising small objects in large pictures. Inside the approach, most ideas and algorithms can work automatically, but two parts, that is, network architecture creation/improvement and threshold selection, still need some hand-crafted work.

5 Future work and open questions

Here, we propose some open questions for future work. Firstly, in order to make this approach become an automatic method of object recognition, the following two questions should be considered:

- Is there a general method of automatically selecting the threshold for object coordinate finding? The relationships between thresholds and precision, recall, and the degree of network training need to be investigated.
- Is there a general method of creating the neural network architecture automatically? The formulas and rules given in this paper can be considered.

Secondly, only several examples have been used to test this approach. More experiments are needed for further research; in particular, it is necessary to collect some sample image databases which contain more complicated background and some classes of small objects with translation and rotation variance but no size variance.

Thirdly, the trained network is swept pixel by pixel through a large picture. This is very inefficient when the picture is very large and only contains a few desired small objects. The techniques to solve this problem need to be investigated.

Moreover, this approach uses squares (containing small objects) as training examples. Can general rectangles or circles be used instead of squares?

Finally, regarding the training efficiency in this domain, is there a training algorithm which is faster to train the network than this back error propagation?

6 Acknowledgement

I would like to thank Dr. Vic Ciesielski, my first supervisor, who gives me a lot of guidance and correction of this paper. I also thank Dr. James Thom, my second supervisor, who gives me some advice to this work; Steven Hayes, whose programs ('list/multipat', 'nntrain', 'nntest' and the related environment) are used in the first two stages of this work; and Chris Kamusenski, who provides some retina image data.

References

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

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

[3] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Trans on PAMI*, Volume 15, Number 10, pages 1042–1052, 1993.

[4] Olivier Faugeras. *Three-Dimensional Computer vision – A Geometric Viewpoint*. The MIT Press, 1 edition, 1993. ISBN 0-262-06158-9.

[5] Steven Hayes and Victor Ciesielski. A neural network approach to positional and rotational invariant recognition of 3d objects. Technical report, Royal Molbourne Institute of Technology, Melbourne, Victoria, Australia, 1997.

[6] Robert Hecht-Nielsen. Neural networks and image analysis. In Carpenter and Grossbury (editors), *Neural networks for vision and image processing*. MIT Press, 1992.

[7] Robbins. Recognising 3 dimensional items – a neural net approach. This is a technical notes, 1994.

[8] M. Shirvaikar and M. Trivedi. A network filter to detect small targets in high clutter backgrounds. *IEEE Transactions on Neural Networks*, Volume 6, Number 1, pages 252–257, Jan 1995.

[9] Lilly Spirkovska and Max B. Reid. Higher-order neural networks applied to 2d and 3d object recognition. *Manufactured in the Netherlands*, pages 169–199, 1994.

[10] D. Valentin, H.Abdi, A.J.O'Toole and G.W.Cottrell. Connectionist models of face processing: A survey. *Pattern Recognition*, Volume 27, pages 1208–1230, 1994.

[11] Ross Willington. *Information Retrieval*. 1991. RMIT lecture notes.

[12] P. Winter, S. Sokhansanj, H. Wood and W. Crerar. Quality assessment and grading of lentils using machine vision. In *Agricultural Institute of Canada Annual Conference*, Saskatoon, SK S7N 5A9, Canada, July 1996. Canadian Society of Agricultural Engineering. CASE paper No. 96-310.

[13] A. Yli-Jaaski and F. Ade. Grouping symmetrical structures for object segmentation and description. *Computer vision and image understanding*, Volume 63, Number 3, pages 399–417, May 1996.

[14] Andreas Zell and Gunter Mamier. *SNNS User Manual*. University of Stuttgart, 1995. Version 4.1.