TARGET DETECTION WITH MORPHOLOGICAL SHARED-WEIGHT NEURAL NETWORK: DIFFERENT UPDATE APPROACHES

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

YIXUAN YE

James Keller, Thesis Supervisor

MAY 2018

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

TARGET DETECTION WITH MORPHOLOGICAL SHARED-WEIGHT NEURAL NETWORK: DIFFERENT UPDATE APPROACHES

presented by Yixuan Ye,

candidate for the degree of master of science,

and hereby certify that, in their opinion, it is worthy of acceptance.

JAMES KELLER

JIANLIN CHENG

MIHAIL POPESCU

ACKNOWLEDGMENTS

I would like to thank Dr. James Keller for his patient and thorough guidance throughout this project. His helpful leadership gave me the support I needed to finish this project. I can always be inspired by his ideas and comments. And I want to thank Dr. Mihail Popescu and Dr. Jianlin Cheng for being members of my committee.

I also want to thank my colleagues, Shuxian Shen and Anes Ouadou, whose willingness to discuss the basics of networks and problems in coding was a great help to me.

TABLE OF COTENT

ACKNOWLEDGMENTS ii

ABSTRACT xii

LIST OF FIGURES vi

LIST OF TABLES xi

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 LITERATURE REVIEW 4

2.1 Neural Network 4

  2.1.1 History of Neural Network 4

  2.1.2 Multi-layer Neural Network 6

  2.1.3 Backpropagation Algorithm 8

2.2 Mathematical Morphology 10

  2.2.1 Binary Morphology 10

  2.2.2 Grayscale Morphology 14

2.3 Evolutionary Algorithm 18

  2.3.1 History of Evolutionary Algorithm 19

  2.3.2 Genetic Algorithm and Memetic Algorithm 20

2.4 Shared-Weight Neural Network 25

  2.4.1 Convolutional Neural Network 26

  2.4.2 Morphological Shared-weight Neural Network 30

CHAPTER 3 IMPLEMENTATION 32

3.1 Network Structure 32

3.1.1 Feature Extraction Stage 33

3.1.2 Classification Stage 34

3.2 Details of Morphological Shared-weight Neural Network 34

3.2.1 Training Process 35

3.2.2 Test process 43

3.2.3 Initialization 43

3.2.4 Sub-image Generation 45

3.2.5 Morphological Operation 46

3.2.6 Update Approaches 49

3.2.7 Sub-image Replacement Method 56

3.2.8 Target Boxes 57

CHAPTER 4 RESULTS 59

4.1 Dataset 59

4.2 Results of Single Morphological Layer Network 62

4.2.1 Comparison of Loss Function 63

4.2.2 Comparison of Mutation Methods 68

4.2.3 Comparison of Initialization methods 73

4.2.4 Comparison of Three Update Approaches 78

4.2.5 Results of Parallel Morphological Shared-weight Neural Network 88

4.3 Results of Multiple Morphological Layer Network 88

4.3.1 Backpropagation 93

4.3.2 Evolutionary Algorithm with Partial Backpropagation 97

4.3.3 Comparisons of Two Morphological Layers Networks 101

4.4 Comparisons Between Networks with Different Structures 103

4.5 Comparisons between MSNN and CNN 106

4.5.1 Comparisons of One Layer Networks 106

4.5.2 Comparisons of Two Layer Networks 112

CHAPTER 5 CONCLUSION 120

REFERENCES 122

LIST OF FIGURES

Figure 2.1 An example of multi-layer feedforward neural network with one hidden layer........6

Figure 2.2 Neuron j in a typical neural network........7

Figure 2.3 Translation and reflection.........11

Figure 2.4 An example of erosion........12

Figure 2.5 An example of dilation........13

Figure 2.6 An example of hit-miss transform........14

Figure 2.7 The top or top surface of a set A........15

Figure 2.8 The umbra of the top surface of a set A........16

Figure 2.9 An example of erosion and dilation of a grayscale image........18

Figure 2.10 An example of CNN architecture........26

Figure 2.11 An example of convolution on a binary image........28

Figure 2.12 An example of max-pooling........29

Figure 3.1 Architecture of morphological shared-weight neural network........32

Figure 3.2 Architecture of a one-layer morphological feature extraction stage........33

Figure 3.3 An example of window U, and V of an input image........45

Figure 3.4 Hit operation.........48

Figure 3.5 Architecture of MSNN with two feature extraction layers in the feature extraction stage and one hidden layer for the neural network........49

Figure 3.6 Detection plane of a test image and the target is a white and black cup with a terra-cotta soldiers' pattern. 57
Figure 3.7 An example of boxes of target before elimination and after elimination. .. 58
Figure 4.1 Samples from Dataset I 60
Figure 4.2 Samples from Dataset II 61
Figure 4.3 Samples from Dataset III 62
Figure 4.4 An example of output image with one true positive and one false alarm... 65
Figure 4.5 ROC curves of five networks for two different loss functions. 66
Figure 4.6 Vertical average and concatenated ROC curves for two different loss functions. 67
Figure 4.7 ROC curves of three different mutation methods. 70
Figure 4.8 Vertical average and concatenated ROC curves of three different mutation methods. 72
Figure 4.9 Comparisons between three mutation methods. 73
Figure 4.10 ROC curves of three different initialization methods. 75
Figure 4.11 Vertical average and concatenated ROC curves of three different initialization methods. 76
Figure 4.12 Comparisons between three initialization methods. 77
Figure 4.13 Results of Dataset I and Dataset II with BP. 79
Figure 4.14 Results of Dataset III with BP. 80

Figure 4.15 Results of Dataset I and Dataset II with EAPB. 82

Figure 4.16 Results of Dataset III with EAPB...83

Figure 4.17 Results of Dataset I and Dataset II with MA..85

Figure 4.18 Results of Dataset III with MA. 86

Figure 4.19 Comparisons of the three updated approaches for

Dataset II and Dataset III. 87

Figure 4.20 Architecture of a parallel morphological shared-weight neural network. 89

Figure 4.21 Original and zoomed in ROC curves of parallel networks for Dataset I,

Dataset II and Dataset III. 90

Figure 4.22 Vertical average and concatenated ROC curves of parallel networks for

Dataset I, Dataset II and Dataset III. 91

Figure 4.23 Original and zoomed in ROC curves of two-layer morphological

networks trained by BP for Datasets I–III. 95

Figure 4.24 Vertical average and concatenated ROC curves of two morphological

layers networks trained by BP for Datasets I–III 96

Figure 4.25 Original and zoomed in ROC curves of two-layer morphological

networks trained by EAPB for Dataset I, Dataset II and Dataset III. 99

Figure 4.26 Vertical average and concatenated ROC curves of two-layer

morphological networks trained by EAPB for Dataset I, Dataset II and

Dataset III...100

Figure 4.27 Comparisons of two-layer morphological networks trained by BP and networks trained by EAPB for Dataset II and Dataset III. 102

Figure 4.28 Comparison between a two-layer morphological network trained by BP and a one-layer morphological network trained by EAPB for Dataset II and Dataset III. 104

Figure 4.29 Comparison between cascaded networks and parallel networks for Dataset II and Dataset III. 105

Figure 4.30 Original and zoomed in ROC curves of one-layer MSNN and CNN for Dataset I. 107

Figure 4.31 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset I. 108

Figure 4.32 Original and zoomed in ROC curves of a one-layer MSNN and CNN for Dataset II. 109

Figure 4.33 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset II. 110

Figure 4.34 Original and zoomed in ROC curves of one-layer MSNN and CNN for Dataset III. 111

Figure 4.35 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset III. 112

Figure 4.36 Original and zoomed in ROC curves of two-layer MSNN and CNN for Dataset I. 113
Figure 4.37 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset I. 114
Figure 4.38 Original and zoomed in ROC curves of two-layer MSNN and CNN for Dataset II. 115
Figure 4.39 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset II. 116
Figure 4.40 Original and zoomed in ROC curves of two-layer MSNN and CNN for dataset III. 117
Figure 4.41 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset III. 118

LIST OF TABLES

Table 4.1 Parameters of the single layer morphological network................................63

Table 4.2 Parameters of the single layer morphological network................................69

Table 4.3 Parameters of the single layer morphological network with BP..................78

Table 4.4 Parameters of the single layer morphological network with MA 84

Table 4.5 Parameters of parallel MSNN 89

Table 4.6 Parameters of a two layer morphological network trained by BP 94

Table 4.7 Parameters of a two -layer morphological network trained by EAPB.........98

TARGET DETECTION WITH MORPHOLOGICAL SHARED-WEIGHT NEURAL NETWORK: DIFFERENT UPDATE APPROACHES

Yixuan Ye

Dr. James Keller, Thesis Supervisor

ABSTRACT

Neural networks are widely used for image processing. Of these, the convolutional neural network (CNN) is one of the most popular. However, the CNN needs a large amount of training data to improve its accuracy. If training data is limited, a morphological shared-weight neural network (MSNN) can be a better choice. In this thesis, two different update approaches based on an evolutionary algorithm are proposed and compared to each other for target detection based on the MSNN. Another network training, based on back propagation, is used for comparisons in this thesis, which was proposed by Yongwan Won and applied by my colleague and fellow graduate student, Shuxian Shen and Anes Ouadou. Single-layer and multiple-layer MSNNs are both presented with different approaches. For a dataset, the author created part of a dataset for this thesis and used another dataset created by Shen to make comparisons with her network. Results of the MSNN are compared with CNN results to show the performance. Experiments show that for a single-layer MSNN, the performance of an evolutionary algorithm with partial backpropagation is the best. For a multiple layer MSNN, backpropagation performs better, although the MSNN still has a better performance than the CNN.

CHAPTER 1. INTRODUCTION

Machine learning is a popular topic that a lot of people from different fields focus on. Many have tried to define machine learning. One definition from Professor E. Alpaydin of Boğaziçi University defined machine learning as “programming computers to optimize a performance criterion using example data or past experience.” [1] Machine learning can be classified into two categories according to whether or not there is a reference value “label” for each output, i.e., supervised learning and unsupervised learning [2]. There are a lot of branches of machine learning, such as classification, regression, and clustering. Object detection is one application of machine learning, which uses supervised classification techniques [3]. An algorithm is given by a group of images with targets and other nontarget items. To realize object detection, an algorithm must find a target and label it as the target even though it is surrounded by other nontarget items and background. Sometimes object detection also requires algorithms to recognize targets at different distances. It is harder for an algorithm to recognize targets with different scales.

When developing a neural network, people propose many useful networks to solve object detection problems. One of the most of the widely used algorithms is the Convolutional Neural Network (CNN). CNNs are one of the most popular network architectures in deep learning. They are used for image or video recognition [4], language processing [5] and many other applications. With traditional image recognition algorithms, filters must be preprocessed manually, but a CNN can update filters based on input data and required output. It saves a lot of work and makes the whole processing easier. This is the major advantage of CNNs. Since the first CNN was introduced in the 1990s, programmers have developed many different structures of CNN. LeNet-5 is one of the most popular CNNs designed to recognize handwritten and machine-printed characters. Although CNNs have numerous applications and lots of advantages, CNNs still has their own disadvantages. One trouble with the CNN is that this network needs a large amount of training data to become robust and perform accurately. Another problem is the high computational costs of CNNs because of the enormous amount of input data. Although we have better central processing units (CPUs) and graphics processing units (GPUs) today, it is still a heavy task for a personal computer.

To overcome these kinds of problems, the morphological shared-weight neural network (MSNN) was introduced. The MSNN is a shared weight neural network using mathematical morphology [6]. The structure of MSNN is similar to the CNN, but the feature extraction stage is different. The feature extraction process in MSNN uses hit-miss transforms instead of convolutional transforms. Besides, the amount of training data that MSNN needs is less than CNN to achieve the same performance.

Moreover, MSNN performs better than other shared-weight neural networks when applied to target detection [7].

Both MSNNs and CNNs can update their own weights with backpropagation. Backpropagation is an algorithm which calculates error gradients with respect to network weights [8]. Although backpropagation is widely used for neural networks, it still has many disadvantages, such as gradient vanishing and low ability to avoid local minima. So, in this thesis, an innovative approach is proposed, which combines an evolutionary algorithm with backpropagation. This thesis also introduces another memetic algorithm approach.

CHAPTER 2. LITERATURE REVIEW

2.1 Neural Network

Artificial Neural Networks (ANNs) are computational algorithms which were inspired by how the human nervous system processes information. ANN can learn like the human brain when given input data. In biological neural networks, all biological neural functions are stored in the neurons and in the connections between them. So, people think about constructing a simple artificial neural network and train it to simulate functions of biological neurons. Since ANN was introduced in the 1940s, after several decades of researching, it has undergone many improvements and has become widely used in computer vision, pattern recognition, voice recognition and other problems. The next section briefly introduces the history of ANN.

2.1.1 History of Neural Network

Many people have been introduced to the advantages of neural networks in recent years. However, the first version of the neural network was introduced in 1943 by Warren McCulloch and Walter Pitts [9]. They created a neural network model based on mathematics and called it a threshold logic. Two approaches appeared based on this model. One concentrated on biological processes and the other focused on the application of neural networks.

In 1949, D.O. Hebb [10] created a learning hypothesis, which is now known as Hebbian learning. In 1954, Farley and Clark [11] first used a neural computational machine (referred to as a calculator) to simulate the Hebbian network. Other researchers continued to advance these machines and algorithms, which eventually led to machine learning.

However, in 1969, Minsky and Papert indicated two main issues hindering Rosenblatt's perceptions (algorithms developed for pattern recognition) [16] at that time. The first issue was that perceptrons (algorithms developed for pattern recognition) were unable to solve exclusive-or circuit problems; and the second problem was that computers in that era (the late sixties and early seventies) were not powerful enough to process large neural networks which were needed to meet the ever-growing computational needs.

Research on neural networks seemed to come to a standstill until Werbos's backpropagation algorithm was proposed in 1975 [13]. Multi-layer perceptrons trained by the backpropagation algorithm solved exclusive-or problem and speeded up the computation of networks. Neural networks continue to attract researchers' attention and many experiments have been added to this large body of research.

Many novel structured neural networks have made an impact in recent years. The recurrent neural network (RNN) was developed by Schmidhuber's research group and has won several competitions in pattern recognition and machine learning. The convolutional neural network (CNN), which was developed by Yann LeCun in 1996 [14], has had a remarkable performance in image processing. The morphological shared-weight neural network created by Y. Won also provide an outstanding network for image processing [6].

2.1.2 Multi-layer Neural Network

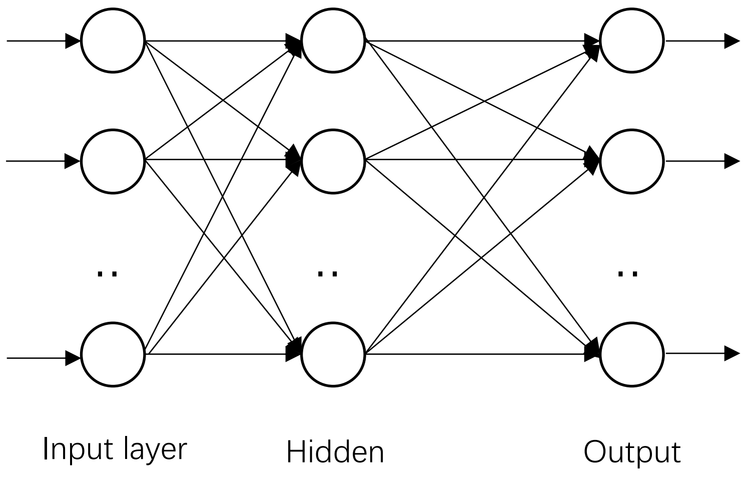


Figure 2.1 An example of multi-layer feedforward neural network with one hidden layer

A typical neural network consists many layers with many neurons in the same layer. The simplest neural network only has one input layer and one output layer, but it can be complicated by adding more hidden layers. Figure 2.1 is an example of a feedforward neural network with one hidden layer.

A feedforward neural network performs a nonlinear input-output mapping. Calculation equations between neurons in the network are shown as follows.

In Figure 2.2, a set of function signals are produced by neighbor neurons located in the left layer, which is the input of neuron j [15].

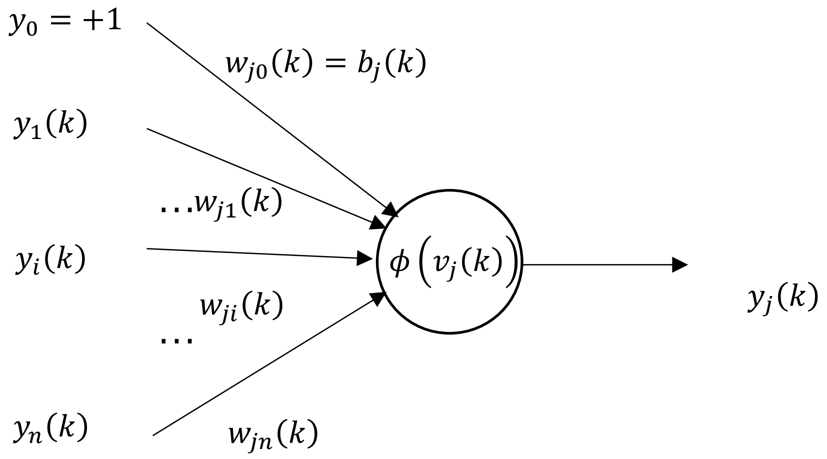


Figure 2.2 Neuron j in a typical neural network

The input of neuron j is $[y_0, y_1(k), y_i(k), \ldots y_n(k)]^T$, where y_0 is the fixed input related to weight $w_{\mathrm{j}0}$, which equals the bias b_j of the neuron j. The weight related to neuron j is $\left[w_{\mathrm{j}0}, w_{\mathrm{j}1}(k), w_{ji}(k), \ldots w_{jn}(k)\right]$. The output of neuron is calculated as

$$y_j(k) = \phi_j\left(v_j(k)\right) \tag{2-1}$$

and

$$v_j(k) = \sum_{i=0}^{n} w_{ji}(k)\, y_i(k) \tag{2-2}$$

where $v_j(k)$ is the activation function of neuron j. Many different activation functions are available including the popular sigmoid activation function and the rectified linear unit.

2.1.3 Backpropagation Algorithm

As mentioned in the last section, each neuron in the neural network has their own associated weights. The network must be trained before using it. Today, the workhorse of learning methods is backpropagation. Backpropagation can be traced back to 1957. The perceptron algorithm was invented in 1957 by Frank Rosenblatt, which is considered as an original form of backpropagation [16]. The basics of backpropagation were introduced by Kelley in 1960 [17]. In 1975, a general method was published by Werbos [13]. This method was related to the general version of backpropagation. In1986, Rumelhart et al. [18], noted that this backpropagation method is useful, efficient, and better than all other traditional learning methods. Their advocacy of backpropagation attracted people's attention, and it became widely used on neural networks.

With backpropagation, weights are updated by gradient descent, which means weights of neurons are adjusted by calculating the gradient of the cost function. Cost function can also be considered as a loss function or error function. In backpropagation, a common cost function is to calculate the difference between the expected label and the real network output.

$$e_j(k) = d_j(k) - y_j(k) \tag{2-3}$$

where $d_j(k)$ is the expected label and $y_j(k)$ is the actual output of neuron j. Given the training sample $\{x(k), d(k)\}$, the total error energy of the whole network is

$$E(k) = \sum_j E_j(k) = \frac{1}{2}\sum_j e_j^2(k) \tag{2-4}$$

To minimize the total error energy, the backpropagation algorithm applies a $\Delta w_{ji}(k)$ to $w_{ji}(k)$. $\Delta w_{ji}(k)$ can be obtained by calculating the partial derivative $\frac{\partial E(k)}{\partial \omega_{ji}(k)}$. According to the chain rule,

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = \frac{\partial E(k)}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial y_j(k)}\frac{\partial y_j(k)}{\partial v_j(k)}\frac{\partial v_j(k)}{\partial w_{ji}(k)} = -e_j(k)\phi_j'\left(v_j(k)\right)y_i(k) \tag{2-5}$$

and

$$\Delta w_{ji}(k) = -\alpha\frac{\partial E(k)}{\partial w_{ji}(k)} = \alpha\delta_j(k)y_i(k) \tag{2-6}$$

where

$$\delta_j(k) = \begin{cases} e_j(k)\phi_j'\left(v_j(k)\right) & if\ neuron\ j\ is\ an\ output\ neuron \\ \phi_j'\left(v_j(k)\right)\sum_l \delta_l(k)w_{lj}(k) & if\ neuron\ j\ is\ a\ hidden\ neuron \end{cases} \tag{2-7}$$

Neuron l is an output neuron connected to the hidden neuron j. And equation 2-6 updates weights based on the delta rule. The delta rule is a gradient descent learning rule with a backprop foundation, used to update the weights of artificial neurons in the neural network.

To increase the rate of learning and avoid instability at the same time, a momentum term was added to the delta rule. So, the update rule becomes

$$\Delta w_{ji}(k) = \alpha\delta_j(k)y_j(k) + \beta\Delta w_{ji}(k-1) \tag{2-8}$$

where β is positive and is also a momentum constant [19].

2.2 Mathematical Morphology

Mathematical morphology is an image processing technique, which uses mathematical theory based on set theory to digital images, graphs, and other spatial structures [20]. Some of the most widely used morphological operators include erosion, dilation, opening and closing. At first, mathematical morphology could only be used on binary images, but it eventually was expanded to grayscale images.

2.2.1 Binary Morphology

In binary morphology, images are viewed as subsets of the integer grid Z^d for dimension *d*. The basic operation of binary morphology is to find the binary image based on a predefined shape, which is called a structuring element (SE). The structuring element probes the binary image with small patterns. SEs are sets that consist of black and white pixels in a 2-dimensional image. Binary morphology is about operations on sets. SEs are used to operate on a source image to produce a destination image.

Before introducing morphological operations of binary images, three basic set operations should be introduced. Let Z^2 be the integer grid, and there are two subsets A and B of Z^2, $A \subset Z^2$, $B \subset Z^2$. Let x be a point or an element of a set [20].

The *translation* of *A* by *x* is defined as

$$A + x = \{a + x : a \in A\} \tag{2-9}$$

The *reflection* of *A* is defined as

$$-A = \{-a : a \in A\} \tag{2-10}$$

The *complement* of *A* is defined as

$$A^c = \{a \in Z^2 : a \notin A\} \tag{2-11}$$

Translation and reflection are shown in Figure 2.3.

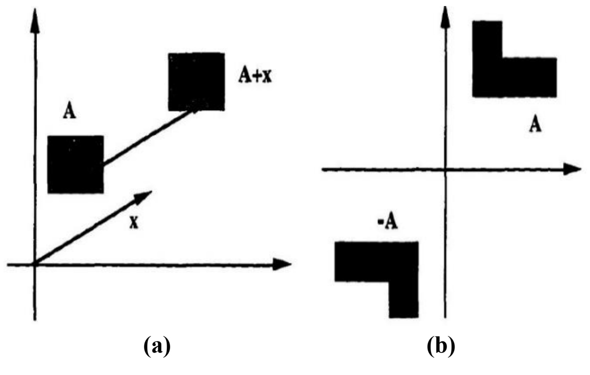


Figure 2.3 Translation and reflection. (a) Translation of a set *A* by *x* and (b) reflection of a binary set.

The erosion of binary image A by structuring element B is defined as [20]

$$E(A, B) = A \ominus B = \{x: B + x \subset A\} \qquad (2\text{-}12)$$

Erosion is a morphological operator, which can detect the locations where the structuring element fits the image. So, $A \ominus B$ should be all points x whose translations of B by x fits inside image A. An example of binary erosion is shown in Figure 2.4 [22].

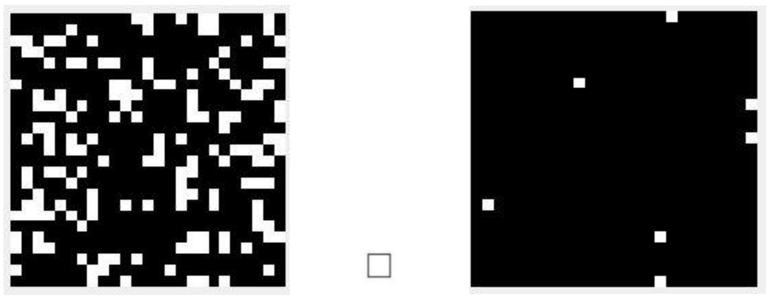
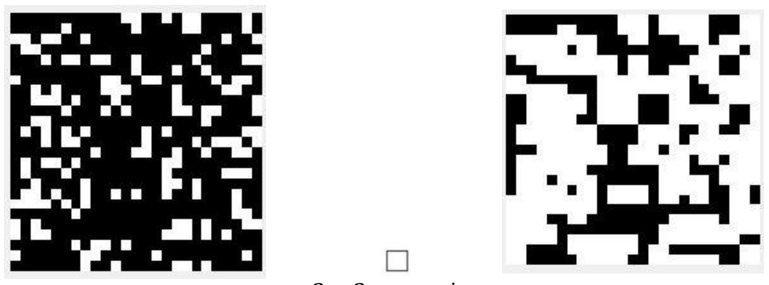


Figure 2.4 An example of erosion

Figure 2.4 shows an example of erosion. Structuring element B is a 2 × 2 image. The top left point of B is selected as the “center point”. After erosion, all points which meet the equation are reserved and the rest are eroded. That is why it is called “erosion.”

The dilation of binary image A by structuring element B is defined as [20]

$$A \oplus B = \{x: (-B + x) \cap A \neq \phi\} \qquad (2\text{-}13)$$



Original image A 2 × 2 structuring element B Dilated image D (A, B)

Figure 2.5 An example of dilation

Figure 2.5 is an example of dilation. Structure element B is the same one used in erosion. With dilation, the structuring element need to be reflected, so the “center point” is the bottom right point now. After dilation, all points which meet the equation are dilated. With dilation, the original image is enlarged by the structure element instead of erosion.

After erosion and dilation, a hit-miss transform is applied for shape detection. It is defined as

$$A \otimes B = A \otimes (E, F) = (A \ominus E) \cap (A^c \ominus F) \tag{2-14}$$

where E and F are two disjoint structuring elements. E describes the shape inside the target, while F describes the shape outside the target, which is the background. Figure 2.6 is an example of hit-miss transform. In the example, E is a structuring element which tries to find the desired foreground and F is a structuring element which tries to find background. Using equation 2-14, after intersection, points which match both structuring elements E and F are our target points.

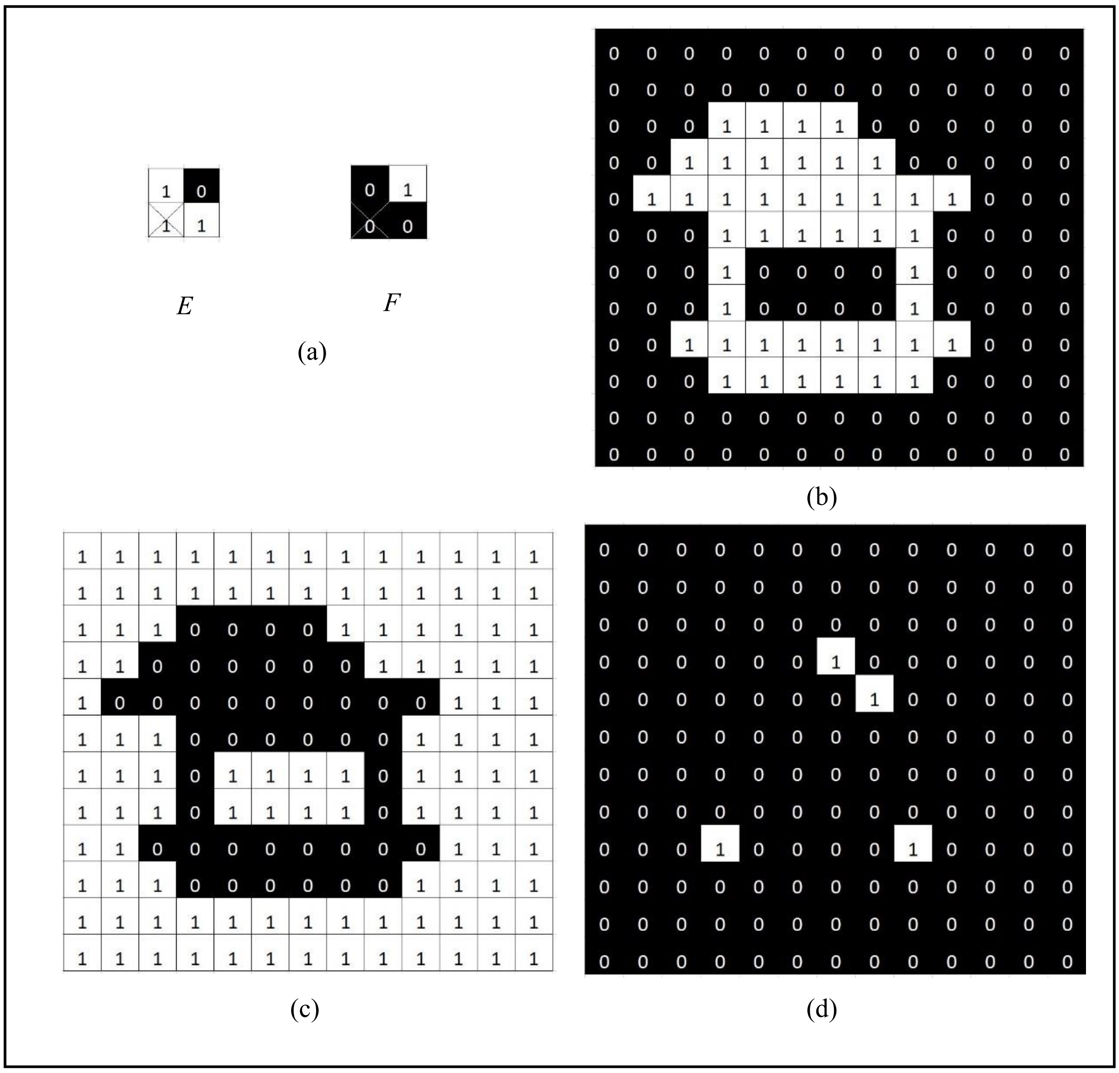


Figure 2.6 An example of hit-miss transform. (a) *E* and *F* structuring element with X as the center of the structuring element, (b) input image, (c) complement of input image, and (d) result of hit-miss transform.

2.2.2 Grayscale Morphology

Binary morphology is widely used for image processing. It can help us recognize a target, detect an outline, reduce noise, and do many other functions. However, operations only for binary images are not enough. What about color images and grayscale images? These images are more common than binary images; so, the morphology techniques were extended to grayscale images. Before introducing definitions of grayscale morphology, concepts of the surface of a set and the umbra of a surface should be described. In Euclidean *N*-space (E^N), suppose there is a set A. The top or top surface of A is a function defined on the projection of A onto its first (*N* – *1*) coordinates [23]. For the grayscale image, *N* = *3*, and for each x in the coordinate (*N* − *1*), the top surface of A at x is the highest value y such that $(x, y) \in A$. Using definitions in [23], let $A \subseteq E^N$ and $F = \{x \in E^{N-1} |\, for\ some\ y \in E, (x, y) \in A\}$. The top or top surface of A, donated by $T(A): F \to E$, is defined by

$$T[A](x) = \max\{y|(x, y) \in A\} \qquad (2\text{-}15)$$

Figure 2.7 show the top or top surface of a set A.

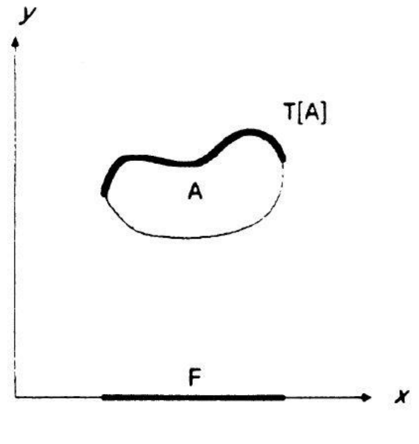


Figure 2.7 The top or top surface of a set *A*

Let $F \subseteq E^{N-1}$ and $f: F \to E$, the umbra of f, donated by $U(f), U(f) \subseteq F \times E$, is defined by

$$U(f) = \{(x, y) \in F \times E | y \le f(x)\} \tag{2-16}$$

Figure 2.8 is an example of umbra of the top surface of A.

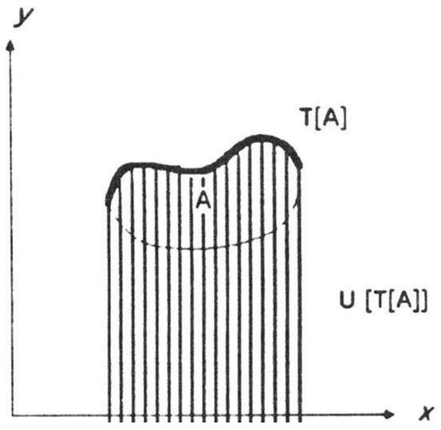


Figure 2.8 The umbra of the top surface of a set A

Based on the top surface and the umbra of the top surface of a set, grayscale erosion and dilation are proposed.

Let $F \subseteq E^{N-1}$ and $K \subseteq E^{N-1}$, Let $f: F \to E$ and $k: K \to E$. The erosion of f by k is denoted by $f \ominus k$, $f \ominus k: F \ominus K \to E$, and is defined by

$$f \ominus k = T\big[U[f] \ominus U[k]\big] \tag{2-17}$$

It can also be computed by

$$(f \ominus k)(x) = min_{z \in K}\{f(x + z) - k(z)\} \tag{2-18}$$

Let $F \subseteq E^{N-1}$ and $K \subseteq E^{N-1}$, Let $f: F \to E$ and $k: K \to E$. The dilation of f by k is denoted by $f \oplus k$, $f \oplus k: F \oplus K \to E$, and is defined by

$$f \oplus k = T\big[U[f] \oplus U[k]\big] \qquad (2\text{-}19)$$

It can also be computed by

$$(f \oplus k)(x) = \max_{\substack{z \in K \\ x-z \in F}} \{f(x-z) + k(z)\} \qquad (2\text{-}20)$$

An example of erosion and dilation of a grayscale image is shown in Figure 2.9. Values of each pixel of the original grayscale image are between 0 and 255. 0 is black and 255 is white. The smaller the value is, the darker the pixel is. The structuring element is a 5-pixel × 5-pixel disk. With erosion equation 2-18, the image after erosion keeps minimum values, which means it always keeps darker pixels. This is why the eroded image looks much darker than the original image, like figure 2.9 (c). And with dilation equation 2-20, it always selects brighter pixels because of maximum operation. So, the dilated image is brighter than the original image, which is shown in figure 2.9 (d).

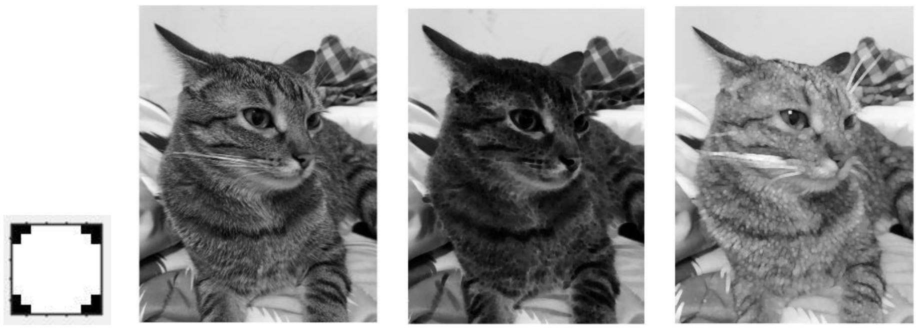

(a) Structuring element (b) original image (c) eroded image (d) dilated image

Figure 2.9 An example of erosion and dilation of a grayscale image

2.3 Evolutionary Algorithm

Evolutionary algorithms are metaheuristic optimization approaches inspired by biological evolution. In the real world, there are many NP (nondeterministic polynomial time) problems. They take too much time to solve for traditional algorithms to try each solution one by one. However, an evolution algorithm can find optimal solutions in a wide range of solutions and it doesn't need to traverse all possible solutions. The main process of an evolutionary algorithm is to create a population of parent individuals, quantify the qualities of the parent individuals with a fitness function, and then generate new individuals based on the parent individuals with crossover, mutation and other operations. Finally, evaluate new individuals and let the best ones take the place old individuals. By repeating this main process, the individual which optimizes the fitness function will be the best choice for a candidate solution of the original problem. Evolutionary algorithms are widely used in engineering, natural science, economics and other fields.

2.3.1 History of Evolutionary Algorithm

In this section, the history of the evolutionary algorithm will be introduced [24]. Evolutionary algorithms can be traced back to the 1950s [25]–[28]. However, due to the lack of powerful computers and the limitations of the initial algorithms, evolutionary algorithms did not show significant improvement until the1970s. The current evolutionary algorithm is inspired from three related approaches: genetic algorithms, evolutionary programming, and evolution strategies. Genetic algorithms were proposed by Holland [29]–[31], after which many studies were made by other researchers, such as De Jong [32], [33], Goldberg [34], [35] and Davis [36]. Applications of genetic algorithms mostly focus on function optimization. Evolutionary programming was developed by Fogel [37], [38] and improved by Burgin [39], [40], Atmar [41], and others. This succession of work helped to develop finite state machines (FSM) to predict results on the basis of former observations. Evolution strategies were introduced by Rechenberg [42], [43] and Schwefel [44], [45] and were designed for solving difficult parameter optimization problems.

In the1980s, development of computers enhanced the ability of perform large scale iterative computation. So, evolutionary algorithms were possible to solve complex real-world optimization problems. Then, researchers put their attention on evolutionary algorithms and more work was done on topics such as genetic algorithms [46], [47], evolutionary programming [48], and other topics in this field. Now, genetic algorithms and memetic algorithms are two popular and widely used techniques in this domain. Details of these two algorithms will be introduced in the next section.

2.3.2 Genetic Algorithm and Memetic Algorithm

As previously mentioned, the genetic algorithm (GA) and memetic algorithm (MA) are popular now and in this experiment, these two algorithms are used to generate results. The main reason for the popularity of the GA and MA is that most existing algorithms for optimization can easily be stopped once they have reached the local optimum instead of the global optimum. However, a GA or MA can overcome the local minimum problem.

The following is a general structure of EA in pseudocode [49].

General Structure of Evolutionary Algorithm

```
    Initialize population;
    Evaluate the fitness of each population;
    Select individuals from populations as parents for first generation;
    While termination condition not satisfied DO
```

```
        Recombine parents;
        Mutate recombined parents to generate offspring;
        Evaluate offspring;
        Select individuals as parents for next generation;
    End While
```

Sometimes for a complex solution domain, an EA was not efficient enough to search all of a solution space. This has led many to attempt combining different techniques, which led to the development of the memetic algorithm (MA). The MA can be considered as a combination of an EA and a local search. It was first proposed by Moscato [50]. Here is the general structure of MA in pseudocode.

General Structure of Memetic Algorithm

```
    Initialize population;
    Evaluate the fitness of each population;
    Select individuals from populations as parents for first generation;
    While termination condition not satisfied DO
        Recombine parents;
        Mutate recombined parents to generate offspring;
        Do local search on each offspring;
        Evaluate offspring;
        Select individuals as parents for next generation;
    End While
```

Designing an evolutionary algorithm requires the determination of representation, initialization, selection and variation operators [14]. These will be introduced in next three sections.

2.3.2.1 Representation

Computers cannot directly process real-world problems without transforming solutions into forms they can deal with. For example, in the traveling salesman problem, assuming there are four cities, a representation could be

$$[1\ 2\ 3\ 4]$$

or another representation with a binary code could be

$$[001010011100]$$

with each three-bit related to the index of a city.

Holland [30] believed that the longer representations have more opportunities to explore the solutions. In fact, there is no best representation for all problems, and sometimes different representations yield similar results [51].

The following advice should be considered when determining a representation [14]:

1. The representation should provide immediate information about the solution.

2. The representation should be amenable to variation operators that are well understood for their mathematical properties and can exhibit a gradation of change.

3. Unless the objective is to explore the utility of a novel representation, utilizing established representations may allow more systematic and meaningful comparisons.

2.3.2.2 Initialization

After determining the representation of solutions, before training, a population of candidate solutions should be initialized as parents in first generation. Initialization ways are also needed to be determined. With different methods of initialization, results of the same structure in an evolutionary algorithm could be different. Kazimipour et al. [52] categorized many well-known initialization methods and the results revealed larger scale problems depending on how sensitive results were to the initialization methods. Moreover, different initialization methods could improve the initial quality and give a better starting point for the evolutionary algorithm [53].

2.3.2.3 Selection

In each generation of evolutionary algorithms, an algorithm needs to select offspring from its population using specific selection methods. Some common selection methods include plus/comma, proportional, and tournament [14].

Plus/comma selection uses notation $(\mu + \lambda)$ and (μ, λ) and these notations refer to the two different cases. The first case is μ parents creating λ offspring and then selecting best μ individuals from $\mu + \lambda$ individuals as parents in next generation. This is called plus selection. The second case is μ parents creating λ offspring and then selecting best μ individuals only from λ offspring as parents in next generation. This is comma selection.

Proportional selection is also called roulette wheel selection. This selection chooses parents in proportion to their fitness value to create offspring. The probability of parents to be selected is

$$p_i = f(i) / \sum_{j=1}^{\mu} f(j) \tag{2-21}$$

where p_i is the probability of ith parent to be selected, $f(i)$ is the fitness value of ith parent, and μ is the number of parents.

Tournament selection has several different versions, but the most common one is to randomly select a subset of size q (often $q = 2$) from the population and then select the best one of q individuals as one parent of the next generation. Then, repeat this process until enough parents are generated.

2.3.2.4 Variation Operators

Variation operators are used to search improved solutions in the solution space. Which means generating offspring from parents in evolutionary algorithm. What kind of operators should be used depends on the representation of the problem, selection methods, and other factors. There are two typical types of variation operators based on their arity. First type is a unary variation operator called mutation. With this operator, an offspring relies on one random selected parent. For example, using Gaussian mutation operator, an offspring is generated by adding a Gaussian random variable with desired mean and standard deviation to the random selected parent. Second type is a binary variation operator called recombination or crossover. This operator merges information from two parents into one or two offspring. For example, with the two-points crossover, suppose we have two parents which are $[p_{11}, p_{12}, p_{13}, p_{14}, p_{15}]$ and $[p_{21}, p_{22}, p_{23}, p_{24}, p_{25}]$. Two random indexes between 1 and 5 are computed. Let the two random indexes be 2 and 4. The offspring of these two parents should be $[p_{11}, p_{22}, p_{23}, p_{24}, p_{15}]$ and $[p_{21}, p_{12}, p_{13}, p_{14}, p_{25}]$.

2.4 Shared-Weight Neural Network

A shared-weight neural network consists of two stages. The first stage is called the feature extraction stage, which is used to extract the feature maps from input data. The second stage is the classification stage, which is a fully connected neural network. A common approach to feature extraction is convolution [54]. Another extraction method is the morphological hit-miss transform [21], which was used in this thesis research. The output of the first stage is the input of the second stage. “Shared-weight” refers to those weights that define the feature map as sharing the same weights. Matrixes consist of these weights are often called kernels or filters. For example, in a convolutional neural network, each kernel is used to process the entire input image instead of using different weights for each sub-image. Instead of having several weights to train, the same weights are used for each sub-image. Thus, the number of parameters is reduced, which makes networks faster and easier to train.

2.4.1 Convolutional Neural Network

The convolutional neural network (CNN) is in the category of a shared-weight neural network, which has proven effective in the image processing field. LeNet architecture was one of the first convolutional neural networks, which propelled deep learning into mass acceptance. It has three different layers: 1) the convolutional layer, 2) the pooling layer and 3) the fully connected layer as shown in Figure 2.10.

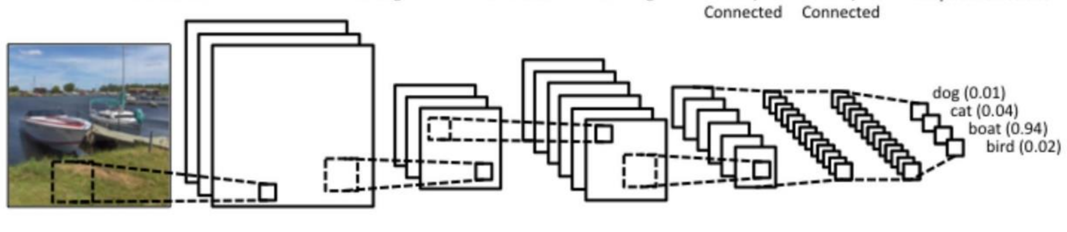


Figure 2.10 An example of CNN architecture [55]

2.4.1.1 Convolutional Layer

The purpose of the convolutional layer is to extract features from input data. For the CNN, inputs are usually images. Convolution maintains the relationship between pixels by learning features with small squares of input images. A feature map is obtained by convoluting the input image with a linear filter, adding a bias, and then applying a nonlinear function. Convolution is a linear operator. But in the real world, most data for CNNs to learn are nonlinear; thus, a nonlinear function is used to introduce nonlinearity to the network. If we have the kth feature map at a given layer, which is h^k, its filters are determined by weights W^k and bias b_k, after which, a feature map h^k is obtained as [56]

$$h_{ij}^k = \varphi\big((w^k * x)_{ij} + b_k\big) \tag{2-22}$$

where $\varphi(v)$ is a nonlinear function, such as sigmoid and ReLU functions.

The definition of convolution for a 1D signal is

$$o[n] = f[n] * y[n] = \sum_{u=-\infty}^{\infty} f[n]g[n-u]$$

$$= \textstyle\sum_{n=-\infty}^{\infty} f[n-u]g[n] \tag{2-23}$$

and convolution for a 2D signal is

$$o[m,n] = f[m,n] * g[m,n]$$

$$= \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[m-u,n-v] \quad (2\text{-}24)$$

Figure 2.11 shows convolution on a binary image. A convolution of a 5 × 5 image and 3 ×3 filter is computed. The filter is moved 1 pixel per step over the input image and convolves with the input image to compute convolved features.

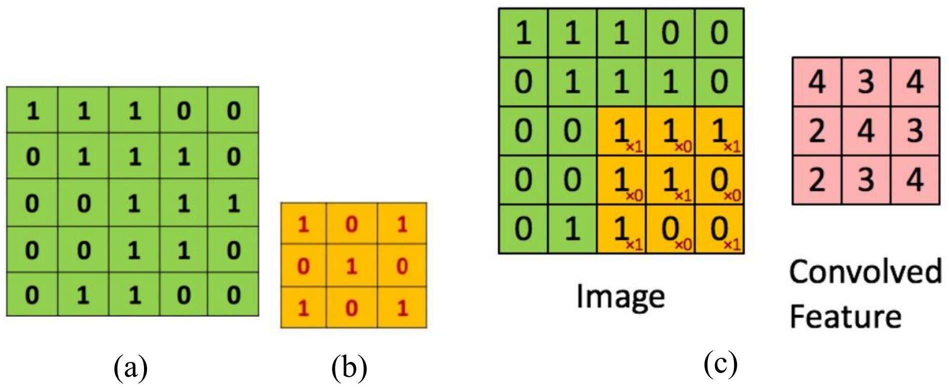


Figure 2.11 An example of convolution on a binary image. (a) input binary image. (b) 3 × 3 filter, and (c) result of convolution [57]

2.4.1.2 Pooling Layer

Pooling layer is used to reduce the dimensionality of feature maps and keep the valuable information at the same time. Different pooling types include max, average, and sum. In this thesis, max-pooling is used. So, details of max-pooling are introduced in this section.

Max-pooling partitions an input image into non-overlapping windows and outputs the maximum value in these windows. Benefits of pooling are reducing the size of representation, reducing the number of parameters, and controlling overfitting.

As shown in Figure 2.12, the input of the pooling layer is a 6 × 6 image; thus, the result of max-pooling with a 2 × 2 filter and a stride of 2 is a 3 × 3 feature map. The location of each maximum is recorded in a matrix.

0.34	0.32	0.8	0.12	0.25	0.76
0.08	0.92	0.13	0.56	0.73	0.26
0.49	0.62	0.15	0.28	0.41	0.84
0.28	0.07	0.73	0.66	0.81	0.47
0.34	0.91	0.48	0.14	0.59	0.42
0.45	0.37	0.12	0.52	0.62	0.05

(a) Input image

0.92	0.8	0.76
0.62	0.73	0.84
0.91	0.52	0.62

(b) Feature map

4	1	3
3	2	3
3	4	2

(c) Location

Figure 2.12 An example of max-pooling

2.4.1.3 Fully Connected Layer

A fully connected layer is a fully connected neural network. The term "fully connected" means that every neuron in the previous layer is connected to every neuron in the next layer. The output from the previous layer is the input of a fully connected layer. The purpose of this layer is to use these features for classifying the input image into various classes based on the training dataset. Details of this layer are the same as a multi-layer neural network.

2.4.2 Morphological Shared-weight Neural Network

The idea of combining mathematical morphology and other networks was proposed by Wilson [58] in 1989, which was the first time that mathematical morphology was combined with other networks. Wilson called these networks as morphological networks. The theory of a morphological network was described by Davidson [59]. Then, morphological networks began to be used on the image processing area. Davidson used it to solve template identification and target classification problems [60], [61]. The literature search conducted in this research could not find where the structuring elements used in a morphological layer were ever updated during backpropagation until 1995. Won introduced morphological shared-weight neural networks (MSNNs) which combines mathematical morphology with neural networks and developed backpropagation for MSNNs[21]. Won indicated that MSNN had better performance in the target detection field. Jin detected vehicles with morphological neural networks [62]. Although some literature has been published making mention of morphological shared-weight neural networks, the amount of research in this area is much less than that devoted to other algorithms. This is one of the reasons the morphological shared-weight neural network is discussed in this thesis.

The structure of a morphological shared-weight neural network is similar to a CNN. It still has a pooling layer and a fully connected layer. However, for the first stage, which is the feature extraction stage, the MSNN uses mathematical morphology instead of convolution. And for each feature map, MSNN has two filters while CNN has only one filter for each feature map. In the morphology layer, the hit-miss transform is applied for feature extraction. Furthermore, to compute the hit-miss transform, erosion and dilation operations need to be calculated first. These operations will be shown in the later sections.

CHAPTER 3. IMPLEMENTATION

3.1 Network Structure

This thesis focuses on an updated approach to the morphological shared-weight neural network. In this section, the structure of MSNN is introduced. As previously mentioned, the morphological shared-weight neural network has two stages, which are similar to the convolutional neural network. The difference between them is that the feature extraction stage uses hit-miss transform in MSNNs instead of convolution, which is used by CNNs. A schematic of the MSNN architecture is shown in Figure 3.1.

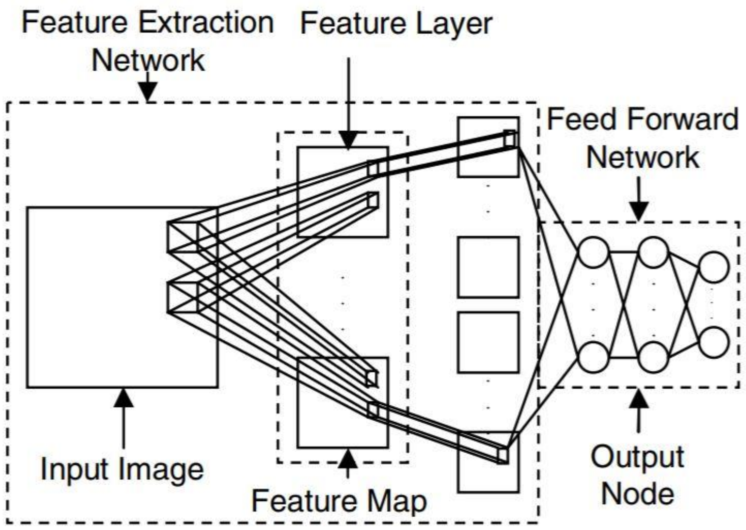


Figure 3.1 Architecture of morphological shared-weight neural network

The feature extraction stage extracts feature maps from the input image with hit-miss transform. After max-pooling, the output of first stage is the input of the classification stage. In classification, a fully-connected neural network recognizes whether the input is a target or not.

3.1.1 Feature Extraction Stage

The first stage is the feature extraction stage. In a morphological shared-weight neural network, the feature extraction stage applies grayscale hit-miss transform. The layer which computes hit-miss transform is also called the morphological layer. A morphological layer needs two kernels to compute the hit-miss transform, one for the hit operator and another one for the miss operator. Then the hit and miss operator results are combined to generate hit-miss transform. A max-pooling layer is applied to reduce the dimensionality of data and avoid overfitting. In this thesis, three different structures of the feature extraction stage are used: 1) the one-morphological-layer network, 2) the two-morphological-layer network and 3) the parallel morphological network. Figure 3.2 shows the one-morphological-layer feature extraction stage.

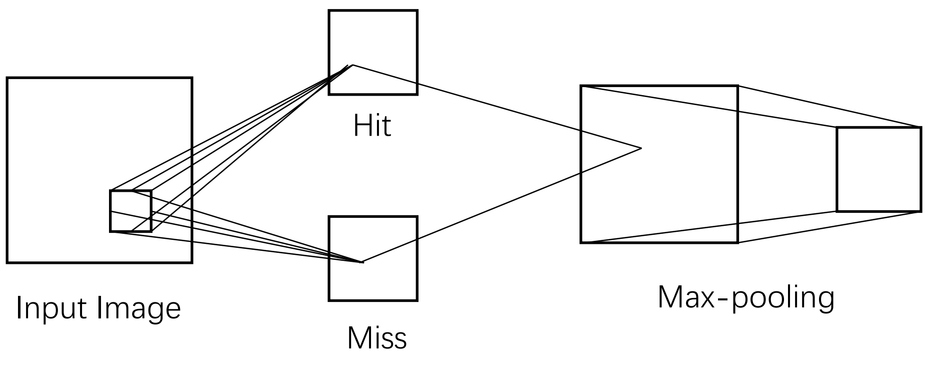


Figure 3.2 Architecture of a one-layer morphological feature

3.1.2 Classification Stage

The classification stage is a fully connected neural network. The output from the feature extraction stage is a 2-D matrix. However, the input of the classification stage should be 1-D vector. Thus, the output of the feature extraction stage will be transferred to a 1-D vector as the input of the classification stage. The architecture of the classification stage is a multiple-layer neural network with one input layer, one hidden layer and one output layer. In the output layer, there is only one neuron, and the output value is between 0 and 1. 0, which represents the input image background, while 1 represents the input image target.

3.2 Details of Morphological Shared-weight Neural Network

In this section, details of the morphological shared-weight neural network (MSNN) are introduced. First, a training process and a test process are proposed. For the training process, the author uses three different algorithms. They include this thesis's novel contribution to MSNN development, i.e., the evolutionary algorithm with partial backpropagation (EAPB), as well as backpropagation (BP) and the memetic algorithm (MA). Then, process details are introduced, such as sub-image generation, morphological operation and update approaches.

3.2.1 Training Process

The purpose of the training process is to settle all weights, which include kernels in morphological layers and neurons in the fully connected neural network. In this thesis, three diverse ways are used to train MSNNs.

3.2.1.1 Backpropagation Approach

The first approach, backpropagation, is one of the most popular and traditional methods, which is widely used to train most kinds of neural networks. In this thesis, Shen's backpropagation algorithm is used for MSNN with single morphological layer [22]. However, the author of this thesis expands it to a multiple morphological layer MSNN and makes a few small changes; moreover, algorithms are tested on new test dataset. So, some new details of backpropagation must be introduced to establish a proper context instead of just referring to Shen's thesis.

With the backpropagation algorithm, after settling all parameters and initializing structuring elements and neural network weights, the first step is to generate sub-images from training images, including target sub-images and background sub-images. These sub-images are then passed into the network one by one. After the feature extraction stage, feature maps are generated and input into the classification stage. Errors are calculated by a loss function of the difference between the actual output of a fully connected neural network and the expected output label, which is "1" for target sub-images and “0” for background sub-images. Square error function and cross-entropy function are used as the loss function in this thesis. Then, errors are backpropagated throughout the network. The weights of the neural network and structuring elements are updated by these backpropagated errors. When the error of a sub-image is below a well-trained threshold, the sub-image is replaced by another sub-image with replacement mechanisms. The pseudocode of this algorithm is shown as follows [22]:

Backpropagation Algorithm

```
Read N input image with targets and backgrounds; Epoch = 1; RandSelect = 0;
While (RandSelect < MaxRandSelect & Epoch < MaxEpoch).
    For n = 1 : N
        Randomly select M target sub-images, Tn1, …, TnM;
        Randomly select M background sub-images, Bn1, …, BnM;
    End For
    RandSelect = RandSelect + 1;
    ErrMonitor = 0;
    While (Epoch < MaxEpoch & ErrMonitor < ContinueLow).
        For sub-image in {Tn1, Bn1, …, TnM, BnM}
            Perform forward and backward propagation;
            If (PSS < WellTrained)
```

```
                Replace current sub-image with new one;
            End If
        End For
        Epoch = Epoch + 1;
        If (RMSE < StopErr)
            ErMonitor = ErrMonitor + 1;
        Else
            ErrMonitor = 0;
        End If
    End While
End While
```

3.2.1.2 Evolutionary Algorithm with Partial Backpropagation

Although backpropagation is a widely-used training algorithm for neural networks, and its training speed is faster than many randomly searched algorithms, this algorithm still has many limitations. Backpropagation is not efficient enough to avoid getting stuck at local minima, and it cannot explore a broad range of a solution space. This is because when using gradient descent, weights always move toward the local minima instead of exploring unfamiliar space. Evolutionary algorithms have been established as efficient enough to avoid local minima **and** explore broader solution spaces. However, the new evolutionary algorithm lacks speed when training due to its ability to generate new solutions almost randomly while exploring the broader solution space. This helps the algorithm to explore more but to converge more slowly. The author’s solution was to combine the evolutionary algorithm with backpropagation, which provides the innovative element proposed in this thesis, i.e., the evolutionary algorithm with partial backpropagation (EAPB).

With EAPB, weights of structuring elements are put together to form the chromosome of the evolutionary algorithm. Obviously, they are updated by EA. Thus, in each generation of EA, the best chromosome settles into the role of the current kernel; then, the weights of neural network are updated by backpropagation based on these kernels. That is why this algorithm is known as “partial” backpropagation.

For EAPB, after setting up all parameters, the first step is initializing populations of kernels and weights in neural network and then generating sub-images, including target sub-images and background sub-images. These sub-images are then passed into the feature extraction stage and classification stage one by one. Weights of kernels are updated by the evolutionary algorithm for one generation. Then, the best chromosome is selected as current kernels, and the weights of neural network are updated by backpropagation for one generation. This process repeats until reaching the stop condition.

```
Read N input image with targets and backgrounds;
Set up Max Epoch, probability of mutation and crossover and other parameters;
Initialize weights of neural networks and initial K parents of structuring elements;
For n = 1 : N
    Randomly select M target sub-images, Tn1, …, TnM;
    Randomly select M background sub-images, Bn1, …, BnM;
End For
For epoch = 1 : Max Epoch
    If epoch = 1
        Evaluate loss of initial parents;
    End If
    If rem(epoch,10) = 0
        Reselect sub-images;
    End If
    For j = 1 : K
        Select parents with proportional selection;
        Generate offspring with mutation and crossover;
    End For
    For j = 1 : 2*K
        Evaluate parents and offspring;
        Select the best K individuals from all parents and offspring as new parents;
        Select the best individual as the current structuring element which will be
        used in backpropagation for the neural network;
    End For
    For sub-image in {Tn1, Bn1, …, TnM, BnM}
        Perform forward and do backward propagation only for the neural
        network part;
        If (PSS < WellTrained)
            Replace current sub-image with a new one;
        End If
    End For
End For
```

3.2.1.3 Memetic Algorithm

With EAPB, the evolutionary algorithm works with backpropagation, and its results are compared with the results of traditional backpropagation. Then, another solution comes to mind based on the question: Why not use a memetic algorithm to update all weights including weights of the structuring elements and the fully connected layer? Notably, the memetic algorithm is a genetic algorithm with local search capabilities. It can, therefore, be used as the training algorithm in this section because it converges faster than the traditional genetic algorithm. All weights, which are the weights of structuring elements and the fully connected layer, are put into the chromosome and updated by the memetic algorithm. For a local search in MA, one generation of backpropagation is used.

With a memetic algorithm (MA), the first step is initializing the K populations, which consist of all the weights in the network. The next step is to generate sub-images and pass them into the feature extraction stage and the classification stage one by one. Then, parents are evaluated by loss function, and offspring are generated with mutation, crossover and local search. The best K populations from parents and offspring are selected as new parents in the next generation. This process is repeated until meeting the stop condition.

Memetic Algorithm

```
Read N input image with targets and backgrounds;
Set up Max Epoch, probability of mutation and crossover and other parameters;
Initialize K parents, which consists of all weights;
For n = 1 : N
```

```
        Randomly select **M** target sub-images, ***Tn1, …, TnM***;
        Randomly select **M** background sub-images, ***Bn1, …, BnM***;
    **End For**
    **For epoch = 1 : Max Epoch**
        **If epoch = 1**
            Evaluate loss of initial parents;
        **End If**
        **If rem(epoch,10) = 0**
            Reselect sub-images;
        **End If**
        **For j = 1 : K**
            Select parents with proportional selection;
            Generate offspring with selected parents by mutation and crossover;
            Do local search, which is one generation of backpropagation;
        **End For**
        **For j = 1 : 2*K**
            **For** sub-image in {Tn1, Bn1, …, TnM, BnM}
                Perform forward propagation and evaluate the individual;
                **If (PSS < WellTrained)**
                    Replace current sub-image with new one;
                End If
            End For
            Select best K individuals from all parents and offspring as new parents;
        End For
    End For
```

3.2.2 Test process

In the test process, instead of randomly selecting sub-images, a sliding window is used to go through the whole test image and sub-images from every location of the test images are generated by the window. Then these sub-images are passed into the network one by one, and output values are computed for each position. Thus, a new 2-D matrix is generated, and output values of each sub-image are settled on each corresponding location. This 2-D matrix is called the detection plane.

3.2.3 Initialization

In all three training processes, initialization is a common step. Some researchers found that initialization methods of neural network could affect results significantly [63], and some effective initialization methods have been proposed for specific neural network models [64], [65]. So, in this thesis, three different initialization methods are compared with each other in single morphological layer network. The first initialization method is all zero initialization. As its name indicates, this method initializes weights with all zeroes. The second initialization method is small random numbers initialization. All weights are initialized with random numbers generated by Gaussian distribution with 0 mean and 1 standard deviation.

The third initialization is proposed by the author based on the MSNN. For hit and miss structuring elements, understanding of the physical meaning of the MSNN structuring elements allows hit kernel to be initialized by randomly selecting a target sub-image, which is the same size as the hit kernel. Meanwhile, the miss kernel is initialized by randomly selecting a sub-image of the background, which is also the same size as the miss kernel.

3.2.4 Sub-image Generation

Input images are pre-processed by the sub-image generation algorithm before they are passed into the network. To compare results with Shen's network, we both used the same sub-image generation algorithm. With the sub-image generation algorithm, from each input training image, the M target sub-images with Label 1 and M background sub-images with Label 0 are generated. Input images are read into the algorithm with locations of target centers. So, target sub-images are selected randomly from the input image with centers located in a small window U at the target center. Background sub-images are selected randomly from the input image with centers located outside of window V, which includes the entire target. Figure 3.3 is an example of window U, with the window V as an input image. Centers of the target sub-images are selected from the orange area while centers of background sub-images are selected from blue area.

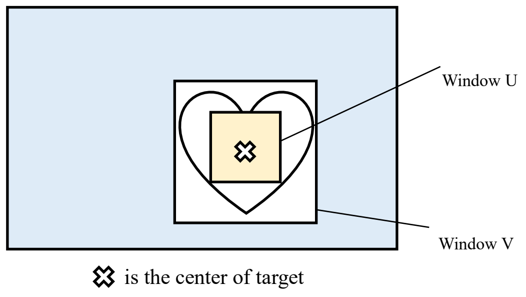


Figure 3.3 An example of window U, and V of an input image

The sub-image generation is described as follows: [22]

```
Sub-image generation algorithm

Data: Training image S, target center C, sub-image size winSize, M sub-images
Result: Target sub-images, background sub-images, labels
Define a small window U centered at the target center C;
Define a large window V which contains the entire target and is centered at the target
center C;
For i = 1 : M
    Randomly select a position Pt inside of the window U;
    Use position Pt as the sub-image center to cut a target sub-image Tn with the
    size winSize;
    Randomly select a position Pb outside of the window V;
    Use position Pb as the sub-image center to cut a background sub-image Bn with
    the size winSize;
End
Generate labels: target label = 1, background label = 0;
```

3.2.5 Morphological Operation

In the feature extraction stage, hit-miss transform is computed in the morphological layer. To calculate hit-miss transform, two morphological operations are used. One is the hit operation, which applies grayscale erosion on the input image. Another one is the miss operation, which applies grayscale dilation on the input image. The final calculation involves a combination of these two operations to generate the hit-miss operation.

The notations and equations of the morphological operation are defined as [21:]

✧ $a(x)$: The input to a node which is the output of node *x*

✧ $t_y^h(x)$: Hit structuring element weight associating node y with node *x*.

✧ $t_y^m(x)$: Miss structuring element weight associating node y with node *x*.

✧ net_y^h: Net input for Hit (erosion) operation.

✧ net_y^m: Net input for Miss (dilation) operation.

$$\text{Hit operation: } net_y^h = \min_{x \in D[t_y]} \{a(x) - t_y^h(x)\} \tag{3-1}$$

$$\text{Miss operation: } net_y^m = \max_{x \in D[t_y]} \{a(x) - t_y^m(x)\} \tag{3-2}$$

$$\text{Hit-miss transform: } a(y) = net_y = net_y^h - net_y^m \tag{3-3}$$

Take hit operation as an example: A structuring element slides onto the input image one pixel per step. At each position where the structuring element is located, the hit operation is computed, which is the subtraction between the local area of the image and the structuring element. This allows the minimum value to be selected.

To realize these operations in the algorithm and speed up the computation, Shen's second update mechanism is used [22]. With an input image and a hit structuring element, we slid the whole input image instead of the structuring element. This is because sliding the structuring element causes an overlap between steps. Almost all pixels in the input image are associated with each element in the structuring element, and they are also subtracted by it. So, all pixels associated with the same element in the structuring element are put together to form a new 2-D matrix. If the size of the structuring element is *[K, K]*, $K \times K$ 2-D matrixes are generated, and they are combined together as a 3-D matrix. Each 2-D matrix is subtracted by the associated element in SE one by one and at the same time, a 2-D location map is generated with a corresponding location of minimum value through all the 2-D matrices. If more than one location with minimum value appears, only the first location should be recorded. An example of hit operation is shown in Figure 3.4.

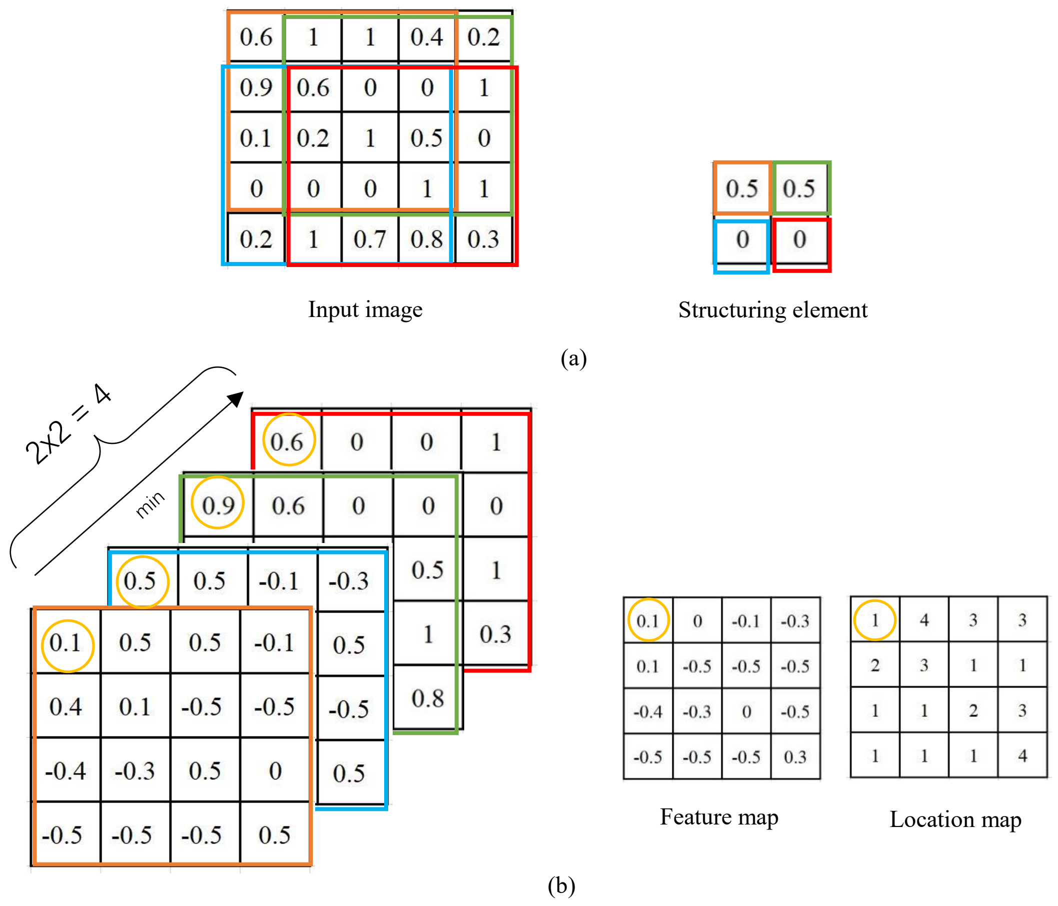


Figure 3.4 Hit operation. (a) A window slides one pixel per step on the input image to generate 2-D matrices with each matrix corresponding to the same color element in SE and then subtracts the element from the whole matrix, which has the same color as the element. (b) The minimum must be found through the 3rd dimension to get the minimum value for the feature map and to record the location.

The miss operation follows the same process as the hit operation. The only difference is selecting the maximum value from all the 2-D matrices instead of the minimum value from all the 2-D matrices.

3.2.6 Update Approaches

As previously mentioned, the three different training approaches for weights are backpropagation (BP), evolutionary algorithm with partial backpropagation (EAPB) and memetic algorithm (MA). Each has its own update rules for weights. In this section, details of the three different update approaches are introduced. All update rules are based on the MSNN architecture as shown in Figure 3.5.

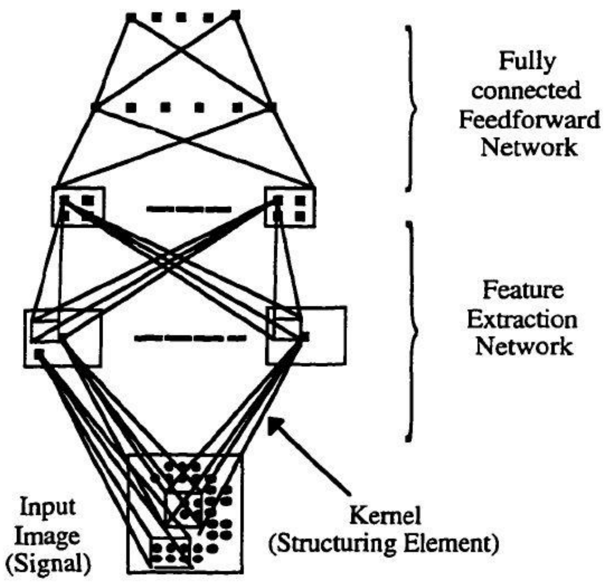

Fully connected Feedforward Network
Feature Extraction Network
Input Image (Signal)
Kernel (Structuring Element)

Figure 3.5 Architecture of MSNN with two feature extraction layers in the feature extraction stage and one hidden layer for the neural network

3.2.6.1 Backpropagation Approach

With BP, all structuring elements' weights and fully connected neural networks are updated by gradient descent. First, let's review the notations of the MSNN [21]:

- ✧ $a(x)$: The input to node a, which is the output of node x.

- ✧ $t_y^h(x)$: Hit structuring element weight associating node y with node *x*.

- ✧ $t_y^m(x)$: Miss structuring element weight associating node y with node *x*.

- ✧ net_y^h: Net input for Hit (erosion) operation.

- ✧ net_y^m: Net input for Miss (dilation) operation.

For the feature extraction stage, the update rules are expressed as:

$$\Delta t_h^y(x) = \eta \delta_y \frac{\partial net_y^h}{\partial t_y^h(x)} \tag{3-4}$$

$$\Delta t_h^m(x) = -\eta \delta_y \frac{\partial net_y^m}{\partial t_y^{m*}(x)} \tag{3-5}$$

$$\delta y = -\sum_k \left(\frac{\partial E}{\partial a_k} . \frac{\partial a_k}{\partial net_k} . \frac{\partial net_k}{\partial a(y)}\right) \tag{3-6}$$

where

$$\delta y = \delta(y) = \sum_k \delta_k w_k(y) \tag{3-7}$$

For the nodes in the top feature extraction layer

$$\delta y = \delta(y) = \sum_k \delta_k \left(\frac{\partial net_k^h}{\partial a(y)} - \frac{\partial net_k^m}{\partial a(y)}\right) \tag{3-8}$$

and for the nodes in other feature extraction layers.

Note that in equation 3-8, the last factor $\frac{\partial net_k}{\partial a(y)}$ will be equal to zero if $y \neq \arg\max_{q \in D[t_k]} \{a(q) - t_k^{m*}(q)\}$. When implementing this algorithm, this fact should be used to choose the index *k* over which to form the sum.

$$\frac{\partial net_y^h}{\partial t_y^h(x)} = \frac{\partial}{\partial t_y^h}\left[\min_{q\in D[t_y]}\{a(q) - t_y^h(q)\}\right]$$

$$= \begin{cases} -1 & if\ x = \arg\min_{q\in D[t_y]}\{a(q) - t_y^h(q)\} \\ 0 & otherwise \end{cases} \tag{3-9}$$

$$\frac{\partial net_y^m}{\partial t_y^{m*}(x)} = \frac{\partial}{\partial t_y^{m*}}\left[\min_{q\in D[t_y]}\{a(q) - t_y^{m*}(q)\}\right]$$

$$= \begin{cases} -1 & if\ x = arg\ \underset{q\in D[t_y]}{max}\{a(q) - t_y^{m*}(q)\} \\ 0 & otherwise \end{cases} \tag{3-10}$$

and

$$\frac{\partial net_k^h}{\partial a(y)} = \frac{\partial}{\partial a(y)}\left[\min_{q\in D[t_k]}\{a(q) - t_k^h(q)\}\right]$$

$$= \begin{cases} 1 & if\ y = \arg\min_{q\in D[t_k]}\{a(q) - t_k^h(q)\} \\ 0 & otherwise \end{cases} \tag{3-11}$$

$$\frac{\partial net_k^m}{\partial a(y)} = \frac{\partial}{\partial a(y)}\left[\min_{q\in D[t_k]}\{a(q) - t_k^{m*}(q)\}\right]$$

$$= \begin{cases} 1 & if\ y = \arg\max_{q\in D[t_k]}\{a(q) - t_k^{m*}(q)\} \\ 0 & otherwise \end{cases} \tag{3-12}$$

The neuron network consists of an input layer, a hidden layer and an output layer with its notations designated as

$E_j(k)$: The instantaneous error energy of neuron *j*.

w_{ji}: The synaptic weight between neuron *j* and neuron *i*.

$v_j(k)$: The input of the activation function associated with neuron *j*.

$y_j(k)$: The output of neuron *j*.

$e_j(k)$: The error signal produced at the output of neuron *j*.

$\phi_j(k)$: The activation function of neuron *j*.

Loss function, which is used to calculate the error signal, is the square error loss function in BP.

$$E_j(k) = \frac{1}{2} e_j^2(k) \tag{3-13}$$

So, the total instantaneous error energy of the whole network is

$$E(k) = \sum_j E_j(k) = \frac{1}{2} \sum_j e_j^2(k) \tag{3-14}$$

With the chain rule, the correction $\Delta w_{ji}(k)$ applied to the synaptic weight connecting neuron *i* to neuron *j* is given by

$$\Delta w_{ji}(k) = \begin{cases} \alpha e_j(k) \phi_j'\left(v_j(k)\right) y_i(k) & if\ neuron\ j\ is\ an\ output\ node \\ \alpha \phi_j'\left(v_j(k)\right) \sum_l \left(e_l \phi_j\left(v_l(k)\right) w_{lj}(k)\right) y_i(k) & if\ neuron\ j\ is\ a\ hidden\ node \end{cases} \tag{3-15}$$

For the activation function of the hidden layer and output layer in the backpropagation approach with single morphological layer, the sigmoid function is used on both the hidden layer and the output layer. With two morphological layers, a rectified linear unit (ReLU) function is used on the hidden layer to avoid gradient vanishing, and the sigmoid function is used on the output layer.

Sigmoid function: $$\phi_j\left(v_j(k)\right) = \frac{1}{1+\exp\left(-av_j(k)\right)} \tag{3-16}$$

RuLU function: $$\phi_j\left(v_j(k)\right) = \max\left(0, v_j(k)\right) \tag{3-17}$$

3.2.6.2 Evolutionary Algorithm with Partial Backpropagation

With EAPB, the weights of the neural network are still updated with backpropagation, so update rules are same as the neural network part of BP, which includes equation 3-13 to 3-15. However, the weights of the structuring elements are updated by the evolutionary algorithm. In this section, we will focus on the update rules of SE.

Take a single morphological layer network as an example. Suppose we have a pair of structuring elements. One is the hit structuring element $[h_{11},\ h_{12} \cdots h_{1N};\ h_{21},\ h_{22} \cdots h_{2N};\ \cdots;\ h_{N1},\ h_{N2} \cdots h_{NN}]$ with size [*N, N*], and another one is the miss structuring element $[m_{11},\ m_{12} \cdots m_{1N};\ m_{21},\ m_{22} \cdots m_{2N};\ \cdots;\ m_{N1},\ m_{N2} \cdots m_{NN}]$ with size [*N, N*]. Then, we put all $2 \times N^2$ weights into a vector considered as the EA chromosome $[\ h_{11},\ h_{12} \cdots h_{1N},\ h_{21},\ h_{22} \cdots h_{2N},\ \cdots,\ h_{N1},\ h_{N2} \cdots h_{NN},\ m_{11},\ m_{12} \cdots m_{1N},\ m_{21},\ m_{22} \cdots m_{2N},\ \cdots,\ m_{N1},\ m_{N2} \cdots m_{NN}\ ]$. This is the representation of our solution, and our goal is to find the best chromosome which can minimize the loss function. After that, we initialize *K* populations of chromosome with a selected initialization method and pass them on as initial parents into the next generations.

For each generation of EA, offspring of parents are generated with crossover and mutation operators. A two-point crossover is used in EAPB. For example, if we have two parents which are $[p_1^1,\ p_2^1,\ p_3^1,\ p_4^1,\ p_5^1]$ and $[p_1^2,\ p_2^2,\ p_3^2,\ p_4^2,\ p_5^2]$, then two random indexes between 1 and 5 are computed. Suppose the two random indexes are 2 and 4. With the two-points crossover, the offspring of these two parents should be $[p_1^1,\ p_2^2,\ p_3^2,\ p_4^2,\ p_5^1]$ and $[p_1^2,\ p_2^1,\ p_3^1,\ p_4^1,\ p_5^2]$.

For mutation, there are three different versions. The first one adds a small random number between $[-1, 1]$ to each weight of the chromosome. The second one adds a random Gaussian distribution number with 0 mean and 1 standard deviation to each weight of the chromosome. Finally, the last one adds *m*loss*/160, where *m* is a random Gaussian distribution number with 0 mean and one standard deviation and the *loss* is the sum of total errors calculated by the loss function.

In the classification stage, the update rules of a multiple-layer neural network with one input layer, one hidden layer, and one output layer are same as the rules in the BP approach. The only difference is the loss function. The cross-entropy function is added to compare with the square error function, which is

$$E_j(k) = -[\, d_j(k)\ln y_j(k) + \left(1 - d_j(k)\right)\ln\left(1 - y_j(k)\right)\,] \tag{3-18}$$

where $d_j(k)$ is the desired output of neuron j, and $y_j(k)$ is the actual output of neuron j.

So, with the chain rule, the correction $\Delta w_{ji}(k)$ of the cross-entropy function applied to the synaptic weight connecting neuron i to neuron j is changed to

$$\Delta w_{ji}(k) = \begin{cases} \alpha e_j(k) y_i(k) & if\ neuron\ j\ is\ an\ output\ node \\ \alpha \phi_j'\left(v_j(k)\right)\sum_l \left(e_l w_{lj}(k)\right) y_i(k) & if\ neuron\ j\ is\ a\ hidden\ node \end{cases} \tag{3-19}$$

3.2.6.3 Memetic Algorithm

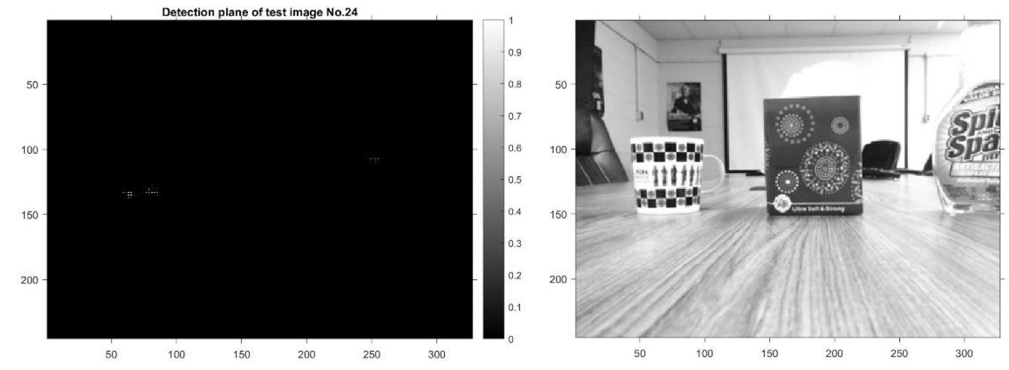
In a memetic algorithm, all weights of the structuring elements and neural network are put together into the chromosome. The update process is much like the update process of the feature extraction stage in EAPB. First, initialize the *K* populations and pass them into the algorithm. Then, in each generation, generate *K* offspring with crossover, mutation and a local search. Here, the local search is the main technique used in the memetic algorithm to speed up convergence of the algorithm. After doing crossover and mutation, offspring will do several generations of backpropagation to slightly update all weights in the offspring. Then these offspring and parents compare with each other and select the best *K* populations as new parents. A local search can be considered an operation of trying to find local minima around each offspring and to then replace these offspring with the local minima to compare them with each other and thereby find the minimum which is closest to the global minimum state or is exactly the global minimum state. There is no backpropagation in this approach.

3.2.7 Sub-image Replacement Method

If there are N training images, for each training image, *M* target sub-images and *M* background sub-images are selected to pass into the algorithm. The number of sub-images passed into the algorithm is $2 \times M \times N$. In this experiment, $M = 40$, $N = 8$, and the size of input training image is [245, 327], which means 640 sub-images were selected to pass into the algorithm. But for a training image, there are 54,963 sub-images in total. So, it is not enough to train the network with just 640 sub-images. This is the reason for a sub-image replacement operation in an algorithm. When the error of a sub-image is smaller than the well-trained value, which is set up manually, this sub-image will be replaced by a new sub-image. The new sub-image has the same label as the replaced one. The sub-image replacement method used in this thesis is randomly selecting a new sub-image, which is similar to the sub-image generation process.

3.2.8 Target Boxes

The output of MSNN is a detection plane, which is a continuous confidence map. If the confidence is close to 1, it means the sub-image of the corresponding location has a high confidence to qualify as the target. If the confidence is close to 0, it means the sub-image of the corresponding location has a low confidence to be the target, which indicates that this sub-image has a high confidence, making it a good choice for the background. Figure 3.6 represents a test image detection plane. The target is a white and black cup with a terra-cotta soldiers' pattern.

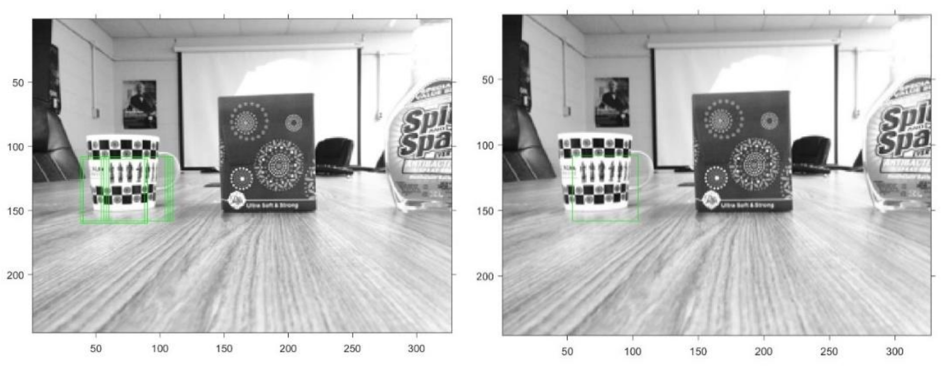


(a) Detection (b) Test image

Figure 3.6 Detection plane of a test image and the target is a white and black cup with a terra-cotta soldiers' pattern.

In Figure 3.6, a few bright points in the detection plane are reported based on the targets in these corresponding sub-images. However, these reported targets come from the same actual target. When we draw detection boxes, which contain targets based on these points, too many repeated boxes will appear on the same target. Thus, a method, called non maximal suppression, to eliminate these repeated boxes is introduced.

The first step of this method is to find a maximum on the detection plane, record it, and set its neighbor area to zeros. Then, find another maximum, repeat the first step until all points with values are recorded points, and the other points are zero. After this operation, target boxes are eliminated, and important boxes are maintained. An example of target boxes before and after elimination is shown in Figure 3.7.



(a) Boxes of target before elimination and (b) boxes of target after elimination

Figure 3.7 An example of boxes of target before elimination and after elimination.

CHAPTER 4 RESULTS

4.1 Dataset

In this thesis, three groups of datasets have a white and black cup with a terra-cotta soldiers' pattern as target. One group of datasets was created by Shen [22], which was used to compare our network results with those of Shen. Another two groups of datasets were created by the author under the advisement of Dr. James Keller at the University of Missouri-Columbia. We selected a cup as our target and took a lot of images in different instances and with diverse backgrounds. Afterwards, these images were resized and converted into grayscale. Then, we set up the centers of these images manually which were used to locate targets. The size of the images is [245, 327].

For Dataset I, all images were placed at the same distance from the camera. The target was a white and black cup with a terra-cotta soldiers' pattern. Figure 4.1 shows some samples from Dataset I.

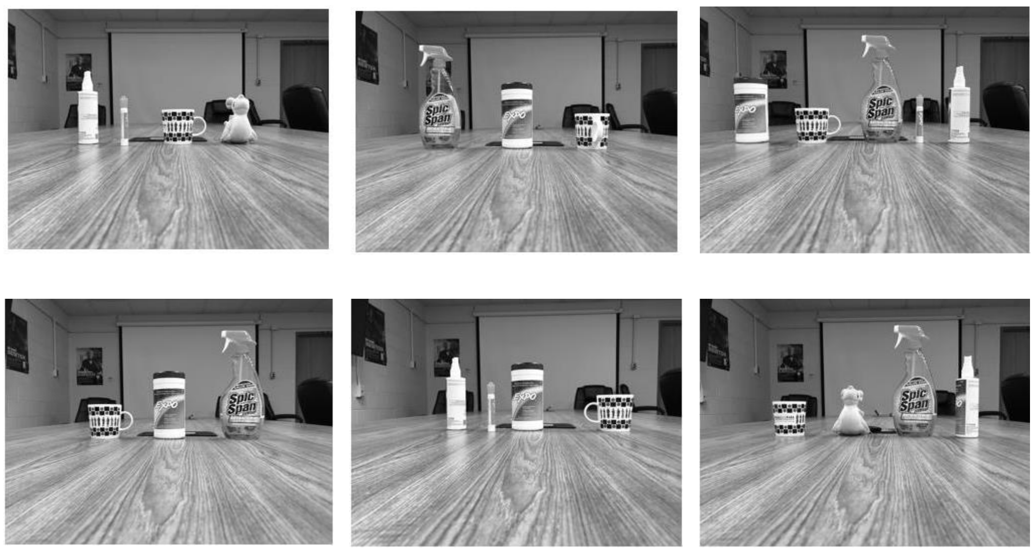


Figure 4.1 Samples from Dataset I

This dataset has 10 groups with eight images per group. The background of each group is different. The background of each image in one group is the same, but the target has eight different directions in total. This dataset is the “simplest” group for the algorithm.

Dataset II still has 10 groups with eight images per group. Figure 4.2 shows some samples from this dataset.

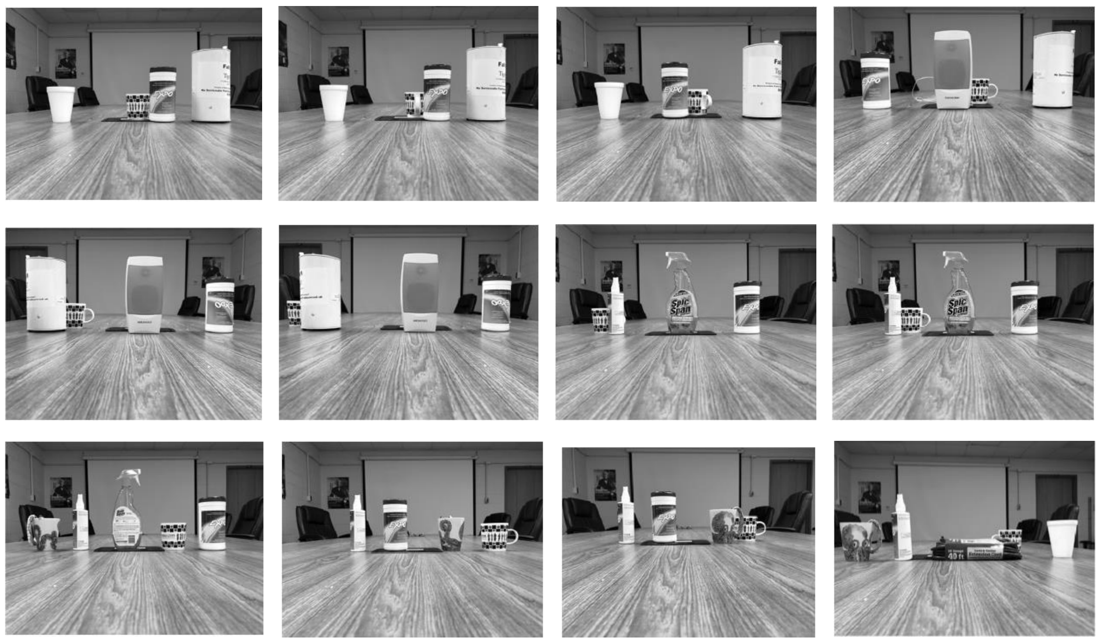


Figure 4.2 Samples from Dataset II

All images in this dataset have the same distance as in Dataset I, and all have the same target as Dataset I. The difference between them is that in Dataset II, most images have occlusion situations. As shown in Figure 4.2, the target is occluded by other background items in Groups 1 to 4 and in Groups 7 to 10. In Groups 5 to 10, a new cup is added to the image and there is no occlusion in Groups 5 and 6. These two groups are used to test whether the network can identify “the cup” or just cup shapes.

Dataset III is the dataset which Shen created. Figure 4.3 shows some samples from that dataset. Dataset III has 16 groups of images with eight images in each group. The target is still the black and white cup.

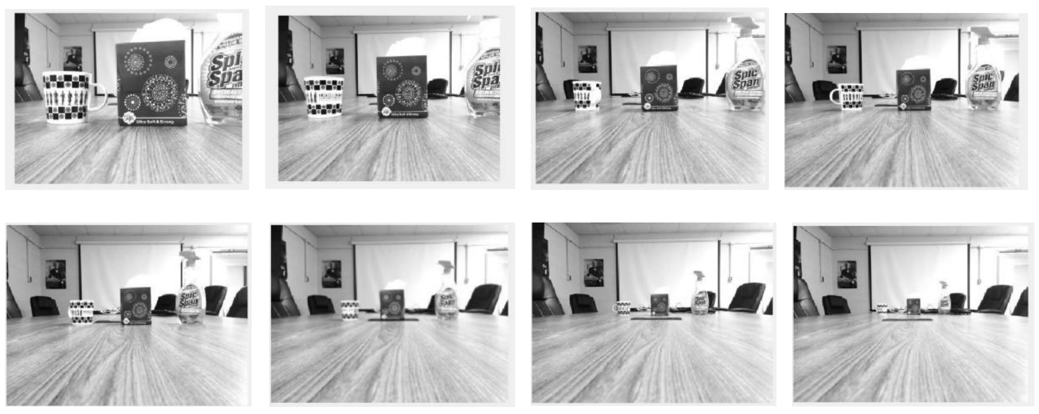


Figure 4.3 Samples from Dataset III

In Dataset III, the three objects in each image are the cup, a tissue box and a sprayer. None of the items in these images are occluded by any other items. These 16 groups of images have different distances from the camera—both far and near. In each group, the cup has eight different directions.

All images are split into one training set and three testing sets. Group 9 of Dataset III is selected as the training set. Dataset I, Dataset II and the rest images of Dataset III are considered as testing set I, testing set II and testing set III separately.

4.2 Results of Single Morphological Layer Networks

The experiments of a single morphological layer network are focused on comparisons between three different update approaches, which are backpropagation (BP), the evolutionary algorithm with partial backpropagation (EAPB) and the memetic algorithm (MA). First, with EAPB, which is the main algorithm discussed, three experiments were conducted for comparisons. The first one compares performances of updating networks with the square error loss function and the cross-entropy loss function. The second one compares the performances of the different mutation methods mentioned in Section 3.2.6.2. The third one compares performances of the different initialization methods mentioned in Section 3.2.3. Then, the results of the three different update approaches are compared. After one morphological layer with one pair of kernels trained by three update approaches are compared with each other, one more pair of kernels is added into the morphological layer. Thus, a parallel morphological shared-weight neural network is trained to see if the results are better than the results of the networks with one pair of kernels.

4.2.1 Comparison of Loss Function

Single morphological layer networks are trained separately by the EAPB approach with two different loss functions. The first loss function is the square error loss function, and the second one is the cross-entropy loss function. The parameters of two networks are shown in Table 4.1.

Table 4.1 Parameters of a single layer morphological network

Iteration	**200**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
Structuring element size	5×5
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
β (momentum constant)	0.1
Well trained	10^{-5}
Learning rate	0.3
Size of population	20
Probability of mutation	0.8
Probability of crossover	0.8

Dataset III is used for this comparison. Group 9 of Dataset III is used as training data, and the remaining images of Dataset III are used as test data. For both loss functions, we trained each network five times and the only difference between two networks is different loss functions. These two networks use the same initialization method, crossover and mutation method. The reason why we trained each network five times is operations include initialization, crossover and mutation could make results of the same network different each time because of their randomness. So we ran each network five times and averaged or concatenated results to make results more reliable. The accuracy of the network is evaluated by the ROC curve. The y label of the ROC curve is the true-positive (TP) rate, which is the number of correct points detected as targets by the algorithm divide the number of total targets. The x label of the ROC curve is the false-positive (FP) rate, which is the false alarm per image and the false alarm is the number of wrong points detected as targets. Colors on the color bar correspond to thresholds from 0 to 1. These thresholds are used to distinguish targets and backgrounds. For example, we have a detection plane of an input image, and 0.9 is selected as the threshold. So, the corresponding sub-image of the point which has a confidence (output) higher than 0.9 on the detection plane is labeled as 1. The corresponding TP rate and FP rate are calculated based on this threshold and all information is related to one point on the ROC curve with the corresponding color of this threshold.

Figure 4.4 is an example output image. There are two green boxes, which are the bounding boxes where the algorithm indicates a target exists. These boxes are drawn with centers, which are points with TP rates higher than the threshold, after non maximal suppression. The red point in the output image is the point which has the highest confidence as a target. So, the center of the left box has the correct point detected as a target by the algorithm, and the center of the right box is a false alarm. If the purpose of those working with this experiment is to find targets with no false alarms, then the preferred algorithms have ROC curves possessing a higher TP rate when the false alarm per image is 0. However, if we want to find targets, the more the better. The target-seeking algorithms can tolerate some false alarms per image, but the less false alarms, the better. Thus, the preferred algorithms are those with ROC curves possessing a higher TP rate when false alarms per image are in a desired acceptable range.

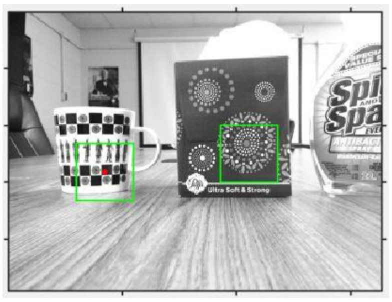
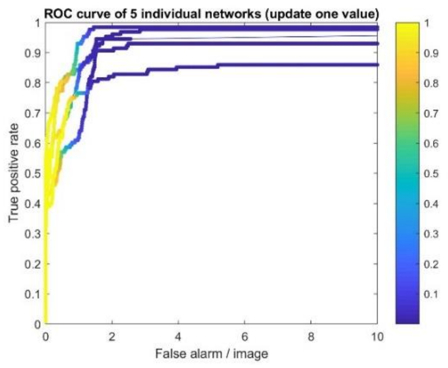
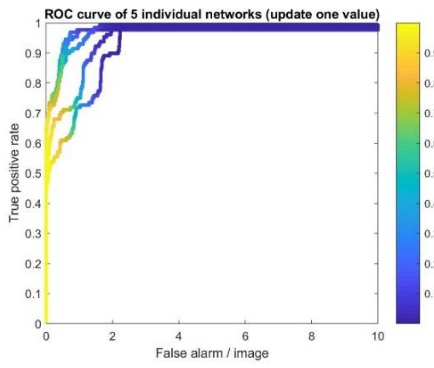


Figure 4.4 An example of output image with one true positive and one false alarm.

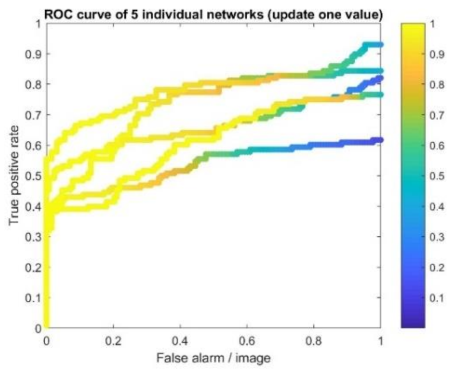
Figure 4.5 shows all ROC curves of the five networks for both functions. The left column has the results of the cross-entropy loss function, and the right column has the results of the square error loss function. Figure 4.6 shows the vertical average ROC curve and the concatenated ROC curve of five individual networks for both functions. Averages of the TP rates are calculated in vertical-average ROC curves to observe the fluctuation of the network. Moreover, the concatenated ROC curves are computed by merging all test data together and considering them as one entire test dataset.



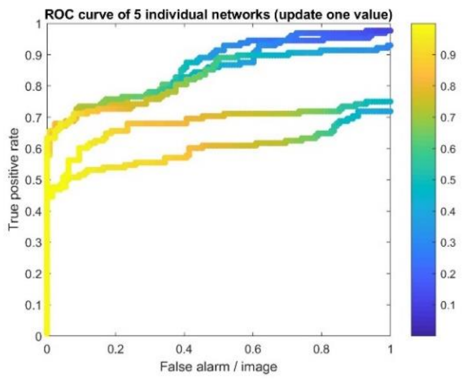
(a) ROC curves of cross-entropy function



(b) ROC curves of square error function

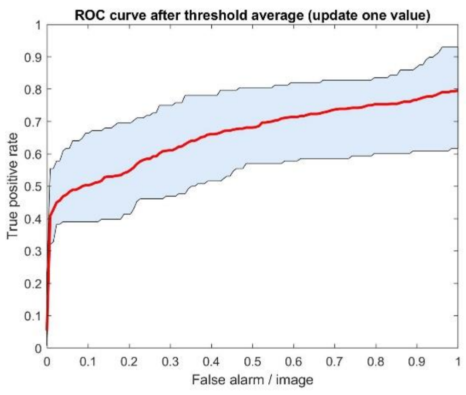


(c) Zoomed in ROC curves of cross-entropy function

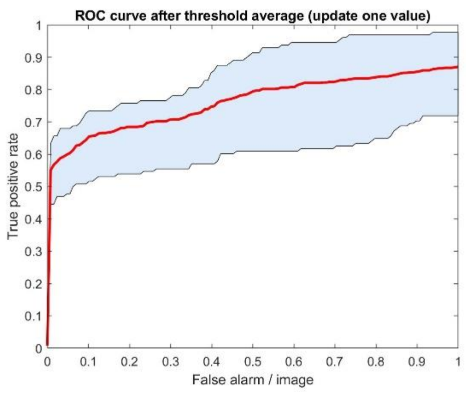


(d) Zoomed in ROC curves of square error function

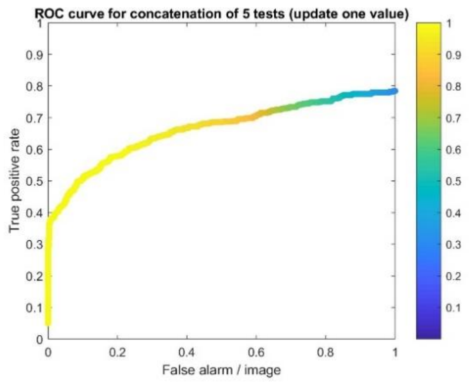
Figure 4.5 ROC curves of five networks for two different loss functions.



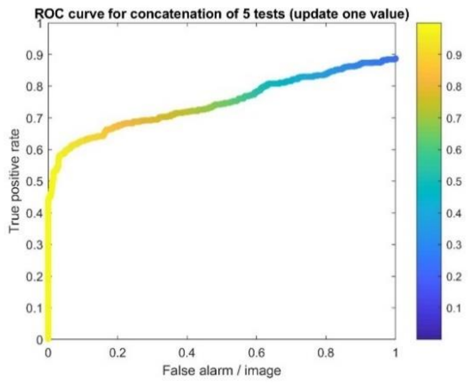
(a) Vertical average ROC curve of cross-entropy function



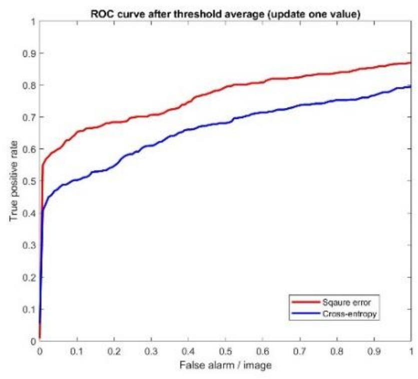
(b) Vertical average ROC curve of square error function



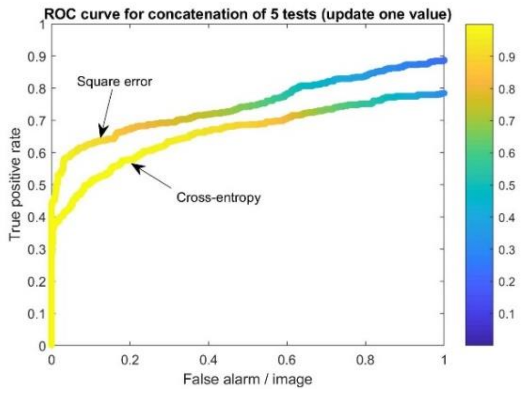
(c) Concatenated ROC curve of cross-entropy function



(d) Concatenated ROC curve of square error function



(e) Vertical average ROC curves of two loss functions



(f) Concatenated ROC curves of two loss functions

Figure 4.6 Vertical average and concatenated ROC curves for two different loss functions.

From the figures, it is obvious that using square error loss function gives a better performance. Figure 4.6 (e) and (f), show that with the same FP rate, the TP rate of the square error function is always higher than the TP rate of the cross-entropy function. The reason could be that the cross-entropy function makes the algorithm converge too fast, and the current weights of neural network could be stuck with the current best structuring elements. So the chance of finding other better structuring elements may be lower. With these results, the square error function is clearly better for the EAPB approach, which is why it was chosen. Its use will be shown later. Samples of detection results are shown in Appendix Figure A-1 and Appendix Figure A-2.

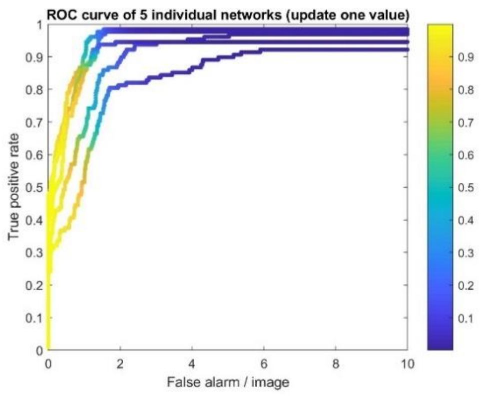
4.2.2 Comparison of Mutation Methods

Three different mutation methods are compared with each other. For networks to be trained, we still use one single morphological layer network trained separately by EAPB with different mutation methods. The first mutation method adds a small random number between $[-1, 1]$ to each weight of the chromosome. The second one adds a random Gaussian distribution number with 0 mean and 1 standard deviation to each weight of the chromosome, and the third one adds m*loss/160, where m is a random Gaussian distribution number with 0 mean and 1 standard deviation. Network parameters are listed as follows:

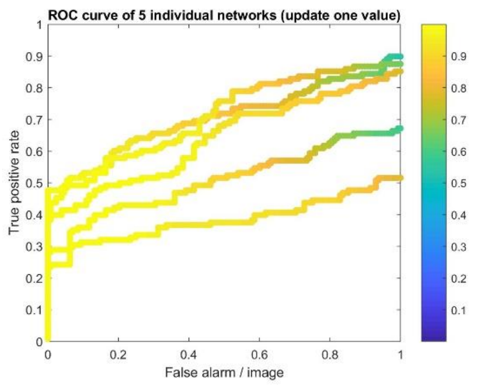
Table 4.2 Parameters of the single-layer morphological network

Iteration	**200**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
Structuring element size	5×5
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
β (momentum constant)	0.1
Well trained	10^{-5}
Size of population	20
Probability of mutation	0.8
Probability of cross over	0.8

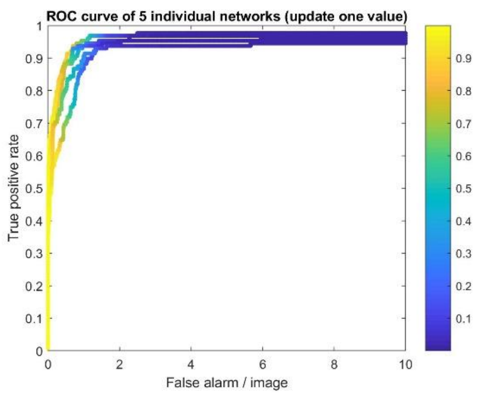
The dataset used in these comparisons is still Dataset III. Group 9 of Dataset III is used as training data, which includes eight images. And the rest images of Dataset III are used as testing data.



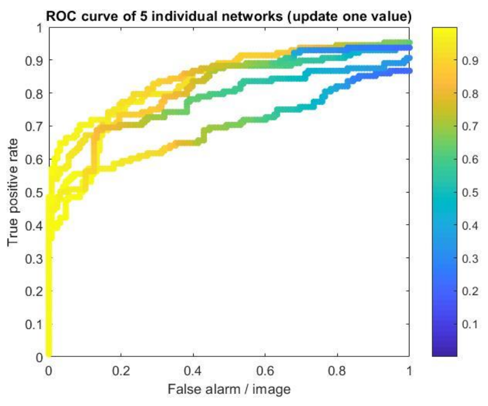
(a) Original ROC curves of first mutation method



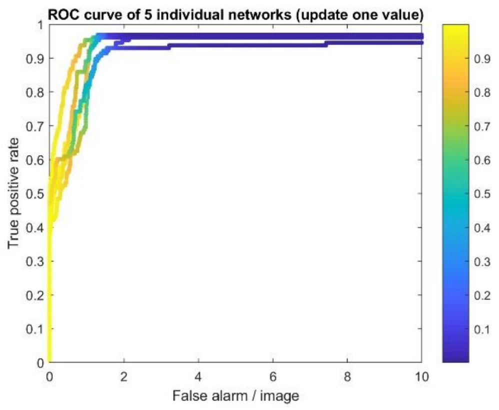
(b) Zoomed in ROC curves of first mutation method



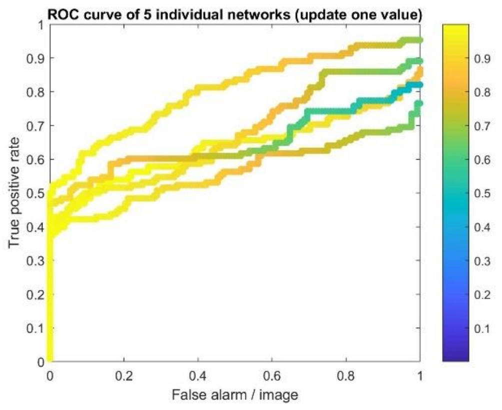
(c) Original ROC curves of second mutation method



(d) Zoomed in ROC curves of second mutation method



(e) Original ROC curves of third mutation method



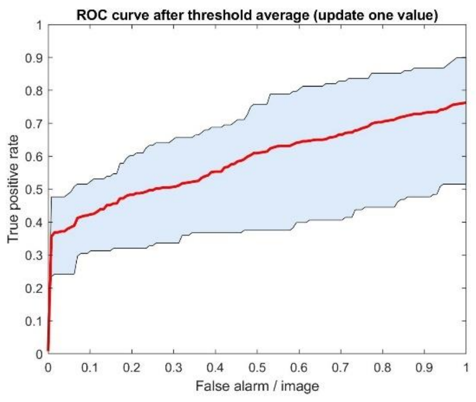
(f) Zoomed in ROC curves of third mutation method

Figure 4.7 ROC curves of three different mutation methods.

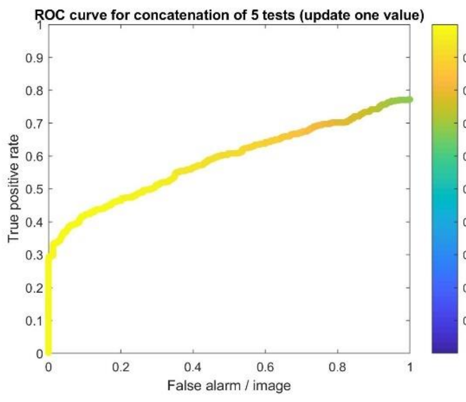
Figure 4.7 shows the results of three different mutation methods. Each row of figures presents the original and zoomed in ROC curves for corresponding mutation methods.

In Figure 4.8, the vertical average ROC curves and concatenated ROC curves of three different mutation methods are shown.

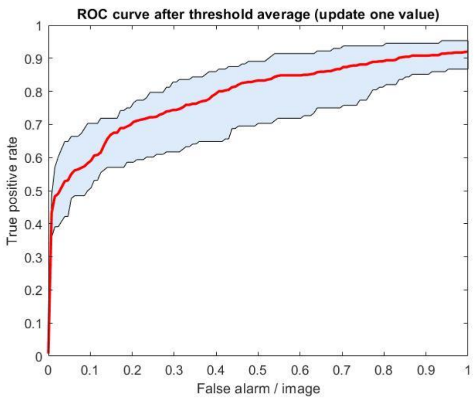
In Figure 4.8, we can see that with the second mutation method, which adds a random Gaussian distribution number with 0 mean and 1 standard deviation to each weight of the chromosome, results are more stable than those found with the other two mutation methods, and they have a higher FP rate than the others. To observe results more clearly, the vertical average ROC curves and the concatenated ROC curves of three mutation methods are put into Figure 4.9 for comparison.



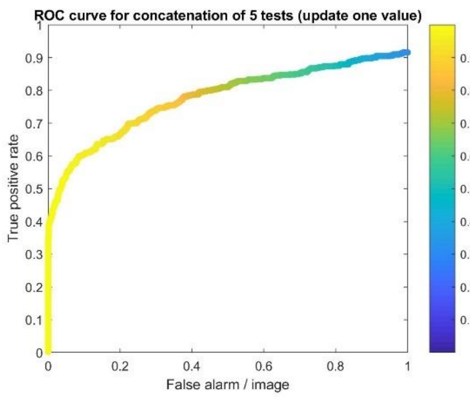
(a) Vertical average ROC curve of first mutation method



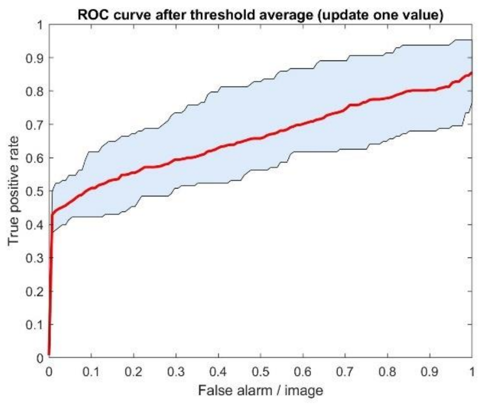
(b) Concatenated ROC curve of first mutation method



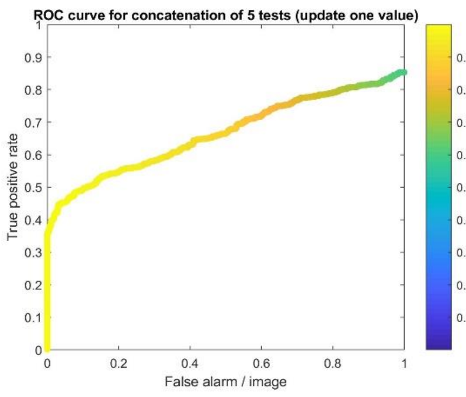
(c) Vertical average ROC curve of second mutation method



(d) Concatenated ROC curve of second mutation method

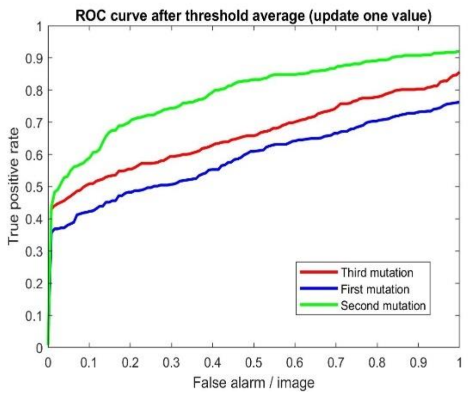


(e) Vertical average ROC curve of third mutation method

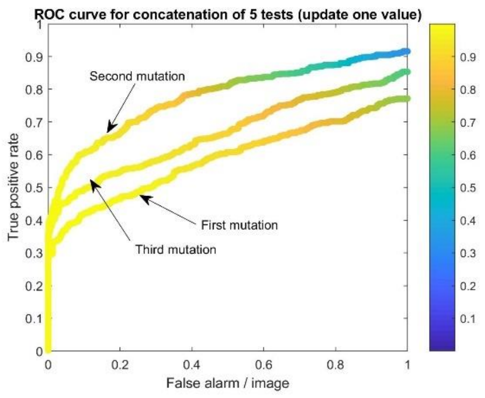


(f) Concatenated ROC curve of third mutation method

Figure 4.8 Vertical average and concatenated ROC curves of three different mutation methods.



(a) Vertical average ROC curves of three mutation methods



(b) Concatenated ROC curves of three mutation methods

Figure 4.9 Comparisons between three mutation methods.

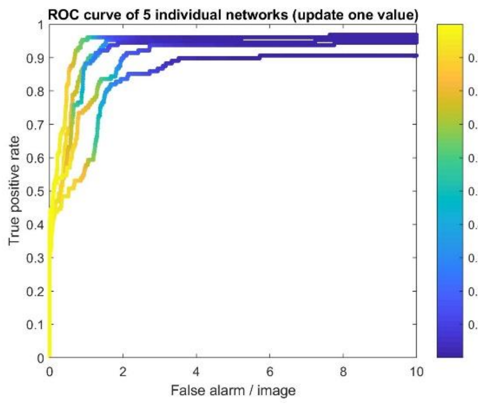
From the previous figures, it is obvious that the second mutation which adds a random Gaussian distribution number with 0 mean and 1 standard deviation to each weight of the chromosome has a better performance than the other two mutation methods. With the same FP rate, the ROC curve of the second mutation has a higher TP rate value, which is what we want. So, the second mutation method is selected as the mutation method for the following experiment. Examples of detections results of three mutation methods are shown in Appendix Figure A-3 and Appendix Figure A-4.

4.2.3 Comparison of Initialization methods

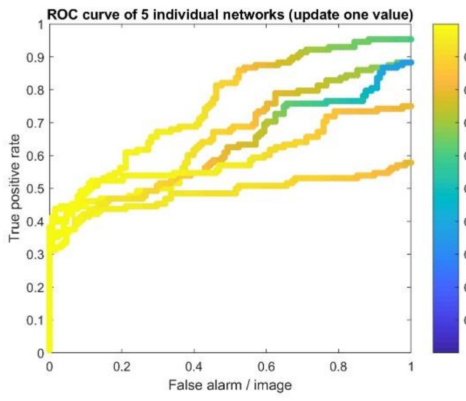
As mentioned in Section 3.2.3, three initialization methods are compared with the EAPB update approach. The first initialization method is an all-zero initialization. The second initialization method is a small random numbers initialization. The third one is proposed by the author based on the MSNN. For the hit-miss structuring elements, the hit kernel is initialized by randomly selecting a sub-image of the target, which has the same size as the hit kernel, while the miss kernel is initialized by randomly selecting a sub-image of the background, which also has the same size as the miss kernel. Dataset III is still used in these comparisons with group 9 as the training data and the remaining images as testing data. Parameters used in these networks are the same as parameters in Section 4.2.2.

Original and zoomed in ROC curves are shown in Figure 4.10. Each row of figures shows ROC curves of the first initialization method, the second initialization method, and the third initialization method in sequence.

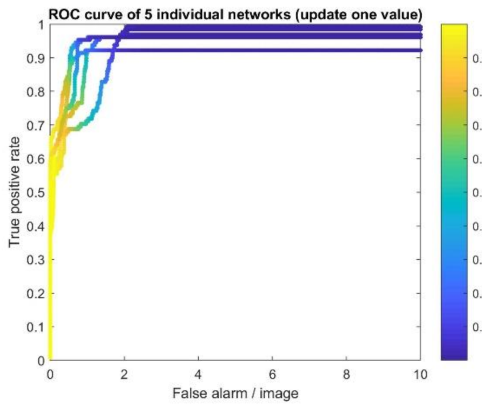
From Figure 4.10, we can see that the performances of the second and third initialization methods are better than the first initialization methods. We cannot figure out which one is better among the second and third initialization methods from these figures, so the vertical average and concatenated ROC curves are shown in Figure 4.11. But from Figure 4.11, there is still no significant distinguishing difference between them. Then, the vertical average and concatenated ROC curves from the three initialization methods are put into one figure separately for observing clearly in Figure 4.12.



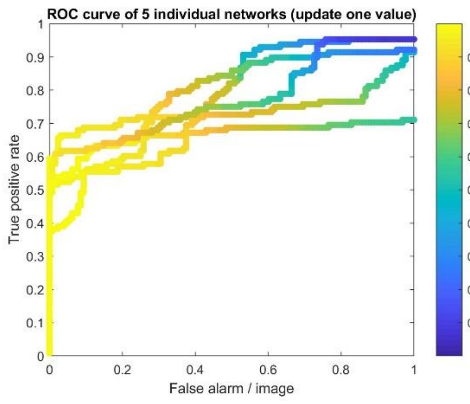
(a) Original ROC curves of
first initialization method



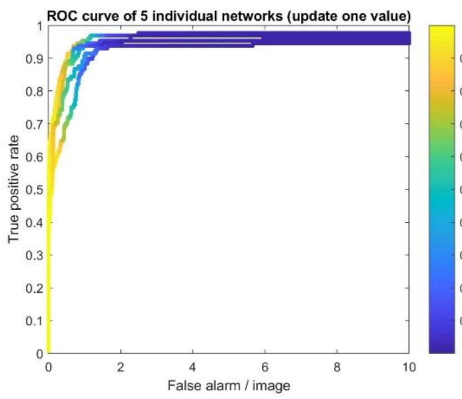
(b) Zoomed in ROC curves of
first initialization method



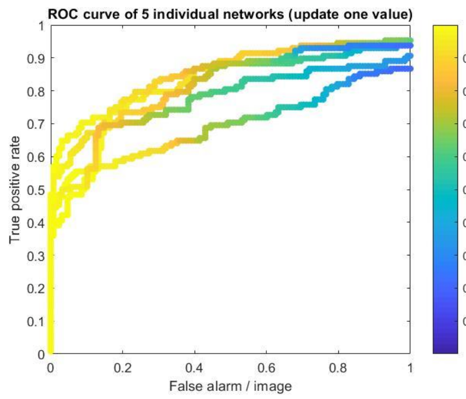
(c) Original ROC curves of
second initialization method



(d) Zoomed in ROC curves of
second initialization method

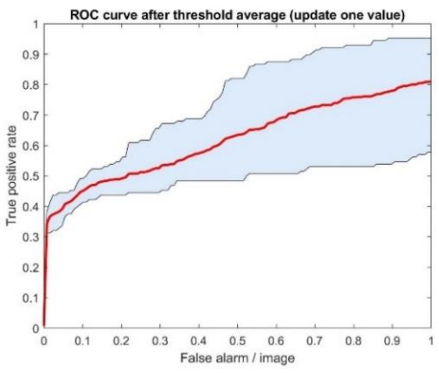


(e) Original ROC curves of third
initialization method

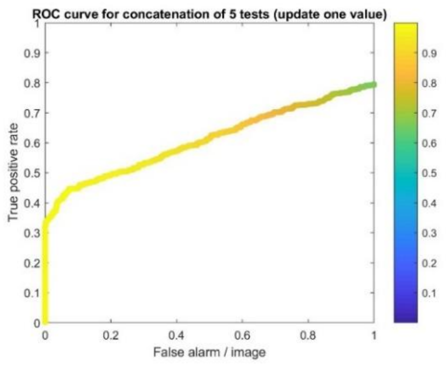


(f) Zoomed in ROC curves of third
initialization method

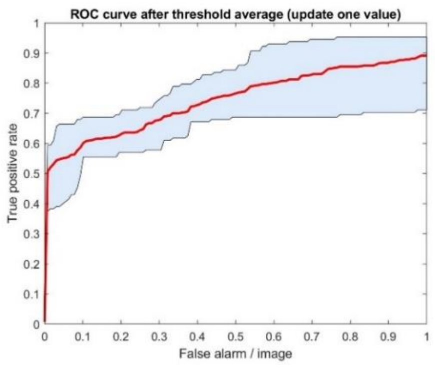
Figure 4.10 ROC curves of three different initialization methods.



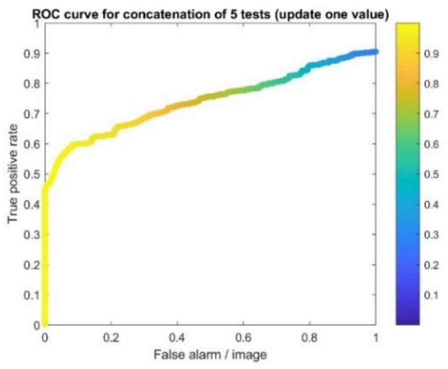
(a) Vertical average ROC curve of first initialization method



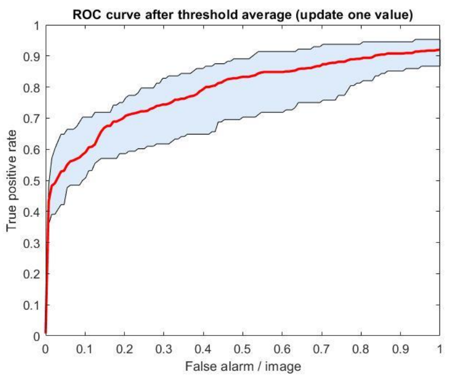
(b) Concatenated ROC curve of first initialization method



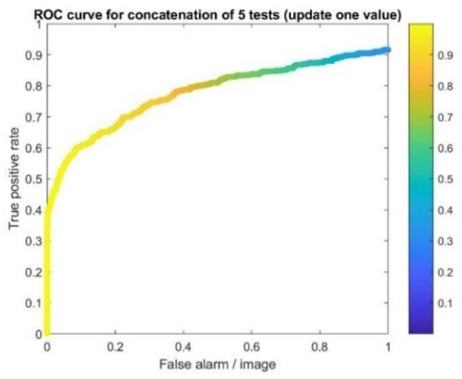
(c) Vertical average ROC curve of second initialization method



(d) Concatenated ROC curve of second initialization method

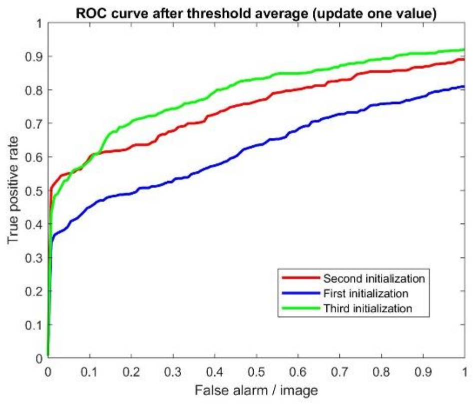


(e) Vertical average ROC curve of third initialization method

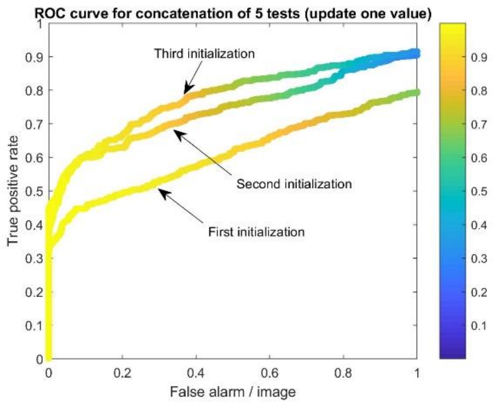


(f) Concatenated ROC curve of third initialization method

Figure 4.11 Vertical average and concatenated ROC curves of three different initialization methods.



(a) Vertical average ROC curves of three initialization methods



(b) Concatenated ROC curves of three initialization methods

Figure 4.12 Comparisons between three initialization methods.

From Figure 4.12, the vertical average ROC curve of the third initialization method is slightly higher than curve of second initialization method and the concatenated ROC curve of the third initialization method is also higher than the curve of the second initialization method between [0.2, 0.8] of the FP rate. The reason for the different results with different initialization methods is that the initial chromosomes start at various locations within the solution space, so the closer the initial chromosomes are to the global minimum, the easier it is for the algorithm to find the best solution. So, in this case, the third initialization method was selected as the initialization method used in all experiments for EAPB. Examples of detections results of three initialization methods are shown in Appendix Figure A-5 and Appendix Figure A-6.

4.2.4 Comparison of Three Update Approaches

In this section, three different update approaches are compared with each other. They are backpropagation (BP), evolutionary algorithm with partial backpropagation (EAPB) and the memetic algorithm (MA). The network used is MSNN with a single morphological layer. For the dataset, group 9 Dataset III is selected as the training data. Dataset I, Dataset II and the remaining images of Dataset III are used as three groups of test data.

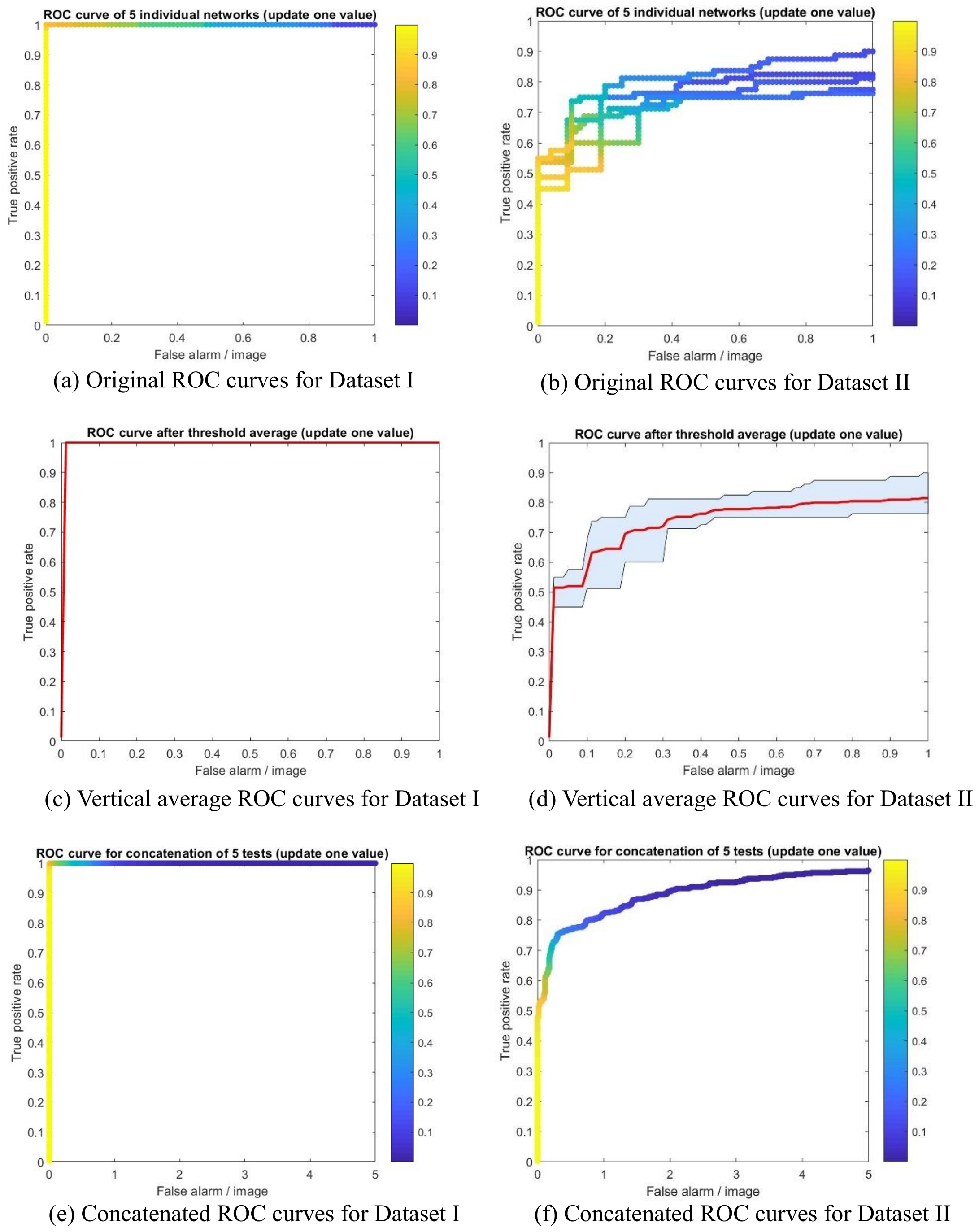
4.2.4.1 Backpropagation (BP)

First, a single morphological layer network is trained by BP five times to assure more reliable results. Weights of structuring elements are initialized as zero and weights of the neural network are initialized with random small numbers between [−1, 1], which are the same as those found in Shen's networks, so we can easily compare our results with her results. Parameters for this network are listed in Table 4.3.

Table 4.3 Parameters of the single morphological layer network with BP

Iteration	**1000**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
Structuring element size	5×5
Hidden neurons in neural network	20
a_1 (parameter in sigmoid function in hidden layer)	0.3
a_2 (parameter in sigmoid function in output layer)	0.4
β (momentum constant)	0.2
MaxRandSelect	5
Well trained	10^{-5}

Group 9 of Dataset III is selected as training data. After training, these networks are tested on three groups of test data, which are Dataset I, Dataset II and Dataset III without group 9. Figure 4.13 shows the test results of Dataset I and Dataset II.



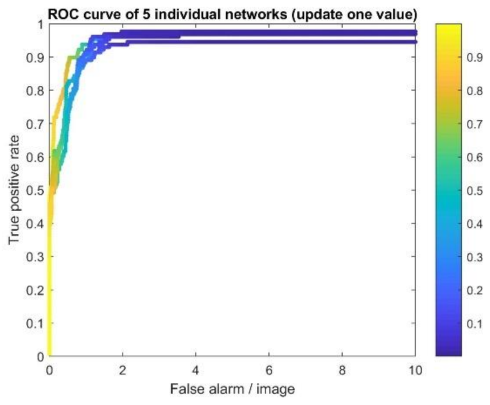
(a) Original ROC curves for Dataset I (b) Original ROC curves for Dataset II

(c) Vertical average ROC curves for Dataset I (d) Vertical average ROC curves for Dataset II

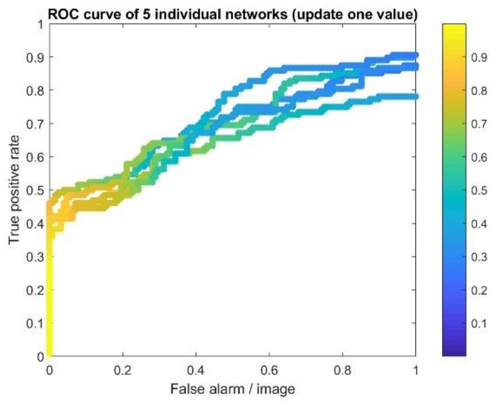
(e) Concatenated ROC curves for Dataset I (f) Concatenated ROC curves for Dataset II

Figure 4.13 Results of Dataset I and Dataset II with BP.

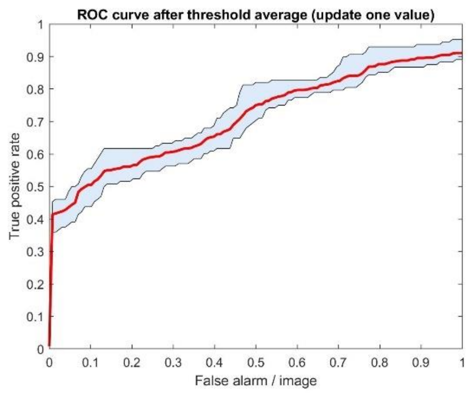
From Figure 4.13, it is obvious that test results of Dataset I are exceptional, i.e., 100 percent accuracy. This means MSNN trained by BP has excellent performance on test images which have the same distance as training images. For Dataset II results, which is the dataset with occlusion situations, performance results cannot be concluded now without comparisons. These results will be compared with two other update approaches later. Dataset III results are shown in Figure 4.14.



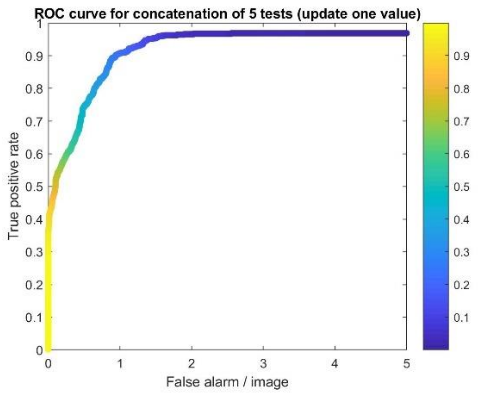
(a) Original ROC curves



(b) Zoomed in ROC curves



(c) Vertical average ROC curve



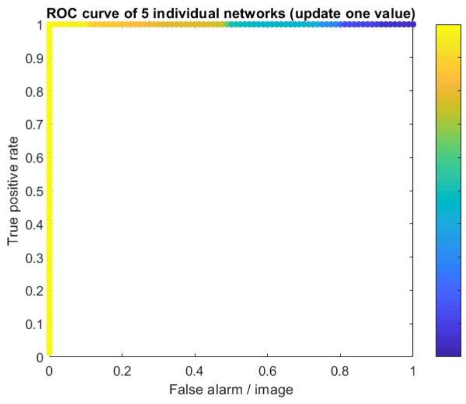
(d) Concatenated ROC curve

Figure 4.14 Results of Dataset III with BP.

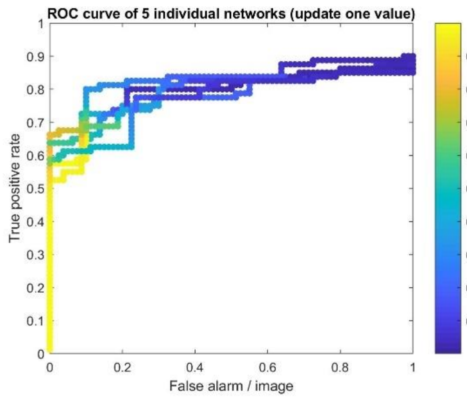
Figure 4.14 (c) indicates that the results of networks trained by BP are stable. The reason could be that initialization method of weights were the same in each network and each network has the same starting point with zero initialization. Thus, the algorithm search solutions are in the same neighborhood space with backpropagation.

4.2.4.2 Evolutionary Algorithm with Partial Backpropagation (EAPB)

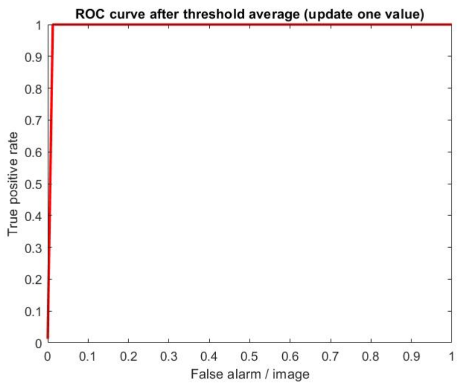
Five individual networks, which were also trained by EAPB, are featured in this section. The third initialization method mentioned in Section 3.2.3 and the second mutation method are used. These networks were trained by group 9 of Dataset III. Dataset I, Dataset II and the remaining images in Dataset III represent the three featured groups of test data in this section. Parameters used in this section are the same as parameters in Table 4.2. Figure 4.15 shows the results of Dataset I and Dataset II. Figure 4.16 shows results of Dataset III.



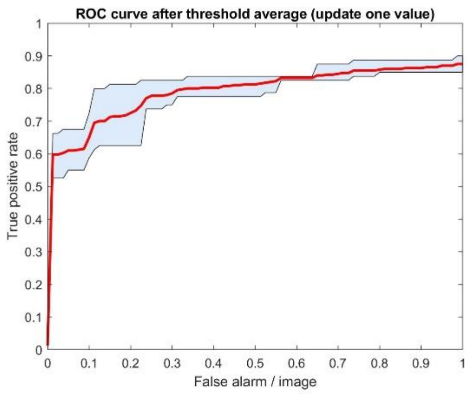
(a) Zoomed in ROC curves for Dataset I



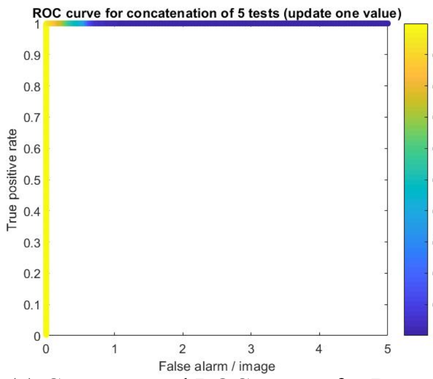
(b) Zoomed in ROC curves for Dataset II



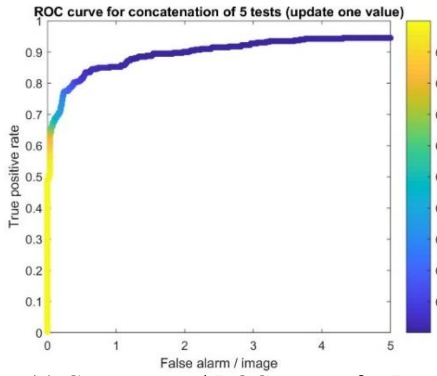
(c) Vertical average ROC curves for Dataset I



(d) Vertical average ROC curves for Dataset II

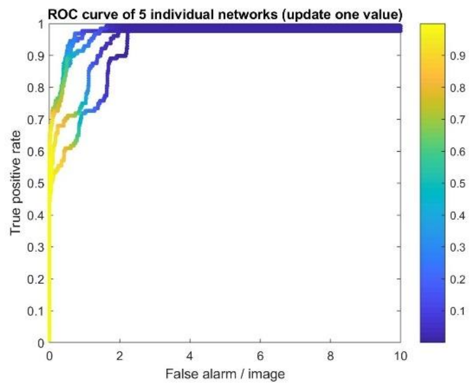


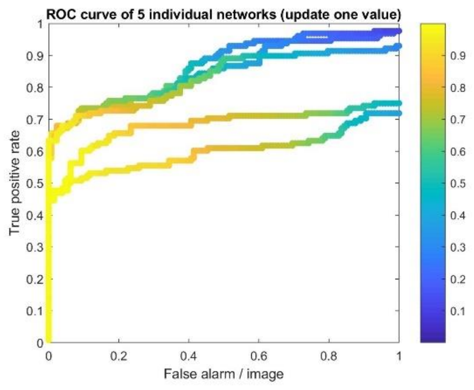
(e) Concatenated ROC curves for Dataset I



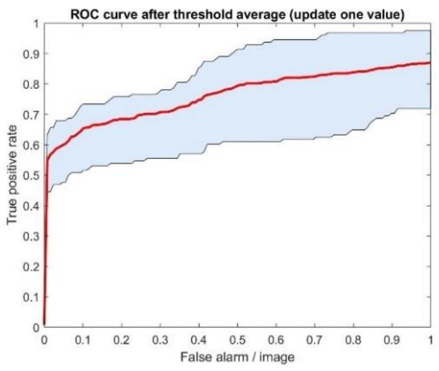
(e) Concatenated ROC curves for Dataset II

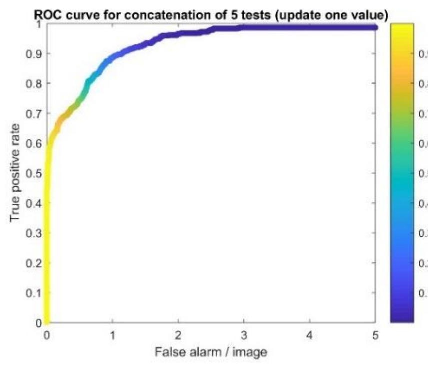
Figure 4.15 Results of Dataset I and Dataset II with EAPB.





(a) Original ROC curves (b) Zoomed in ROC curves





(c) Vertical average ROC curve (d) Concatenated ROC curve

Figure 4.16 Results of Dataset III with EAPB

Figure 4.15 (a), (c) and (e) show results of networks trained by EAPB also have perfect performances on Dataset I, which means networks trained by EAPB can exactly detect the target from images which have the same distance as the training images. In figure 4.15 (b), (d) and (f), for test images with occlusion situations, when false alarm per image is 0, the TP rate reaches 0.6 and when false alarm turns to 1, the TP rate is around 0.9. Networks trained by EAPB have a great capacity to detect targets from different distances when we train them with images from only one distance. But networks trained by EAPB take more time for training. The reason could point to the evolutionary algorithm part of this approach, which not only exploits the search space, but also conducts an exploratory search that needs much more time to expand and search the solution space.

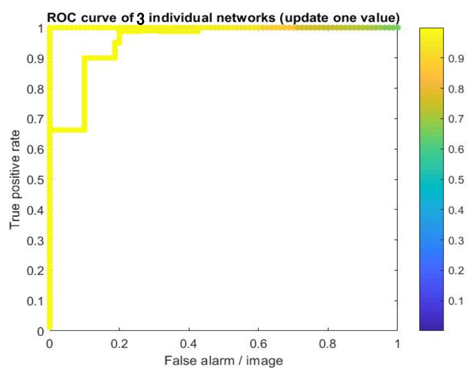
4.2.4.3 Memetic Algorithm (MA)

For the memetic algorithm, a single morphological layer network is trained by MA. The third initialization method mentioned in Section 3.2.3 and the second mutation method are used. All weights are put together as one chromosome and a local search is added to speed up the training process. With a local search, instead of searching solutions totally at random, each population converges to the nearest local minimum and compares with each other, which indicates a faster path to the global minimum. But even with a local search, training networks with MA still requires a lot of time. Thus, only three individual networks are trained with MA instead of five. Table 4 lists the parameters used for these networks.

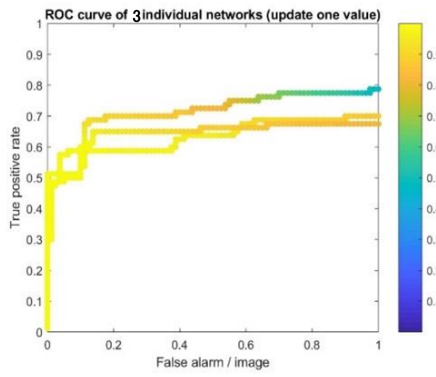
Table 4.4 Parameters of the single morphological layer network with MA

Iteration	**100**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
Structuring element size	5×5
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
Well trained	10^{-5}
Size of population	10
Probability of mutation	0.8
Probability of crossover	0.8

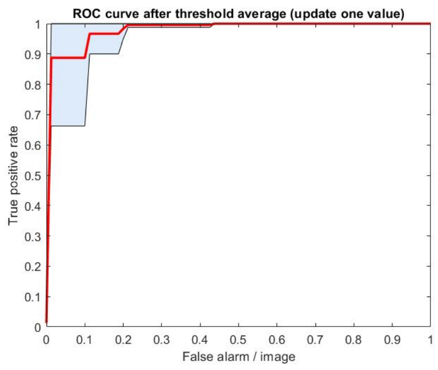
Figure 4.17 shows the results of Dataset I and Dataset II. The left column shows figures from Dataset I and the right column shows figures from Dataset II.



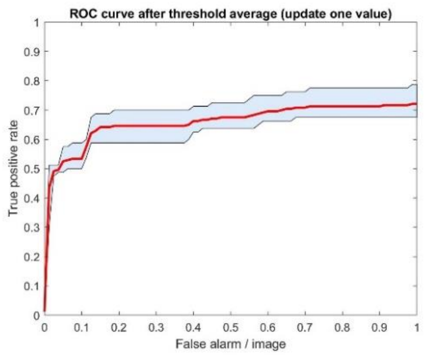
(a) Zoomed in ROC curves for Dataset I



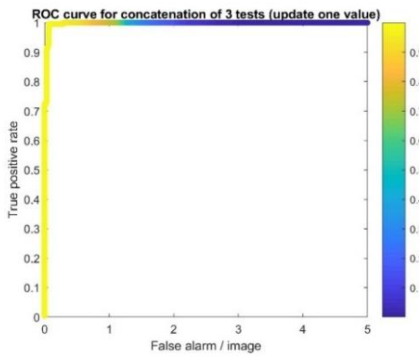
(b) Zoomed in ROC curves for Dataset II



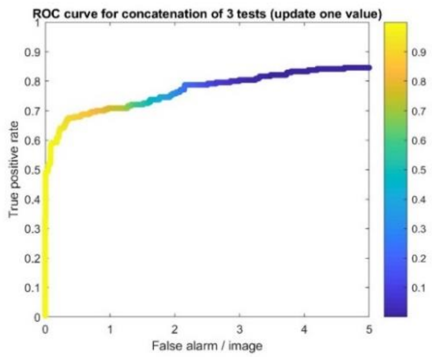
(c) Vertical average ROC curves for Dataset I



(d) Vertical average ROC curves for Dataset II



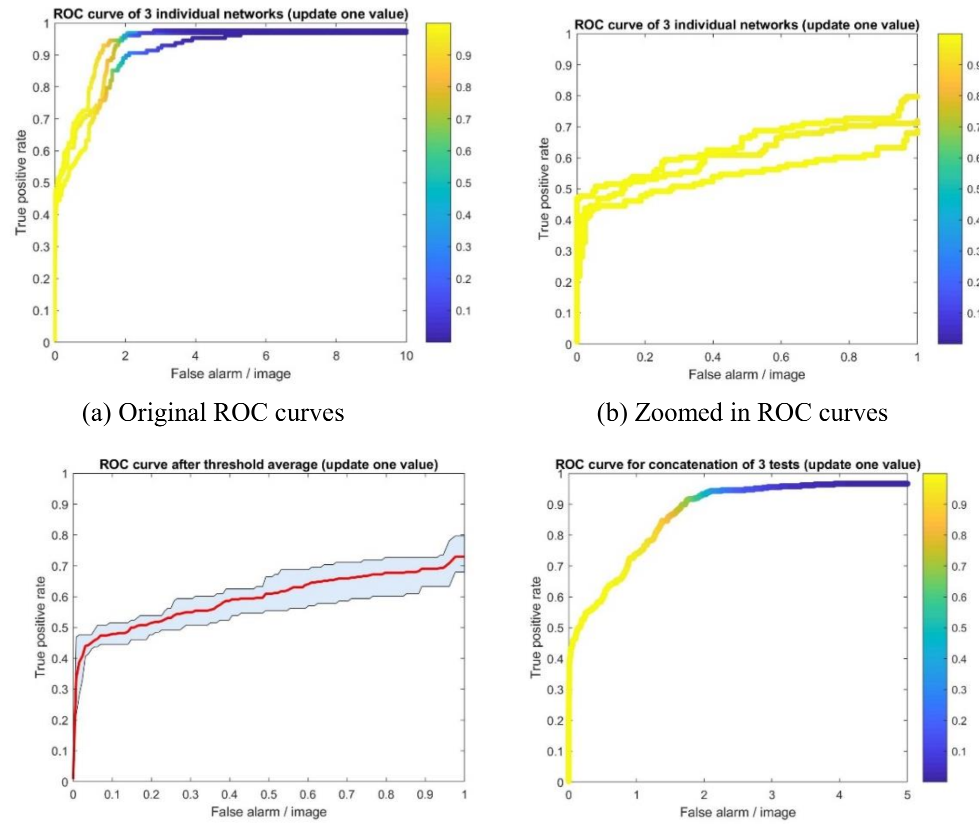
(e) Concatenated ROC curves for Dataset I



(f) Concatenated ROC curves for Dataset II

Figure 4.17 Results of Dataset I and Dataset II with MA.

Figure 4.18 shows test results of Dataset III. The ROC curves of Dataset I show that networks trained by MA perform poorly compared to the other two update approaches. When the FP rate is 0, the MA-trained network TP rates cannot always reach 1. For Dataset II, the highest TP rate is around 70 percent, which is lower than the others. Results of Dataset III are shown in Figure 4.18. It is obvious that the MA-trained performance of networks is not as good as the other two update approaches. The reason for this could lie in all weights being put into one chromosome making the chromosome too complicated, thereby rendering it unable to find the best solution within the appropriate time.



(a) Original ROC curves

(b) Zoomed in ROC curves

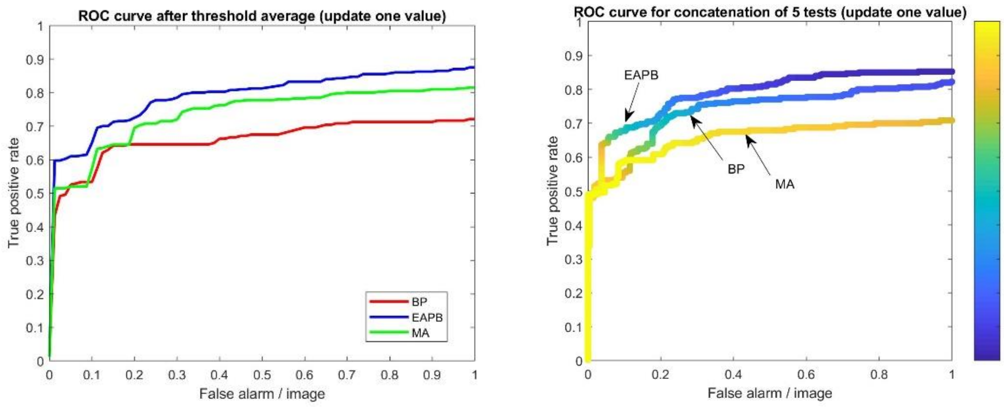
(c) Vertical average ROC curve

(d) Concatenated ROC curves

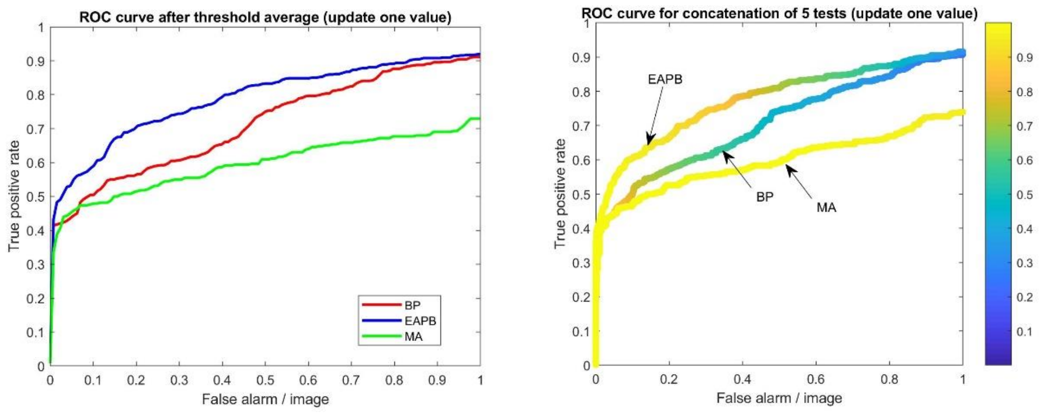
Figure 4.18 Results of Dataset III with MA.

4.2.4.4 Three update approaches

Based on the results of the three updated approaches, this section will discuss their differences. Figure 4.19 compares the approaches for Datasets II and III. No comparisons are shown on Dataset I due to its superior performance. The left column compares the vertical average ROC curves and the right column compares concatenated ROC curves.



(a) Vertical average ROC curves for Dataset II (b) Concatenated ROC curves for Dataset II



(c) Vertical average ROC curves for Dataset III (d) Concatenated ROC curves for Dataset III

Figure 4.19 Comparisons of the three updated approaches for Dataset II and Dataset III.

Figure 4.19 shows that the performances of networks trained by EAPB are better than the performances of the other two update approaches. With the same FP rate value, networks trained by EAPB have a higher TP rate, which means that when networks have the same number of false alarms per image, the networks trained by EAPB are better able to detect the target. From Figure 4.19 (a) and (b), the vertical average of the ROC curve and the concatenated ROC curve of BP for Dataset II are close to the curves of EAPB. The ability of networks trained by EAPB to detect the target in occlusion situations is slightly better than networks trained by BP, and the ability of networks trained by EAPB to detect the target from different distances is much better than networks trained by BP between intervals [0, 0.8]; then, the ROC curve of BP catches up with the ROC curve of EAPB. What we require for good networks is the same FP rate; thus, the higher the TP rate is, the better. Based on these results, networks trained by EAPB were determined to have the best performance, and networks trained by MA were determined to have the worst performance. Because of the long computing times and terrible performance of MA, MA will not be considered as an algorithm from this point on. Examples of detections results of three update approaches are shown in Appendix Figure A-7 and Appendix Figure A-8.

4.2.5 Results of Parallel Morphological Shared-weight Neural Network

All results of morphological one-layer networks with just one pair of kernels have already been posted. At this point, the question that comes to mind is: What happens when two parallel pairs of kernels work together in one morphological layer? Will results be better? To answer this question, a parallel morphological shared-weight neural network was trained with EAPB because of EAPB's superior network performance. The structure of this MSNN network is shown in Figure 4.20. Instead of one pair of kernels, the network has two parallel pair of kernels in its feature extraction stage. So, there are two feature maps after the hit-miss transform. These feature maps were put together, converted into a one-dimension vector and passed into the neuron network.

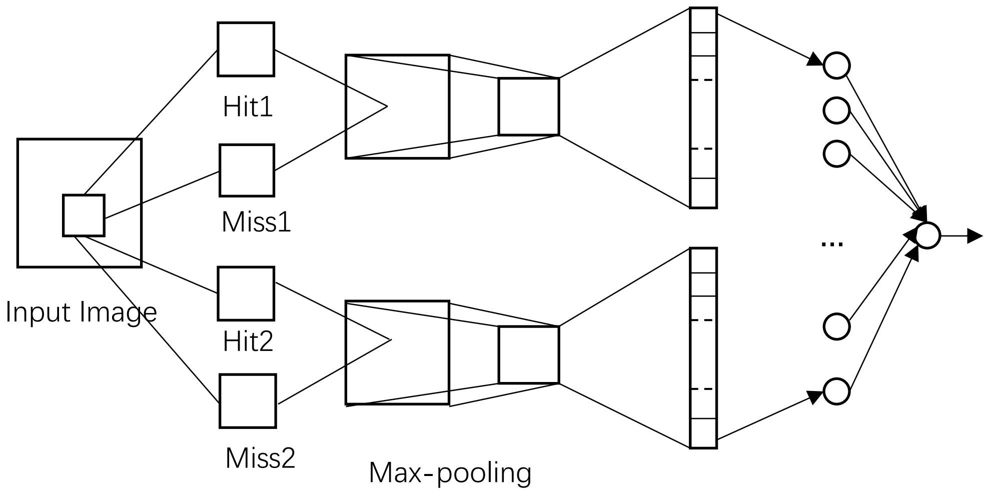


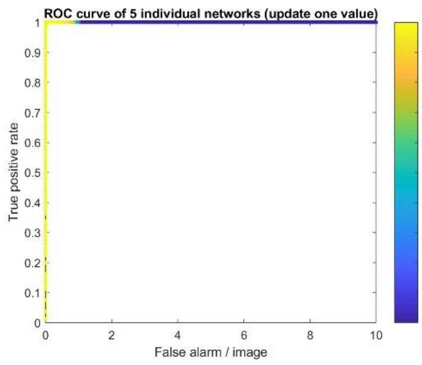
Figure 4.20 Architecture of a parallel morphological shared-weight neural network.

Table 4.5 Parameters of parallel MSNN

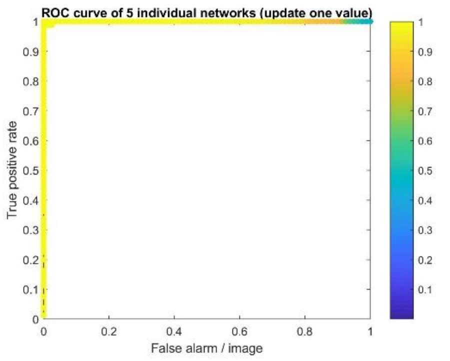
Iteration	**200**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
1st pair of structuring elements size 2nd pair of structuring elements size	5×5 5×5
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
β (momentum constant)	0.1
Well trained	10^{-5}
Learning rate	0.2
Size of population	20
Probability of mutation	0.4
Probability of cross over	0.9

The network was trained five times to make results more reliable. These networks were trained by group 9 of Dataset III and tested on three testing sets, which consists of Dataset I, Dataset II and the remaining images of Dataset III. Figure 4.21

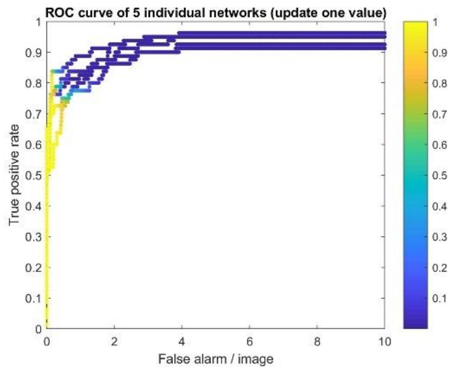
shows the original ROC curves and zoomed in ROC curves of these networks for Dataset I–III.



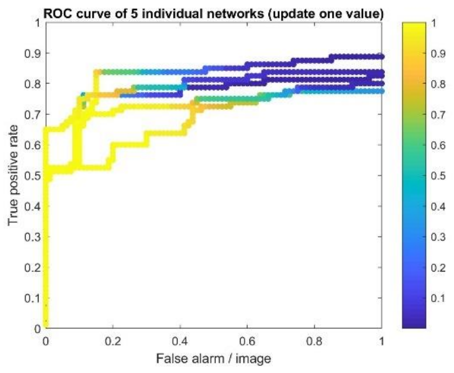
(a) Original ROC curves for Dataset I



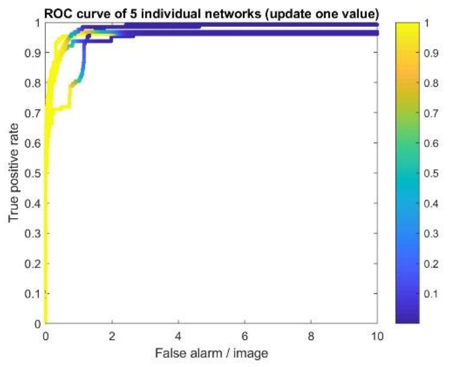
(b) Zoomed in ROC curves for Dataset I



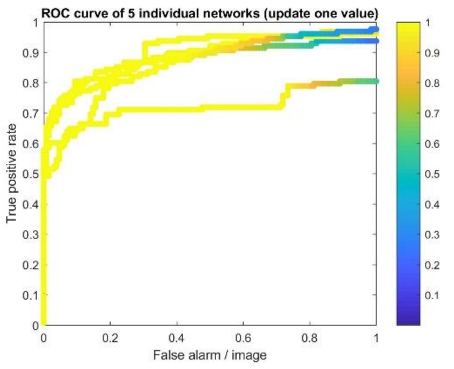
(c) Original ROC curves for Dataset II



(d) Zoomed in ROC curves for Dataset II



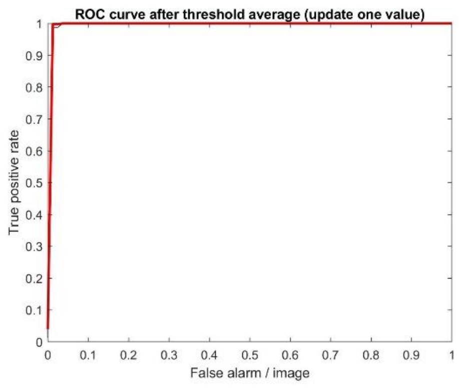
(e) Original ROC curves for Dataset III



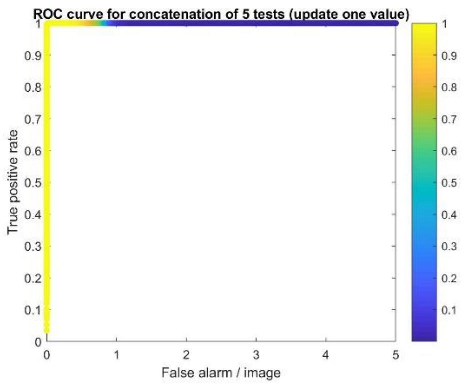
(f) Zoomed in ROC curves for Dataset III

Figure 4.21 Original and zoomed in ROC curves of parallel networks for Dataset I, Dataset II and Dataset III.

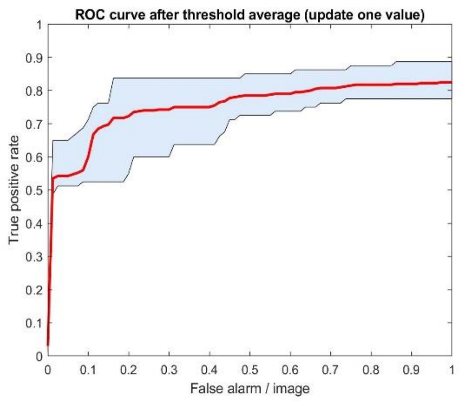
Figure 4.22 shows vertical average ROC curves and concatenated ROC curves of these networks for Datasets I–III.



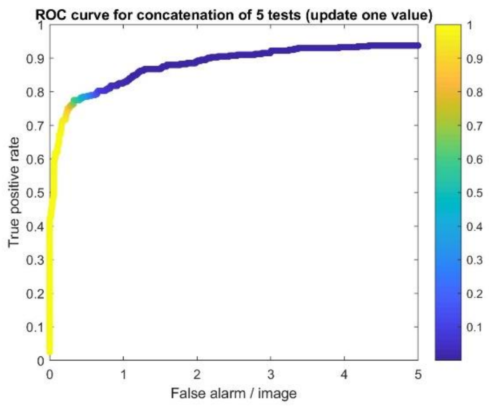
(a) Vertical average ROC curve for dataset I



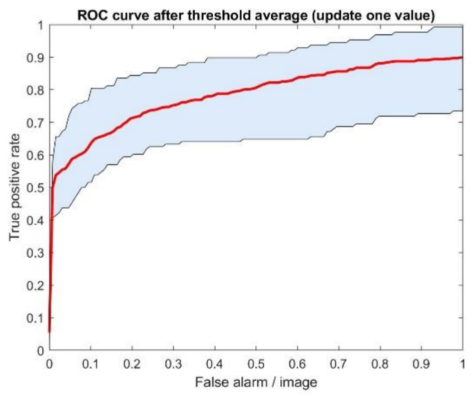
(b) Concatenated ROC curve for dataset I



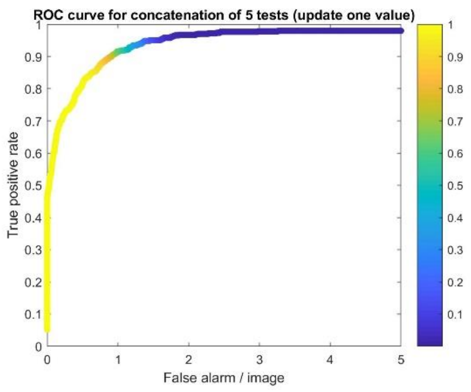
(c) Vertical average ROC curve for dataset II



(d) Concatenated ROC curve for dataset II



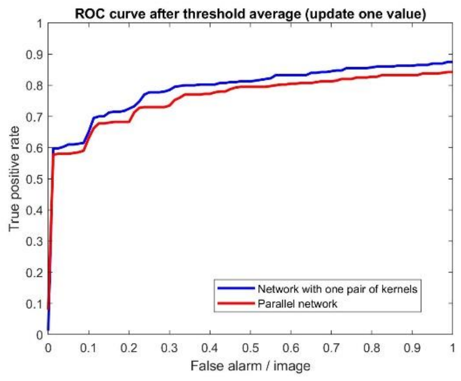
(e) Vertical average ROC curve for dataset III



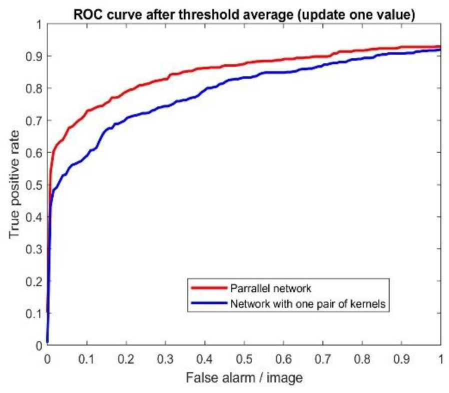
(f) Concatenated ROC curve for dataset III

Figure 4.22 Vertical average and concatenated ROC curves of parallel networks for Dataset I, Dataset II and Dataset III.

From Figures 4.21 and 4.22, we can find that parallel networks trained by EAPB perform perfectly on Dataset I, and all networks reached the 1 TP rate for Dataset I. This means that parallel networks can detect targets correctly from all images with the same distance as the training images. In figure 4.22 (c) and (d), for dataset II, which are images with occlusion situations, when the FP rate is 0, the TP rate is over 50 percent and the TP rate reaches 80 percent while the FP rate turns to 1. From figure 4.22 (e) and (f), the TP rate turns from 50 percent to 90 percent when the FP rate changes from 0 to1.



(a) Vertical average ROC curve for Dataset II



(b) Vertical average ROC curve for Dataset

Figure 4.23 Comparisons between parallel networks and networks with one pair of kernels for Dataset II and Dataset III.

In figure 4.23, results of the parallel network are compared with results of the single layer networks with one pair of kernels for Dataset II and Dataset III. Figure 4.23 (a) shows that performances of both networks are similar, but the single layer network with one pair of kernels performs slightly better than the parallel network. From figure 4.23 (b), it is clearly that the parallel network performs better than the single layer network with one pair of kernels. Examples of detections results of parallel networks are shown in Appendix Figure A-9.

4.3 Results of Multiple Morphological Layer Network

Deep learning has become a hot topic. Many people believe that the deeper the better for networks. So, with MSNN, we also considered making it deeper. Instead of having one pair of kernels in one morphological layer, we used two morphological layers, and each layer has one pair of kernels. It is a cascaded network. We trained it with two update approaches: 1) the backpropagation approach and 2) the evolutionary algorithm with partial backpropagation approach. Our goal was to determine which update method works best in a cascaded network.

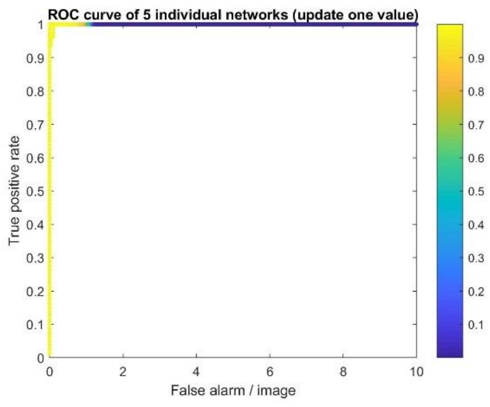
4.3.1 Backpropagation

First, we trained a two-layer morphological network with backpropagation five times. For training data, we continued to use images from group 9 of Dataset III. These networks were also tested on Datasets I–III. In the two-layer morphological networks, there are two morphological layers and after each morphological layer, there is a max-pooling layer. So, after the feature extraction stage, the feature map of two-layer morphological networks is much smaller. Parameters used in this network are listed in Table 4.6.

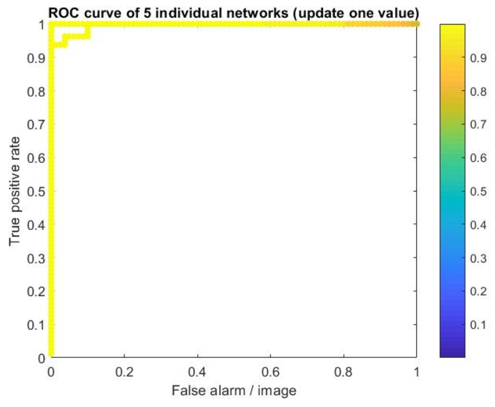
Table 4.6 Parameters of two-layer morphological network trained by BP

Iteration	**500**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
1st pair of structuring elements size 2nd pair of structuring elements size	5×5 3×3
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
β (momentum constant)	0.2
Well trained	10^{-5}
MaxRandSelect	5

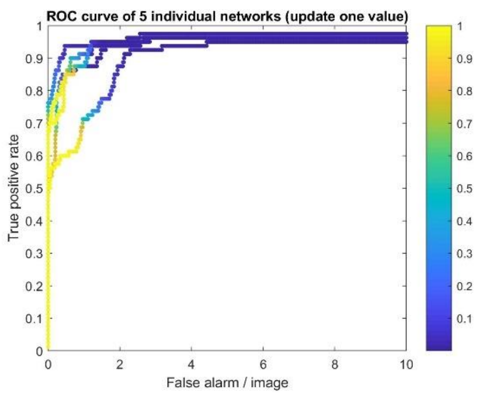
We used all zero initialization for kernels in first and second morphological layers and initialized all weights of the neuron network with small random numbers in the range of $[-1, 1]$. For hidden layers, rectified linear activation function was used to avoid gradient vanishing. Original and zoomed in ROC curves of two-layer morphological networks trained by BP are shown in Figure 4.23. Vertical average ROC curves and concatenated ROC curves are given in Figure 4.24.



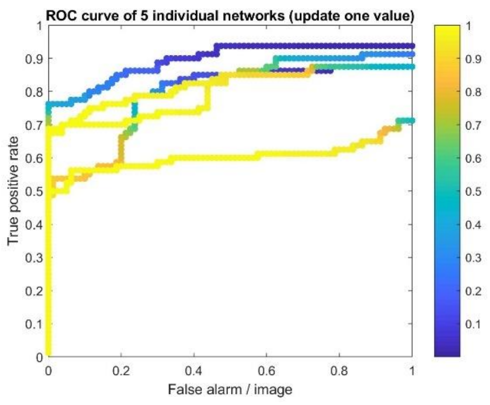
(a) Original ROC curves for Dataset I



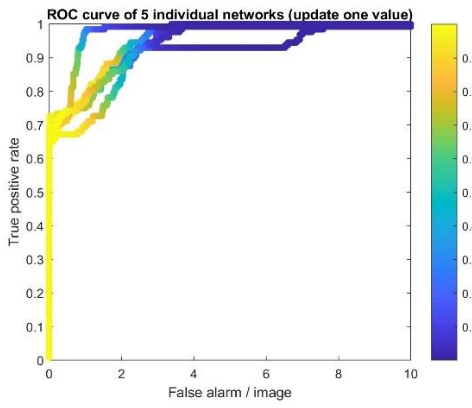
(b) Zoomed in ROC curves for Dataset I



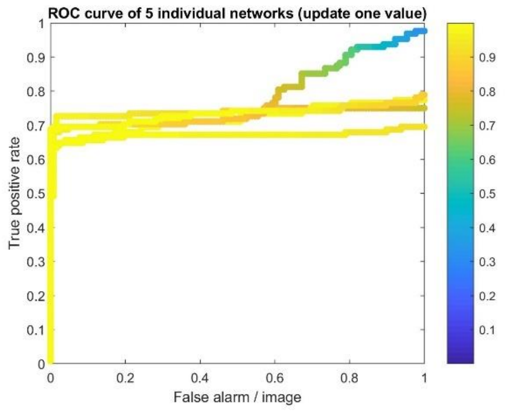
(c) Original ROC curves for Dataset II



(d) Zoomed in ROC curves for Dataset II

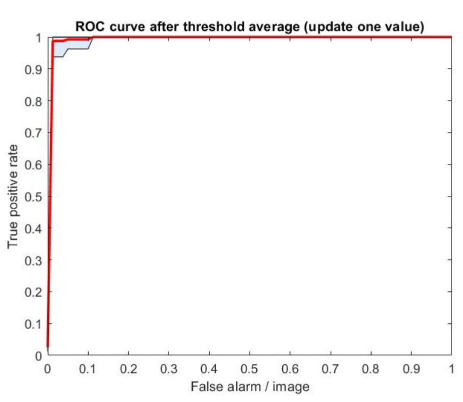


(e) Original ROC curves for Dataset III

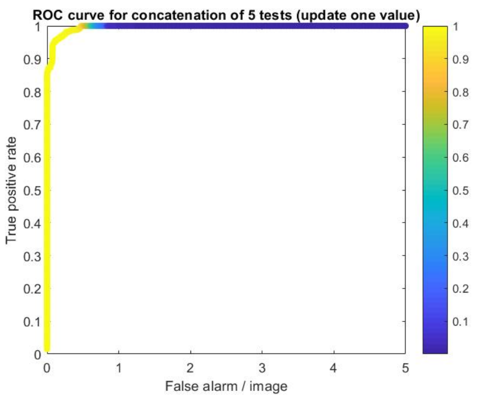


(f) Zoomed in ROC curves for Dataset III

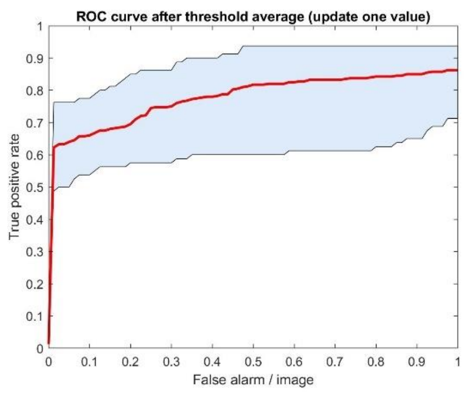
Figure 4.23 Original and zoomed in ROC curves of two-layer morphological networks trained by BP for Datasets I–III.



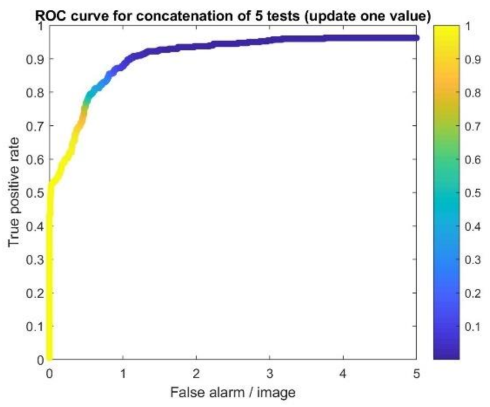
(a) Vertical average ROC curve for dataset I



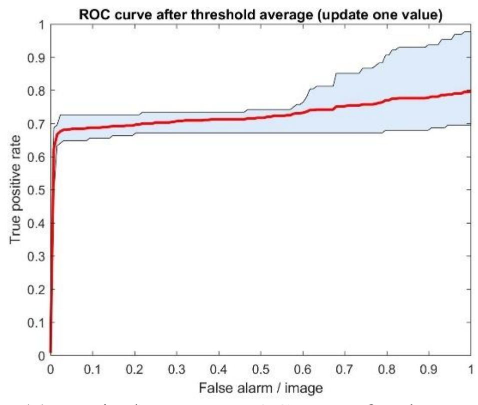
(b) Concatenated ROC curve for dataset I



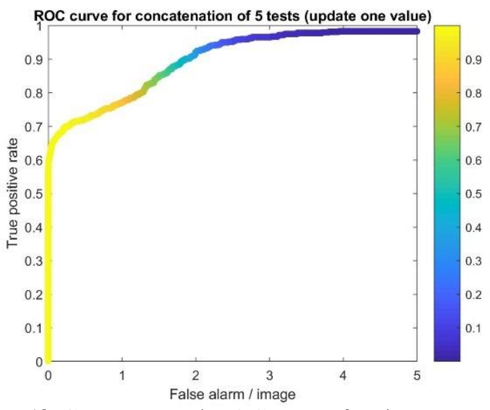
(c) Vertical average ROC curves for dataset II



(d) Concatenated ROC curve for dataset II



(e) Vertical average ROC curve for dataset III



(f) Concatenated ROC curve for dataset III

Figure 4.24 Vertical average and concatenated ROC curves of two morphological layers networks trained by BP for Datasets I–III

Figure 4.23 and Figure 4.24 show the performance of a two-layer morphological network in Dataset I has a slightly inferior performance than that of the other networks. When the FP rate is 0, the TP rate does not reach 1 every time, but it is still very close to 1; thus, this performance is still satisfying. For Dataset II, the color of the threshold quickly turns from yellow to blue when the FP rate is around 0.6 or 0.7. With this turning point, we can use the blue color to set the threshold as 0.7. We can also get an acceptable performance with a 0.6 false alarm per image and an accuracy of over 75%, which is an excellent result for an occlusion situation. Figure 4.24 (e) and (f) show networks with an accuracy of almost 70% when the FP rate is 0, which means networks can detect 70% of the targets correctly without a false alarm. This has been the highest accuracy possible when the FP rate equals 0 for Dataset III until now. Examples of detections results of multiple layer networks trained by BP are shown in Appendix Figure A-10.

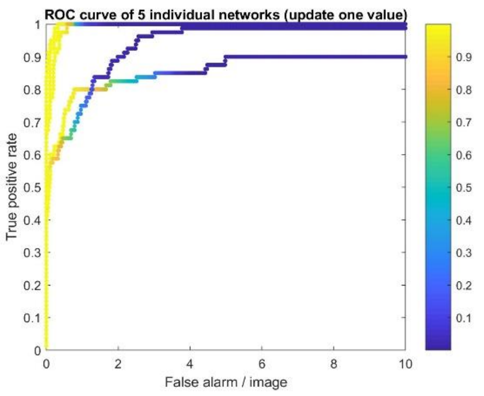
4.3.2 Evolutionary Algorithm with Partial Backpropagation

An evolutionary algorithm (EA) with partial backpropagation performs great when training a one-layer morphological network. So, we decided to use EA with partial backpropagation (EAPB) on two-layer morphological networks to test its performance. With EAPB, instead of putting the weights of one pair of kernels into the chromosome, we the put the weights of two pairs of kernels into the chromosome and initialized them with random Gaussian distribution numbers with 0 mean and 1 standard deviation. All other processes were the same as a one-layer morphological network trained by EAPB. Parameters used are listed in Table 4.7.

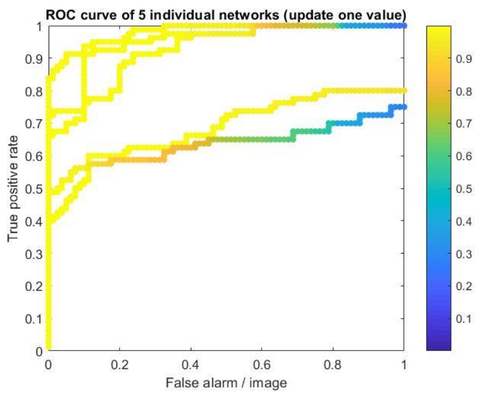
Table 4.7 Parameters of two-layer morphological network trained by EAPB

Iteration	**100**
N (number of training images)	8
M (number of sub-images/image)	40
Sub-image size	50×50
1st pair of structuring elements size 2nd pair of structuring elements size	5×5 3×3
Hidden neurons in neural network	20
a (parameter in sigmoid function in output layer)	0.3
β (momentum constant)	0.1
Well trained	10^{-5}
Size of population	20
Probability of mutation	0.8
Probability of cross over	0.8

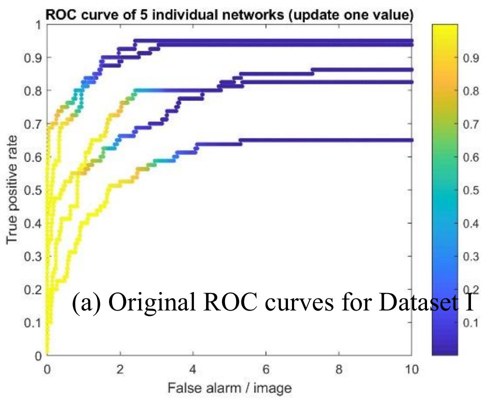
The two-layer morphological network was still trained five times. Training data came from group 9 of Dataset III images and networks were tested using three test sets which consist of Datasets I, Dataset II and the remaining images in Dataset III. Original and zoomed in ROC curves are shown in Figure 4.25. Vertical averages and concatenated ROC curves are given in Figure 4.26. Each row presents the featured paired parameters for each dataset in sequence (I–III).



(a) Original ROC curves for Dataset I

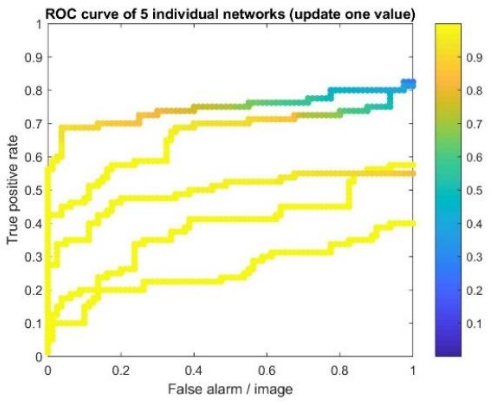


(b) Zoomed in ROC curves for Dataset I

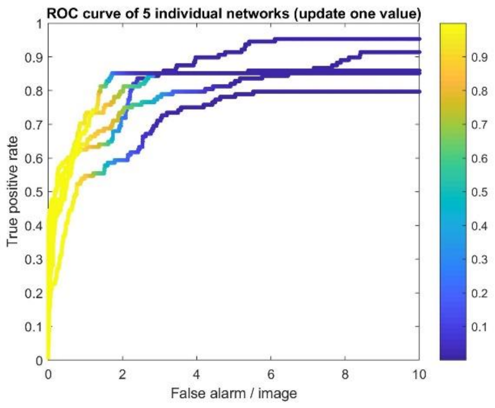


(a) Original ROC curves for Dataset I

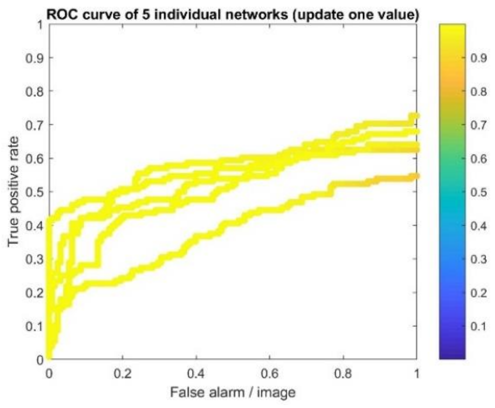
(c) Original ROC curves for Dataset II



(d) Zoomed in ROC curves for Dataset II

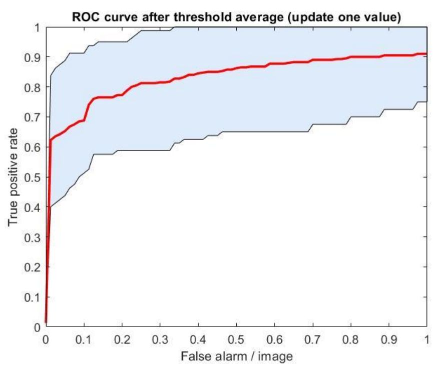


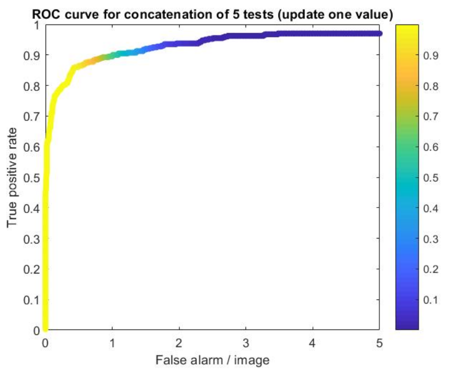
(e) Original ROC curves for Dataset III

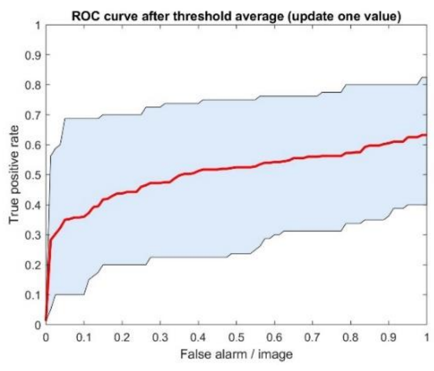


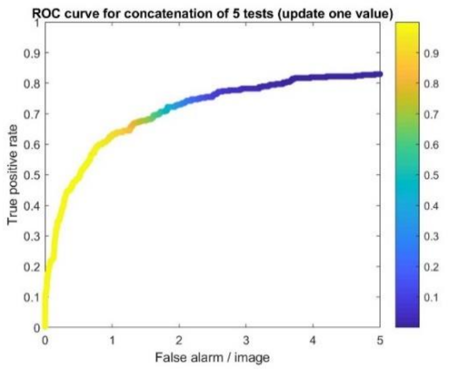
(f) Zoomed in ROC curves for Dataset III

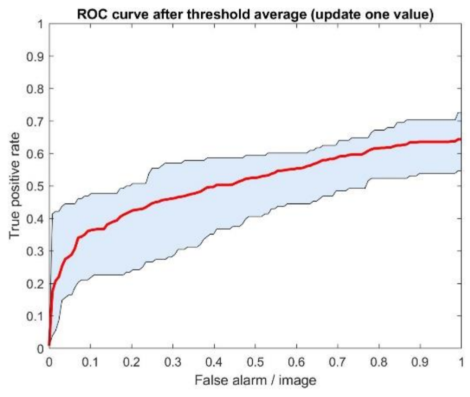
Figure 4.25 Original and zoomed in ROC curves of two-layer morphological networks trained by EAPB for Dataset I, Dataset II and Dataset III.


(a) Vertical average ROC curve for dataset I


(b) Concatenated ROC curve for dataset I


(c) Vertical average ROC curve for dataset II


(d) Concatenated ROC curve for dataset II


(e) Vertical average ROC curve for dataset II

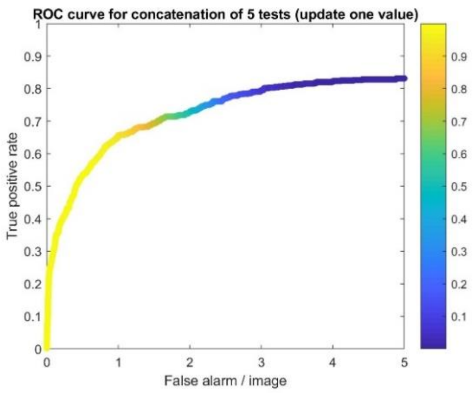
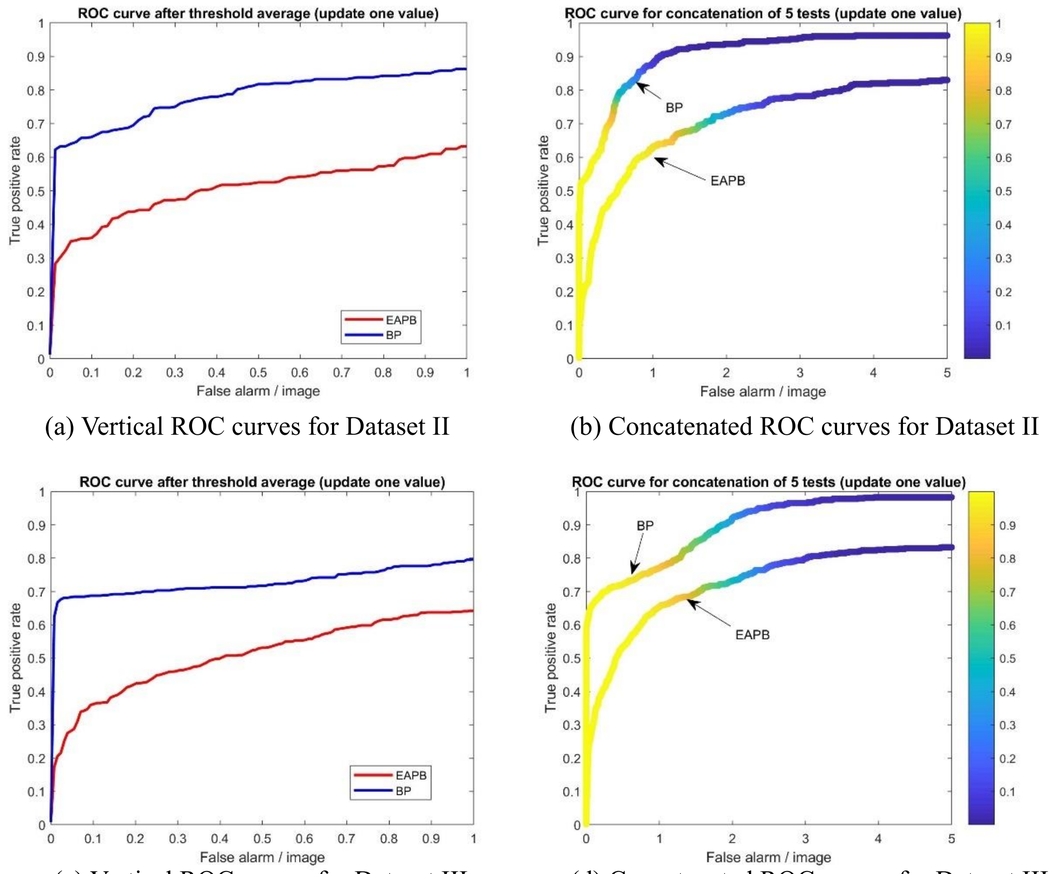

(f) Concatenated ROC curve for dataset II

Figure 4.26 Vertical average and concatenated ROC curves of two-layer morphological networks trained by EAPB for Dataset I, Dataset II and Dataset III.

From Figures 4.25 and 4.26, it is obvious that two morphological neural networks trained by EAPB performed very poorly compared to other previously trained networks. In Figure 4.26 (a) and (b), for Dataset I, the TP rate could reach 1 before the FP rate reached 1, which means the two-layer morphological neural networks trained by EAPB cannot detect all targets correctly from test images, which have the same distance as the training images and have no occlusion situation. In Figure 4.26 (c)–(f) representing Dataset II and III, in both cases, the two morphological neural networks trained by EAPB perform unsatisfactorily. When the false alarm per image is 0, the true positive rates for the two datasets are around 0.3 and 0.2, which is much lower than the true positive rates recorded for the other networks. Moreover, when the false alarm per image reaches 1, the true positive rates (TP rate) for two datasets are only around 0.6. Reasons for the terrible performance could be due to the evolutionary part. We added weights of another pair of kernels into the chromosome, which made the chromosome more complicated and more difficult to find in the search. We were unable to find the best initialization method for the multiple-layer morphological networks, which could have caused the unacceptable results. Examples of detections results of multiple layer networks trained by EAPB are shown in Appendix Figure A-10.

4.3.3 Comparisons of Two-layer Morphological Networks

In this section, we present the results of putting two-layer morphological networks trained by BP and trained by EAPB together for comparison. Figure 4.27 shows the vertical average and concatenated ROC curves of both updated approaches for Dataset II and Dataset III.



(a) Vertical ROC curves for Dataset II (b) Concatenated ROC curves for Dataset II

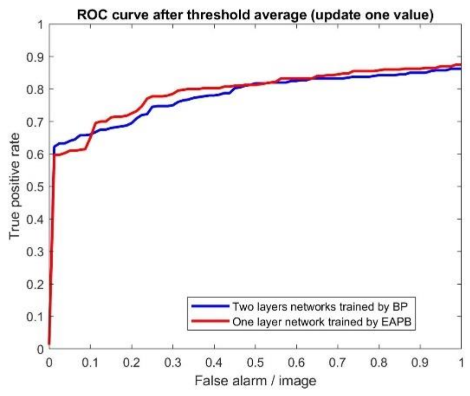
(c) Vertical ROC curves for Dataset III (d) Concatenated ROC curves for Dataset III

Figure 4.27 Comparisons of two-layer morphological networks trained by BP and networks trained by EAPB for Dataset II and Dataset III.

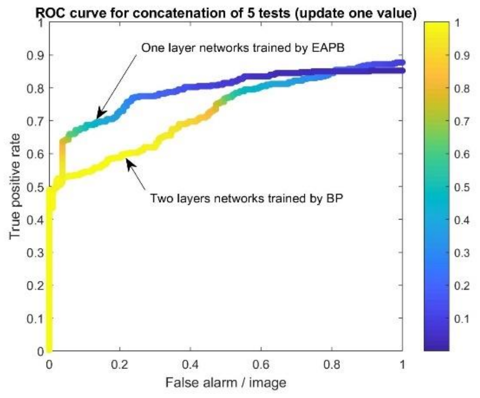
From Figure 4.27, we can see that performance of networks trained by BP is much better than networks trained by EAPB for both datasets. This is interesting because the EAPB-trained networks with a one-layer morphological network performed better than the BP-trained one-layer morphological networks; however, after adding another morphological layer, the networks trained by EAPB performed much worse than before, and the networks trained by BP performed better than before. For EAPB, chromosomes become more complicated after adding another morphological layer, which makes the solution space much wider and more difficult to explore. Although the EAPB uses elitism, it cannot guarantee to find the global minimum each time. Because of random evolutions of generating offspring, the EAPB has big chances to get rid of the local minimum. But it needs "luck" to make moves near the global minimum. It could move from a local minimum to another local minimum with "bad luck". And if the solution space is too wide, it is difficult for the EAPB to find the global minimum within the limited generations. Therefore, results of networks trained by EAPB are not stable. For the two-layer morphological networks trained by EAPB, we have not yet found an appropriate initialization method. Initialization methods can visibly affect results. This was confirmed in Section 4.2.3. Meanwhile, BP, solutions are becoming better and better in each generation. Although BP cannot avoid the local minimum, BP finds a "minimum" around the start point at least. This could explain why the two-layer networks trained by EAPB performed so poorly compared to those trained by BP.

4.4 Comparisons Between Networks with Different Structures

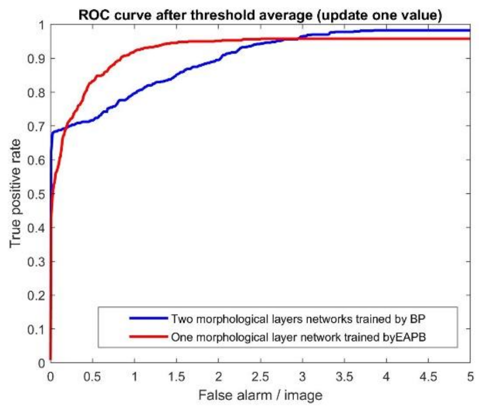
Now we have all results of one-layer morphological networks, parallel networks and two-layer morphological networks. This section compares all these networks. The best update approach for each structure is selected as a representative. So, one-layer morphological networks trained by EAPB are compared with two-layer morphological networks trained by BP to see if more layers make the performance better. Moreover, parallel networks trained by EAPB are compared with two-layer morphological networks trained by BP to observe which performance is better—the parallel network or cascaded network. We compare results of Dataset II and Dataset III because all networks performed satisfactorily on Dataset I. Figure 4.28 compares an EAPB-trained one-layer morphological network with a BP-trained two-layer morphological network. Figure 4.29 compares a parallel network and a cascaded network (two-layer morphological networks trained by BP).



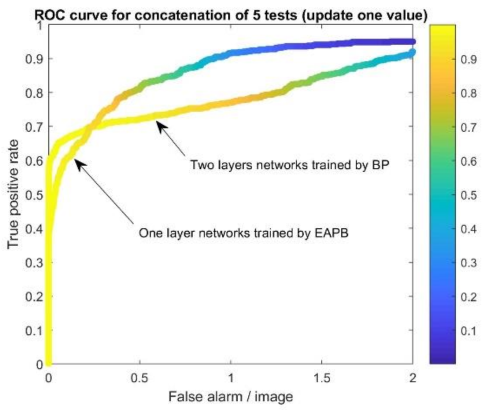
(a) Vertical average ROC curves for Dataset II



(b) Concatenated ROC curves for Dataset II



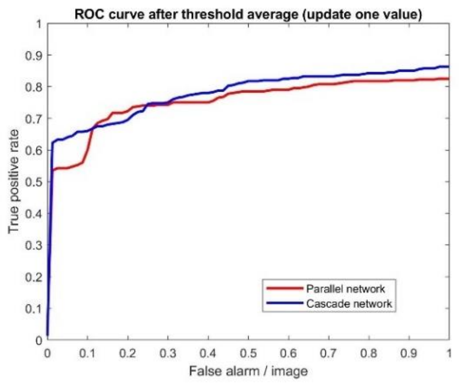
(c) Vertical average ROC curves for Dataset III in range [0, 5]



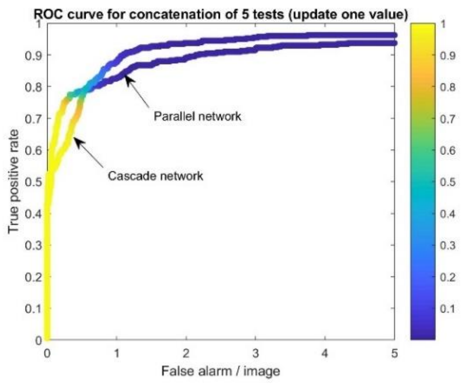
(d) Concatenated ROC curves for Dataset III in range [0, 2]

Figure 4.28 Comparison between a two-layer morphological network trained by BP and a one-layer morphological network trained by EAPB for Dataset II and Dataset III.

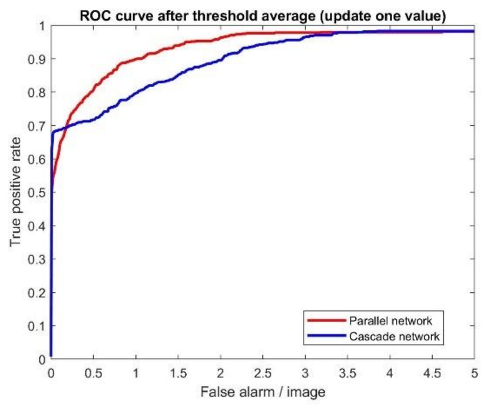
From Figure 4.28, we can see that when the FP rate is in the range of [0, 0.3], the performance of the two-layer morphological network trained by BP is better. Furthermore, in the preferred range [0.3, 2.5], the FP rate of a one-layer morphological network is higher, which indicates that if we want 0 false alarm per image with a higher TP rate, a two-layer morphological network trained by BP is a better choice. However, if we can accept a false alarm per image in range [0.3, 2.5] and need the TP rate to be as high as possible, a one-layer morphological network could better meet our needs.



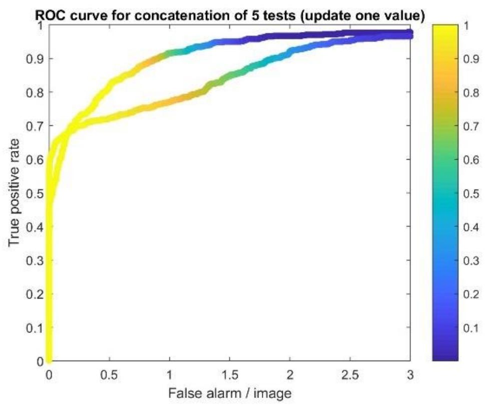
(a) Vertical average ROC curves for Dataset II



(b) Concatenated ROC curves for Dataset II



(c) Vertical average ROC curves for Dataset III



(d) Concatenated ROC curves for Dataset III

Figure 4.29 Comparison between cascaded networks and parallel networks for Dataset II and Dataset III.

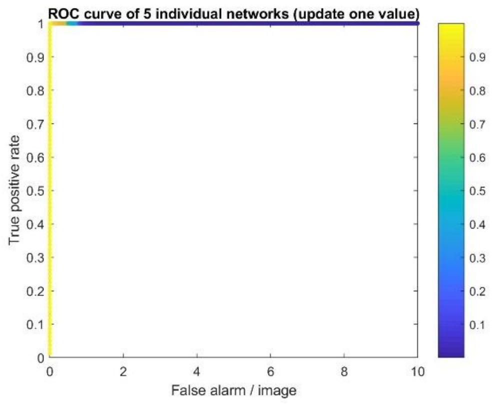
From Figure 4.29 (a) and (b), there is no significant difference between performances of cascade networks and parallel networks for Dataset II. They perform similarly on images which have an occlusion situation. For Dataset III, which consists of images with different distances, when we need 0 false alarm per image, the TP rate of the cascade network is almost 0.7 while the TP rate of the parallel network is around 0.5, which in this case means that the cascade network is a better choice. However, if we can accept some false alarms per image, the parallel network could be preferred.

4.5 Comparisons Between MSNN and CNN

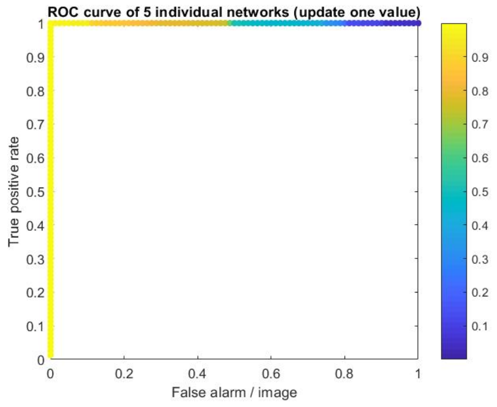
The convolutional neural network (CNN) is a widely used and popular neural network used for image processing. In this thesis, the morphological shared-weight neural network (MSNN) is introduced. A major goal of this research was to compare MSNNs with CNNs and see which performed better with our datasets. For the MSNN, there are one-layer morphological networks and two-layer morphological networks. So, we also have one-layer convolutional networks and two-layer convolutional networks for CNN.

4.5.1 Comparisons of One-Layer Networks

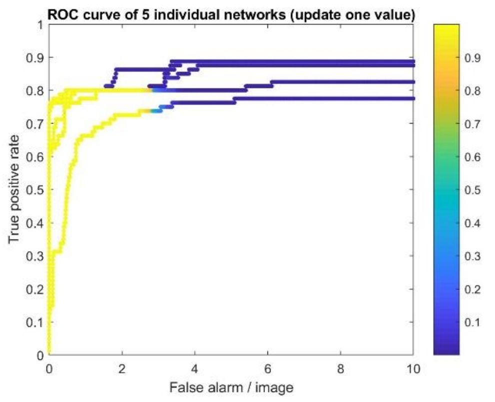
First, a convolutional one-layer network is set up to compare with a morphological one-layer network. In MSNN, a pair of structuring elements is used to compute a one-feature map that corresponds to one convolutional kernel in CNN. We trained CNN with backpropagation and trained MSNN with EAPB. One group of images from Dataset III is used as training data and these networks were tested on Dataset I–III. Figure 4.30 shows the original and zoomed in ROC curves for Dataset I. Figure 4.31 shows vertical average and concatenated ROC curves for Dataset I.



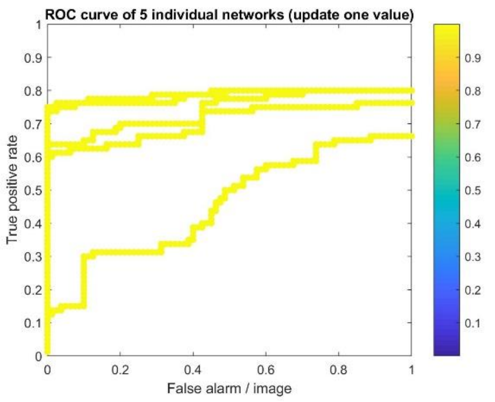
(a) Original ROC curves for Dataset I (MSNN)



(b) Zoomed in ROC curves for Dataset I (MSNN)

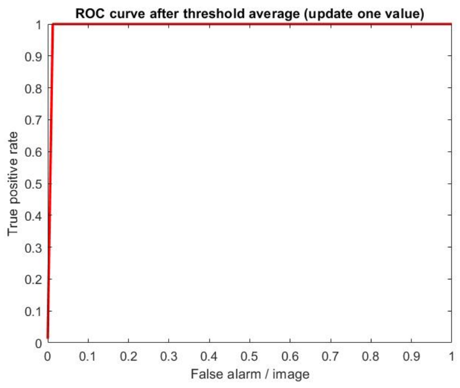


(c) Original ROC curves for Dataset I (CNN)

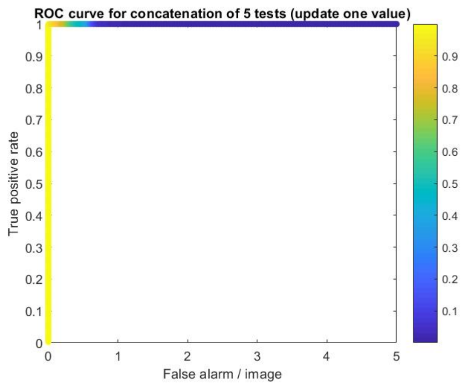


(d) Zoomed in ROC curves for Dataset I (CNN)

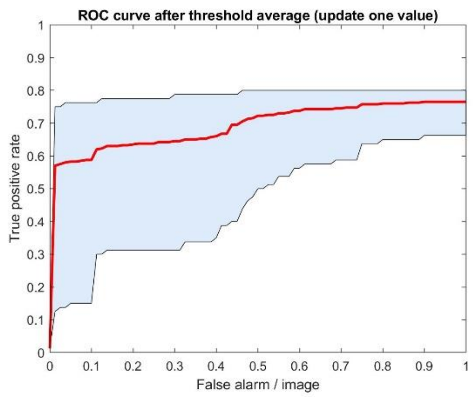
Figure 4.30 Original and zoomed in ROC curves of one-layer MSNN and CNN for Dataset I.



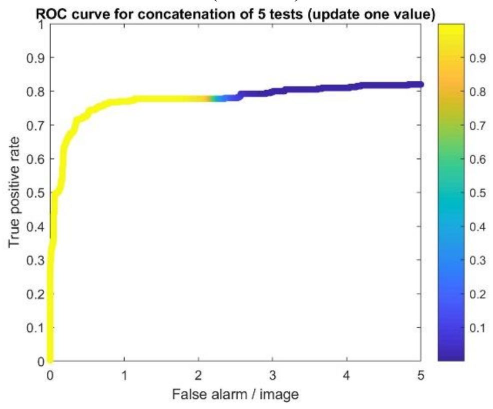
(a) Vertical average ROC curve for Dataset I (MSNN)



(b) Concatenated ROC curve for Dataset I (MSNN)



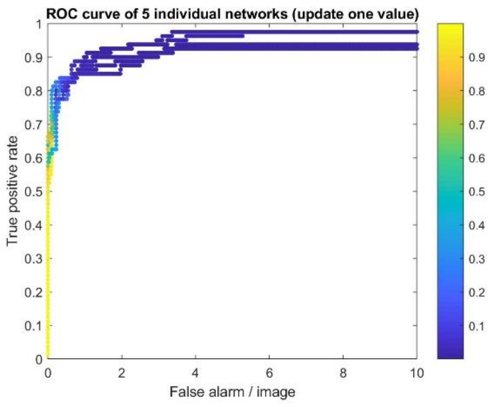
(c) Vertical average ROC curve for Dataset I (CNN)



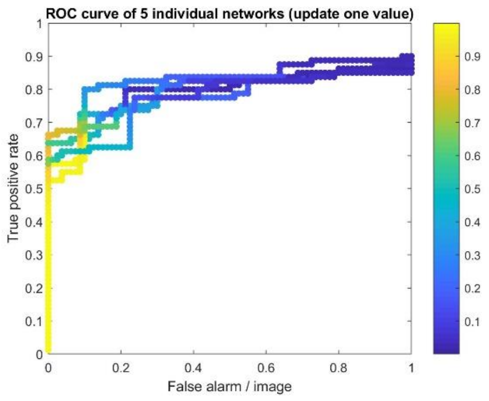
(d) Concatenated ROC curve for Dataset I (CNN)

Figure 4.31 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset I.

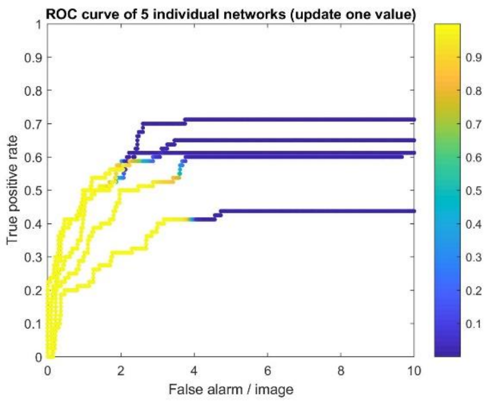
Figures 4.30 and 4.31 show the highly unsatisfactory performance of a one-layer CNN for Dataset I. The TP rate of all CNNs cannot reach 1 and the highest TP rate of CNN is around 0.8, which means a one-layer convolutional neural network has a lower ability than a one-layer morphological shared-weight neural network to detect targets from images, which have the same distance as training images. From Figure 4.31 (c), it is obvious that the results of CNN are unstable. At the same time, MSNN performed perfectly on Dataset I. When there is 0 false alarm per image, MSNN can detect all targets with 100 percent accuracy. Results for Dataset II are shown in Figures 4.32 and 4.33.



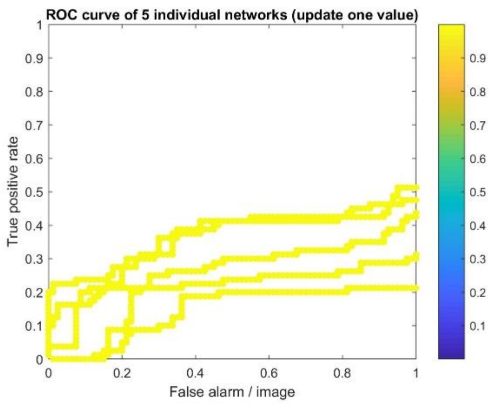
(a) Original ROC curves of MSNN



(b) Zoomed in ROC curves of MSNN

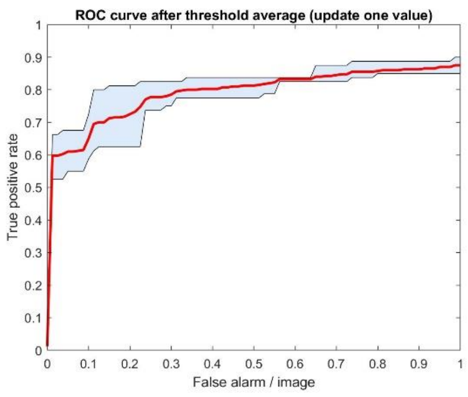


(c) Original ROC curves of CNN

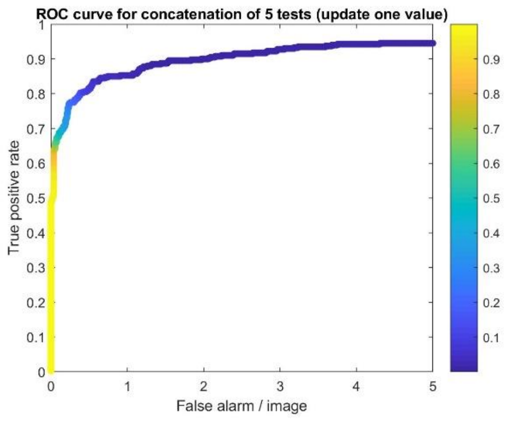


(d) Zoomed in ROC curves of CNN

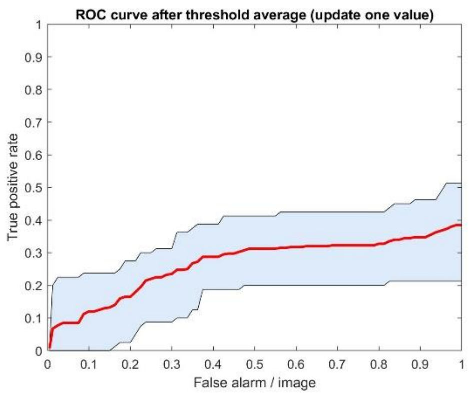
Figure 4.32 Original and zoomed in ROC curves of a one-layer MSNN and CNN for Dataset II.



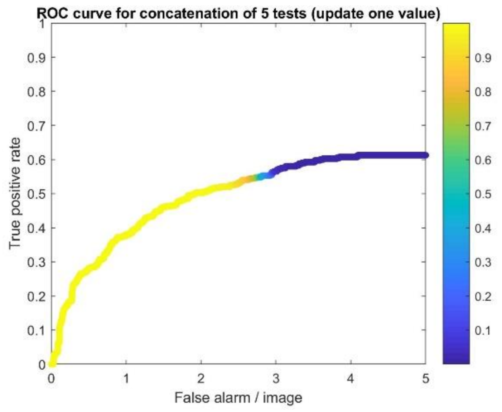
(a) Vertical average ROC curve (MSNN)



(b) Concatenated ROC curve (MSNN)



(c) Vertical average ROC curve (CNN)

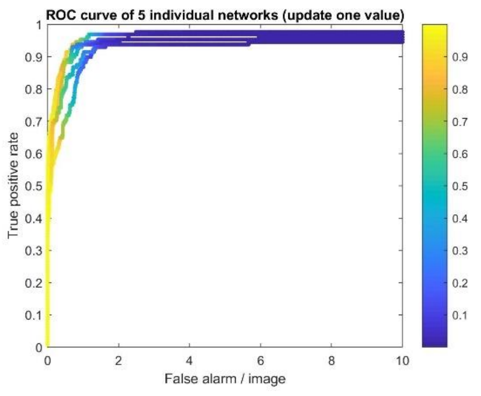


(d) Concatenated ROC curve (CNN)

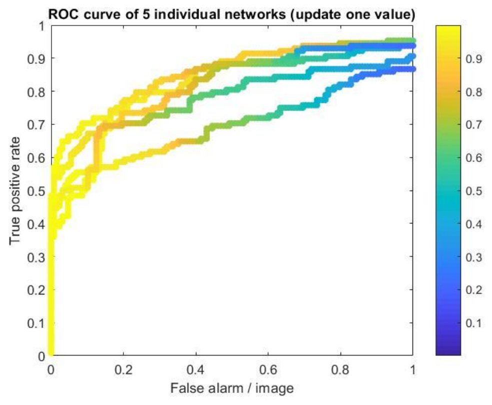
Figure 4.33 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset II.

Figures 4.32 and 4.33 for Dataset II shows the CNN performance as much worse than MSNN on images with occlusion. For the easiest Dataset I, the TP rate of CNN can reach around 0.8, but when data becomes more different, the highest TP rate of CNN is only around 0.6. When the false alarm per image is equal to 1, the TP rate of CNN is only 0.4, which is a very low accuracy rating. CNN cannot efficiently deal with images that have occlusions.

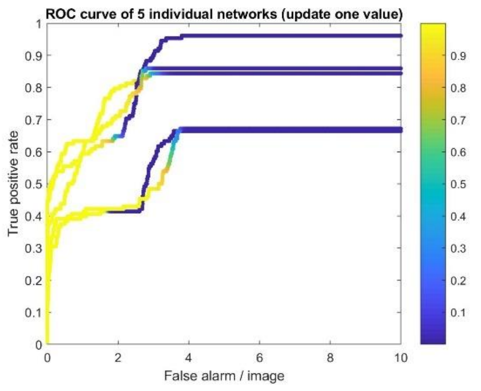
Figures 4.34 and 4.35 show results of a one-layer MSNN and CNN for Dataset III.



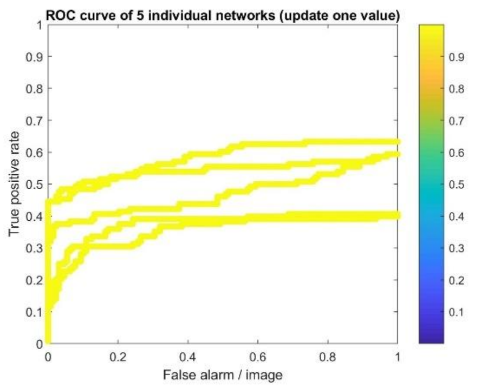
(a) Original ROC curves (MSNN)



(b) Zoomed in ROC curves (MSNN)



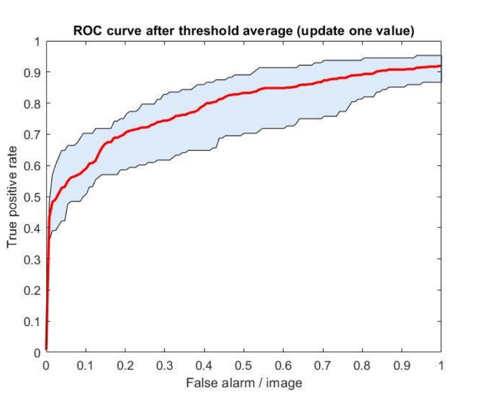
(c) Original ROC curves (CNN)



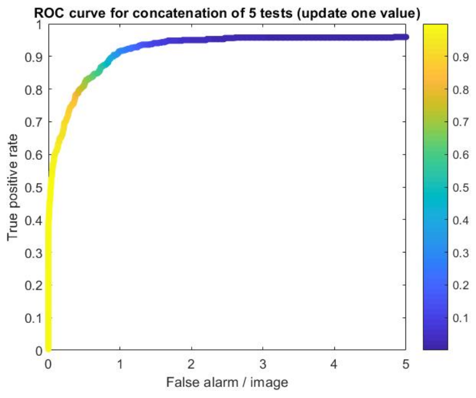
(d) Zoomed in ROC curves (CNN)

Figure 4.34 Original and zoomed in ROC curves of one-layer MSNN and CNN for Dataset III.

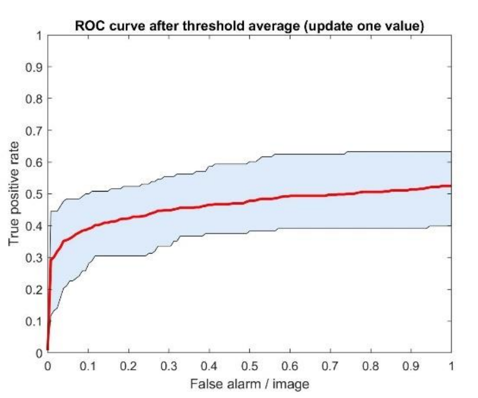
From these two figures, it is not surprising to find the CNN performance to be much worse than the MSNN performance based on prior performances. The poor CNN performance could be caused by not enough training data. Thus, we can conclude that CNN needs a lot of training data to make its accuracy higher. Examples of detections results of single layer networks trained by MSNN and by CNN are shown in Appendix Figure A-11 and Appendix Figure A-12.



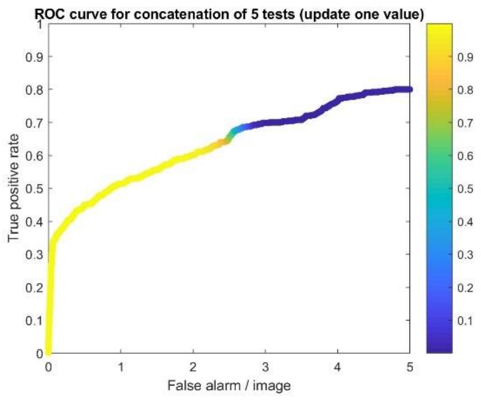
(a) Vertical average ROC curve (MSNN)



(b) Concatenated ROC curve (MSNN)



(c) Vertical average ROC curve (CNN)

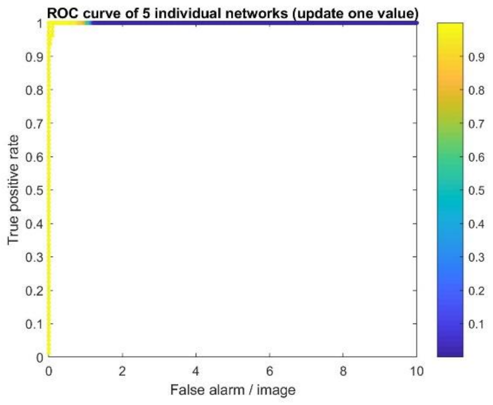


(d) Concatenated ROC curve (MSNN)

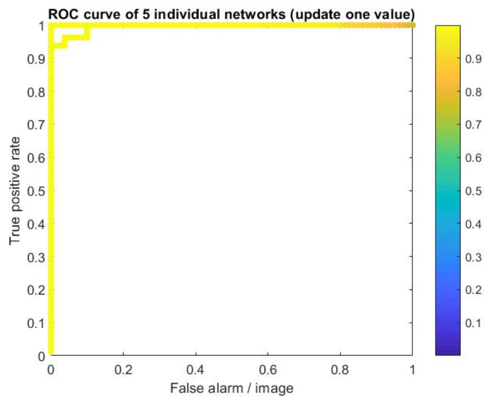
Figure 4.35 Vertical average and concatenated ROC curves of one-layer MSNN and CNN for Dataset III.

4.5.2 Comparisons of Two-Layer Networks

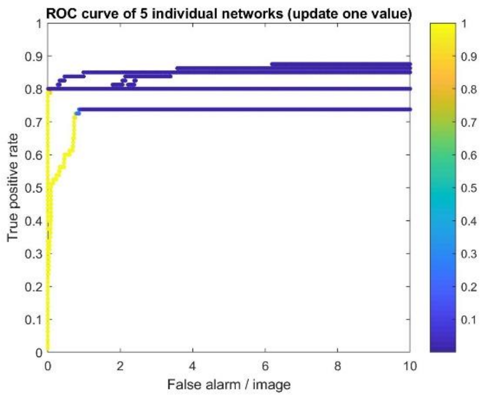
This section compares two-layer networks of the CNN and the MSNN. A two-layer morphological network trained by BP is used for comparison. For CNN, there are two convolutional layers and each layer has one convolutional kernel. The update approach for CNN is still backpropagation. Figure 4.36 - 4.41 show results of two-layer MSNN and CNN for Datasets I, II, and III.



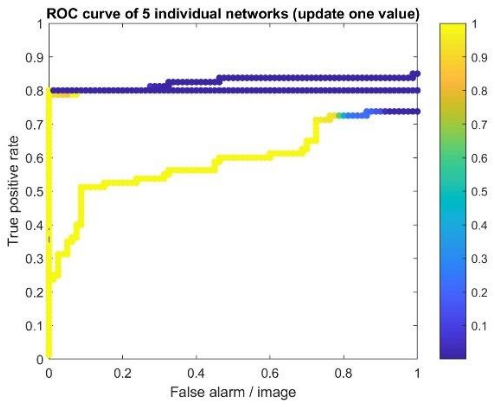
(a) Original ROC curves (MSNN)



(b) Zoomed in ROC curves (MSNN)

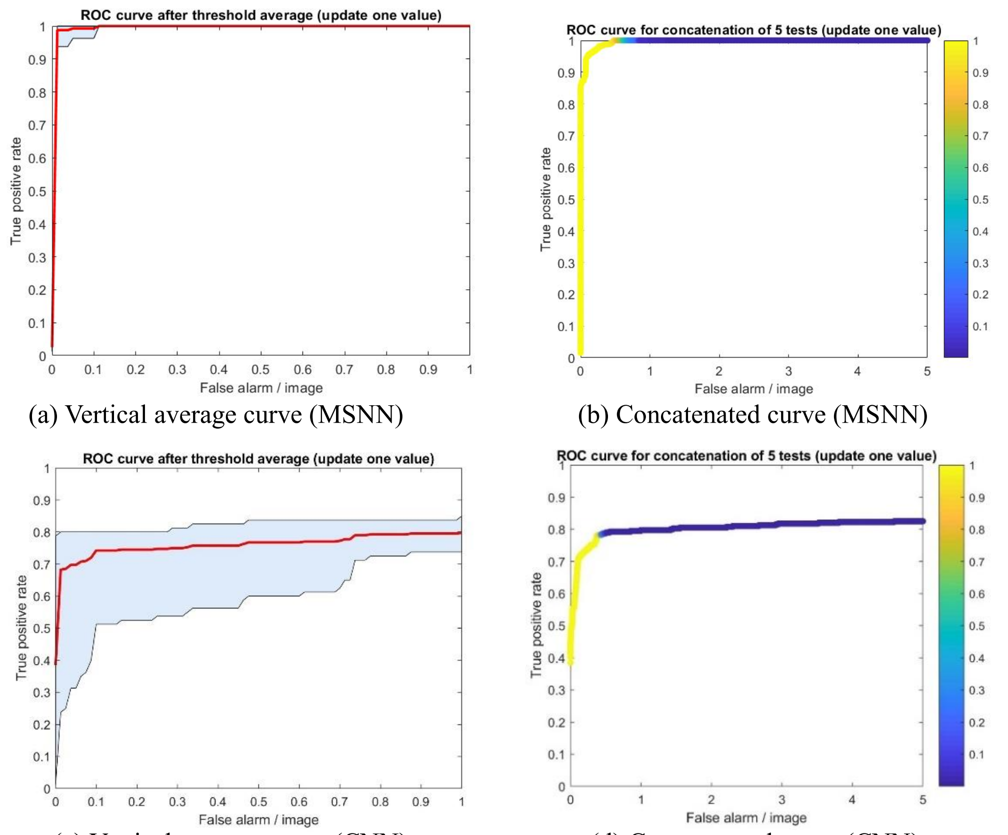


(c) Original ROC curves (CNN)



(d) Zoomed in ROC curves (CNN)

Figure 4.36 Original and zoomed in ROC curves of two-layer MSNN and CNN for Dataset I.

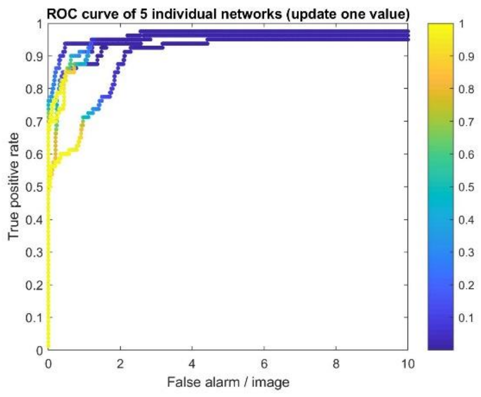


(a) Vertical average curve (MSNN) (b) Concatenated curve (MSNN)

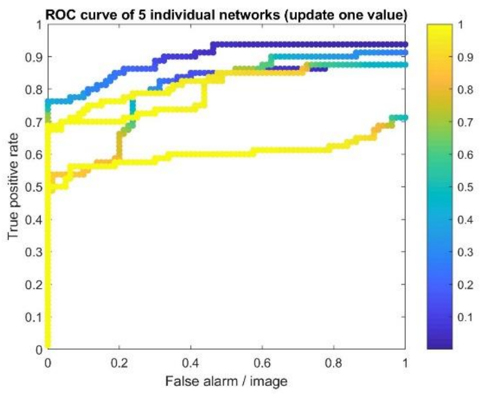
(c) Vertical average curve (CNN) (d) Concatenated curve (CNN)

Figure 4.37 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset I.

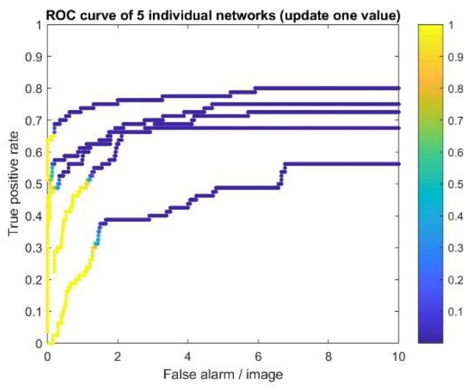
In Figures 4.36 and 4.37, for Dataset I, the highest FTP values of MSNN and CNN are 1 and 0.8, respectively. Both networks have a turning point, at which the threshold quickly turns from 1 to 0.1. So, we can pick 0.5 as our threshold to distinguish the target. Moreover, with this threshold, MSNN has a top TP rate of 1 with 0.6 false alarm per image. CNN has a top TP rate of 0.8 with 0.4 false alarm per image. Notably, 0.6 and 0.4 false alarm per image are both acceptable TP rate values, but with MSNN, the accuracy reaches 100%, which is the optimum goal for our target detection algorithm. So, for Dataset I, performance of a two-layer MSNN is better than CNN based on our needs.



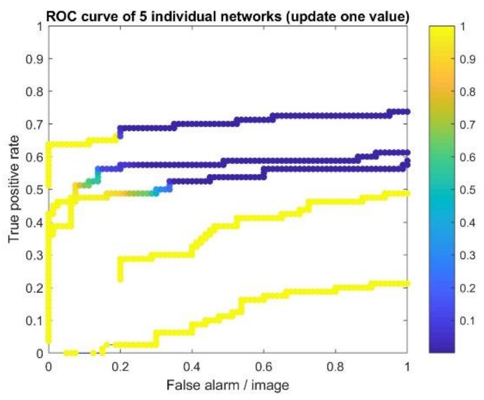
(a) Original ROC curves (MSNN)



(b) Zoomed in ROC curves (MSNN)

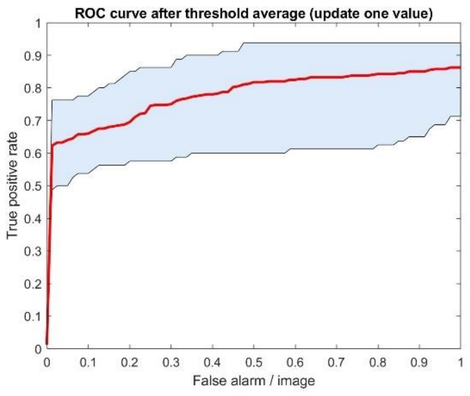


(c) Original ROC curves (CNN)

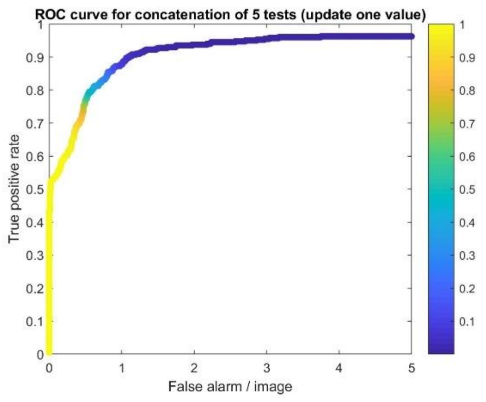


(d) Zoomed in ROC curves (CNN)

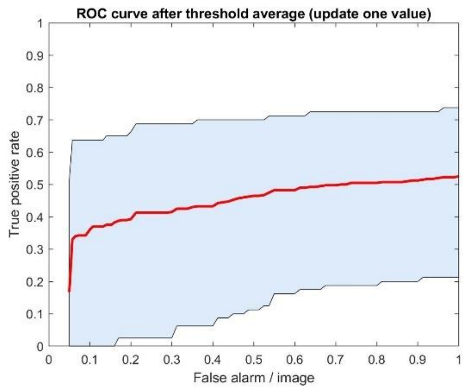
Figure 4.38 Original and zoomed in ROC curves of two-layer MSNN and CNN for Dataset II.



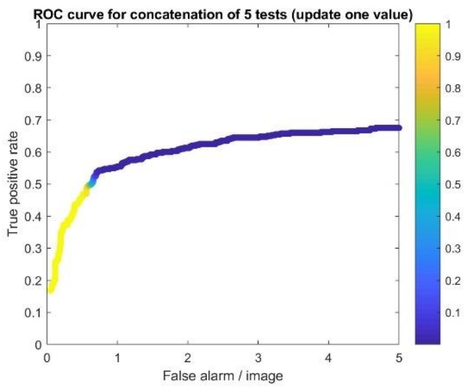
(a) Vertical average ROC curve (MSNN)



(b) Concatenated ROC curve (MSNN)



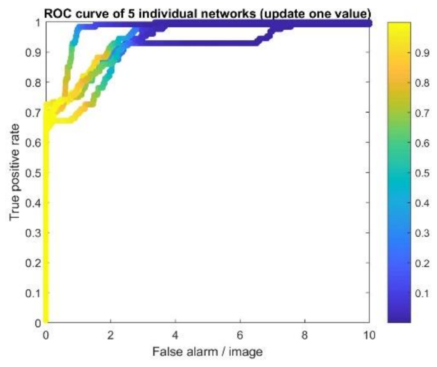
(c) Vertical average ROC curve (CNN)



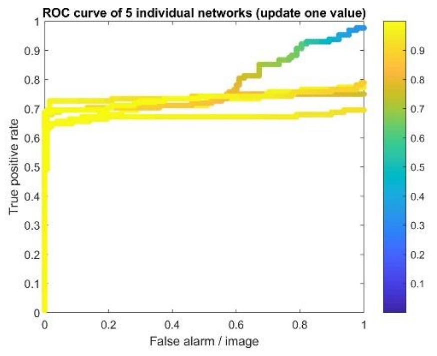
(d) Concatenated ROC curve (CNN)

Figure 4.39 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset II.

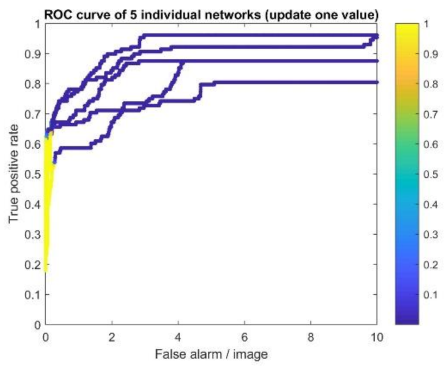
From Figures 4.38 and 4.39, it is obvious that CNN's performance on Dataset II is terrible. In Figure 4.39 (c), in the range of [0, 0.05], there is no TP rate, which means with CNN, false alarms per image cannot be lower than 0.05. Furthermore, this vertical average ROC curve of CNN for Dataset II has a large blue area which indicates a large fluctuation of CNN. The TP rate of MSNN can rise to over 90%, but the TP rate of CNN can only go up to 70%. At the turning point, which is the 0.5 threshold, MSNN has a TP rate which is around 0.75 with a 0.6 false alarm per image, while CNN has a TP rate which is 0.5 with a 0.8 false alarm per image. With these figures, we can conclude that CNNs have a super low ability to detect the target from images which have occlusion situations. MSNNs do a better job on occlusion-containing datasets.



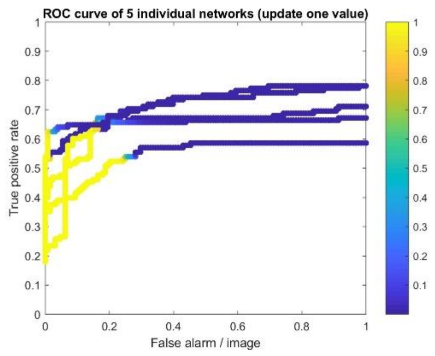
(a) Original ROC curves (MSNN)



(b) Zoomed in ROC curves (MSNN)

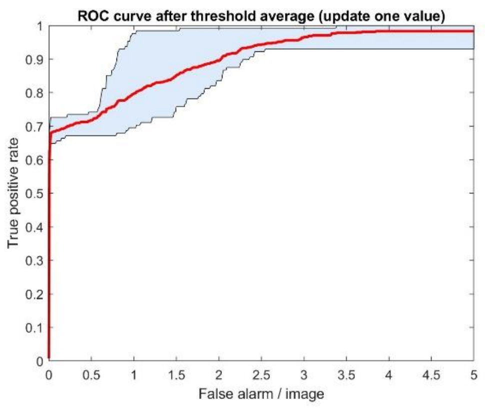


(c) Original ROC curves (CNN)

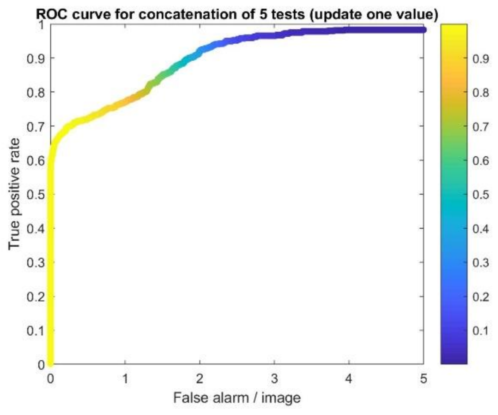


(d) Zoomed in ROC curves (CNN)

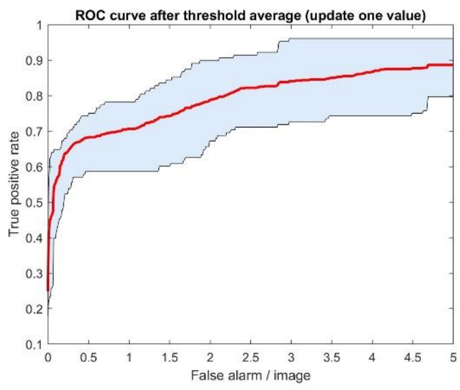
Figure 4.40 Original and zoomed in ROC curves of two-layer MSNN and CNN for dataset III.



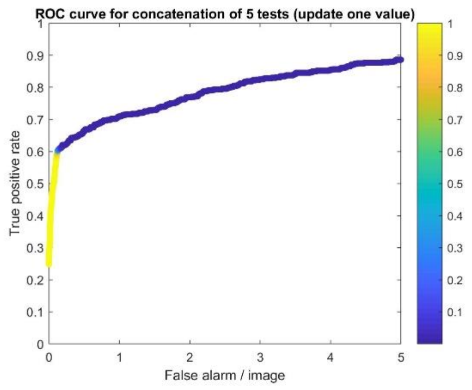
(a) Vertical average ROC curve (MSNN)



(b) Concatenated ROC curve (MSNN)



(c) Vertical average ROC curve (CNN)



(d) Concatenated ROC curve (MSNN)

Figure 4.41 Vertical average and concatenated ROC curves of two-layer MSNN and CNN for Dataset III.

Figures 4.40 and 4.41 show results of two-layer MSNN and CNN for Dataset III. From Figure 4.41 (a) and (b), the TP rate of two-layer MSNN turns from 0.7 to 1 when the FP rate turns from 0 to 5. When the TP rate of a two-layer CNN turns from 0.4 to 0.9. the FP rate turns from 0 to 5. In Figure 4.40 (b), the MSNN color of the threshold changes smoothly; indicating that it is not easy to select a turning point for MSNN. So, let’s look at the requests in practice. When we request 0 alarm per image and a higher TP rate, the two-layer MSNN with a 0.7 TP rate is better than a two-layer

CNN with a 0.45 TP rate. If we need a higher TP rate with an acceptable amount of false alarm, a MSNN is also better than a CNN because with the same amount of false alarms per image, the TP rate of a MSNN is always higher than the TP rate of a CNN. So, for networks with two-layer structures, the MSNN is a better choice for target detection. Examples of detections results of single layer networks trained by MSNN and by CNN are shown in Appendix Figure A-13 and Appendix Figure A-14.

CHAPTER 5. CONCLUSIONS

Among the three update approaches used to train one-layer MSNNs, the performance of networks trained by EAPB was the best. Networks trained by MA had the worst performances. Networks trained by BP had the most stable performances.

Based on comparisons of one-layer MSNNs and two-layer MSNNs, a two-layer MSNN trained by BP has a better performance if no false alarm per image is requested. However, if a few false alarms can be accepted, a one-layer MSNN trained by EAPB would be better. Although a one-layer MSNN trained by EAPB performs well, it is a surprise when a two-layer MSNN trained by EAPB has a highly unsatisfactory performance.

With three groups of datasets, when test images have the same distances as training images, a MSNN can detect all targets correctly with no false alarms, while a CNN can detect 80% of targets with no false alarm. The CNN's worst performance in this research was on test images with occlusion situations among all the datasets tested. Notably, the MSNN had a good ability to detect the target from its test image even when the target was occluded by other items. Both MSNN and CNN can detect targets from images at different distances from the camera. MSNN performed better than CNN for Dataset III, and MSNN had a higher ability to detect a target from an image which was at an overly long distance from the camera.

Although a convolutional neural network is a popular and widely used neural network for image processing, there are lots of new networks that challenge it. The morphological shared-weight neural network introduced in this thesis is a potential candidate to beat CNNs. According to experiments in this thesis, with a limited amount of training data, (eight training images in all the experiments), the performance of the MSNN was much better than the CNN, without consideration of one-layer or two-layer networks. The reason for MSNN's superior performance could be that the MSNN is not only exploitive but also exploratory in its search methods.

Like all things in the real world, MSNN has both a good and bad side. One disadvantage of MSNN is that when training MSNN with EAPB, it takes longer than when training with BP because of the MSNN's exploratory search. If there is a way to shorten the EAPB training time of MSNN without sacrificing other advantages, the exploratory search's drag on timing would be an interesting problem to solve in future research. As for multiple-layer networks, this author sees a need for a more appropriate MSNN training methods. This can be done through changing initialization approaches and variation operators. One research avenue would be to add more layers and kernels to each layer. Making MSNN deeper is another possibility for future research.

REFERENCES

[1] Alpaydin, E. (2014). Introduction to machine learning. Cambridge (USA): MIT Press.

[2] Mohri, M., Rostamizadeh, A., & T,alwalkar, A. (2012). Foundations of machine learning. MIT Press.

[3] Grenander, U. (1998). Foundations of Object Detection and Recognition. doi:10.21236/ada352287

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. doi:10.1145/3065386

[5] Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing. Proceedings of the 25th international conference on Machine learning - ICML 08. doi:10.1145/1390156.1390177

[6] Won, Y., Gader, P., & Coffield, P. (1997). Morphological shared-weight networks with applications to automatic target recognition. IEEE Transactions on Neural Networks, 8(5), 1195-1203. doi:10.1109/72.623220

[7] Khabou, M., Gader, P., & Keller, J. (n.d.). Morphological shared-weight neural networks: a tool for automatic target recognition beyond the visible spectrum. Proceedings IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications (CVBVS99). doi:10.1109/cvbvs.1999.781099

[8] Werbos, P. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560. doi:10.1109/5.58337

[9] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115-133. doi:10.1007/bf02478259

[10] Hebb, Donald (1949). The Organization of Behavior. New York: Wiley. ISBN 978-1-135-63190-1.

[11] Farley, B. W. A. C., & Clark, W. (1954). Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, *4*(4), 76-84.

[12] Minsky, M., Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press. ISBN 0-262-63022-2.

[13] Werbos, P.J. (1975). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.

[14] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[15] Keller, J. M., Fogel, D. B., & Liu, D. (2016). *Fundamentals of computational intelligence: neural networks, fuzzy systems and evolutionary computation*. Hoboken: John Wiley & Sons.

[16] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

[17] Kelley, Henry J. (1960). "Gradient theory of optimal flight paths". Ars Journal. 30 (10): 947–954. doi:10.2514/8.5282.

[18] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533.

[19] Haykin, S. S. (2011). Neural networks and learning machines. New Dehli: PHI Learning.

[20] Serra, J. (1982). Image analysis and mathematical morphology. London: Academic Press.

[21] Won, Y. (1995). Nonlinear correlation filter and morphology neural networks for image pattern and automatic target recognition (Doctoral dissertation, University of Missouri - Columbia)

[22] Shen, S. (2017). Multi-scale target detection based on morphological shared-weight neural network (Thesis, University of Missouri - Columbia).

[23] Haralick, R. M., Sternberg, S. R., & Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, (4), 532-550.

[24] Back, T., Hammel, U., & Schwefel, H. P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation*, *1*(1), 3-17.

[25] Bremermann, H. J. (1962). Optimization through evolution and recombination. *Self-organizing systems*, *93*, 106.

[26] Friedberg, R. M. (1958). A learning machine: Part I. *IBM Journal of Research and Development*, *2*(1), 2-13.

[27] Friedberg, R. M., Dunham, B., & North, J. H. (1959). A Learning Machine: Part II. *IBM Journal of Research and Development*, *3*(3), 282-287.

[28] Box, G. E. (1957). Evolutionary operation: A method for increasing industrial productivity. *Applied statistics*, 81-101.

[29] Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, *9*(3), 297-314.

[30] Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: University of Michigan Press.

[31] Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In *Pattern-directed inference systems* (pp. 313-329).

[32] De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, Univ. of Michigan, Ann Arbor.

[33] De Jong, K. A. (1987). On using genetic algorithms to search program spaces. in *Proc. 2nd Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum, pp. 210–216.

[34] Goldberg, D. E. (1985, July). Genetic algorithms and rule learning in dynamic system control. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 8-15).

[35] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*.

[36] Davis, L. (1991). Handbook of genetic algorithms.

[37] Fogel, Lawrence J. (1962). Autonomous Automata. *Industrial Research*. **4** (2): 14–19.

[38] Fogel, L.J. (1964) On the Organization of Intellect. Ph.D. Thesis, University of California, Los Angeles.

[39] Burgin, G. (1969). On playing two-person zero-sum games against nonminimax players. *IEEE Transactions on Systems Science and Cybernetics*, *5*(4), 369-370.

[40] Burgin, G. H. (1973). Systems Identification by Quasilinearization and by Evolutionary Programming. *J. Cybern.*, vol. 3, no. 2, pp. 56–75, 1973.

[41] Atmar III, J. W. (1976). Speculation on the evolution of intelligence and its possible realization in machine form. Ph.D. dissertation, New Mexico State Univ., Las Cruces, 1976.

[42] Rechenberg, I. (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog.

[43] Rechenberg, I. (1994) *Evolutionsstrategie '94, in Werkstatt Bionik und Evolutionstechnik*. Stuttgart, Germany: Frommann-Holzboog , vol. 1.

[44] Schwefel, H. (1975) *Evolutionsstrategie und numerische Optimierung* Dissertation. Technische Universit¨at Berlin, Germany.

[45] Schwefel, H.-P. (1995) *Evolution and Optimum Seeking*. New York: Wiley, (Sixth-Generation Computer Technology Series).

[46] Grefenstette, J. J. Ed. (1985) *Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum.

[47] Whitley, L. D., Ed. (1993) *Foundations of Genetic Algorithms 2.* San Mateo, CA: Morgan Kaufmann.

[48] Fogel, D. B. and Atmar, W. Eds. (1992) *Proc 1st Annu. Conf. on Evolutionary Programming*. San Diego, CA: Evolutionary Programming Society.

[49] De Oliveira, H. C. B., Alexandrino, J. L., & De Souza, M. M. (2006, December). Memetic and genetic algorithms: A comparison among different approaches to solve vehicle routing problem with time windows. In *Hybrid Intelligent Systems, HIS'06. Sixth International Conference on* (pp. 55-55). IEEE.

[50] Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*.

[51] Fogel, D. B., & Ghozeil, A. (1997). A note on representations and variation operators. *IEEE Transactions on Evolutionary Computation*, *1*(2), 159-161.

[52] Kazimipour, B., Li, X., & Qin, A. K. (2013, June). Initialization methods for large scale global optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (pp. 2750-2757). IEEE.

[53] Burke, E. K., Newall, J. P., & Weare, R. F. (1998). Initialization strategies and diversity in evolutionary timetabling. *Evolutionary computation*, *6*(1), 81-103.

[54] Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, *8*(1), 98-113.

[55] What is visual recognition? Retrieved February 16, 2018, from https://www.clarifai.com/technology

[56] Convolutional Neural Networks (LeNet). Retrieved February 16, 2018, from http://deeplearning.net/tutorial/lenet.html

[57] An Intuitive Explanation of Convolutional Neural Networks. Retrieved February 16, 2018, from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[58] Wilson, S. S. (1989, November). Morphological networks. In *Visual Communications and Image Processing IV* (Vol. 1199, pp. 483-496). International Society for Optics and Photonics.

[59] Davidson, J. L., & Ritter, G. X. (1990, July). Theory of morphological neural networks. In *Digital Optical Computing II* (Vol. 1215, pp. 378-389). International Society for Optics and Photonics.

[60] Davidson, J. L., & Hummer, F. (1993). Morphology neural networks: An introduction with applications. *Circuits, Systems and Signal Processing*, *12*(2), 177-210.

[61] Davidson, J. L. (1992, June). Simulated annealing and morphology neural networks. In *Image Algebra and Morphological Image Processing III* (Vol. 1769, pp. 119-128). International Society for Optics and Photonics.

[62] Jin, X., & Davis, C. H. (2007). Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks. *Image and Vision Computing*, *25*(9), 1422-1431.

[63] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, February). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147).

[64] Yam, J. Y., & Chow, T. W. (2000). A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, *30*(1-4), 219-232.

[65] Kumar, S. K. (2017). On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*.

APPENDIX

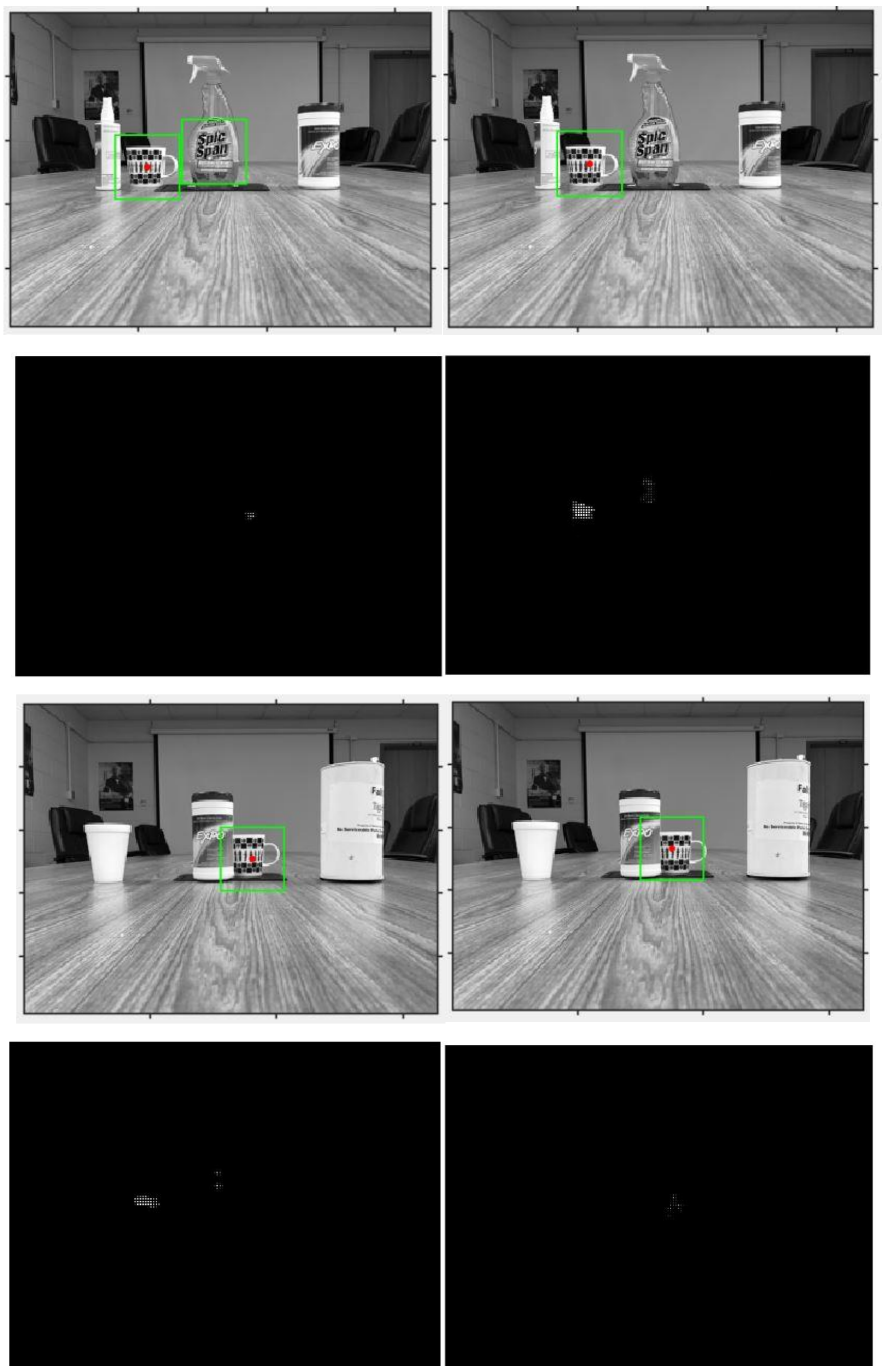


Figure A-1 Samples of detection results of two loss functions (Dataset II). The left column is from the cross-entropy function, and the right column is from the sqare error function.

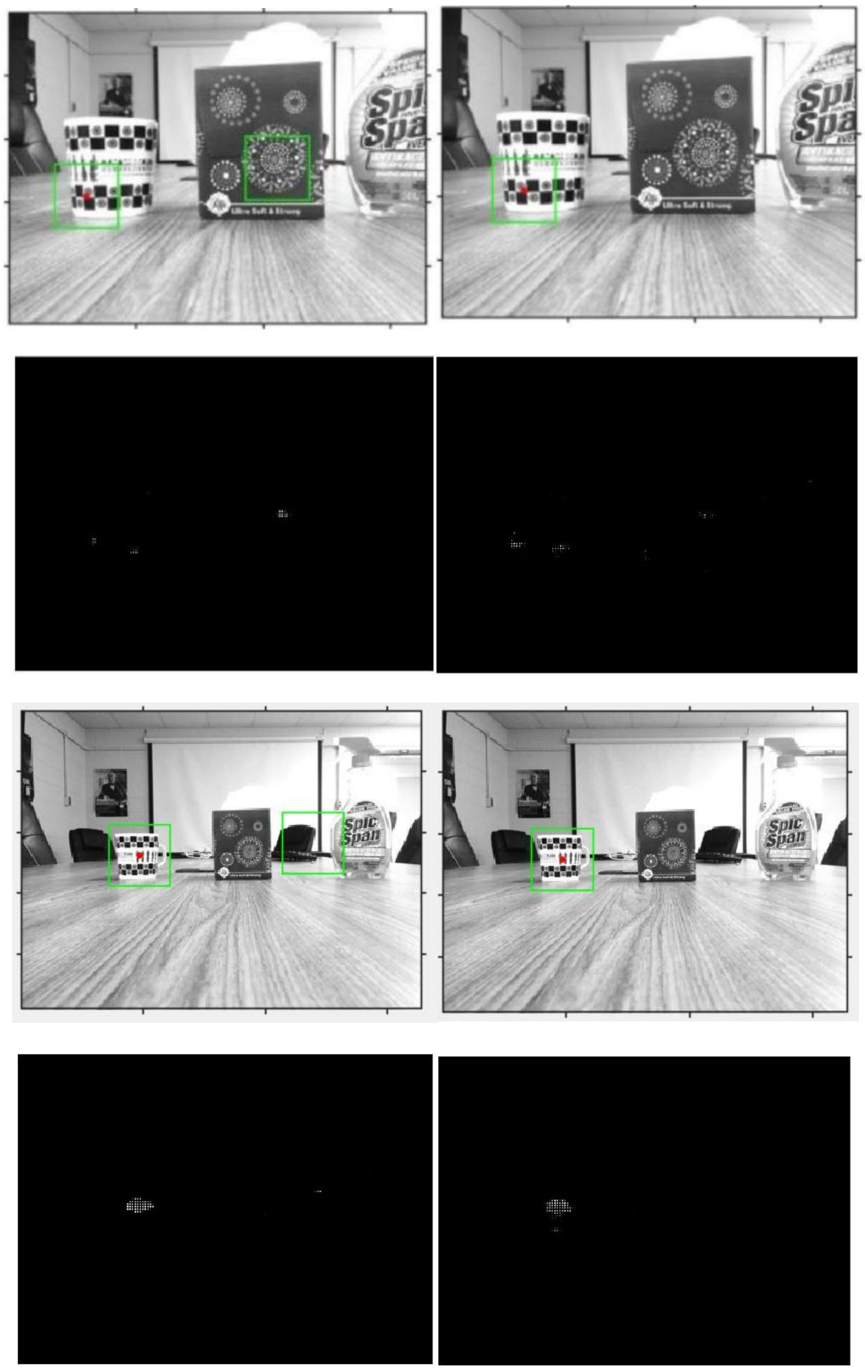


Figure A-2 Samples of detection results of two loss functions (Dataset III). The left column is from the cross-entropy function, and the right column is from the sqare error function.

Figure A-1 and Figure A-2 show detection results of the experiment in section 4.2.1, which is the comparison of loss functions. Figure A-1 represents results of two randomly seleted test images in Dataset II, including output images with bounding boxes and detection planes. Figure A-2 represents results of Dataset III. The left column images are results of the network using cross-entropy loss function and the right column images are results of the network using square error loss function.

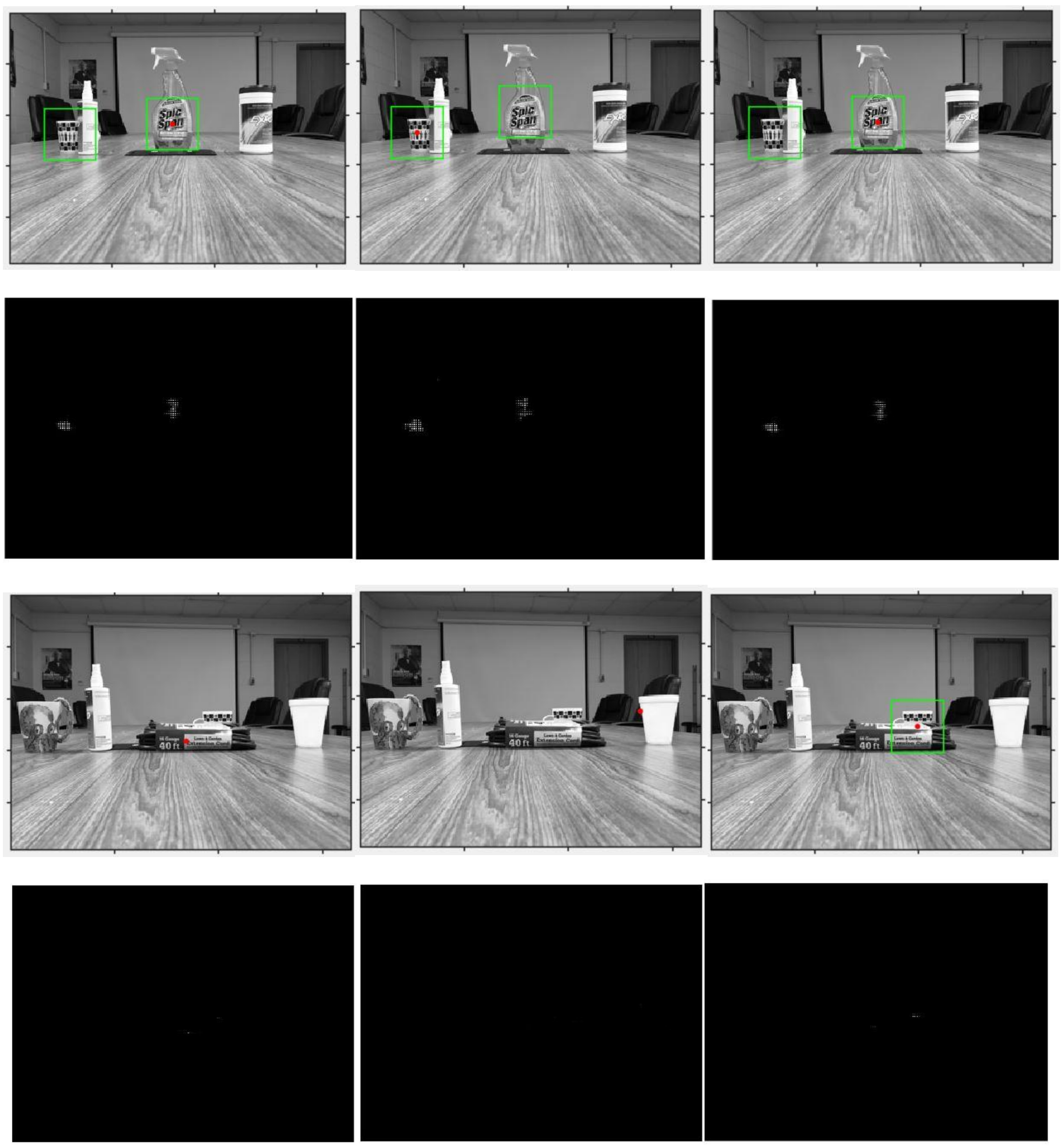


Figure A-3 Samples of detection results of three mutation functions (Dataset II).

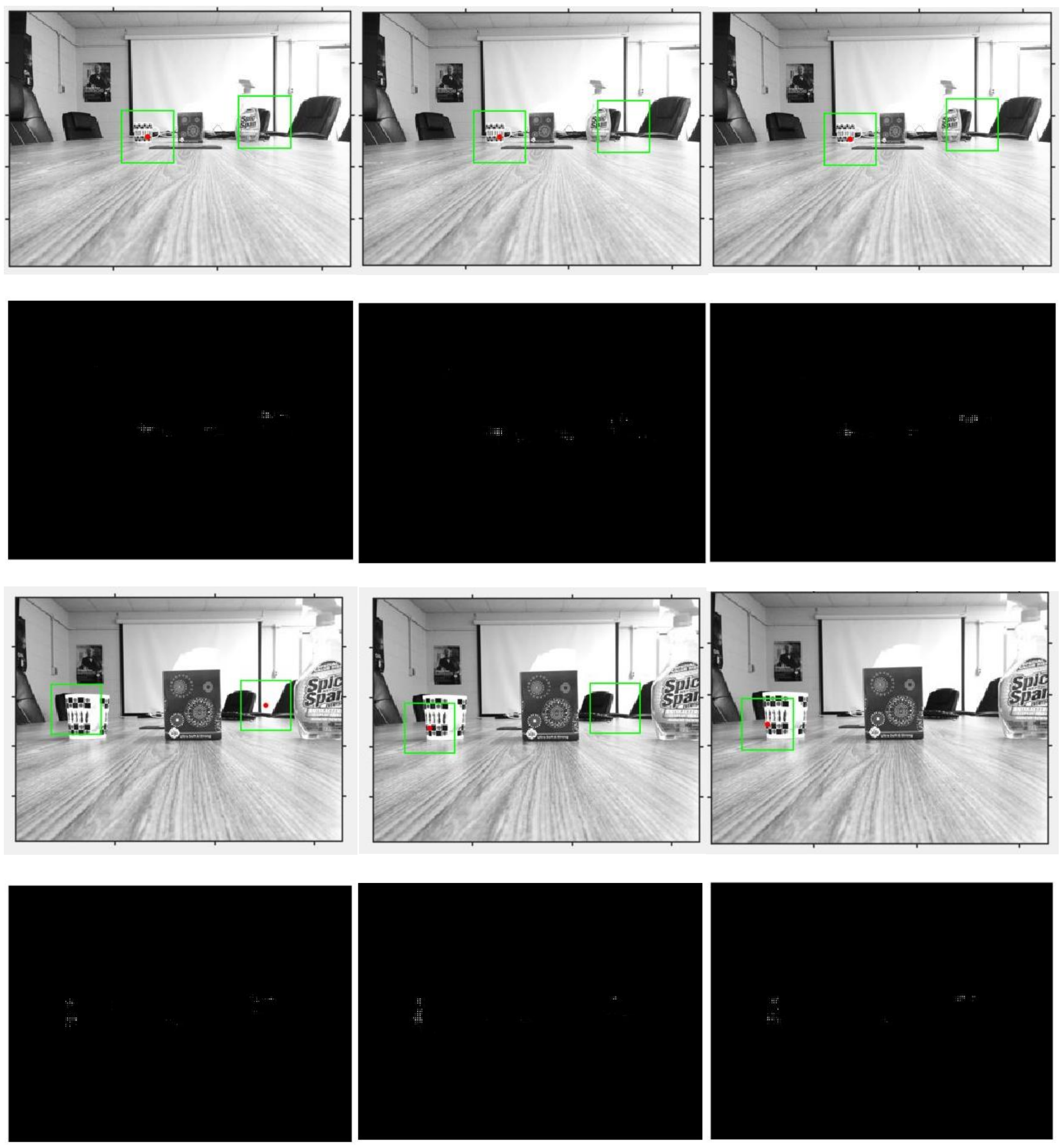
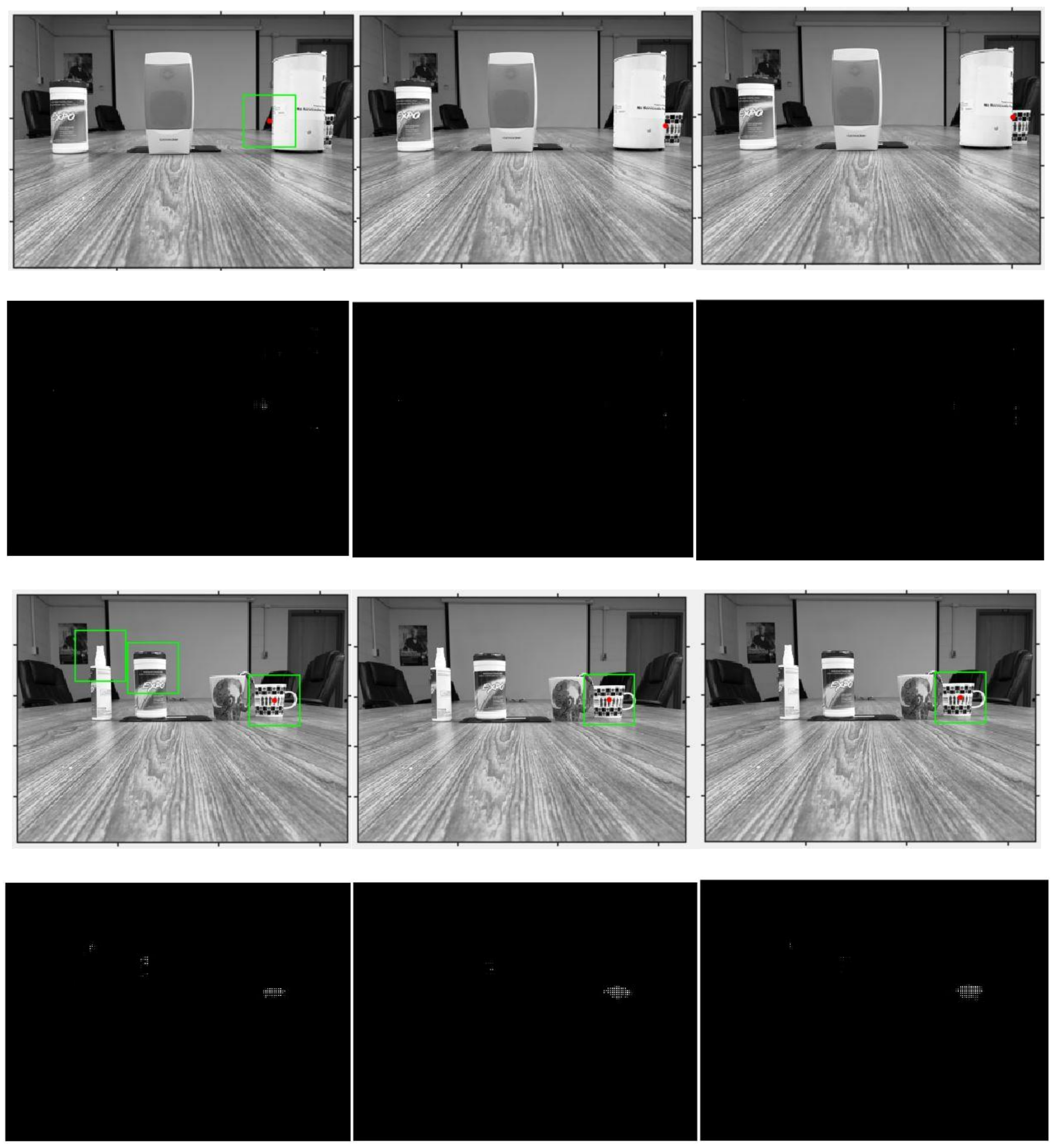


Figure A-4 Samples of detection results of three mutation functions (Dataset III).

Figure A-3 and Figure A-4 show detection results of the experiment in section 4.2.2, which is the comparison of mutation methods. Figure A-3 represents results of two randomly seleted test images in Dataset II, including output images with bounding boxes and detection planes. Figure A-4 represents results of Dataset III.



Initialization 1 Initialization 2 Initialization 3

Figure A-5 Samples of detection results of three initialization methods (Dataset II).

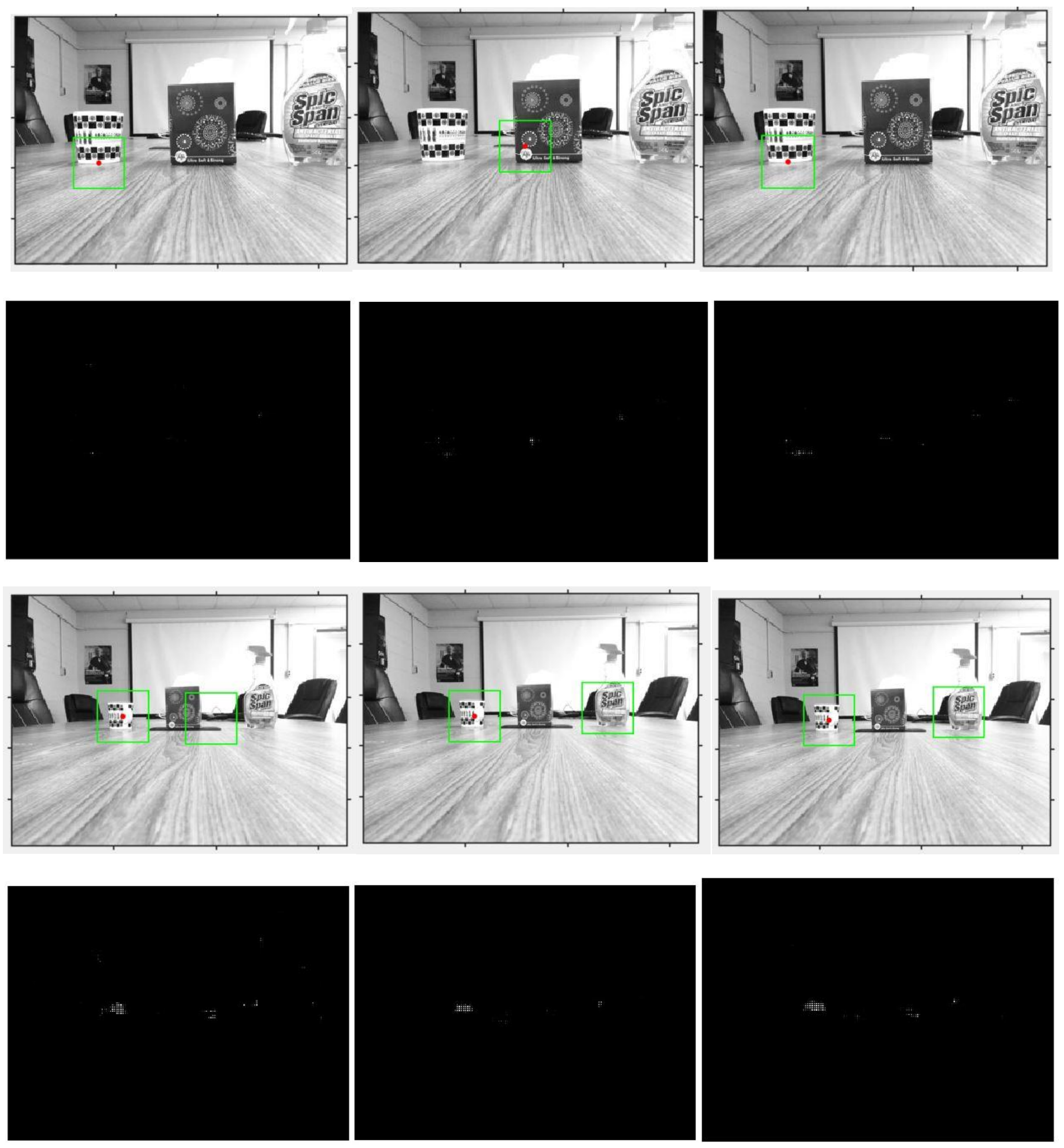


Figure A-6 Samples of detection results of three initialization methods (Dataset III).

Figure A-5 and Figure A-6 show detection results of the experiment in section 4.2.3, which is the comparison of initialization methods. Figure A-5 represents results of two randomly seleted test images in Dataset II, including output images with bounding boxes and detection planes. Figure A-6 represents results of Dataset III.

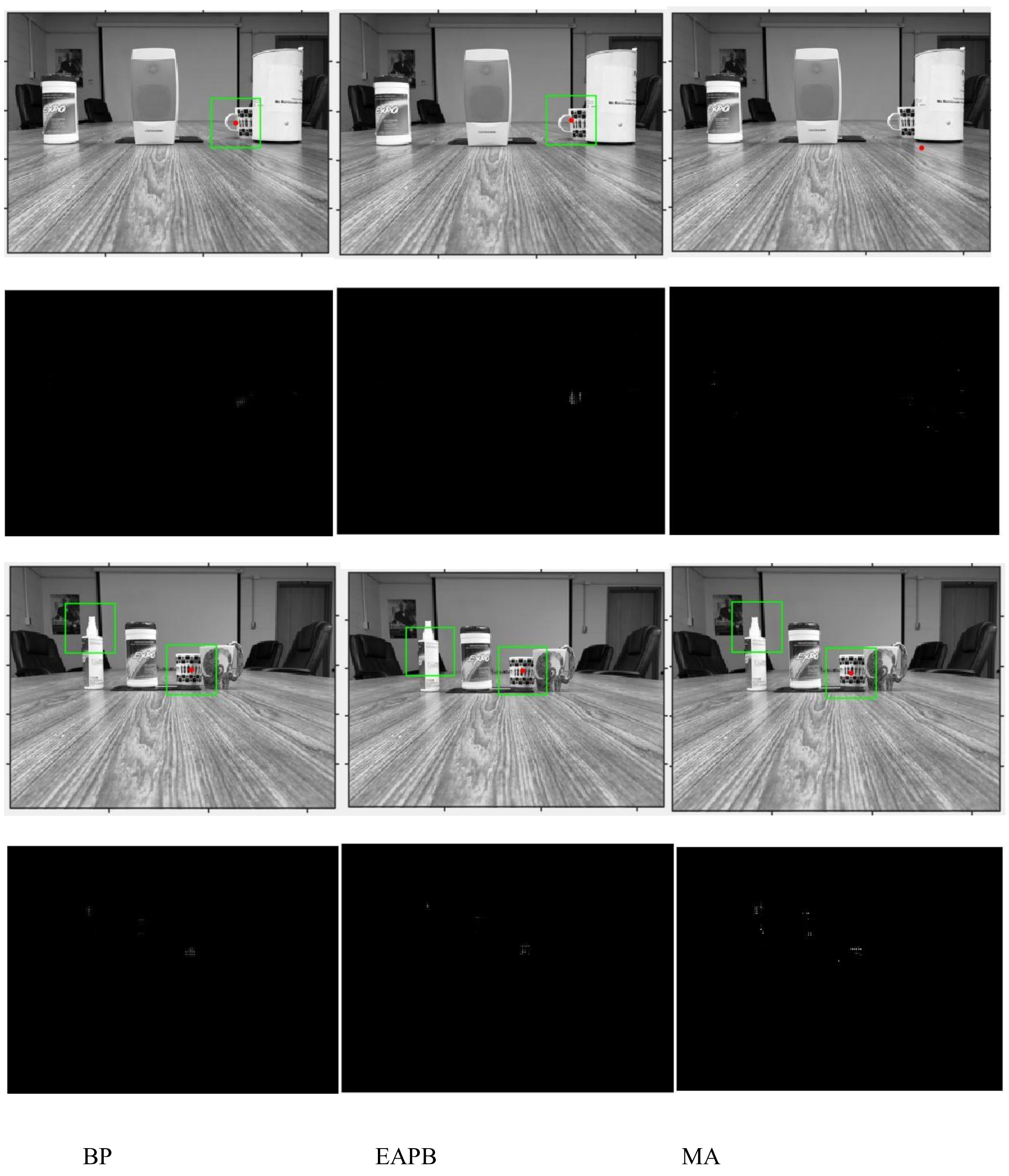


Figure A-7 Samples of detection results of three update approaches (Dataset II).

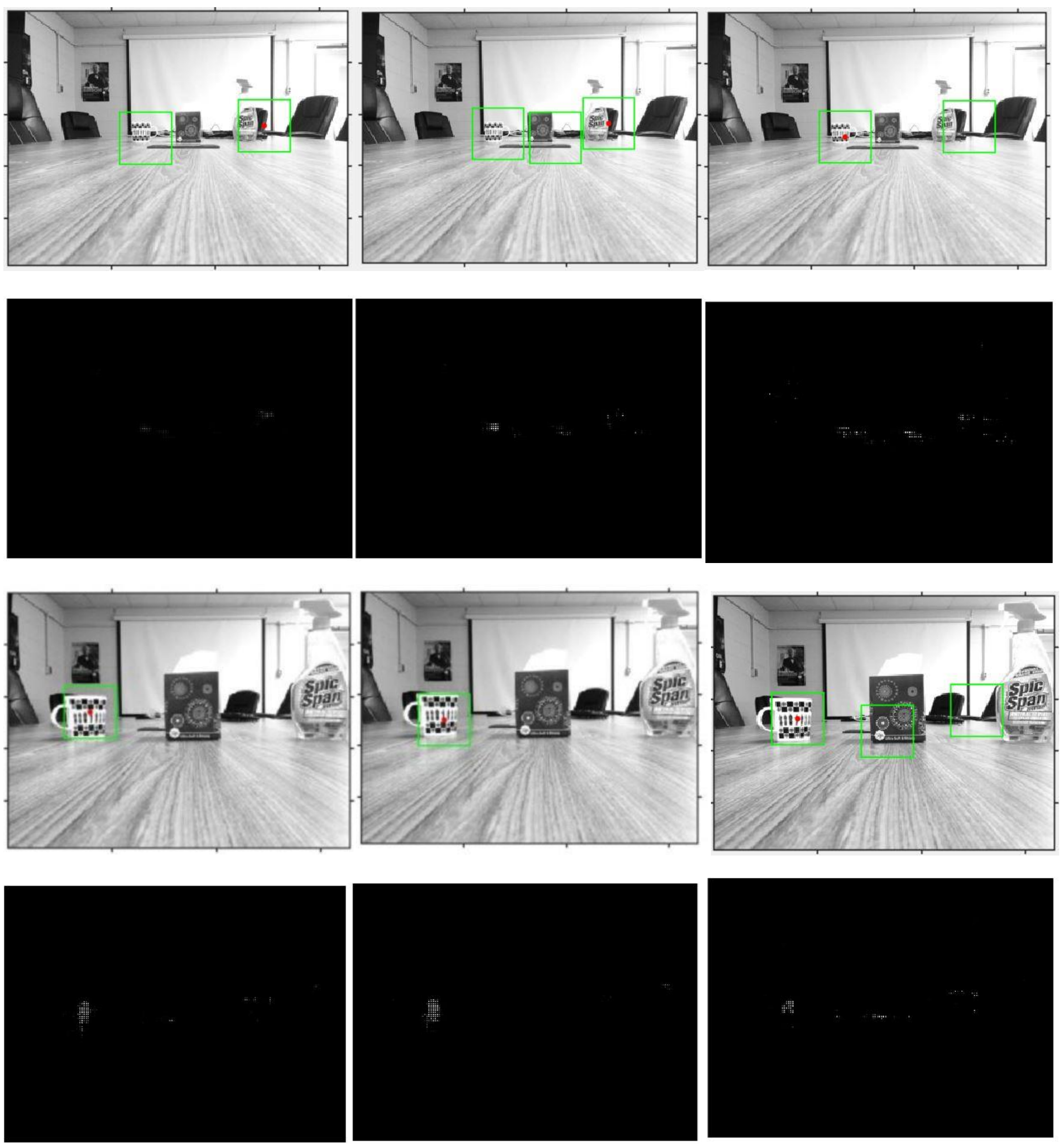


Figure A-8 Samples of detection results of three update approaches (Dataset III).

Figure A-7 and Figure A-8 show detection results of the experiment in section 4.2.4, which is the comparison of three update approaches. Figure A-7 represents results of two randomly seleted test images in Dataset II. Figure A-8 represents results of Dataset III.

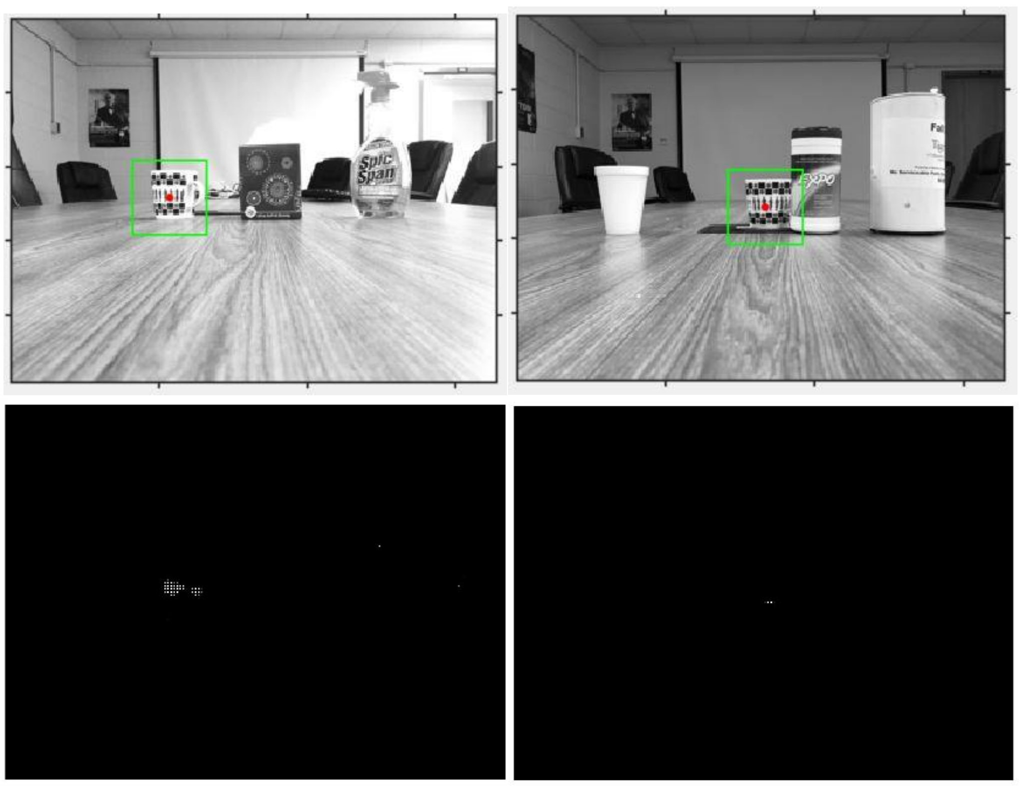


Figure A-9 Samples of detection results of parallel networks.

Figure A-9 shows detection results of the experiment in section 4.2.5, which is the results of parallel networks. It represents results of one randomly seleted test image in Dataset II and another randomly seletcted test image in Dataset III.

Figure A-10 shows detection results of the experiment in section 4.3, which is the results of multiple morphological layer networks trained by BP and trained by EAPB. The top two rows represent results of one randomly seleted test image in Dataset III and the bottom two rows are results of one randomly seletcted test image in Dataset II.

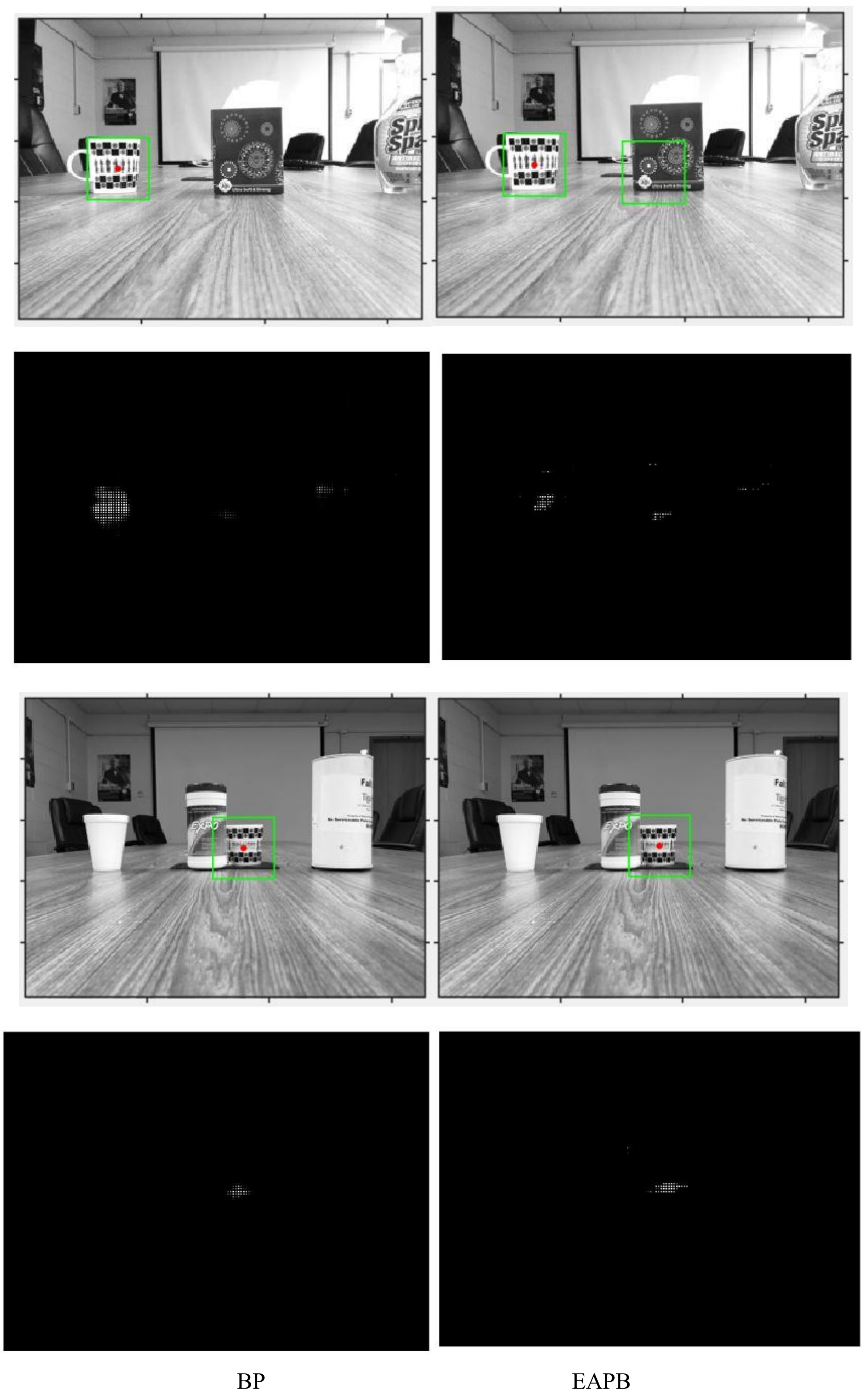


Figure A-10 Samples of detection results of multiple morphological layer networks.

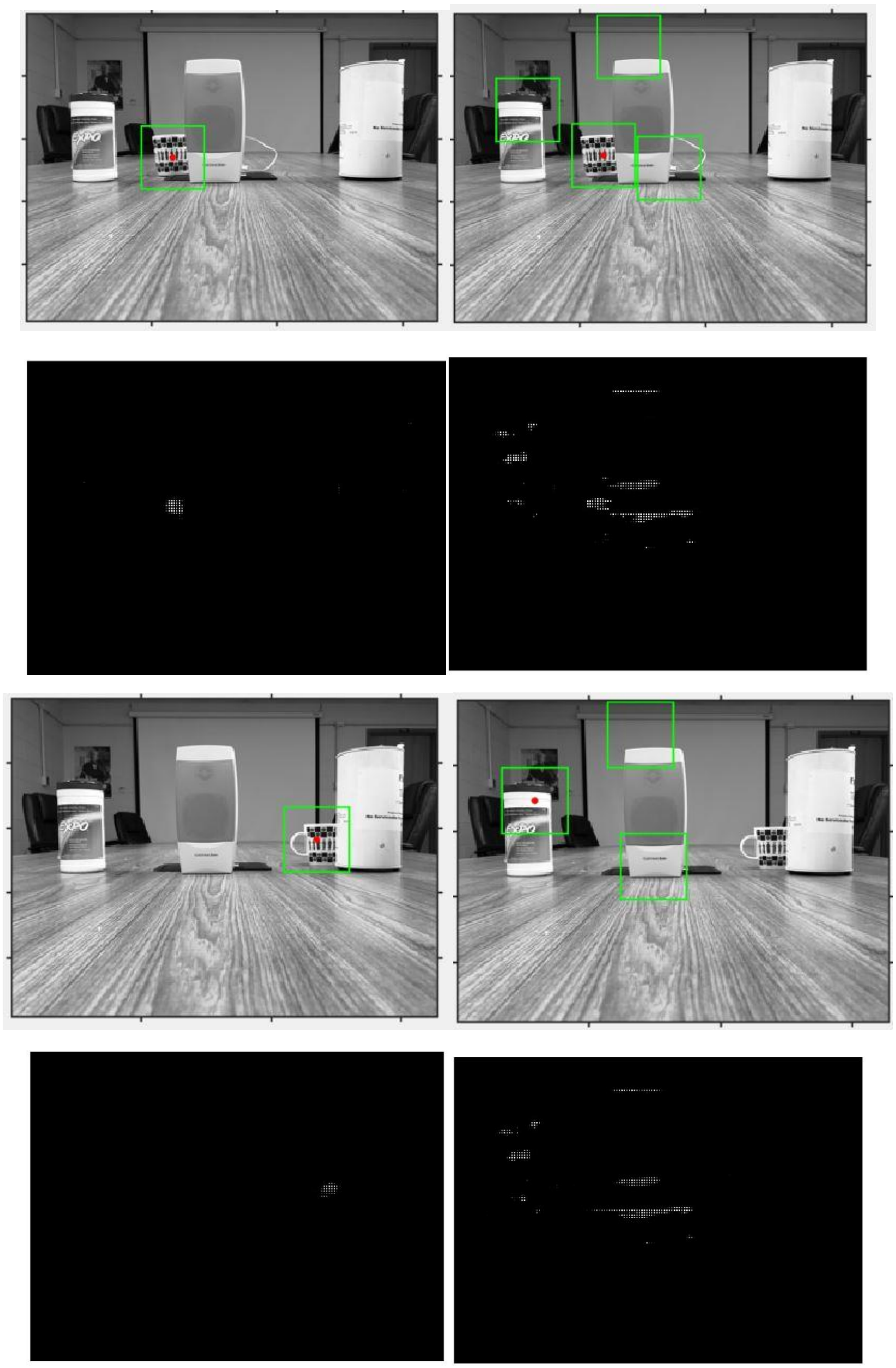


Figure A-11 Samples of detection results of single morphological layer networks trained by MSNN and CNN (Dataset II).

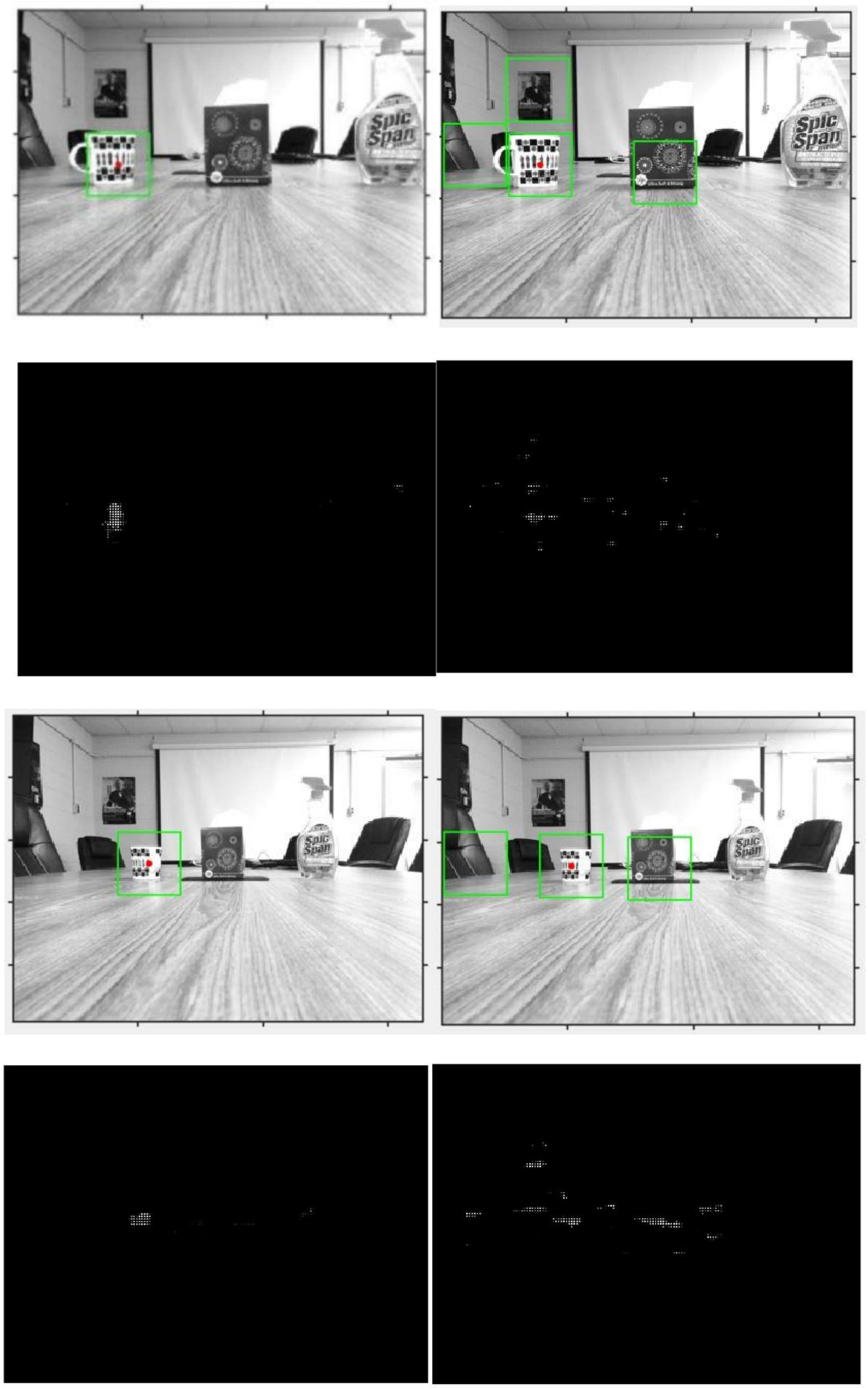


Figure A-12 Samples of detection results of single morphological layer networks trained by MSNN and CNN (Dataset III).

Figure A-11 and Figure A-12 show detection results of the experiment in section 4.5.1, which is the comparisons of single morphological layer networks trained by EAPB and trained by CNN. Figure A-11 represents results of two randomly seleted test images in Dataset II. Figure A-12 represents results of Dataset III.

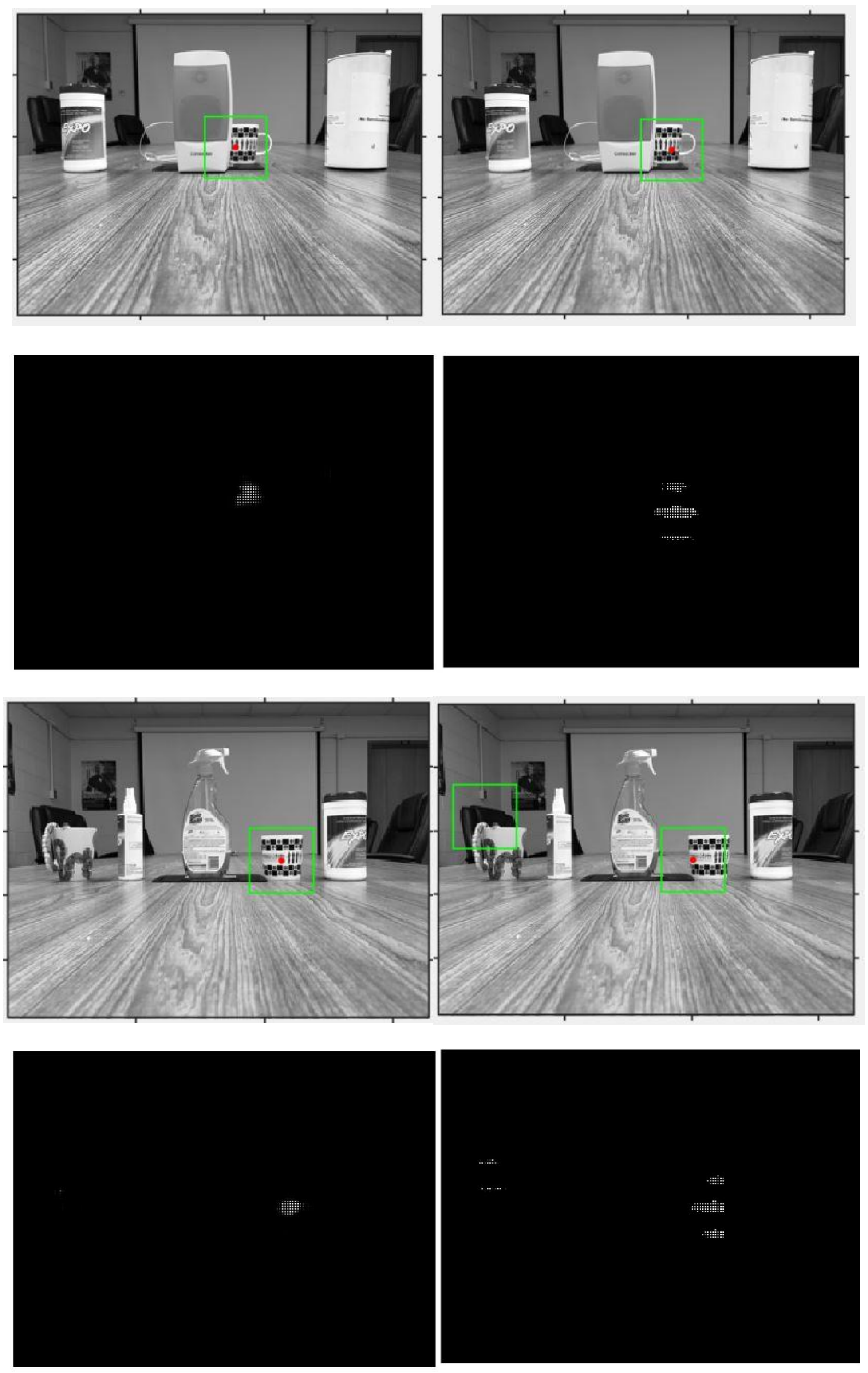


Figure A-13 Samples of detection results of two morphological layer networks trained by MSNN and CNN (Dataset II).

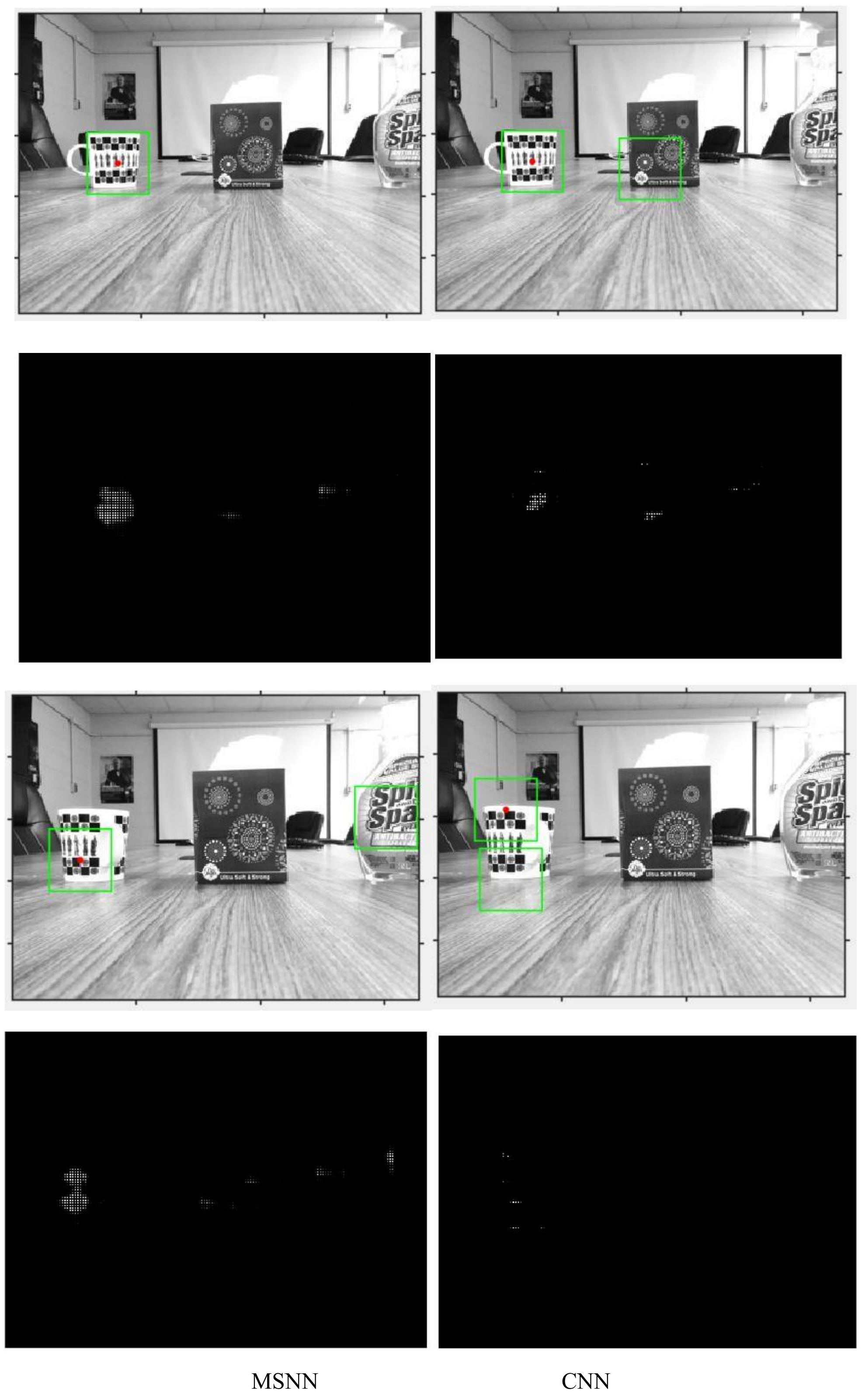


Figure A-14 Samples of detection results of two morphological layer networks trained by MSNN and CNN (Dataset III).

Figure A-13 and Figure A-14 show detection results of the experiment in section 4.5.2, which is the comparisons of two morphological layer networks trained by EAPB and trained by CNN. Figure A-13 represents results of two randomly seleted test images in Dataset II. Figure A-14 represents results of Dataset III.