Thai Banknote Recognition Using Neural Network

1055119　　SAKOOBUNTHU, Lalita　　【Takeda Lab.】

1. Introduction

Recently, neural network (NN) algorithm is widely used for pattern recognition because of its abilities of self-organization, parallel processing and generalization [3]. With these 3 abilities, the NN can recognize patterns effectively and robustly. Therefore, the NN has been applied to invent a new recognition system and its hardware using a DSP unit for banknote reader and sorter machines. In recognition processes, there are 3 important steps that are (i) a slab values extraction, (ii) the NN system, and (iii) the DSP application.

In our previous research [3], this proposed system had been applied for various kinds of worldwide banknote such as Japanese Yen, US Dollar and Euro banknote. In this research, a new kind of banknote, Thai banknote (Baht = B) is proposed as the object of recognition. Since, the recognized banknote target is different from the previous research. Therefore, new threshold values for edges detection and a suitable mask set which used for the slab values extraction must be selected again. Generally, Thai banknote has 20 patterns (5 types × 4 conveyed directions). These 4 conveyed directions are head upright, head reversed, tail upright, and tail reversed. However, in this research, an axis-symmetry mask set has been applied. Then, the total number of patterns, which are recognized, is decreased to 10. Next section, the recognition system overview, the slab values extraction, and the NN system are briefly described. Then, the experimental results are presented. Finally, the DSP application of this system is explained.

2. Proposed Recognition System

This recognition system consists of preprocesses, the NN system, and the DSP unit as shown in Figure 1. The first part is preprocesses which performed on PC. On the slab values extraction, the slab values, which represented as characteristics of banknote, are extracted from each banknote image by using an axis-symmetry mask set. Then, all slab values are inputted to the NN system for learning and recognition. To confirm the recognition ability of the system before applied for real banking machines, the NN system is firstly performed on PC. Finally, when yield the effective recognition system, all banknote images, an axis-symmetry mask set, and PC converged NN weights are transferred to the DSP unit to execute the NN learning and recognition for the real world system. In this section, all preprocesses and the NN system are described.

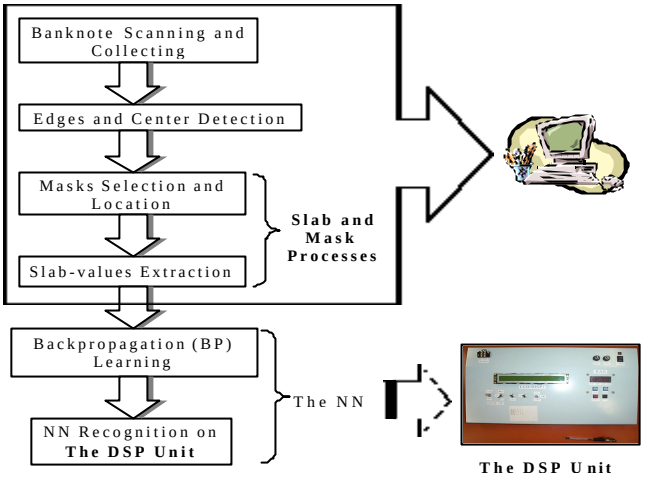


Figure 1　The Recognition System Overview.

2.1. Data Collecting and Edges Detection

The proposed recognition system classifies Thai banknote types from banknote images. Therefore, Thai banknote images are collected as training data for the NN system. To begin, the banknote image is collected by scanner and saved as bit map data. Next, each bit map data is transformed to the NN data format and saved into PC for applying as input to next processes.

Since, a mask set is applied for the slab values extraction, so edges and center of banknote images must be detected to locate the mask. For this detection, 2 threshold values, threshold "cho" and "tan", are manually selected until yield suitable values for all Thai banknote types. These threshold values are offsets of gray scale level projection in long and short side of the banknote. The first and last column (or line) that its gray scale level is more than the threshold values is decided as the edge of banknote image. Finally, banknote center is automatically detected.

2.2. Slabs Values and Axis-symmetry Masks

In this research, the characteristics of each banknote image are slab values that extracted by using an axis-symmetry mask set. This kind of mask set consists of 50 axis-symmetry mask patterns. Thus, each banknote image has 50 slab values and each slab value is summation of non-masked pixel values of each mask pattern. Using the slab values, the NN scale is reduced, which is useful for commercial products, because each pixel value of banknote image is not necessary inputted to the NN. To yield the slab values, firstly, the mask set is manually selected.

Second, center of each mask pattern is located at the same position as the banknote center as shown in Figure 2. Finally, all slab values are extracted and save as database on PC. As a result of using the axis-symmetry mask set, 2 conveyed directions of banknote image are recognized simultaneously (Figure 2). Therefore, total number of recognized patterns for Thai banknote is only 10. Namely, 5 types × 2 conveyed directions (head and tail).

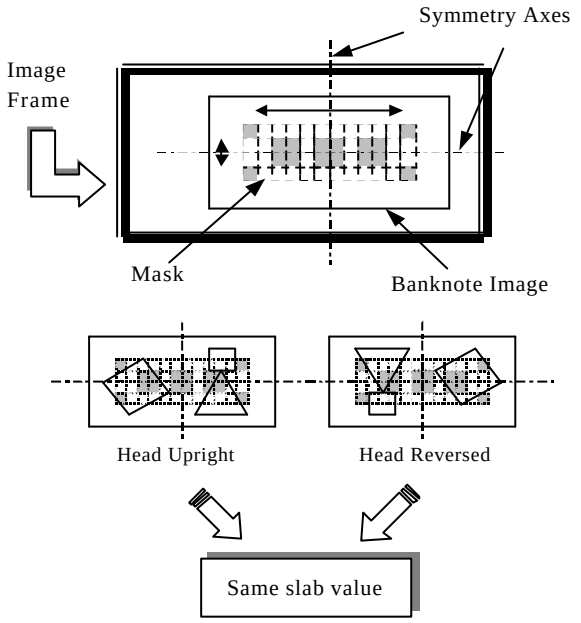


Figure 2　Axis-symmetry Mask and Slab Values.

2.3. Neural Network

In this research, the NN system has 3 layers, input, hidden, and output layer. Table 1 shows the NN structure and learning conditions. Since, one mask set has 50 mask patterns then input nodes number of the input layer is also 50. And the total number of banknote patterns that are recognized is 10 that equal to output nodes number of the output layer. For the NN learning, 20 pieces of each banknote pattern are applied as the training data and its learning algorithm is the backpropagation method. The training data are learned to adjust the NN weights until the error will be reduced below the minimum mean squared error or the iteration number will be reached the maximum iteration number.

Table 1　The NN Structure and Learning Conditions.

Mask no.	50
No. of banknote types	10
Hidden node no.	30
Learning data / pattern	20
Testing data / pattern	80
Mean square error	0.0001
Maximum iteration no.	20000
Learning algorithm	Backpropagation method

3. Experimental Results

After the NN learning process is completed, the converged NN weights and the mask set are saved in PC and then 80 pieces of banknote image per patterns are tested on PC to evaluate recognition ability by using those NN weights. A recognition ability of all banknote patterns is 100% excepting the 50B head and the 100B head, which their abilities are 95.56% and 98.89%, respectively. Moreover, there is still problem on reliability of the system because of fluctuation of the output response of some patterns. These fluctuations are mainly affected by 2 factors that are (i) the mask positions and (ii) the threshold values for edges and center detection.

4. DSP Application

To apply this recognition system for real banking system, the DSP unit is proposed. After the system ability was confirmed by testing on PC, all banknote images, the mask set, and PC converged NN weights are transferred to the DSP unit. The DSP unit starts to execute the NN learning again to confirm the converged NN weights and then the recognition process is executed. The NN learning and recognition are performed by NN calculations which saved on flash memory of the DSP unit. By performing the experiment, it seems that the ability of recognition system on the DSP unit is same as the recognition ability of PC experiment.

5. Conclusion

The NN recognition system with the axis-symmetry mask set for Thai banknote has been proposed and its recognition ability was confirmed by evaluating several pieces of banknotes on both PC and the DSP unit. However, there are some problem in the system reliability because of the output fluctuation by mask set and threshold values. Therefore, in the future, we will improve the threshold-values selection method and apply a genetic algorithm (GA) for mask selection to improve the reliability of this recognition system.

References

[1] Sakoobunthu, L., Takeda, F., and Sato, H., "Thai Banknote Recognition Using Neural Network and Applied to the Neuro-Recognition Board", The International Workshop on Signal Processing Application and Technology, pp.107-114, 2002.

[2] Sakoobunthu, L., Takeda, F., and Sato, H., "Online Continuous Learning by DSP Unit", SICE System Integration Division Annual Conference (SI2002), Vol.3, pp. 321-322, 2002.

[3] Takeda, F. and Omatu, S., "High Speed Paper Currency Recognition by Neural Networks", IEEE Trans. on Neural Networks, Vol.6, No.1, pp.73-77, 1995.

平成 14 年度

修士学位論文

ニューラルネットワークを用いたタイ紙幣識別

Thai Banknote Recognition Using Neural Network

1055119　　Lalita SAKOOBUNTHU

指導教員：竹田史章教授

2003 年 1 月 31 日

高知工科大学大学院基盤工学専攻情報システム工科コース

Acknowledgements

I would like to express my profound gratitude to Prof. Takeda Fumiaki for a great opportunity of master study in Kochi University of Technology, especially, his advice and leadership in the neural network laboratory during my study. Additionally, I deeply appreciate to his attempt in every explanation for through knowledge of neural network, image processing, DSP operation and permission to study his intelligent recognition system. Excepting Prof. Takeda Fumiaki, I also would like to thank Mr. Hironobu Sato for his assistance on the recognition experiment of the DSP unit. Finally, I would like to give the special thank to the GLORY Co., Ltd. for their funds of the recognition system development.

Abstract

Nowadays, neural networks (NNs) are widely used in many fields of engineering and the most famous application is pattern recognition. In previous researches of Prof. Takeda, a banknote recognition system using a NN has been developed for various applications in worldwide banking systems such as banknote readers and sorters. In this report, a new kind of banknotes, Thai banknotes, are being proposed as the objects of recognition. There are 5 types of Thai banknotes that are 20B, 50B, 100B, 500B, and 1000B (B = bath) and there are 4 conveyed directions for inserting them, namely head upright, head reversed, tail upright, and tail reversed. So, there are totally 20 patterns for recognition. For the recognition, there are three important steps which are (i) image preprocesses, (ii) a neural algorithm, and (iii) digital signal processing (DSP).

To begin, slab values are collected by applying an axis-symmetry mask set which is one of image preprocesses. These slab values are the digitized characteristics of a banknote. The slab value is equal to the summation of non-masked pixel values of each banknote image. This kind of mask set has symmetry-masked areas for both vertical and horizontal axes of banknote. As a result of using this mask set, both upright and reversed direction of the same banknote can be recognized, simultaneously. Therefore, in this case, the total number of patterns that are recognized is 10. Next, a neural algorithm, slab values are input to the NN to execute its learning process. After the learning process is completed, the mask set and the NN weights are transferred to a DSP unit that consists of a DSP device for recognition in real banking machines. Furthermore, for commercial usability, the NN algorithm is also transferred to this DSP unit in order to execute learning and recognition. Finally, several hundred banknotes samples are analyzed to show efficiency of the proposed system. By following the three steps of the proposed system that have been explained above, the recognition ability of the proposed system was confirmed. It seems possible that this system can be use in worldwide banking systems.

Key words: Neural networks, Digital signal processor, Image preprocesses, Continuous learning, Banknote recognition.

Table of Contents

List of Figures

List of Figures

List of Tables

Chapter 1: Introduction

This report presents "Thai banknote recognition by using neural network" which is one of Takeda laboratory's researches at Kochi University of Technology. This chapter shows purpose and brief description for a proposed system of this research. Moreover, since, Thai banknotes have been proposed as the objects of recognition. Therefore, Thai banknote characteristics are also discussed on this chapter.

1.1 Study purpose

In worldwide banking systems, there are several kinds of banknotes then it takes much time to classify all banknote kinds by human. Thus, automatic banknote readers and sorters machines are needed. For this reason, many recognition techniques such as a discriminative inequality technique and a neural network (NN) technique have been proposed for the reader and sorter machines. Based on several previous research studies, it seems that the NN technique is more suitable and robust for recognition than any other techniques because of its self-organization, parallel processing, and generalization abilities [13]. Therefore, the NN technique has been applied to invent recognition system for banknote readers and sorters machines. However, using only the NN technique for recognition is not effective for real world banking system. Consequently, image preprocessing techniques and a Digital Signal Processor (DSP) implementation also have been applied to improve the recognition ability of such system. Therefore, this recognition system is composed of three core techniques that are image preprocesses, the NN technique, and DSP implementation.

This recognition system has been applied to recognize many kinds of banknotes such as US dollar and Japanese yen. In this report, a new kind of banknotes, Thai banknotes, have been proposed as objects of recognition because there is no machine to recognize Thai banknotes now. Moreover, in Thailand, there are only coin-recognized machines at train stations that are not convenient for ticket buying. Therefore, Thai banknotes recognition machines are needed and it is possible that this research will be helpful for Thai banking system in the near future.

1.2 Thai banknotes

For Thai banknotes (Baht = B), there are 5 types that are 20B, 50B, 100B, 500B, and 1000B. Figure 1.1 shows all types of Thai banknotes in 2 directions (Head and Tail). Each type has same width, 7.2 cm, but has different length and color. For example, 20B is green and 13.8 cm long while 500B is violet and its length is 16.6 cm. Furthermore, there are 4 conveyed directions of each type for inserting them to the banking machine which are head upright, head reversed, tail upright, and tail reversed. Figure 1.2 shows 4 conveyed directions for 20B. Therefore, there are totally 20 patterns for inserting all Thai banknotes to the banking machine.

(a) 20B Head (20HB)

(b) 20B Tail (20TB)

(c) 50B Head (50HB)

(d) 50B Tail (50TB)

(e) 100B Head (100HB)

(f) 100B Tail (100TB)

(g) 500B Head (500HB)

(h) 500B Tail (500TB)

(i) 1000B Head (1000HB)

(j) 1000B Tail (1000TB)

Figure 1.1 5 Types of Thai Banknotes in Head and Tail Direction.

(a) 20B Head Upright

(b) 20B Head Reversed

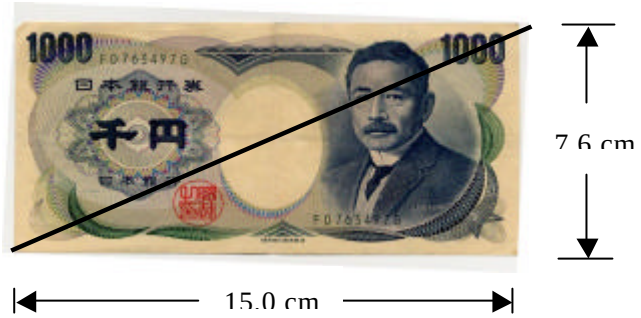
(c) 20B Tail Upright

(d) 20B Tail Reversed

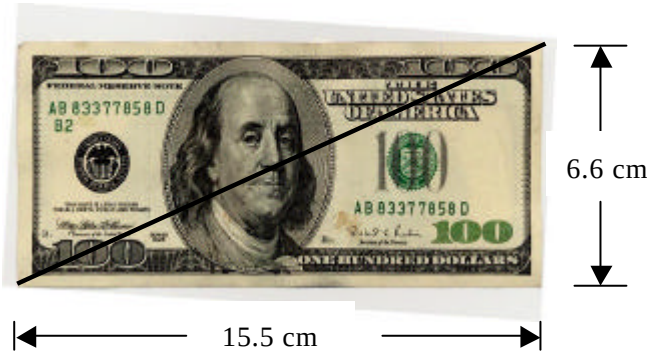
Figure 1.2 4 Conveyed Directions of the 20B Banknote.

Since, this recognition system has been used for US dollar and Japanese yen then differences between those banknotes and Thai banknotes are discussed. There are 2 obviously differences among these 3 banknotes kinds. The first difference is banknote width. While Thai banknotes are 7.2 cm wide, US dollar and Japanese yen width are 6.6 cm and 7.6 cm, respectively. Figure 1.3 shows width comparison among these 3 kinds of banknotes. Another important difference is color of banknote. Thai banknotes color is generally vivid as shown in Figure 1.1. In contrast, color of US dollar and Japanese yen are usually pale when comparing with Thai banknotes such as green, gray, pale blue, and brown as shown in Figure 1.3. By these 2 differences, it concludes that Thai banknotes obviously differ from US dollar and Japanese yen.

(a) Thai Banknote (1000B)



(b) Japanese Yen (¥1000)



(c) US Dollar ($100)

Figure 1.3 Thai Banknote, Japanese Yen, and US Dollar Comparisons.

Chapter 2: Literature Review

Several years ago, many researchers applied various recognition techniques to invent banknote recognition machines such as a discriminative inequalities technique and a NN technique [13]. For the discriminative inequalities technique, it requires discriminative points, which contain characteristics of each banknote, and threshold values to distinguish purpose banknote from other banknotes. This technique requires expert engineering designers to manually determine those required values by trail and error that take much time to design banknote recognition machines. Therefore, the NN technique, which is widely used in several fields of engineering such as pattern recognition, system identification and control problem, has been applied for banknote recognition. The NN is applied for banknote recognition, which is part of pattern recognition, because of its abilities of self-organization, parallel processing, and generalization [13]. By these 3 abilities, the NN can recognize patterns effectively and robustly.

According to previous researches of Prof. Takeda, a three-layers NN has been applied to recognize banknotes by inputting FFT data of banknotes which extracted from 4 sensors to the NN [13]. However, this kind of inputs made a large scale of the NN and then made a scale problem for inventing banknote recognition machines. Therefore, a slab-like architecture [19] and a mask preprocess [13] have been applied to reduce scale of the NN and improve ability of the recognition system, respectively. For the mask preprocess, mask positions are selected randomly [13] and then applied to banknotes image for making a slab value which used as input for the NN. Moreover, excepting the slab-like architecture, the NN scale is also decreased by applying axis-symmetry mask positions [7] instead of random mask positions. According to the axis-symmetry mask positions, the output scale of the NN is twice reduced as will be shown on next chapter. To confirm recognition ability of this system, it has been applied to recognize many banknotes types such as US dollar, Japanese yen, and Euro banknotes [11]-[18]. In this study, this recognition system with the axis-symmetry masks is applied to recognize a new kind of banknotes, Thai banknotes.

Chapter 3: Methodology

In this chapter, the proposed recognition system is described. To begin, overview of the system is presented on the first section of this chapter. Next, each preprocesses are described on other sections. Finally, on the last section, the NN structure and its processes that have been used for this research are explained.

3.1 System overview

Figure 3.1 shows the proposed system configuration. This system consists of 3 main parts, which are preprocesses, the NN, and the DSP unit. For preprocesses, the objective of this part is slab-values extraction. These slab values are characteristics of a banknote image and used as inputs for the NN. Therefore, the first step of preprocesses is banknote images collecting by using scanner. Then, edges and center of all banknote images are detected for masks location. After that an axis-symmetry mask set is applied to each banknote images to extract slab values. All preprocesses are performed on PC before transferring all banknote images and the axis-symmetry mask set to the DSP unit for learning and recognition by the NN. Each preprocess will be described in detail on section 3.2, 3.3, and 3.4 of this chapter.

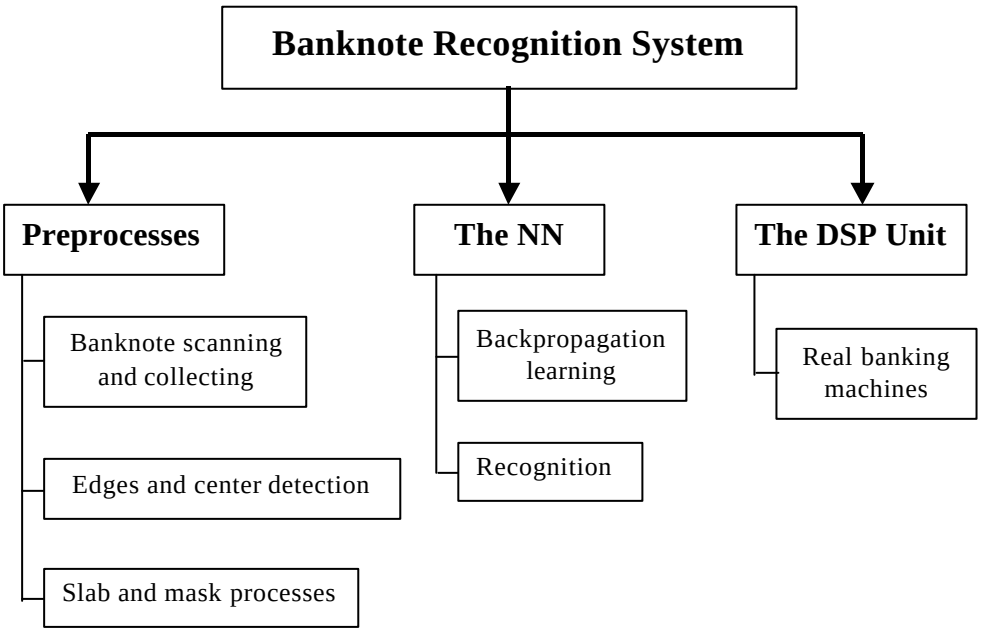


Figure 3.1 Proposed System Configuration.

Next, second part, the NN has 3 layers that are an input layer, a hidden layer, and an output layer which structure of this three-layered NN is 50×30×10, respectively. The number 10 of the output layer is represented a number of banknote patterns that have to be recognized (5 types × 2 conveyed directions). The NN has 2 important functions that are learning and recognition. For NN learning, a Backpropagation (BP) method is applied as a learning algorithm of this research. After finish learning, NN weights, which converged to learning condition, are used for banknote recognition. On the last section of this chapter, the BP method, learning procedure, and the NN recognition will be explained.

Although a system, which use only preprocesses and the NN, can recognize banknotes, it is not sufficient for real banknote recognition machines. Therefore, next part, the DSP unit is applied. This DSP unit can both learn and recognize banknotes by itself because of the NN calculations on its flash memory. This DSP unit will be a main part of the banknote recognition machine. The DSP unit operations will be explained in detail on chapter 5. A banknote recognition flowchart of the proposed system is shown in Figure 3.2.

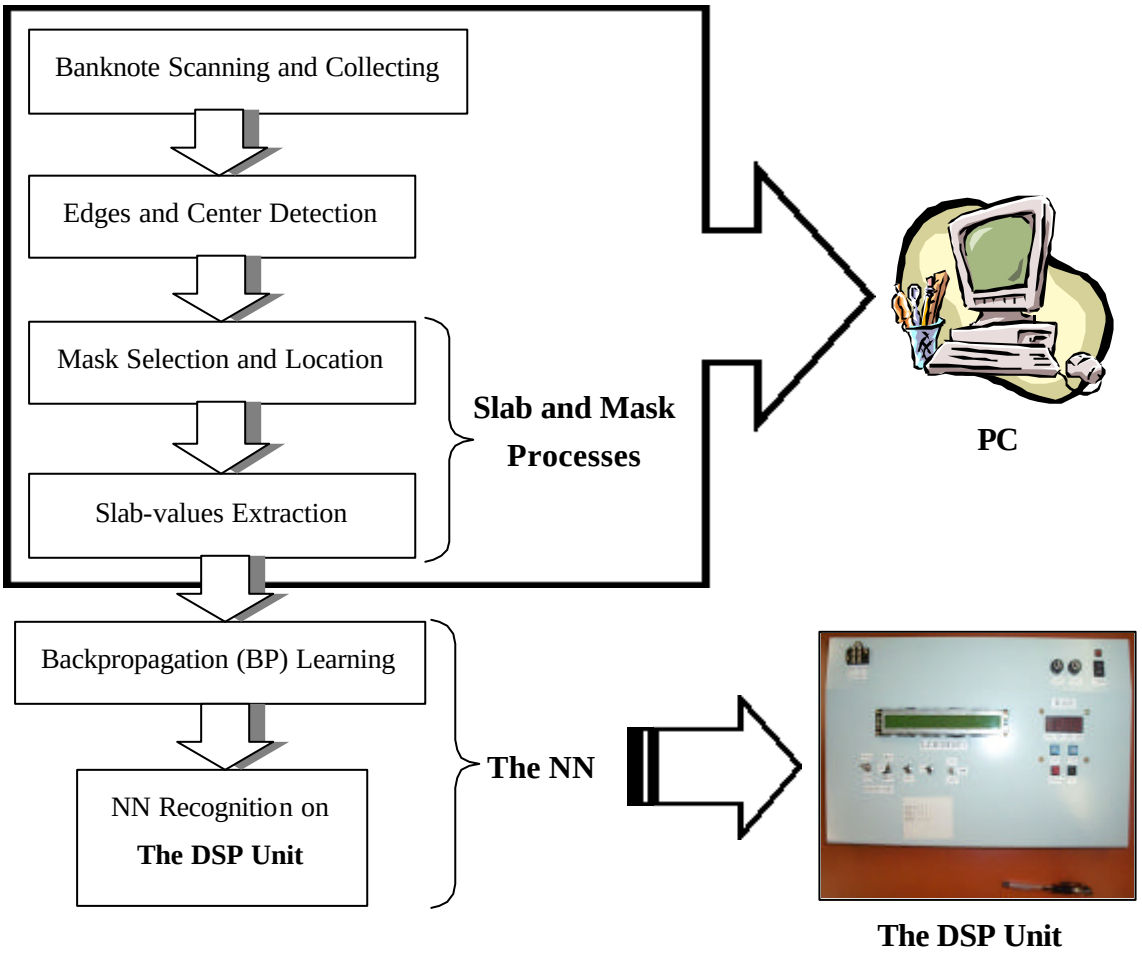


Figure 3.2 A Banknote Recognition Flowchart.

In this research, we perform 2 experiments that are PC experiment and DSP unit experiment. First, we execute all preprocesses and the NN on PC to confirm recognition ability of the proposed system. Next, for the DSP unit experiment, banknote images, the axis-symmetry mask set, and converged NN weights that saved on PC from the PC experiment are transferred to the DSP unit for learning and recognition as a real banknote reader and sorter machine. The experimental results of those experiments are shown on chapter 4 and 5.

3.2 Banknote scanning and collecting

Since, we use slab values, which are characteristics of banknote image, as inputs of the NN then banknote images must be collected. Therefore, the first step of preprocesses is banknotes scanning and collecting. Figure 3.3 shows the banknote scanning and collecting flowchart.

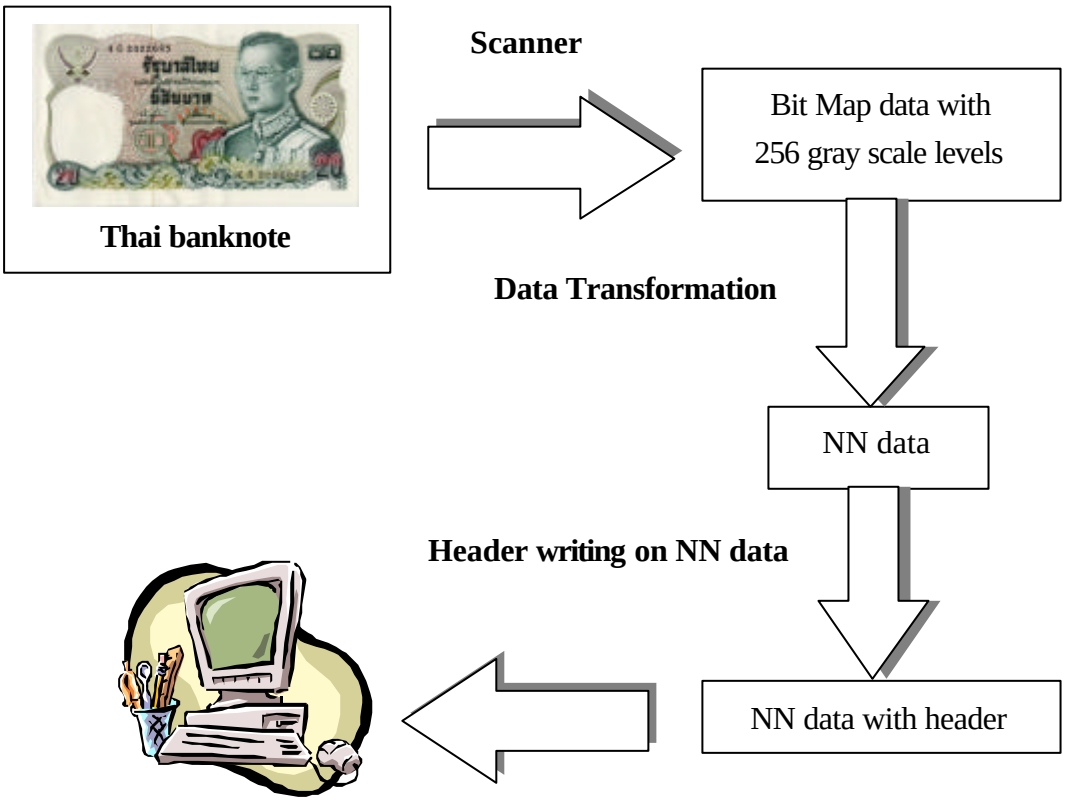


Figure 3.3 Banknote Scanning and Collecting Flowchart.

First, each banknote image is collected by a scanner with 3 conditions that are i) 72 dpi resolution, ii) 256 gray scale level mode, and iii) 60% scanning image and then a scanned image is saved as a bit map data on PC. Second, the bit map data of each banknote image is transformed to a NN data form which used for slab-values extraction. To get the NN data, the bit map data is divided along the vertical and horizontal axis to get 32×216 pixels image frame which is called the NN data as shown in Figure 3.4. The pixel value (N_p) of each pixel of the NN data is equal to the average value of all pixels value (x) of the bit map data which are located in that NN pixel. Namely, one pixel of the NN data is composed of n pixels of the bit map data, therefore; pixel value of that NN pixel is equal to the following equation.

$$\text{pixel value of the NN data } \left(N_p\right) = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (3.1)$$

Third, a header of each NN data image, which is used to indicate specific information of each banknote image such as pattern number, conveyed direction, and country, is written as shown in Figure 3.5. Finally, the NN data, which already include header, are saved on PC for using as banknote images for rest of processes.

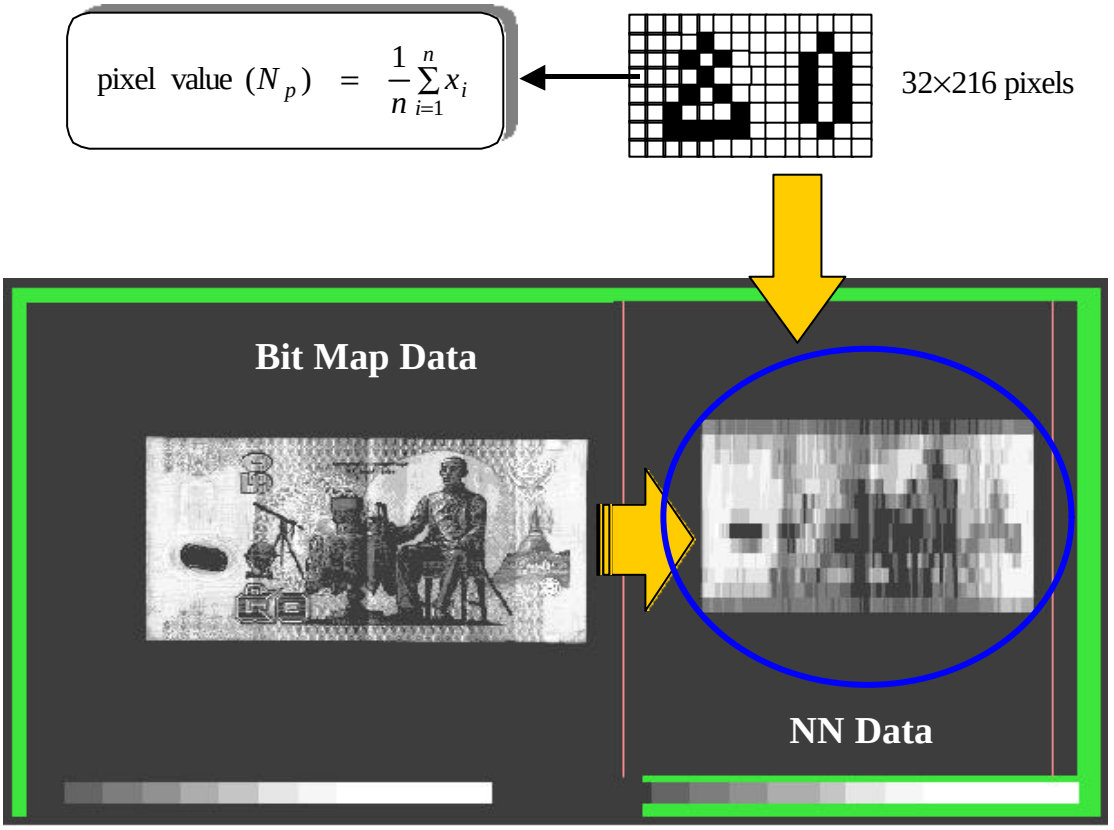


Figure 3.4 Bit Map Data to NN Data Transformation.

Figure 3.5 Specific Information of the Banknote Image (NN data form).

3.3 Edges and center detection

Before locating masks on banknote image, banknote edges and center have to be detected. For edges detection, 2 significant threshold values that are threshold “cho” and “tan” are required. Threshold “cho” and “tan” are offsets of gray scale level projection on the long and wide side of banknote image, respectively. The first and last columns (or line), which their gray scale level is higher than the threshold cho (or tan), are decided as edges of banknote image. Since, each Thai banknote type has different gray scale level from others then these 2 threshold values must be tuned manually until yield suitable threshold values for all Thai banknote types.

After yield the suitable threshold values for the edges detection, edges and center of banknote image are detected, automatically. For Thai banknotes, the threshold “cho” and “tan” are 10 and 35, respectively. By these threshold values, edges and center all Thai

banknote types are detected accurately. Figure 3.6 shows the edges detection of the 20B head direction (20HB). Since, the banknote image is the NN data format then it consists of 32 lines and 216 columns. Edges of this 20HB are line 9 and 21 for a top and bottom edge and column 47 and 195 for a left and right edge, respectively. Namely, line 9 and 21 are the first and last line along the wide side of the 20 HB image which their gray scale level is higher than the threshold "tan" as shown on Figure 3.6.

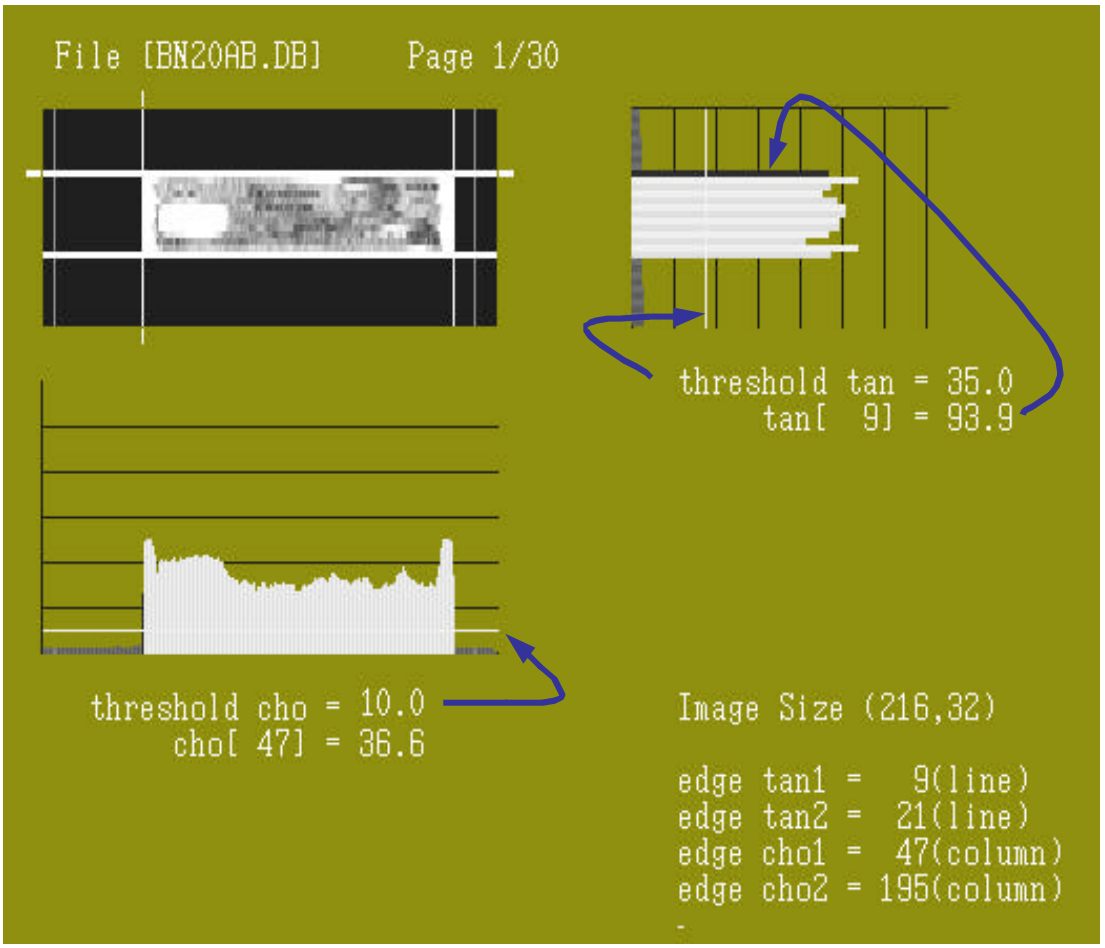


Figure 3.6 Edges Detection with Threshold Values.

As a result of the edges and center detection, the axis-symmetry mask set is manually selected and then applied to banknote image. Figure 3.7 is example of a mask pattern that located on the 20 HB image. To locate mask, center of mask pattern is located at the same position as the center of banknote image. Consequently, the slab values of that banknote image are extracted and used as inputs of the NN system. Details of the slab and mask processes are described on the next section.

3.4 Slab and mask processes

On this section, the slab and mask processes for slab-values extraction are presented. There are 3 subsections for these processes explanation. First, definition of slab value and mask are described. Then, structure and advantage of the axis-symmetry mask set are explained. Finally, selection method and location of a mask set that used in this research are shown.

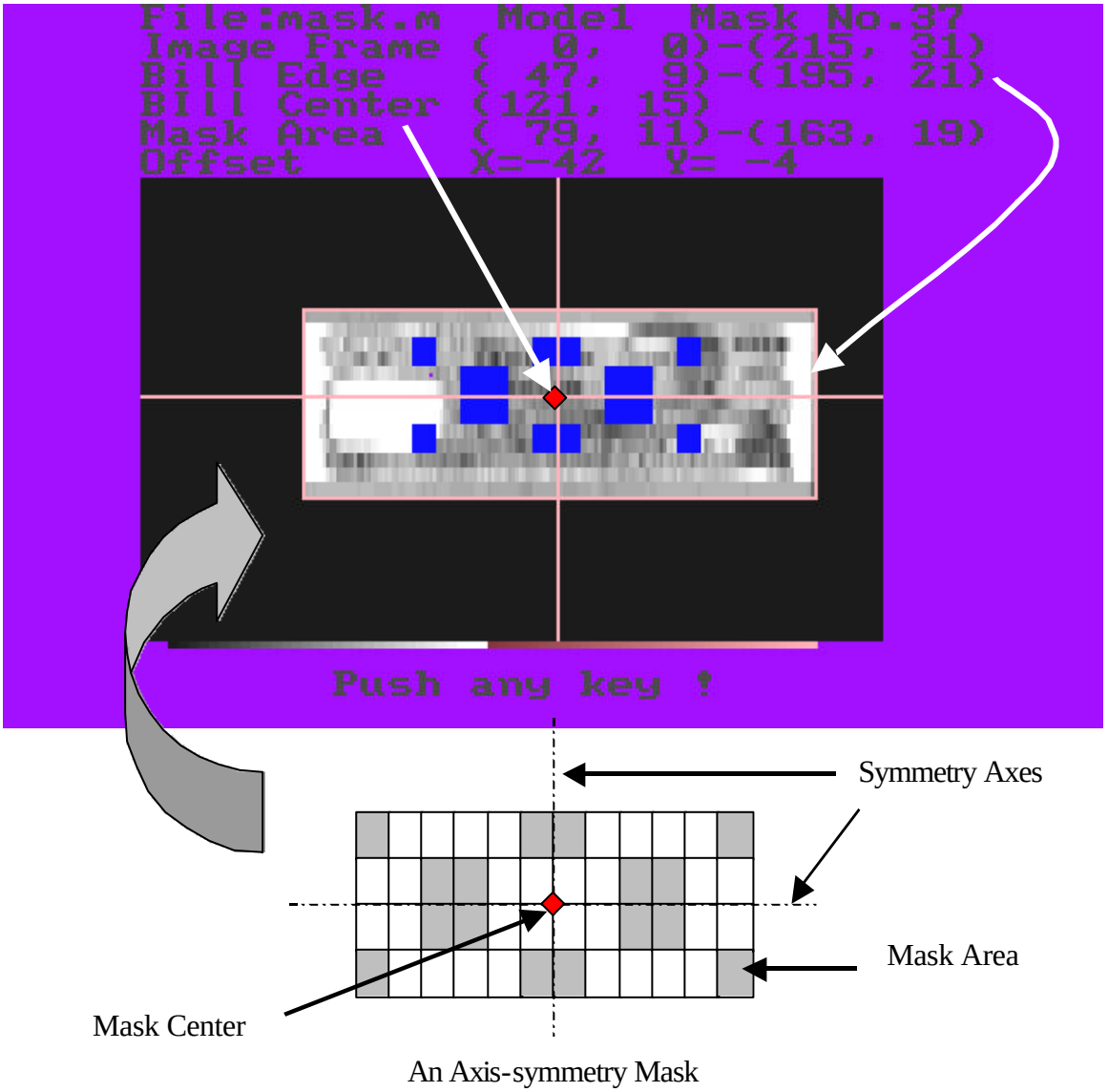


Figure 3.7 Center Detection and Mask Location.

3.4.1 Slab value

The previous researches of Prof. Takeda [11]-[18] proposed the mask process to reduce scale of the NN for effectiveness in practical banking machines. Refer to the invariance network of Widrow[19], slab value, which extracted by using the MAJ (majority) operator, is used as the characteristic of an input image. By using the slab value, output of the NN is robust for any modifications of the input image and the scale of the NN is also reduced. Therefore, this slab method has been applied for the NN scale reduction[11]-[18].

To reduce scale of the NN, pixels value of each image are summed up as so-called a slab value, and then used as the NN input. This slab value is regarded as the digitized characteristic of a banknote. Sometimes, although two images are different, they have the same slab value as shown in Figure 3.8. Therefore, the mask process is applied to solve this problem so that the slab value is the summation of non-masked pixel values as shown in Figure 3.9.

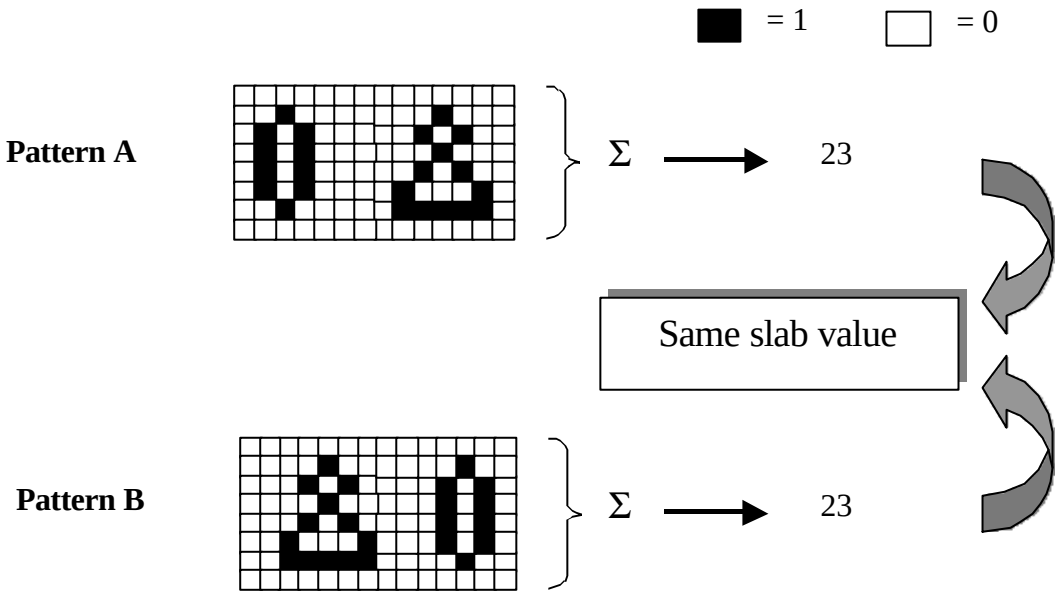


Figure 3.8 Different Images with Same Slab Values.

Moreover, for the effective NN recognition system, several masks, which have different mask patterns, are applied as a mask set to each input image to yield various and different slab values. Thus, by combining this technique with the NN, the NN scale is reduced and number of input nodes of the NN is depended on number of mask patterns in a mask set. Figure 3.10 shows the configuration of the NN recognition system with the slab and mask processes.

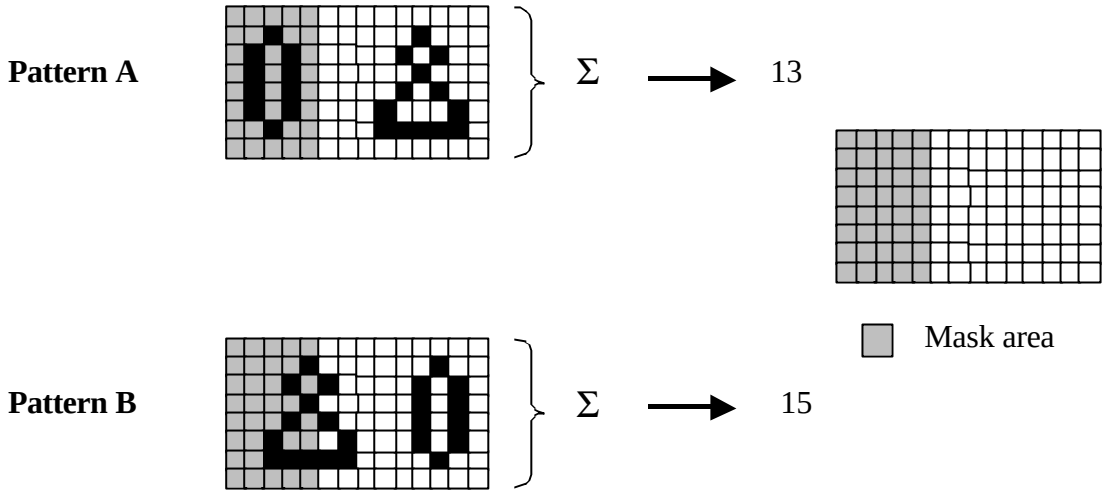


Figure 3.9 Different Images with Mask and Different Slab Values.

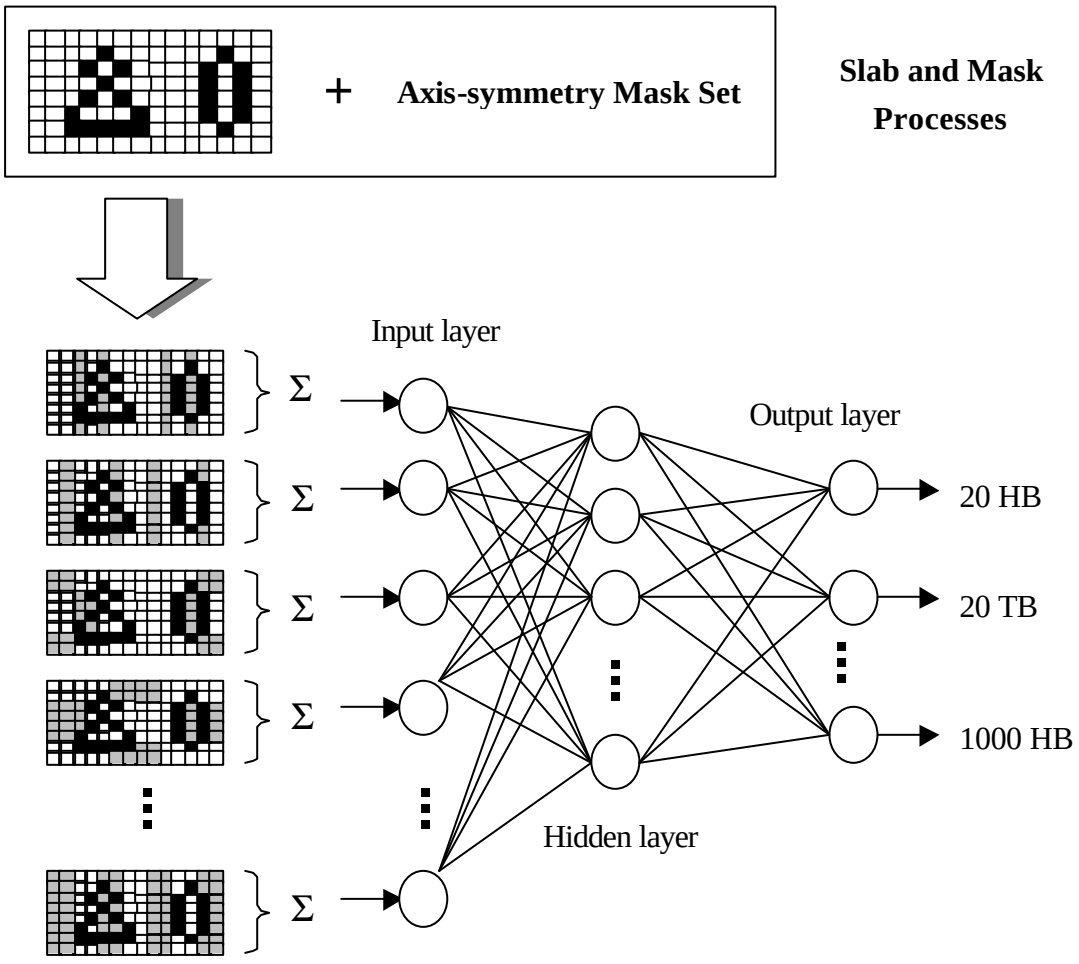


Figure 3.10 The NN Configuration with Slab and Mask Processes.

3.4.2 Axis-symmetry mask

A mask set has been applied to banknote images for the effective recognition system. In this research, one mask set has 50 mask patterns and each mask pattern is divided into 48 blocks (4 × 12), which used to locate mask areas. Figure 3.11 shows configuration of a mask pattern.

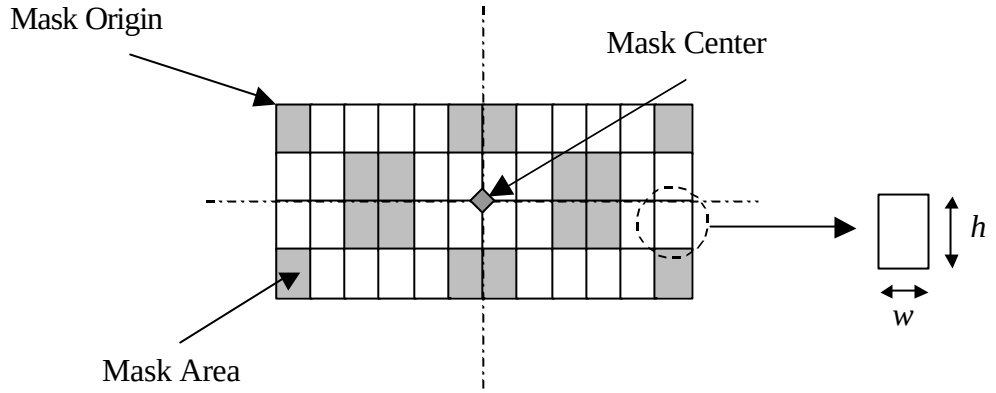


Figure 3.11 Mask Pattern Configuration with (w× h) Block Size.

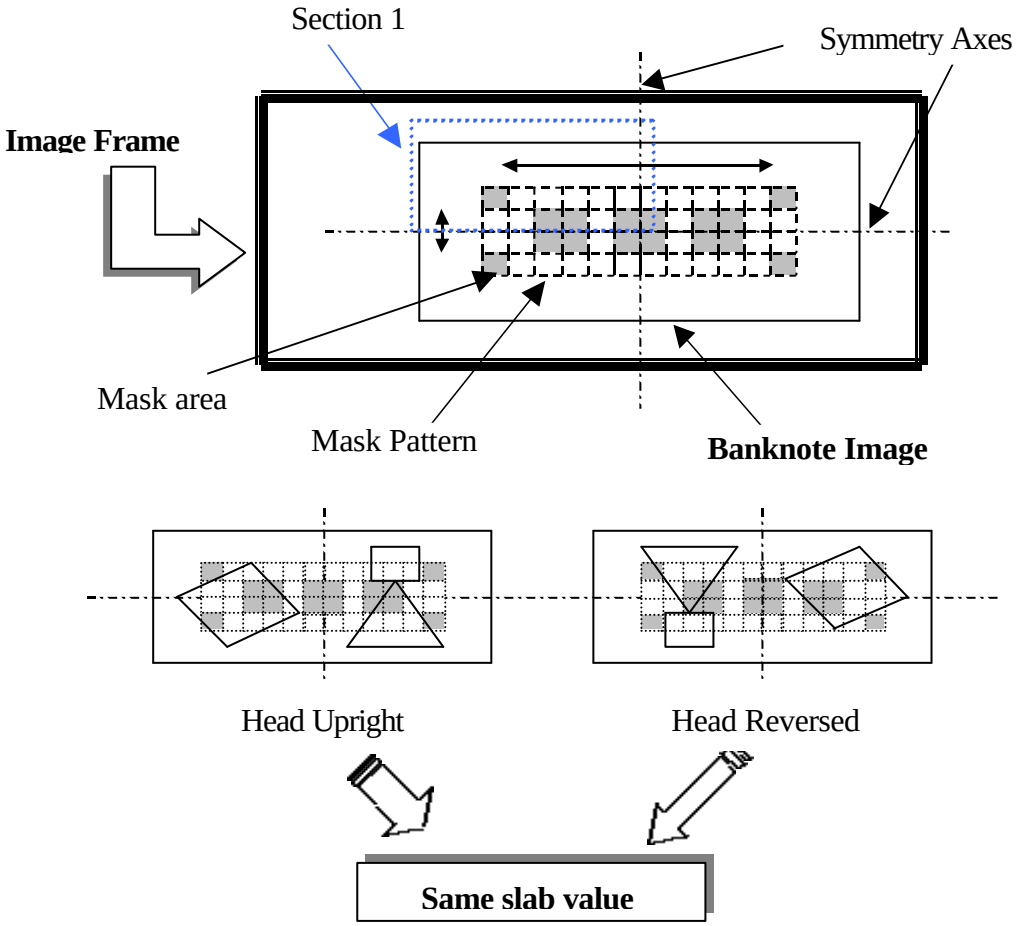


Figure 3.12 Axis-symmetry Mask Structure.

From previous researches, mask areas of a mask pattern are randomly selected. However, in this research, the axis-symmetry mask is applied because of its advantage in multiple kinds banknote recognition[7], [15]. For using this kind of mask, a banknote image and a mask pattern are divided equally by symmetry axes for both vertical and horizontal axis. Next, center of mask pattern is located at the same position as center position of the banknote image. Finally, mask areas of the axis-symmetry mask are also randomly selected but its areas are located by refer to a vertical and horizontal axis of banknote image. Namely, mask areas are randomly selected and firstly locate on a section 1 of the image. After that other mask areas are located at the position, which symmetry with the mask areas in the section 1, on other sections of a mask pattern one by one as shown in Figure 3.12.

As discussed on the chapter 1, there are 5 types for Thai banknotes and each type has 4 conveyed directions which are head upright, head reverse, tail upright, and tail reverse. Thus, there are totally 20 patterns for Thai banknotes. However, as a result of using the axis-symmetry mask, the slab values of both head (tail) upright and head (tail) reversed directions of the same banknote are identical. Then, 2 conveyed directions of a banknote image, upright and reversed, are recognized simultaneously. Therefore, there is only 2 directions of each banknote type, head and tail direction, which should be recognized, and number of total banknote patterns that have to be recognized by the NN is only 10 that are 20HB, 20TB, 50HB, 50TB, 100HB, 100TB, 500HB, 500TB, 1000HB, and 1000TB. Figure 3.12 shows how slab values of upright and reverse direction are the same and present the axis-symmetry mask structure.

3.4.3 Mask selection and location

In this research, we apply a mask set to all banknote images for the slab-values extraction. This mask set consists of different 50 mask patterns. Mask areas of each mask pattern are randomly selected according to significant area to distinguish all patterns of Thai banknotes and located by refer to the vertical and horizontal axis as discussed on last subsection. Since, center of mask pattern must be located at the same position as center of banknote image then we locate mask pattern and define its size by refer to an offset. This offset indicates distance between an origin point of mask pattern and its center point. For example, mask offset is (-48, -4) and center of the banknote image locates at (121, 15) then the origin point of mask pattern is (73, 11) as shown in Figure 3.13. These coordinates are pixel coordinates of the banknote image.

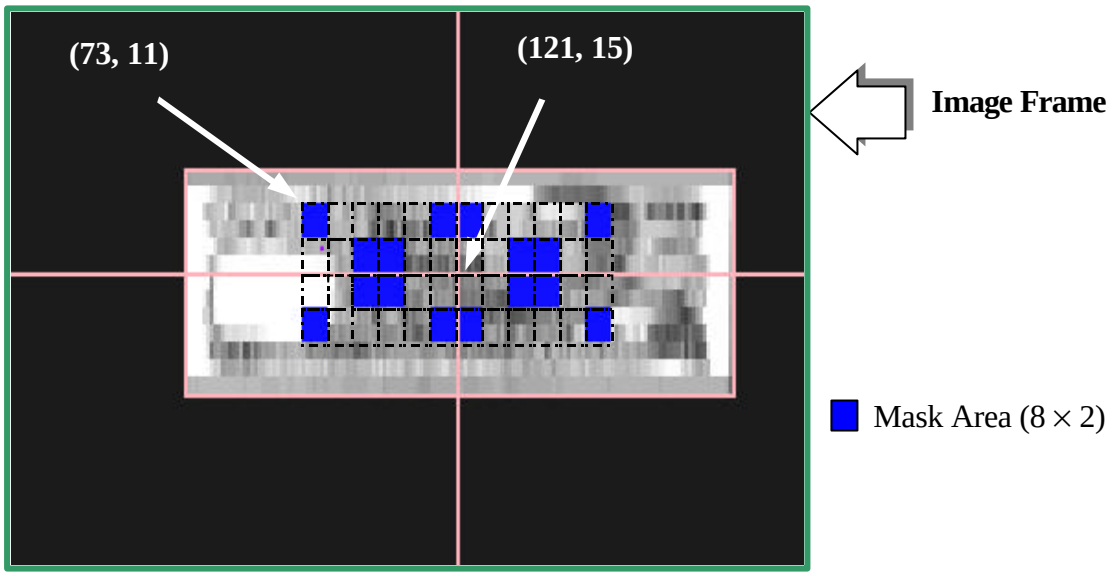


Figure 3.13 Mask Location on the Banknote Image.

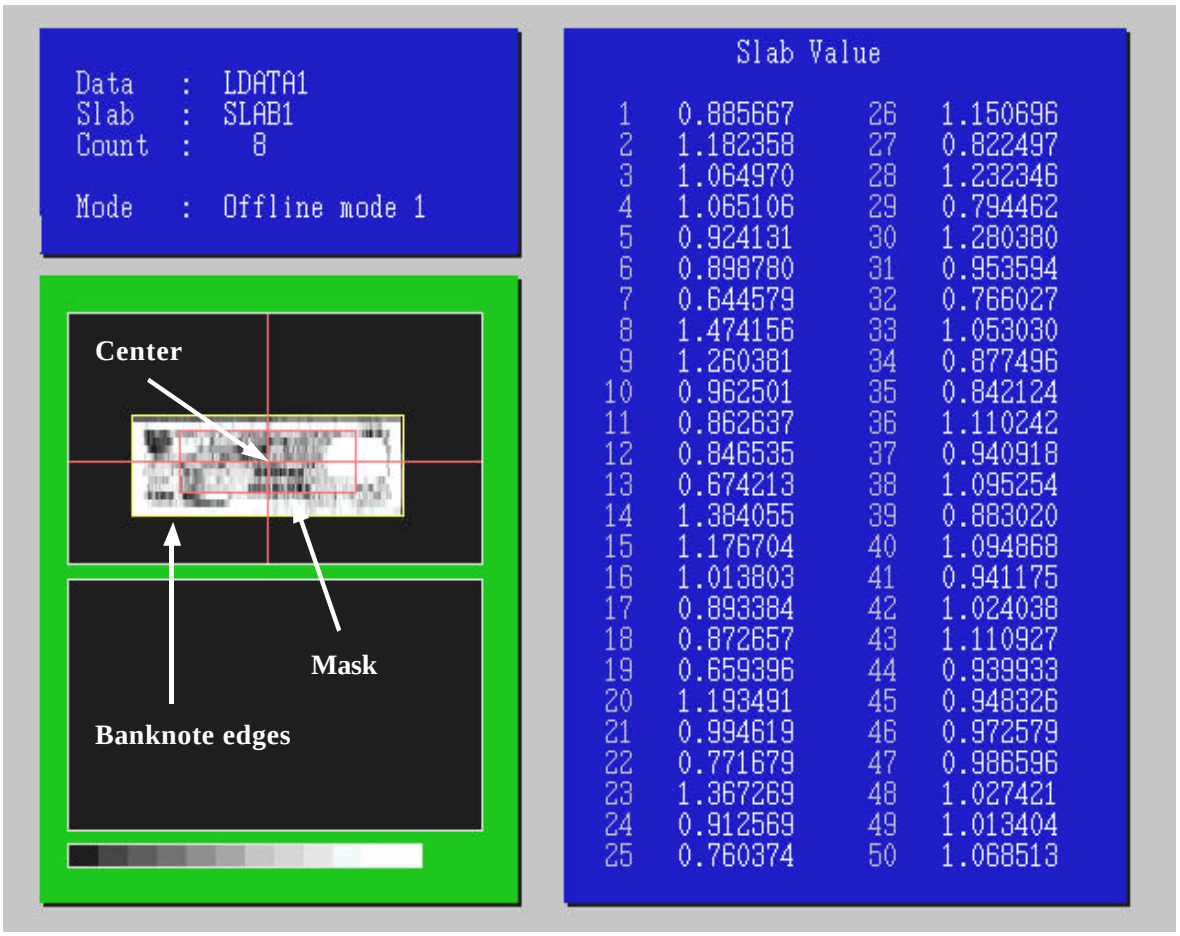


Figure 3.14 50 Slab Values of the 20B Head (20HB).

Each mask pattern consists of 48 blocks that used for mask location. Size of each block is manually select and it is proportional to a mask offset. However, this block size is not bigger than 12×12 because of the limitation of the banknote image size (32×216). For this research, we applied a (-48, -4) mask offset and a (8×2) block size for each mask pattern. A mask set of the axis-symmetry masks that used in this research is shown in Appendix A. After that, this mask set is applied to each banknote image to yield 50 slab values which used as inputs for the NN. Figure 3.14 shows 50 slab values of the 20HB.

3.5 Neural network

Next, slab values of all banknote images are inputted to the NN for learning and recognition. Therefore, the NN that used in this research are described. First, the NN structure is presented. Second, NN learning method of this research and its learning conditions are explained. Third, learning data which used for training the NN are shown. Finally, the NN recognition process is described. Table3.1 shows the NN information of this research. This information is discussed in detail on next subsections.

Table 3.1 The NN Information of Thai Banknote Recognition

Mask no. (Input node no.)	50
No. of banknote types (Output node no.)	10
Hidden node no.	30
Learning data / pattern	20
Testing data / pattern	80
Minimum mean square error	0.0001
Maximum iteration no.	20000
Learning algorithm	Backpropagation method

3.5.1 Neural network structure

A three-layered NN have been applied for this banknote recognition. The NN consists of an input layer, one hidden layer, and an output layer. For Thai banknotes recognition, structure of this three-layered NN is $50 \times 30 \times 10$. The NN has 50 nodes of the input layer because there are 50 mask patterns for the axis-symmetry mask set. The 50 slab values that extracted by the mask set are inputted to the NN for learning. For the hidden layer, number of

the hidden nodes is selected manually and the suitable amount for this research is 30 nodes. Since, there are 10 patterns for Thai banknotes recognition, then number of the output node is 10. Figure 3.15 shows the structure of the NN for Thai banknotes.

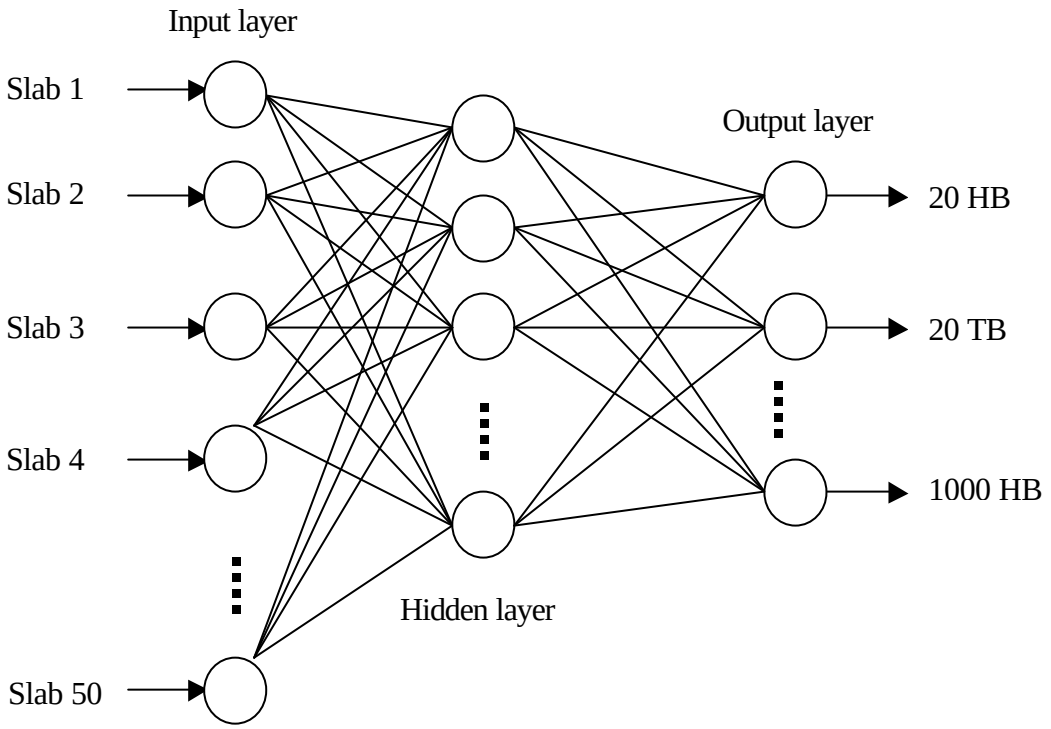


Figure 3.15 The NN Structure for Thai Banknote Recognition.

3.5.2 Learning method

The backpropagation (BP) method has been used as learning algorithm of this research because of its generality [4]. The BP achieves its generality because of the gradient-descent which used for the NN learning. The gradient-descent is analogous to an error minimization process. Error minimization is an attempt to fit a closed-form function to a set of actual data points. Namely, the closed-form function deviates from the exact value by a minimal amount. Figure 3.16 shows the error minimization process of a function through the set of actual data.

The learning process begins by inputting an input value to the NN. This input is propagated through the entire network until its output value is produced. Then the BP uses the generalize delta rule to determine error of the current pattern and contributed by slightly in a direction that reduce its error signal. After that, the process is repeated for next patterns. The typical three-layered backpropagation NN is shown in Figure 3.17. Assume that, for Figure 3.17, the input layer has *p* nodes and the hidden layer has *m* nodes. For the output layer, it has

n nodes. Moreover, there are weight W_{hi} and W_{oh} which represented weights that connect between layers of the NN.

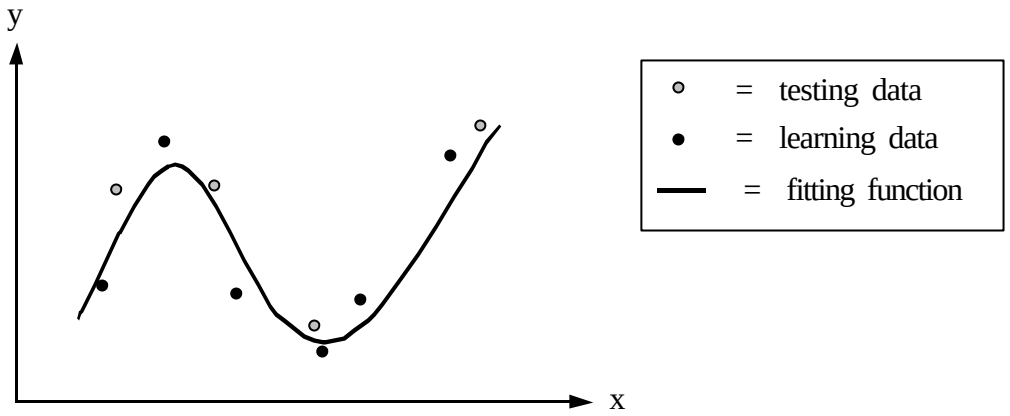


Figure 3.16 The Error Minimization Concept.

The first operation of the NN is called feedforward calculation as shown by solid lines on Figure 3.17. During this operation, the output vector *O(t)* is calculated by feeding the input vector *I(t)* through the hidden layer of the NN. For the given input vector *I(t)*, the output of the node *h* of the hidden layer *H(t)*, is

$$H_h(t) = f\left(Net_h(t)\right) \tag{3.2}$$

$$H_h(t) = f\left(\sum_i W_{hi} I_i(t)\right) \tag{3.3}$$

where $Net_h(t)$ represents the total input of the node *h* in the hidden layer, *t* is iteration number, and *f(x)* is the activate function. In this research, sigmoid function as shown in Eq. (3.4) has been applied as the activate function.

$$f(x) = \frac{1}{1+\exp\left(\frac{-x+\theta}{T}\right)} \tag{3.4}$$

The above equation is the sigmoid function. θ is a threshold value that effects to center of the sigmoid function as shown in Figure 3.18 [3]. For *T*, it is called network temperature constant which effects to slope of the sigmoid graph. When T is large positive number, the sigmoid graph has few slope as shown on Figure 3.19 [1]. In this research, T is automatically selected by our learning program and we applied $\theta = 0$. Graph of the sigmoid function at $\theta = 0$ is shown in Figure 3.19.

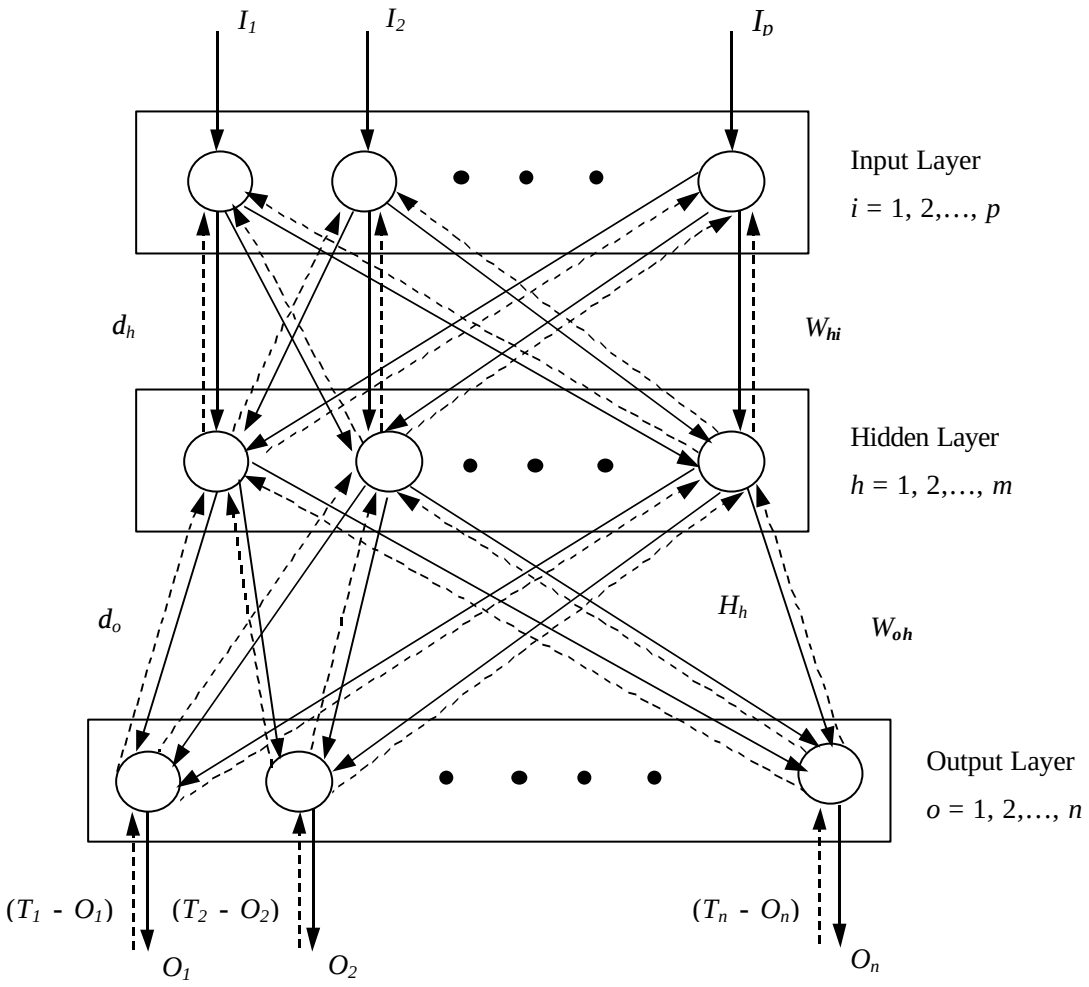
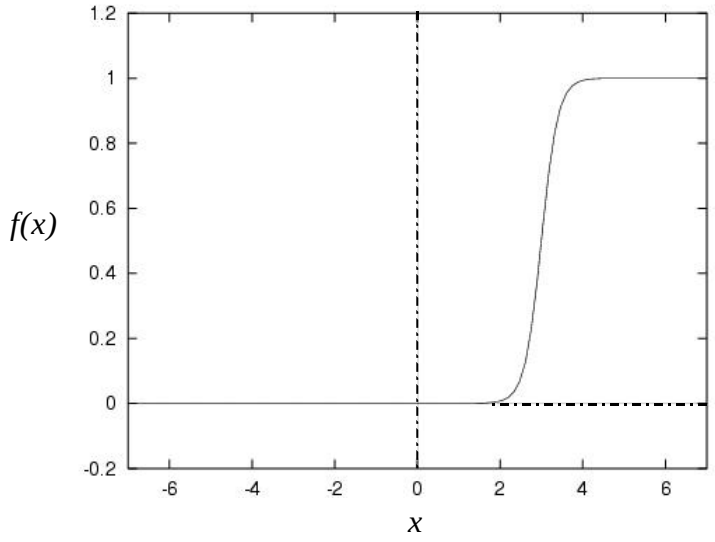


Figure 3.17 Typical three-layered Backpropagation Neural Network.



$$f(x) = \frac{1}{1+\exp\left(\frac{-x+3}{0.2}\right)}$$

Figure 3.18 The Sigmoid Function Graph when $q = 3$ and $T = 0.2$.

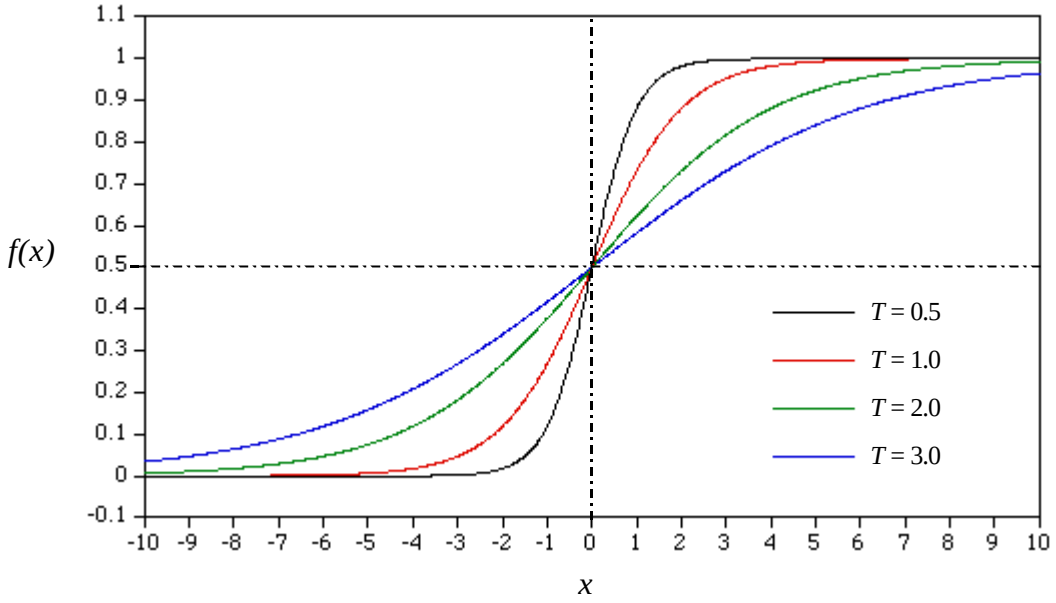


Figure 3.19 The Sigmoid Function Graph with $\theta = 0$.

For the output of node o in the output layer $O_o(t)$ is

$$O_o(t) = f(Net_o(t)) \tag{3.5}$$

$$O_o(t) = f\left(\sum_h W_{oh} f\left(\sum_i W_{hi} I_i(t)\right)\right) \tag{3.6}$$

where $Net_o(t)$ represents the total input of node o in the output layer.

The second operation of the NN is error backpropagation or feedback calculation that marked by dashed lines on Figure 3.17. When inputs of pattern o are inputted to the NN, actual output on the node o of the output layer is $O_o(t)$ while desired output of that pattern is $T_o(t)$. Therefore, error function of this research which defined by mean squared error $E_o(t)$ between the desired output $T_o(t)$ and the actual output $O_o(t)$ is

$$E_o(t) = \frac{1}{2}[T_o(t) - O_o(t)]^2 \tag{3.7}$$

Moreover, when the NN are learned, several learning data are applied. Therefore, the mean squared error of all output nodes of k^{th} learning data of the pattern o, $E_{ko}(t)$, is

$$E_{ko}(t) = \frac{1}{2}\sum_o [T_o(t) - O_o(t)]^2 \tag{3.8}$$

Finally, the mean squared error of all learning data of patter *o*, *E(t)*, is

$$E(t) = \frac{1}{2}\sum_{k}\sum_{o}[T_o(t) - O_o(t)]^2 \tag{3.9}$$

As discussed before, aim of the BP is error minimization. Therefore, the NN weights have to be adapted to minimize the mean squared error. In this research, steepest descent has been applied for the NN weights modification. By this steepest descent, the mean squared error, *E(t)*, is continuous decreased at each iteration [6]. The adaptive rule for the NN weights between the hidden layer and the output layer is

$$W_{oh}(t+1) = W_{oh}(t) + \Delta W_{oh}(t) \tag{3.10}$$

$$\Delta W_{oh}(t) = -\varepsilon\delta_o(t)H_h(t) + \alpha\Delta W_{oh}(t-1) + \beta\Delta W_{oh}(t-2) \tag{3.11}$$

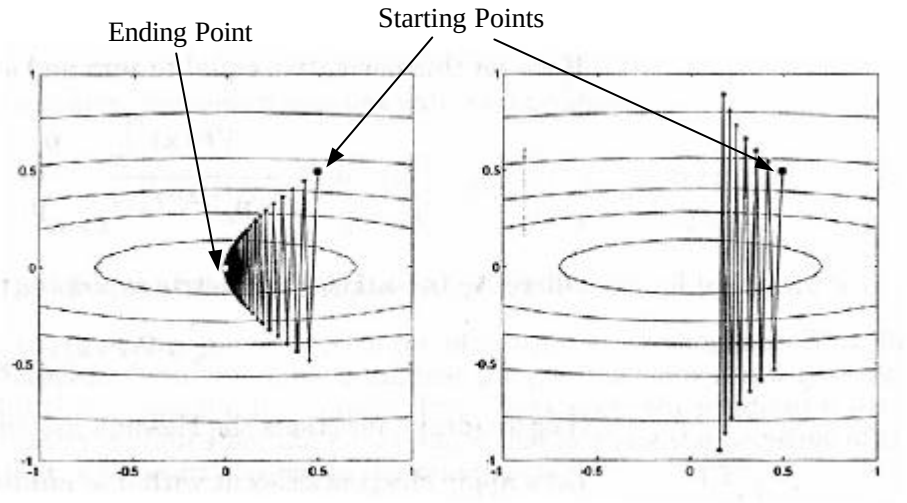
where ε is learning rate, α is inertia constant and β is oscillation constant. For the learning rate ε, if small learning rate is used, it takes much time for learning procedure to reach the minimum mean squared error. Therefore, large learning rate is applied. Then the NN weights modification will take large steps and learning procedure is expected to converge faster. However, if the learning rate is too large, learning process is unstable as shown in Figure 3.20 [2]. In this research, the suitable learning rate range is $0.1 < \varepsilon < 1.0$ and the learning rate that we have been applied for experiments is 0.5.

For inertia constant, α, it reduces oscillation of the mean squared error and accelerates convergence of learning process. An oscillating constant, β, control oscillation of the error and control the error to escape from local minimum. In this research, $\alpha = 0.95$ and $\beta = -0.1$ have been used for this research. For $\delta_o(t)$, it is generalized error between the hidden layer and the output layer, which is determined by the following equation.

$$\delta_o(t) = [T_o(t) - O_o(t)]\, f'(Net_o(t)) \tag{3.12}$$

Similarly, for the NN weights modification between the input layer and the hidden layer, the adaptive rule at t^{th} iteration is

$$W_{hi}(t+1) = W_{hi}(t) + \Delta W_{hi}(t) \tag{3.13}$$



(a) α = 0.039 (Converged Case) (b) α = 0.041 (Unstable Case)

Figure 3.20 The Steepest Descent Trajectories.

$$\Delta W_{hi}(t) = -\varepsilon \delta_h(t) I_i(t) + \alpha \Delta W_{hi}(t-1) + \beta \Delta W_{hi}(t-2) \quad (3.14)$$

$$\delta_h(t) = \left[\sum_o W_{ho} \delta_o(t) \right] f'(Net_h(t)) \quad (3.15)$$

To begin the NN weights modification, initial NN weights are required. In this research, there are 2 types of the initial weights depend on learning type. These 2 initial weights types are i) random initial weights and ii) converged NN weights of last learning. For detail of both types of the learning are discussed in chapter 5. Then, the NN weights are continuously adjusted by using above adaptive rules until reach learning condition which will be discussed on next subsection. Summary of the NN learning method is shown on Figure 3.21.

3.5.3 Learning conditions

For learning procedure, there are learning conditions that used as stopping criteria for the NN weights modification. In general, 2 learning conditions are used which are minimum error and maximum iteration number. Since, the mean squared error has been applied then one of learning condition is the minimum mean squared error which set at 0.0001. Another learning condition, the maximum iteration number is 20000. Therefore, the NN begins to adjust its weights from the initial weights until yield the minimum mean squared error or the iteration number of weights adaptation reaches 20000. Figure 3.22 shows the error graph during learning.

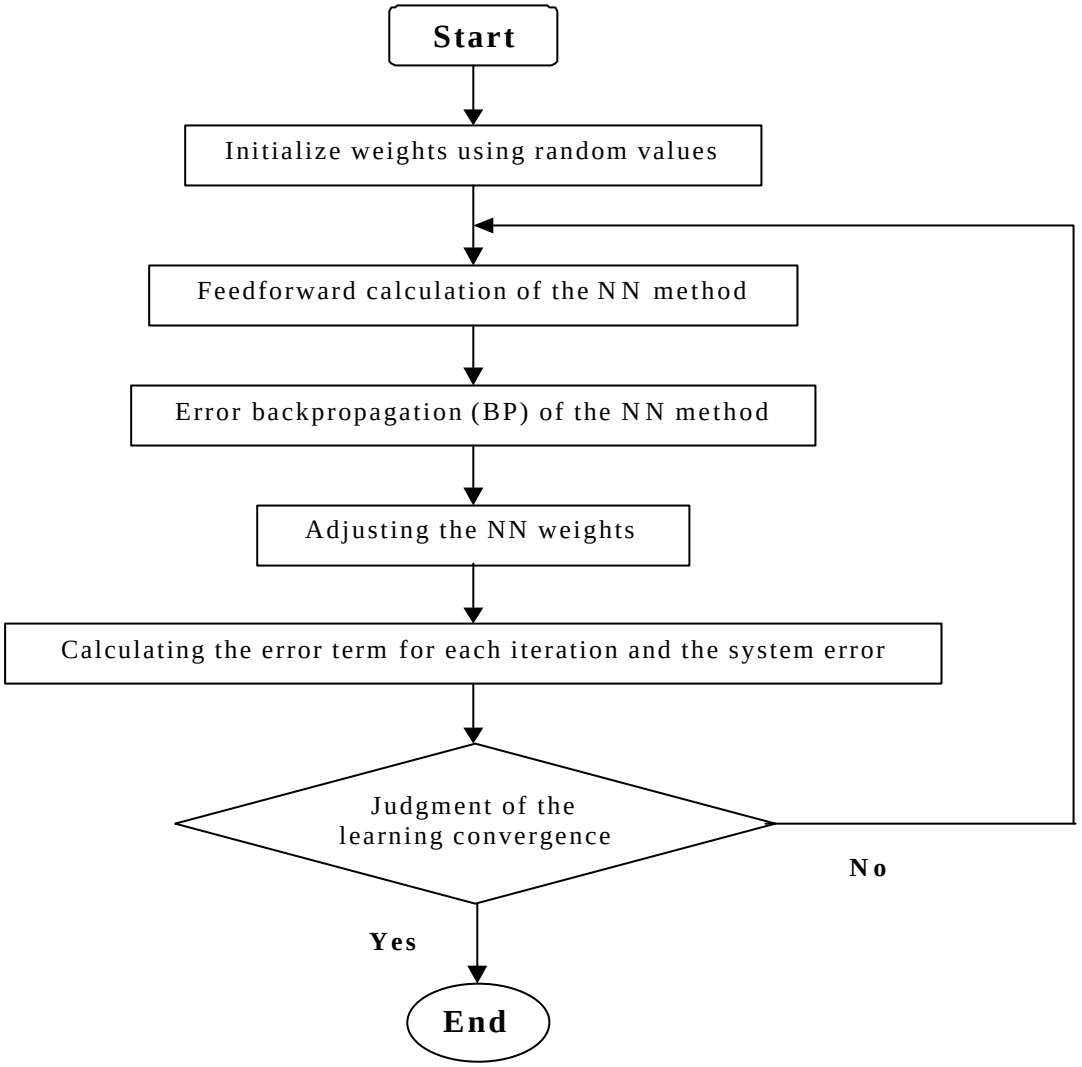


Figure 3.21 Summary of the NN Learning Method.

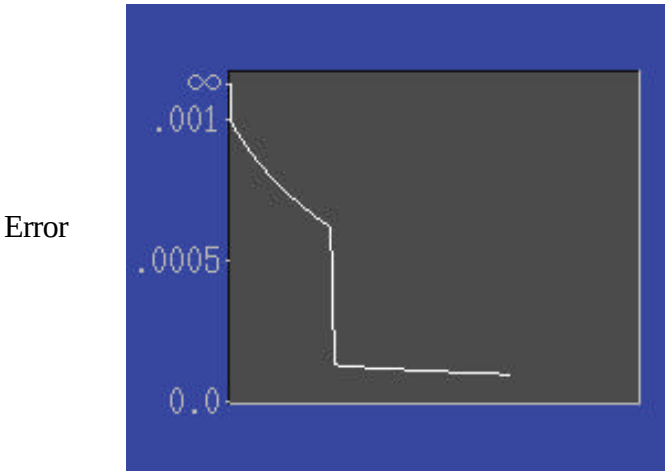
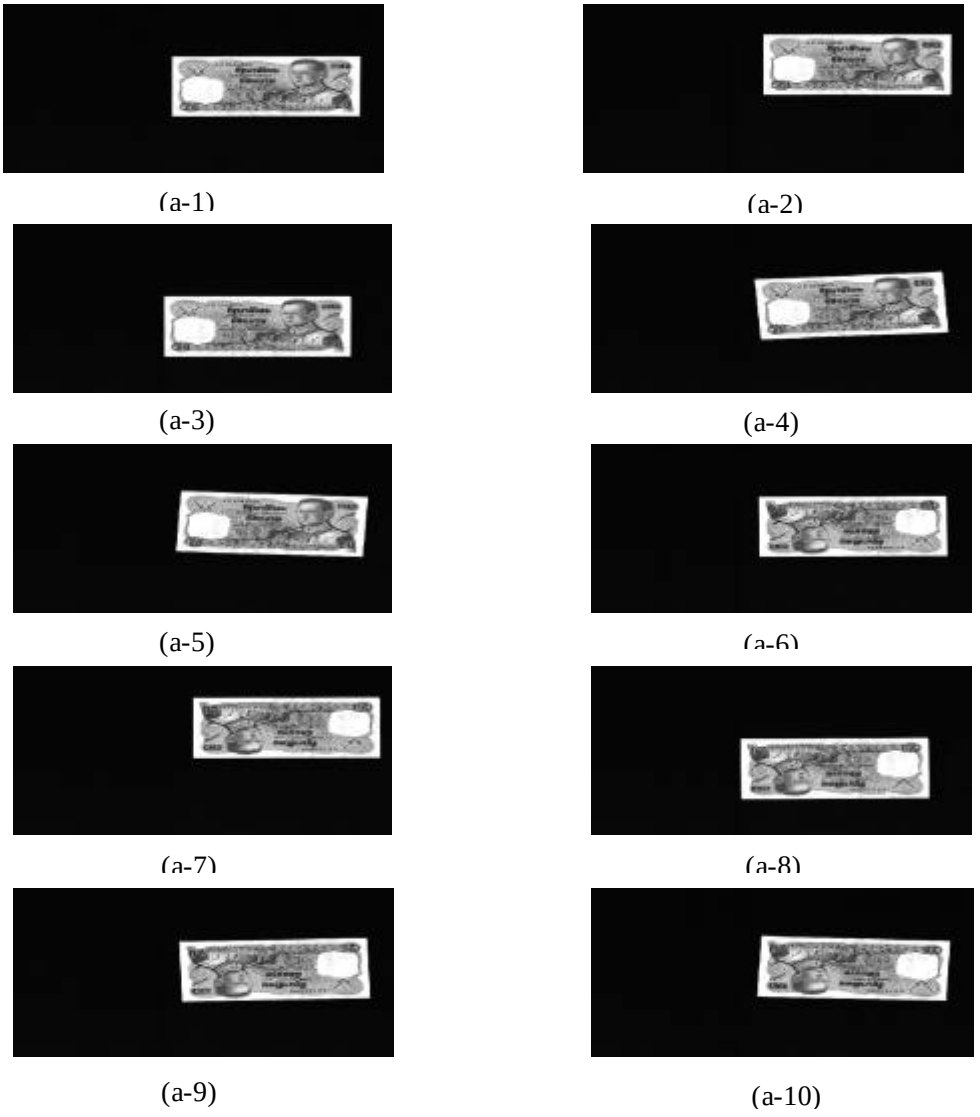


Figure 3.22 Error Graph during Learning Procedure.

3.5.4 Learning data

To adjust the NN weights, learning data are required. In this research, 20 learning data per banknote pattern are applied. The 50 slab values of each learning data are inputted to the NN. During learning, the NN weights are adjusted until the mean squared error between the actual output and the desired output is equal to 0.0001 or the iteration number reaches the limitation as discussed on last subsection. Figure 3.23 shows 20 learning data for the 20HB pattern. These 20 learning data consist of 3 pieces of the 20HB which have different bank number. For banknote piece no. 1, there are 10 learning data with 5 various positions and 2 different conveyed directions as shown in Figure 3.23(a). For banknote piece no. 2 and 3, there are 5 learning data for each piece with 5 different positions. However, learning data of banknote piece no. 2 and 3 differ in the direction as shown in Figure 3.23(b) and 3.23(c).



(a-1) (a-2)

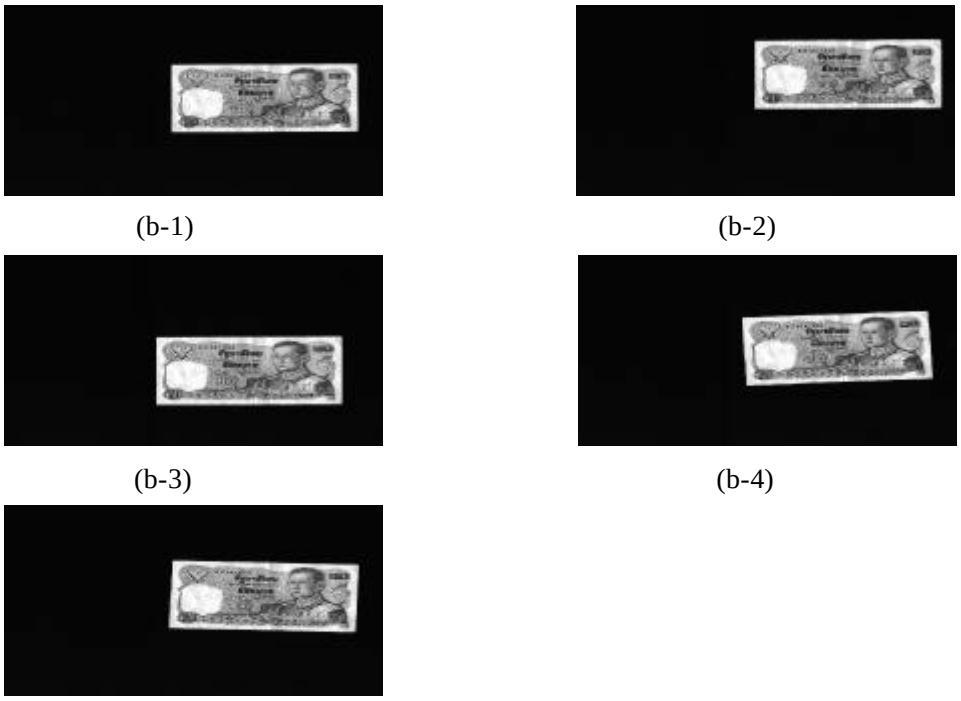
(a-3) (a-4)

(a-5) (a-6)

(a-7) (a-8)

(a-9) (a-10)

3.23(a) 10 Learning Data of Banknote Piece No. 1 (#4G2022695)

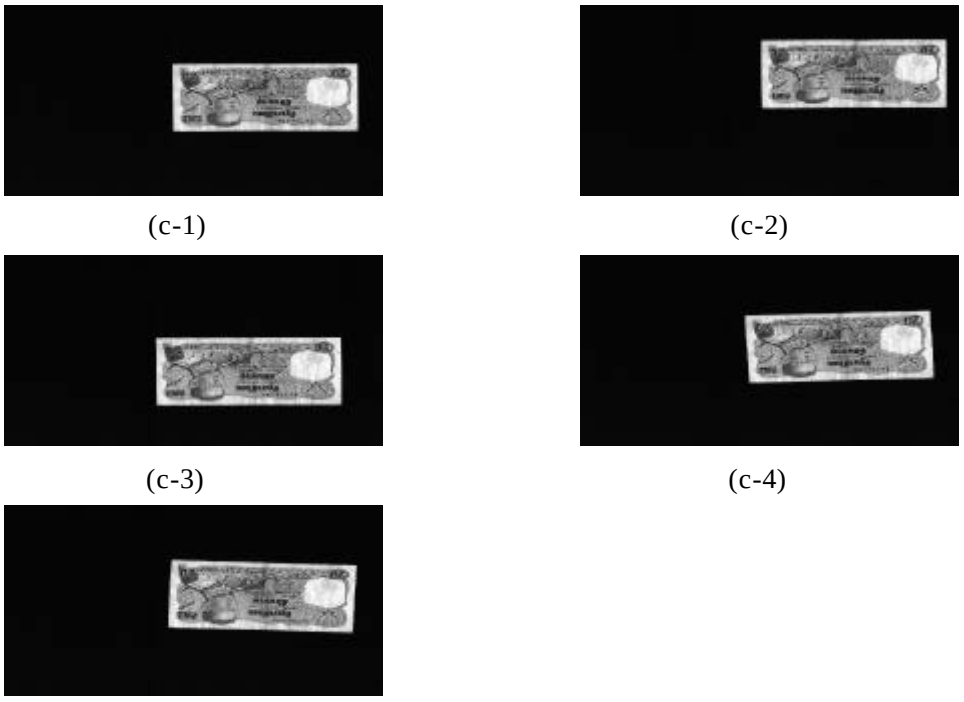


(b-1) (b-2)

(b-3) (b-4)

(b-5)

3.23(b) 5 Learning Data of Banknote Piece No. 2 (#6E8847225)



(c-1) (c-2)

(c-3) (c-4)

(c-5)

3.23(c) 5 Learning Data of Banknote Piece No. 3 (#7E7323161)

Figure 3.23 20 Learning Data of the 20HB.

3.5.5 Recognition procedure

After learning procedure is finished, the converged NN weights are saved on PC for recognition process. On the recognition process, the converged NN weights are used as weights between layers. To begin, banknote that must be recognized is inputted to the slab-extraction process and then the slab values of that banknote image are inputted to the NN. Next, the feedforward calculation of the NN as discussed on previous subsection starts. Then, output responses of all output nodes are produced. Finally, pattern that has maximum output value is decided as a correct pattern of that banknote.

However, sometimes, there are 2 output responses which their output values are little different. Therefore, 2 conditions are set to make the effective recognition. First condition, the maximum output value, which will be decided as the correct pattern, must larger than threshold value TH1. Another condition, the difference between the maximum output value and the second maximum output value must be more than threshold value TH2. Both threshold values that have been used in this research are TH1 = 0.5 and TH2 = 0.2.

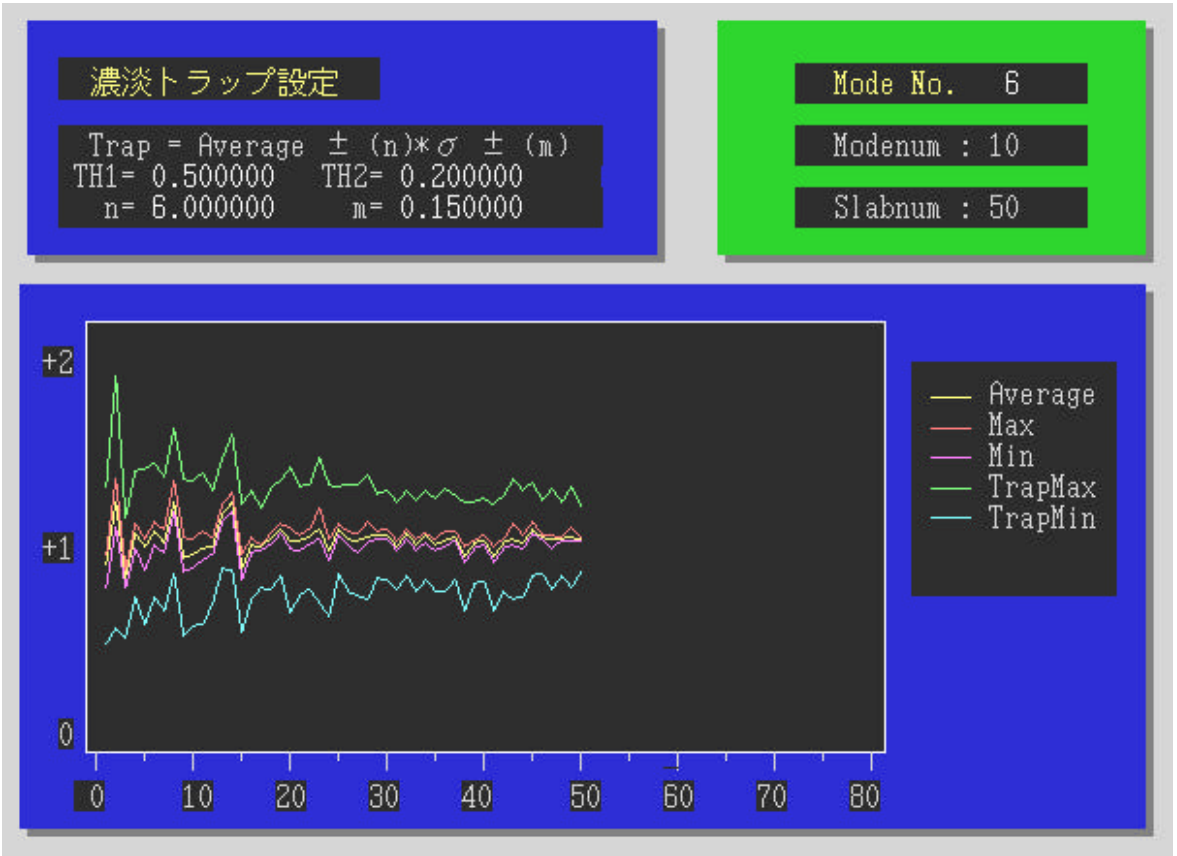


Figure 3.24 Trap Values of the 100TB Pattern.

Moreover, using only the NN is not effective to prevent banknote counterfeit. Therefore, trap values are applied to make boundary for the slab values of each original banknote. If slab values of banknote are exceeding the that banknote will be rejected from the machine. The trap values of each banknote pattern are determined from the following equation.

$$\text{Trap values} = \text{Average slab value} \pm n \times \sigma \pm m \qquad (3.16)$$

where n and m are constant and σ is standard deviation of slab values. In this research, n = 6.0 and m = 0.15. Figure 3.24 shows trap values of the 100TB pattern. Therefore, the pattern that has maximum output values will be decided as the correct pattern if it passes all 3 conditions. The NN recognition flowchart in summary is shown on Figure 3.25

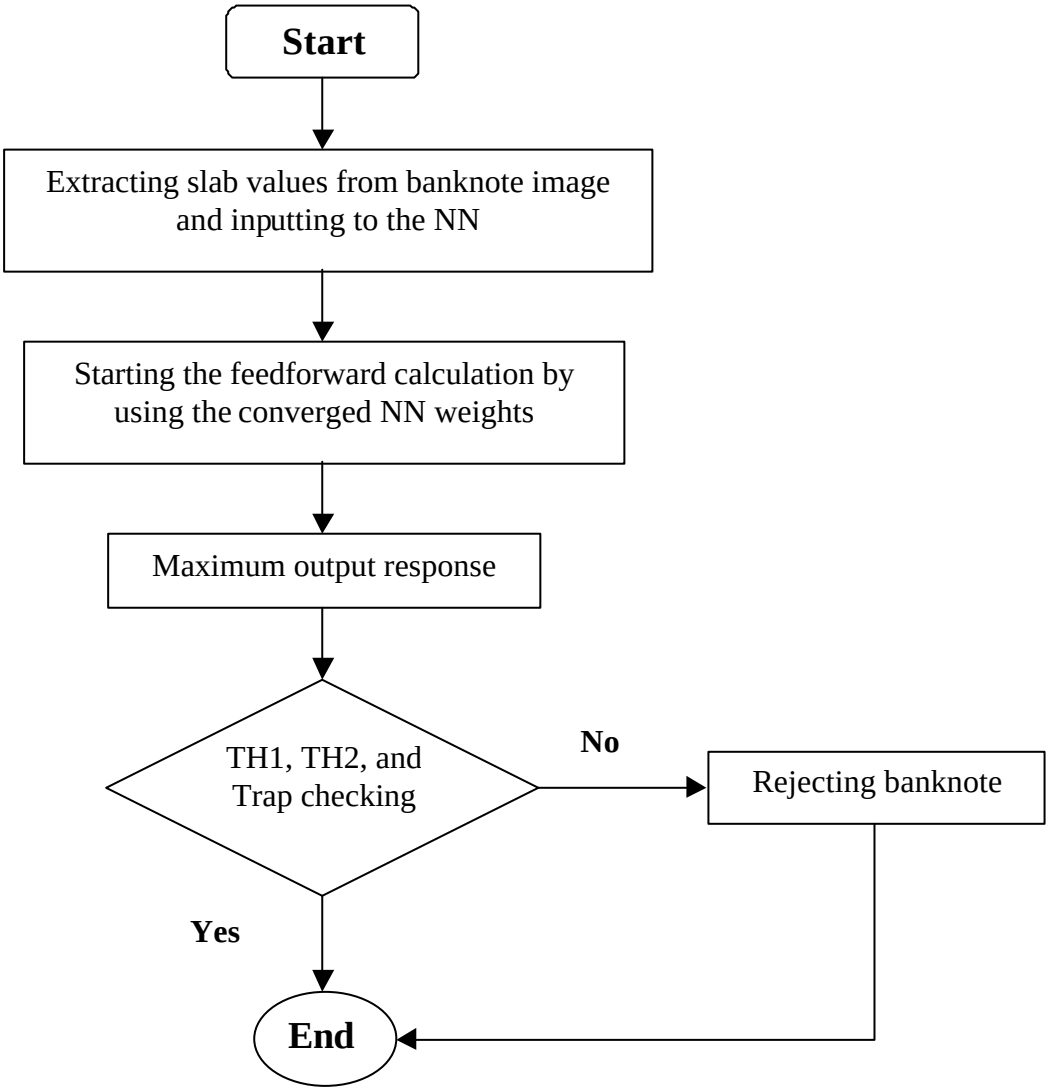


Figure 3.25 The NN Recognition Flowchart.

Chapter 4: Experimental Results and Discussion

Next, the proposed recognition system is tested. Therefore, 80 testing data are applied as will be described on section of this chapter. Then, the recognition results of all 10 patterns are shown. Finally, factors that effect to the recognition results are discussed.

4.1 Testing data

For the NN learning, 20 banknote images have been applied as learning data of each banknote pattern. Next, to test performance of the proposed recognition system, 80 banknote images are used as testing data for each banknote pattern. These 80 testing data are images of 5 pieces of each banknote pattern that have different banknote number in various positions and 2 conveyed directions (upright and reversed). The 80 testing data of the 20HB pattern are shown in Appendix B. These 5 pieces of banknote are composed of usual, worn out, paint, and defect banknotes. Furthermore, all testing data are not the same as images which used for the NN learning.

4.2 Recognition results

After applying all testing data to the proposed system, output responses of all output nodes are calculated. Table 4.1 shows the output responses of first 20 testing data for the 20HB pattern which is pattern no. 1. The output value of the correct pattern should be nearly 1 because the desired output of the NN, which has been discussed on last chapter, is 1. The table shows a decided pattern number which is equal to the correct pattern – 1 on the most right column of the table.

To easily show the recognition results, all output responses are plotted as graphs (Figure 4.1 – 4.10). Most correct patterns have the maximum output values that located at top of graph. However, there are some NN rejected data which will shown by circling on Figure 4.3 and 4.5. For the 50HB pattern (Figure 4.3) that is banknote pattern no. 3, there are 4 NN rejected data which are data no. 37, 39, 67, and 79. For data no.37 and 79, although their

maximum output values were produced on 3rd output node, the NN rejected these 2 data because differences between their output value and the 5th node output values which are the second maximum output value were not larger than the threshold TH2. On the other hand, for data no. 39 and 67, the NN rejected those data because the maximum output values occurred at 5th output node instead of 3rd output node. On Figure 4.5, it is output response graph of the 100HB (pattern no. 5). There is only 1 NN rejected data at data no. 45. It was rejected because of the threshold TH2 for the most and second maximum output values difference. Table 4.2 shows the recognition results of those 5 NN rejected data.

Table 4.1 Output Response of First 20 Testing Data for the 20HB Pattern (Pattern No.1).

pattern no.	learning no.	output #1	output #2	output #3	output #4	output #5	output #6	output #7	output #8	output #9	output #10	decided pattern no.
1	1	**0.998543**	0.004667	0.000000	0.000042	0.000951	0.000073	0.000000	0.001679	0.003614	0.000000	0
1	2	**0.998302**	0.018243	0.000000	0.000079	0.003123	0.000013	0.000000	0.003292	0.001649	0.000000	0
1	3	**0.998640**	0.000937	0.000001	0.000013	0.000206	0.000668	0.000000	0.000791	0.015656	0.000001	0
1	4	**0.998631**	0.004886	0.000000	0.000098	0.000552	0.000071	0.000000	0.003009	0.002986	0.000000	0
1	5	**0.998337**	0.020321	0.000000	0.000009	0.002323	0.000006	0.000000	0.001974	0.004146	0.000000	0
1	6	**0.981631**	0.012491	0.000000	0.000073	0.001966	0.000002	0.000000	0.003847	0.006958	0.000000	0
1	7	**0.994590**	0.006237	0.000000	0.000060	0.001964	0.000012	0.000000	0.004002	0.005413	0.000000	0
1	8	**0.863355**	0.027479	0.000000	0.001141	0.018145	0.000000	0.000002	0.011021	0.002931	0.000000	0
1	9	**0.964968**	0.007978	0.000000	0.000221	0.002254	0.000003	0.000000	0.003714	0.004786	0.000000	0
1	10	**0.991462**	0.005457	0.000000	0.000038	0.001681	0.000019	0.000000	0.000993	0.007311	0.000000	0
1	11	**0.998919**	0.001850	0.000001	0.000009	0.000911	0.000239	0.000000	0.000636	0.006852	0.000001	0
1	12	**0.998919**	0.001870	0.000001	0.000006	0.000553	0.000283	0.000000	0.000469	0.010185	0.000001	0
1	13	**0.998429**	0.001915	0.000002	0.000002	0.000821	0.000177	0.000000	0.000195	0.019114	0.000005	0
1	14	**0.999679**	0.001905	0.000000	0.000008	0.002960	0.000286	0.000000	0.002421	0.005059	0.000000	0
1	15	**0.997638**	0.006851	0.000000	0.000045	0.004324	0.000044	0.000000	0.000976	0.002795	0.000000	0
1	16	**0.949972**	0.022541	0.000000	0.000390	0.013156	0.000001	0.000001	0.005629	0.002792	0.000000	0
1	17	**0.974004**	0.018973	0.000000	0.000684	0.025909	0.000003	0.000000	0.004548	0.001361	0.000000	0
1	18	**0.971737**	0.005226	0.000000	0.000505	0.015973	0.000010	0.000000	0.003917	0.002860	0.000000	0
1	19	**0.981619**	0.008620	0.000000	0.000115	0.007682	0.000002	0.000000	0.008841	0.005277	0.000000	0
1	20	**0.956847**	0.019738	0.000000	0.000816	0.037335	0.000001	0.000001	0.007332	0.002001	0.000000	0

Note: The decided pattern = the correct pattern − 1.

Table 4.2 Output Responses of 5 NN Rejected Data.

pattern no.	learning no.	output #1	output #2	output #3	output #4	output #5	output #6	output #7	output #8	output #9	output #10	decided pattern no.
3	37	0.000001	0.001017	**0.600361**	0.019222	**0.648878**	0.000011	0.001736	0.000000	0.000219	0.000000	4
3	39	0.000486	0.000009	**0.113521**	0.005359	**0.861221**	0.010701	0.003851	0.000000	0.000039	0.000000	4
3	67	0.000003	0.000953	**0.453975**	0.016053	**0.744447**	0.000013	0.001470	0.000000	0.000168	0.000000	4
3	79	0.000006	0.000019	**0.608188**	0.002710	**0.490790**	0.001309	0.007171	0.000000	0.001295	0.000000	4
5	45	0.000295	0.000120	**0.587968**	0.000019	**0.581917**	0.000524	0.000074	0.000000	0.016999	0.000000	2

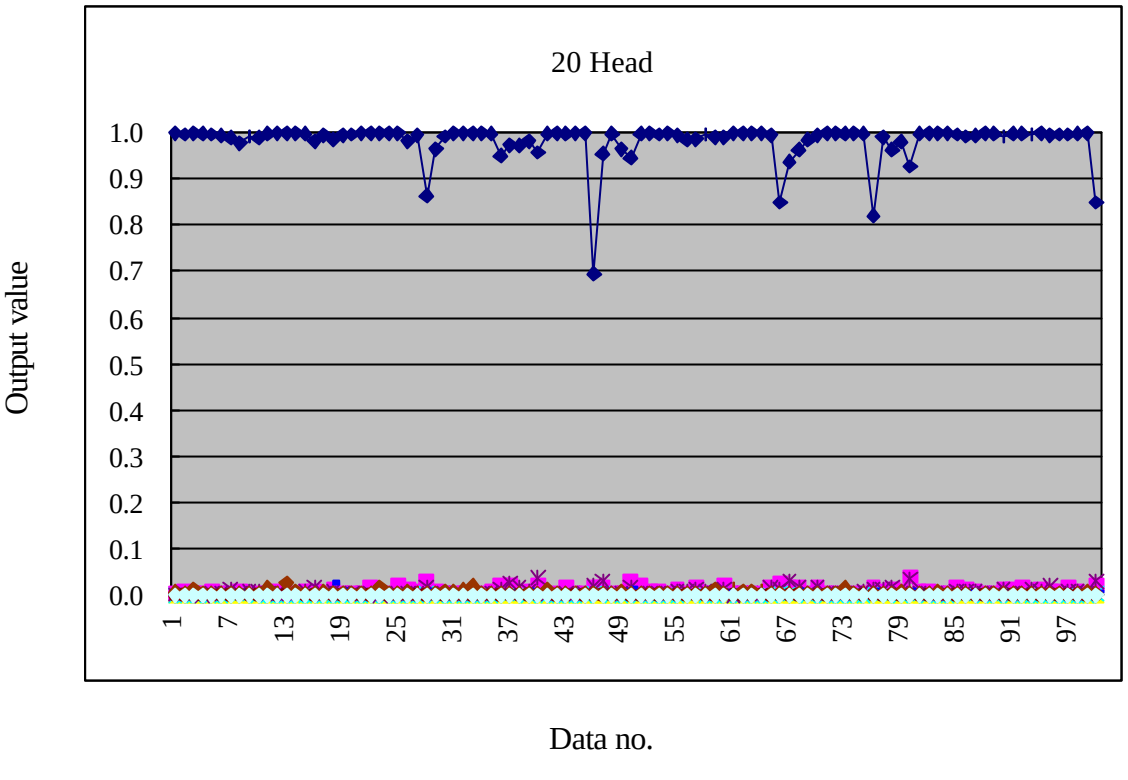


Figure 4.1 Output Responses of the 20HB pattern.

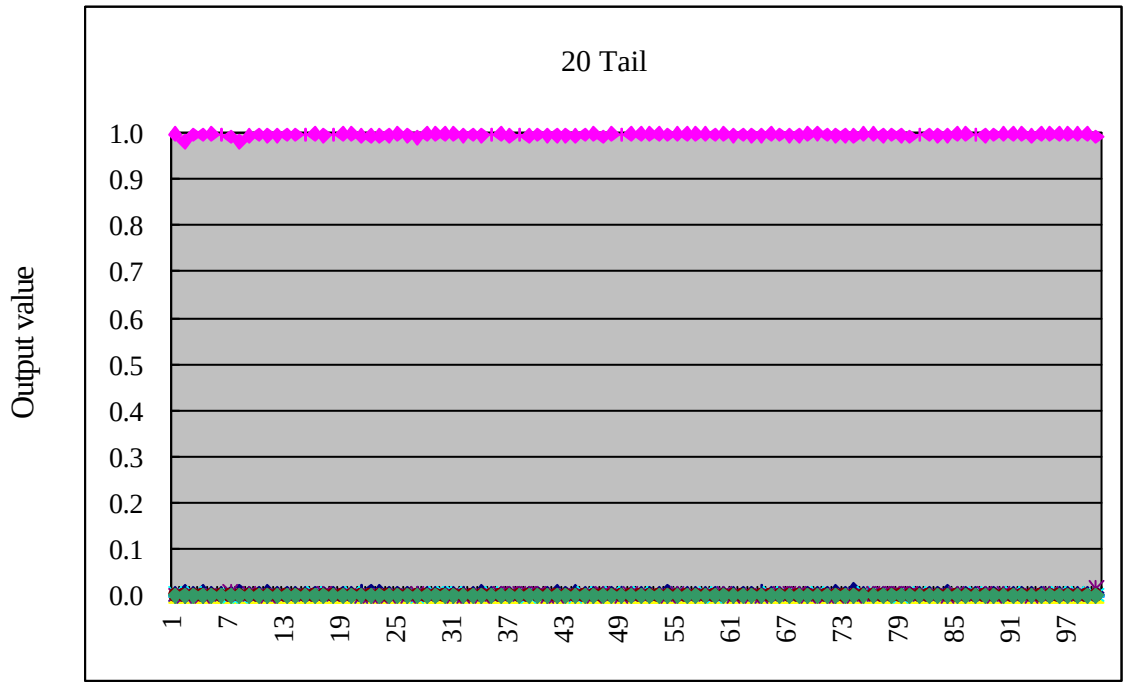


Figure 4.2 Output Responses of the 20TB pattern.

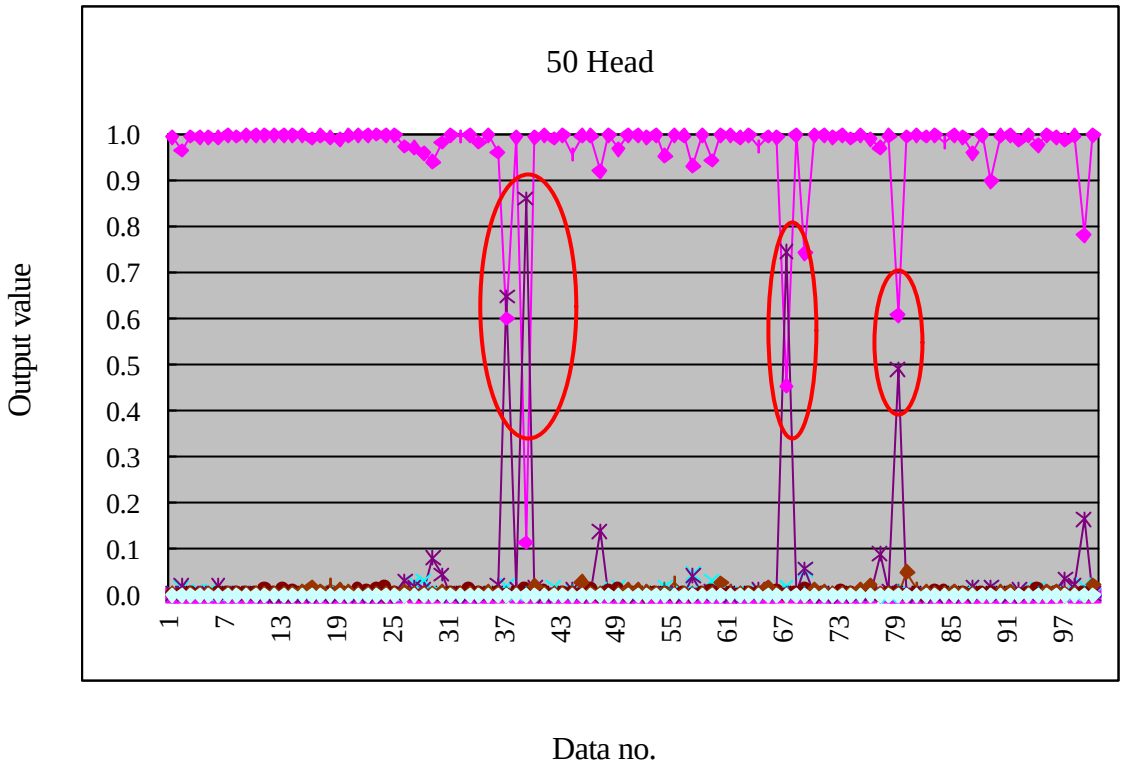


Figure 4.3 Output Responses of the 50HB pattern.

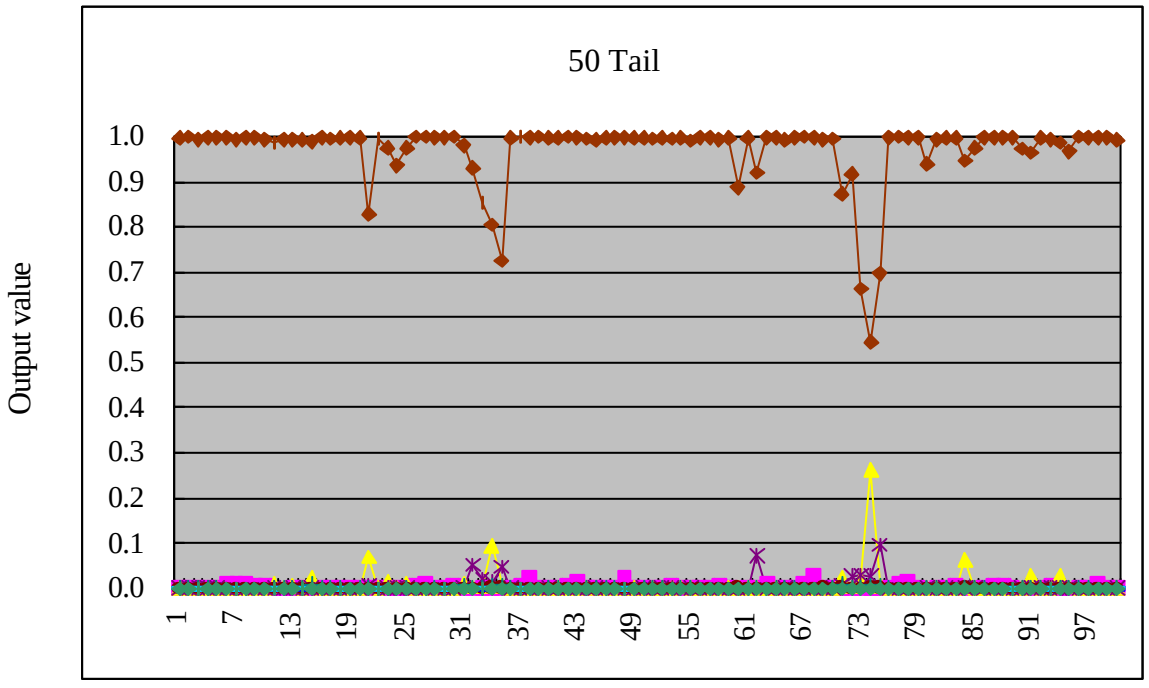


Figure 4.4 Output Responses of the 50TB pattern.

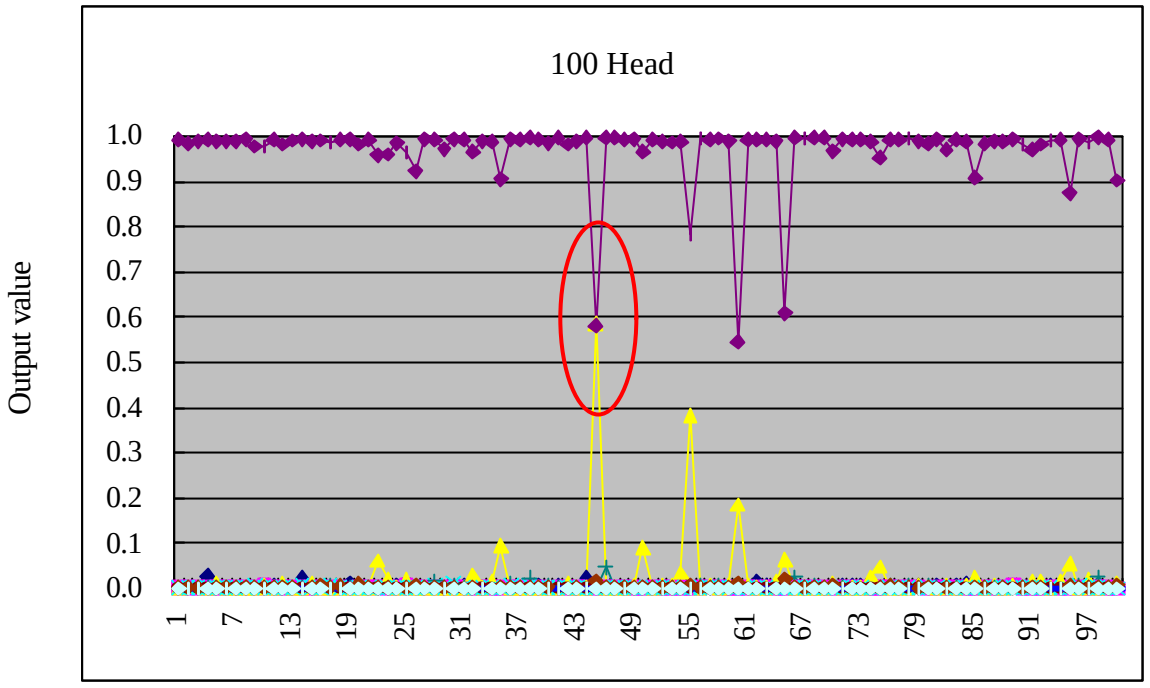


Figure 4.5 Output Responses of the 100HB pattern.

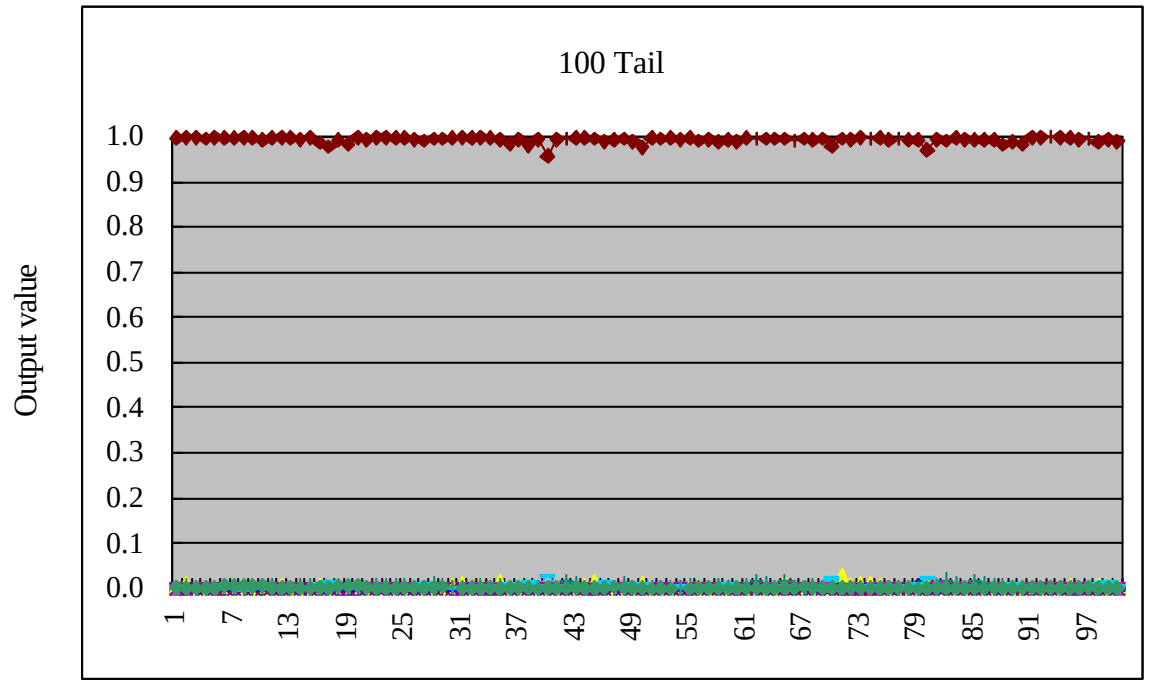


Figure 4.6 Output Responses of the 100TB pattern.

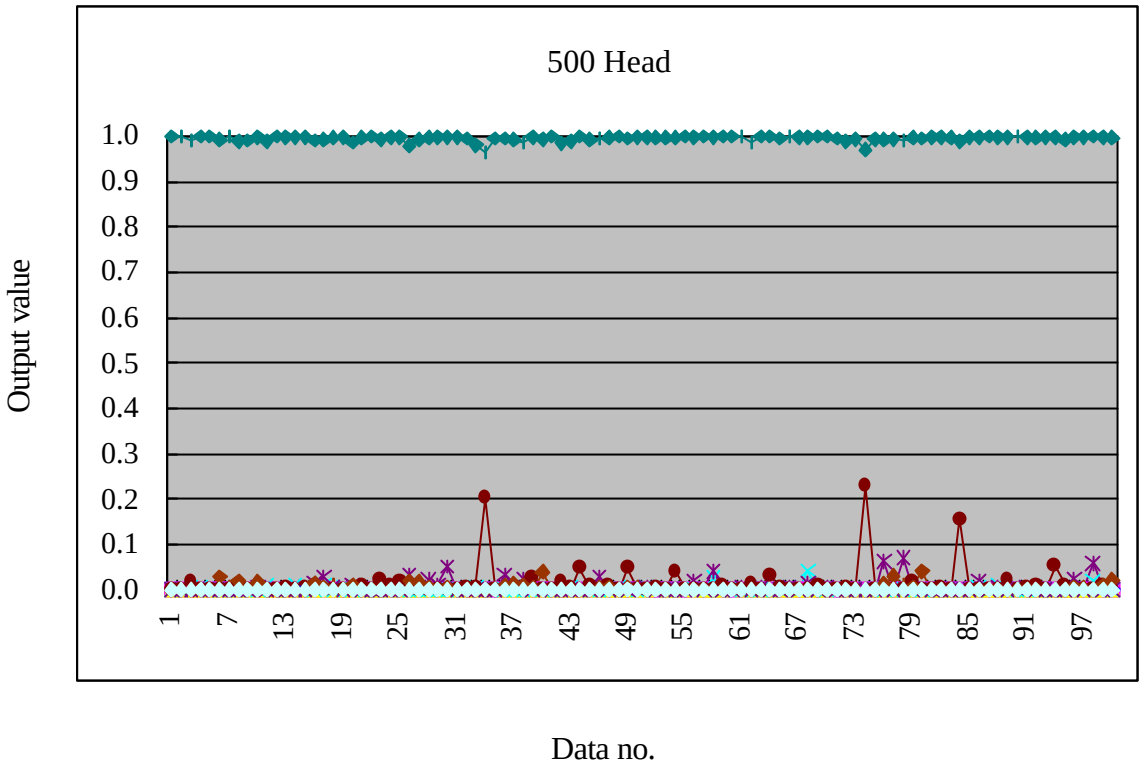


Figure 4.7 Output Responses of the 500HB pattern.

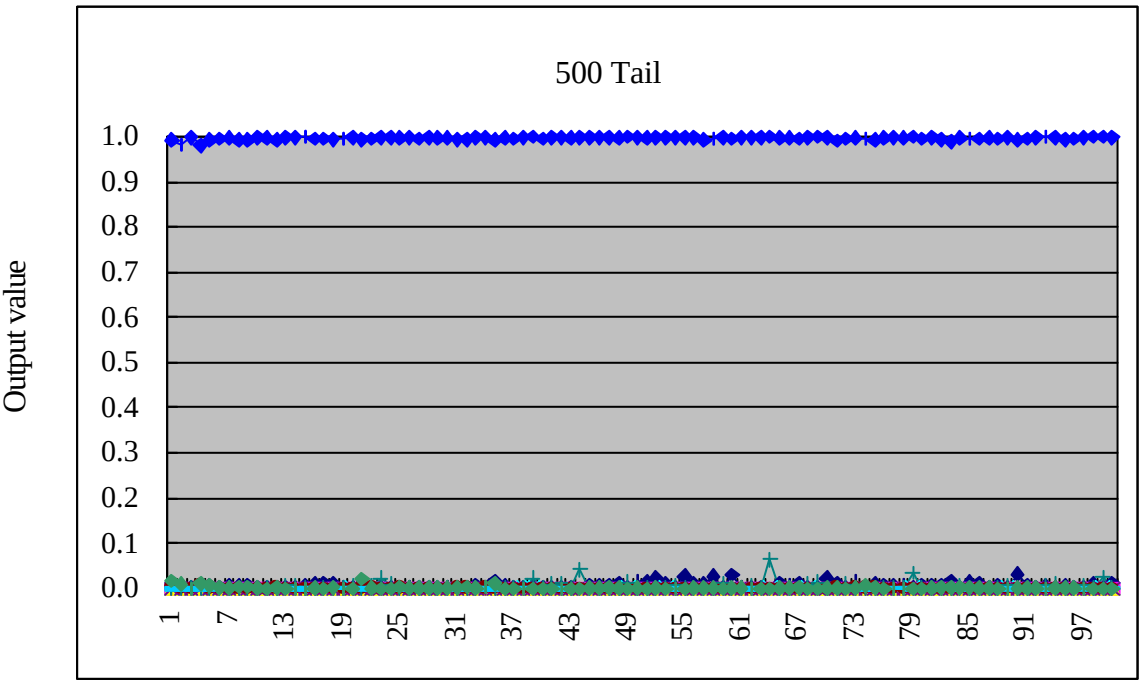


Figure 4.8 Output Responses of the 500TB pattern.

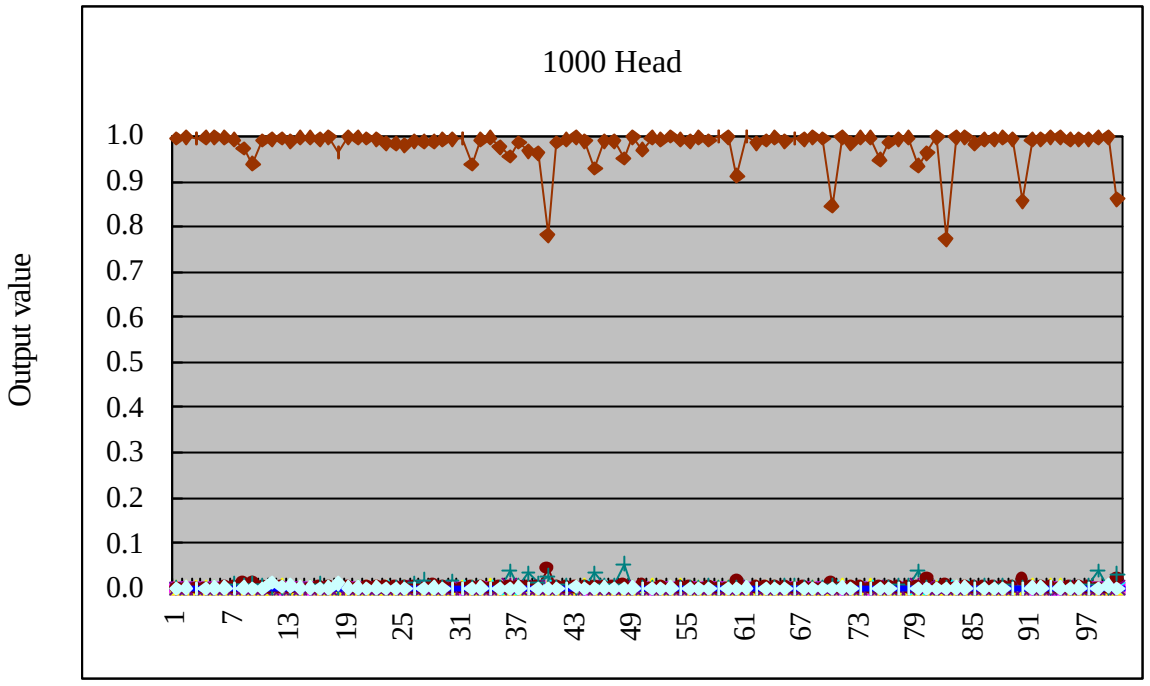


Figure 4.9 Output Responses of the 1000HB pattern.

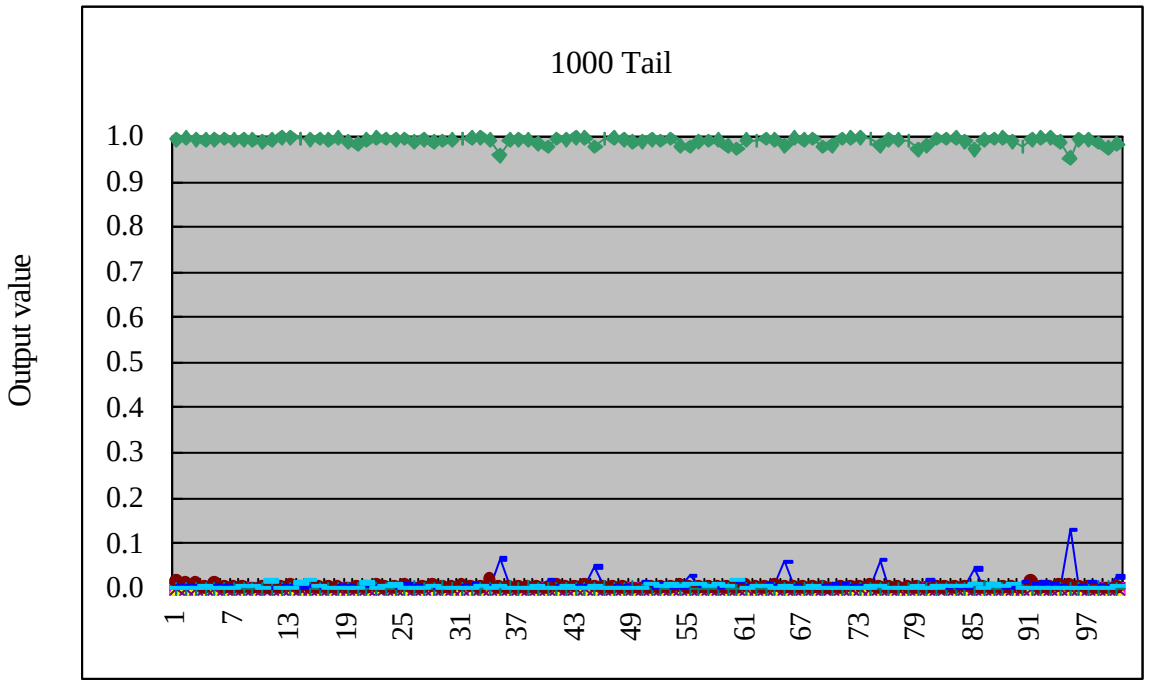


Figure 4.10 Output Responses of the 1000TB pattern.

Figure 4.1 – 4.10 show the output responses graph of all banknote patterns. In those figure, there are 100 data which first 20 data are learning data and rest of them are testing data. Moreover, the effectiveness of the proposed system is confirmed by recognition ability that calculated by

$$Recognition\ ability\ (\%) = \frac{no.\ of\ correct\ recognition\ testing\ data}{no.\ total\ testing\ data} \times 100 \qquad (4.1)$$

The recognition abilities of 10 Thai banknote patterns are shown in Table 4.3.

Table 4.3 Recognition Abilities of 10 Thai Banknote Patterns.

Pattern No.	Banknote pattern	Recognition ability (%)
1	20 HB	100.00
2	20 TB	100.00
3	50 HB	95.56
4	50 TB	100.00
5	100 HB	98.89
6	100 TB	100.00
7	500 HB	100.00
8	500 TB	100.00
9	1000 HB	100.00
10	1000 TB	100.00

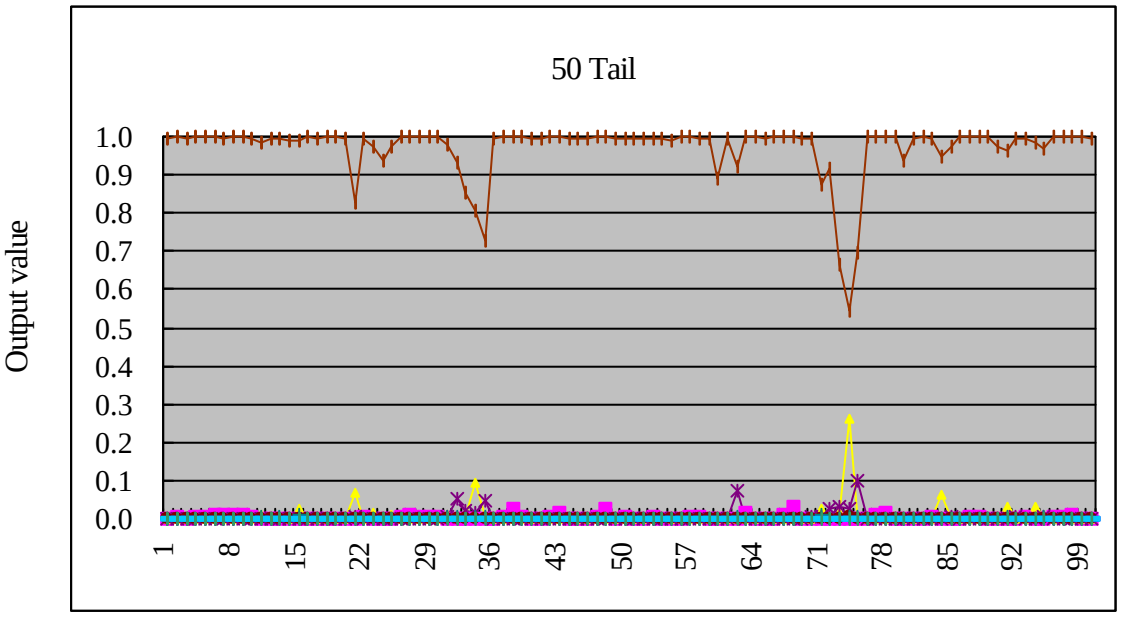
However, there are some variations on output responses as shown on Figure 4.1, 4.4, 4.5 and 4.9. Therefore, various factors that effected on the output response are discussed on the next section.

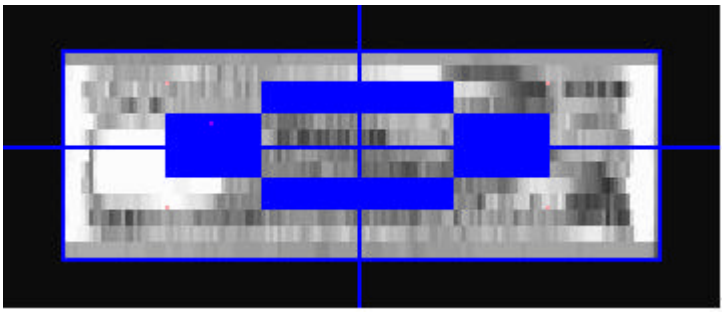
4.3 Recognition result effects

From the experiment, it seems that there are 2 important factors which effect to the output response of the proposed recognition system. First important factor is the mask set. Another key factor is the threshold values for the edges and center detection of banknote. These 2 key factors are discussed on subsection 4.3.1 and 4.3.2. Moreover, there still are other factors that affect the output responses and it will be discussed on the subsection 4.3.3.

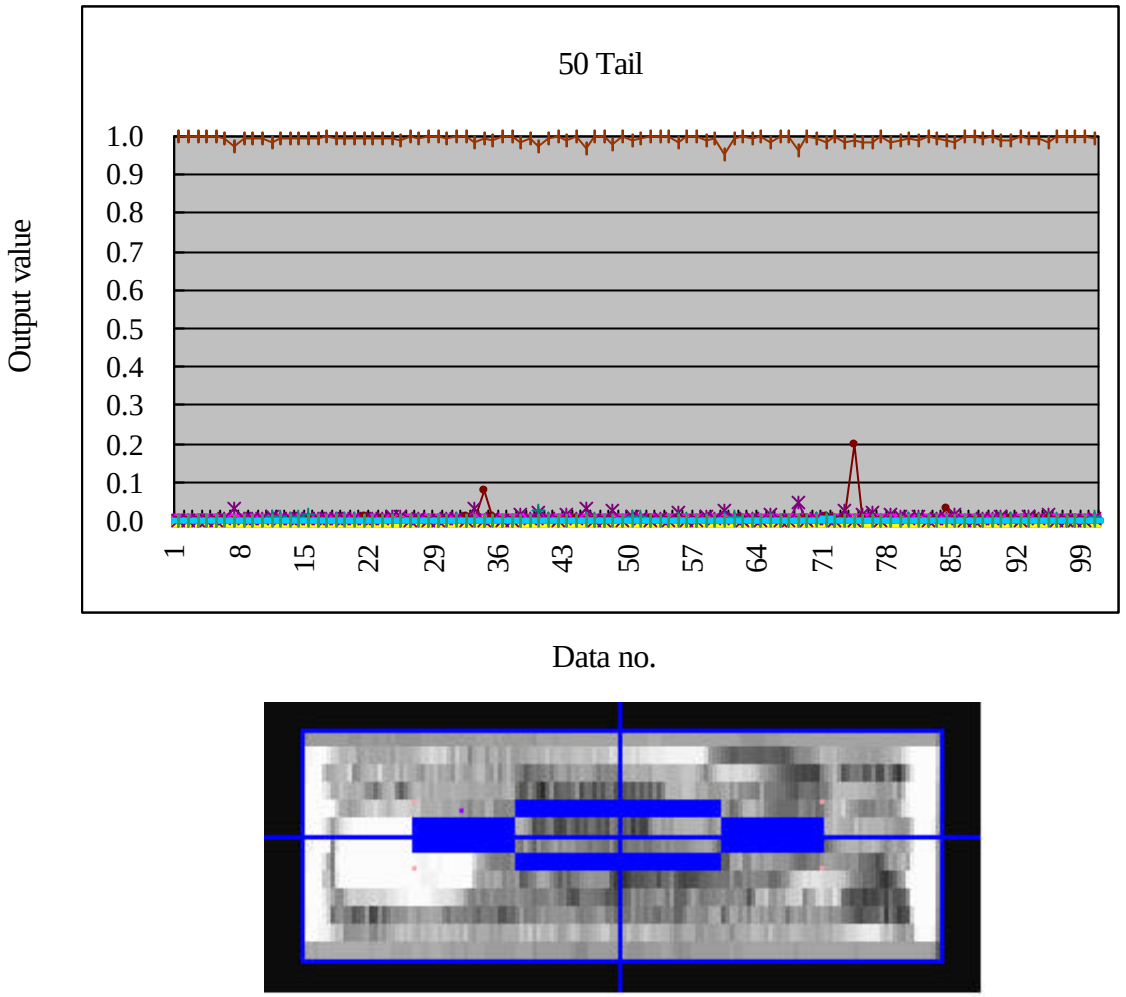
4.3.1 Mask set

The most important factor is the mask set. Since, the slab values, which are the characteristics of banknote and used as inputs of the NN, are extracted from banknote image by using the mask set that has 50 different mask patterns. Then, different mask sets give different slab values. By different slab values, the output responses of the same banknote pattern, which applying dissimilar mask set, also differ as shown in Figure 4.11. Therefore, to get the effective system, we should consider the mask set carefully. For example, areas that have same pixel values in all banknote patterns should be neglected by locating mask. Another locating method of mask set, most pixels of banknote image are marked except only some pixels that their pixel values in all banknote patterns are obviously distinct. Finally, the system yields the effective slab values for the NN learning and recognition. From Figure 4.11, two different mask set with the same mask pattern but different mask offset were applied to show effect of the mask set.





(a) 8×2 block size and (-48, -4) offset

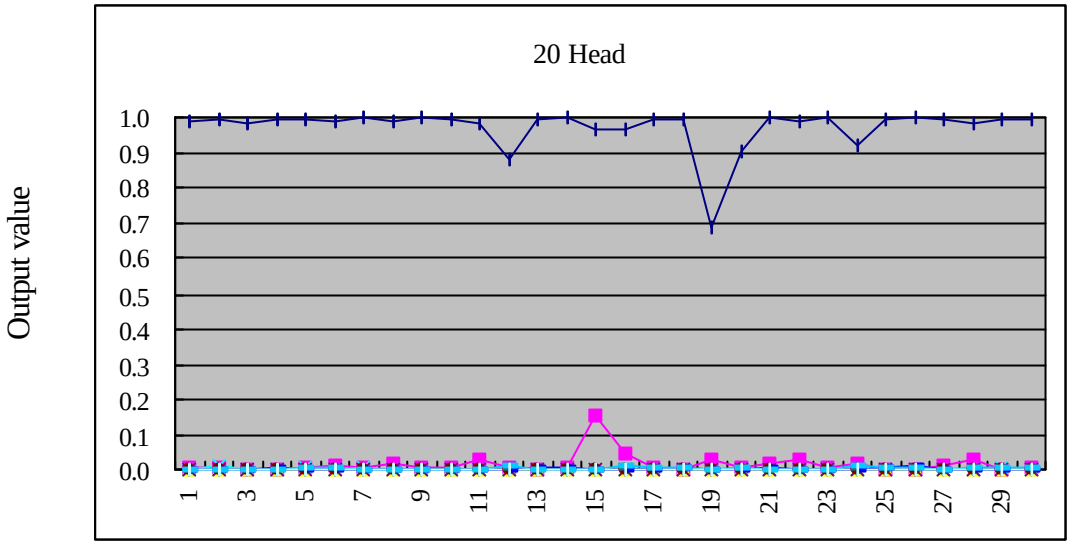


(b) 8×1 block size and (-48, -2) offset

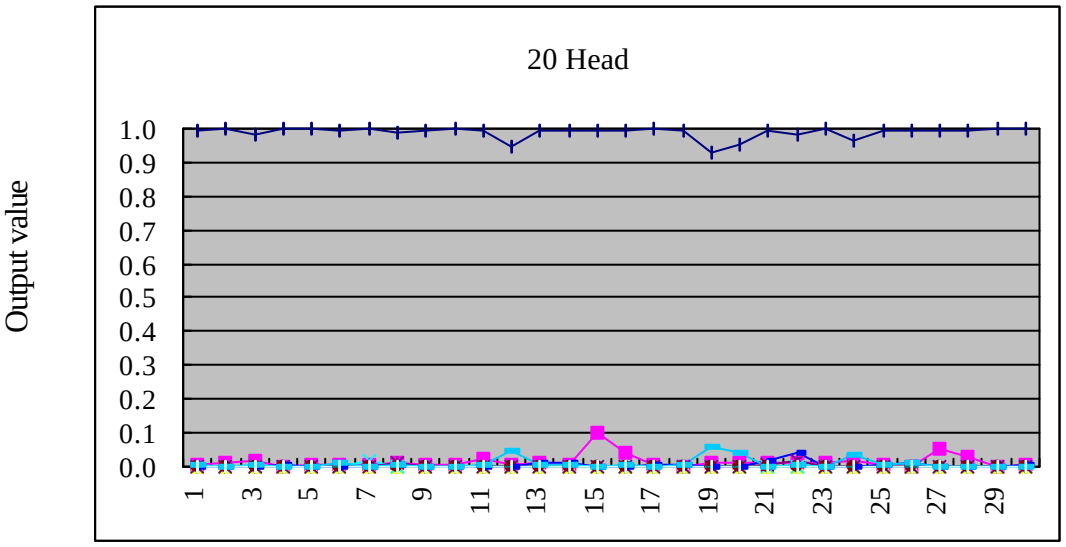
Figure 4.11 Effect of the Mask Set on the Output Response.

4.3.2 Threshold values for the edges and center detection

As discussed in chapter 3, 2 threshold values (threshold "cho" and "tan") are selected manually for the edges and center detection of banknote. With the suitable threshold values, the edges and center of banknote image are detected accurately. However, sometimes, there are many pairs of these two threshold values that can detect edges and center. Therefore, we have to select the threshold values carefully to yield the most suitable threshold values because the center of banknote image effects to mask areas of the mask set. Figure 4.12 shows different output responses of the 50TB pattern with two different pair of threshold values a) threshold "cho" = 15 and threshold "tan" = 35 and b) threshold "cho" = 10 and threshold "tan" = 35, although both of them can detect the same edges of the banknote image. 30 data of banknote images were applied to show effect of the threshold values to the output response.



(a) Threshold Values (15,35)



(b) Threshold values (10,35)

Figure 4.12 Threshold Values Effect to the Output Response of the 20HB Pattern.

4.3.3 Other factors

As well as mask and threshold values factors, there still are other 2 factors that make the variation of the output response. First, noises, which sometimes occur during the image

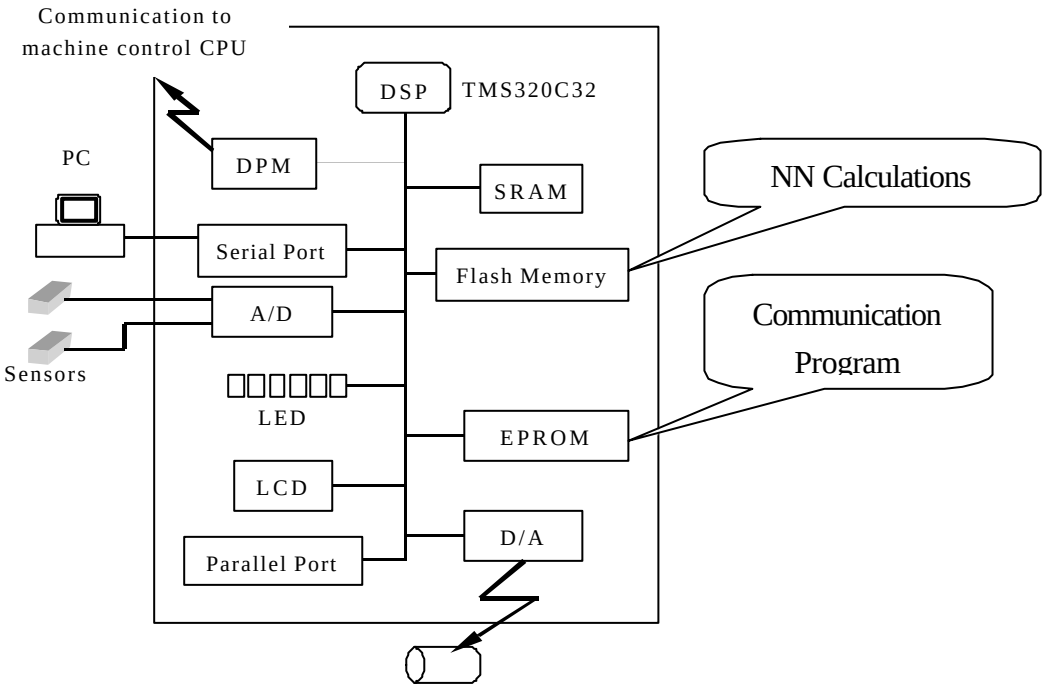
scanning process, make some mistakes in the banknote image. Another factor is some defects on the original banknote itself. For example, very old or extremely destroyed banknote, which may cause very low gray scale level and make wrong edges detection. Although the output responses vary by those factors, the NN recognition system still can recognize banknotes accurately.

Chapter 5: Recognition Using the DSP Unit

After finish PC experiments, the proposed recognition system is applied to the DSP unit for real banking system. The DSP unit configuration and it operation are explained on the first section of this chapter. This DSP unit can execute both learning and recognition procedure by itself that those procedures will be described in detail on section 5.2 and 5.3. Finally, to confirm efficiency of the DSP unit, some banknotes are applied and its recognition results are shown.

5.1 DSP configuration

The hardware construction and overview of the DSP unit is shown in Figure 5.1. The most significant part of the DSP unit for learning and recognition is flash memory. Because the NN calculations that are the feedforward and calculation program the error backpropagation are saved on the flash memory. These 2 NN calculations are main operations of the NN which have been discussed on chapter 3. Moreover, for comfortable in the DSP unit controlling, the communication program is implemented on EPROM of the DSP unit as shown in Figure 5.1.



(a) Configuration of the DSP Unit.

(b) Overview of the DSP Unit.

Figure 5.1 Configuration and Overview of the DSP Unit.

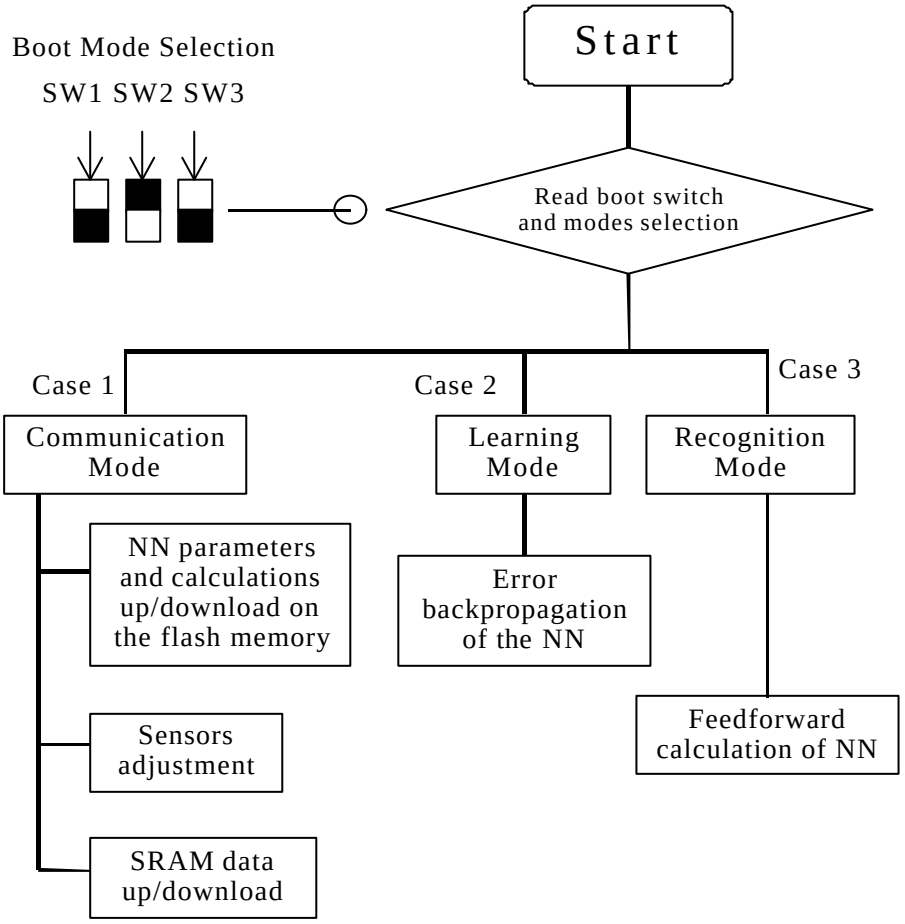
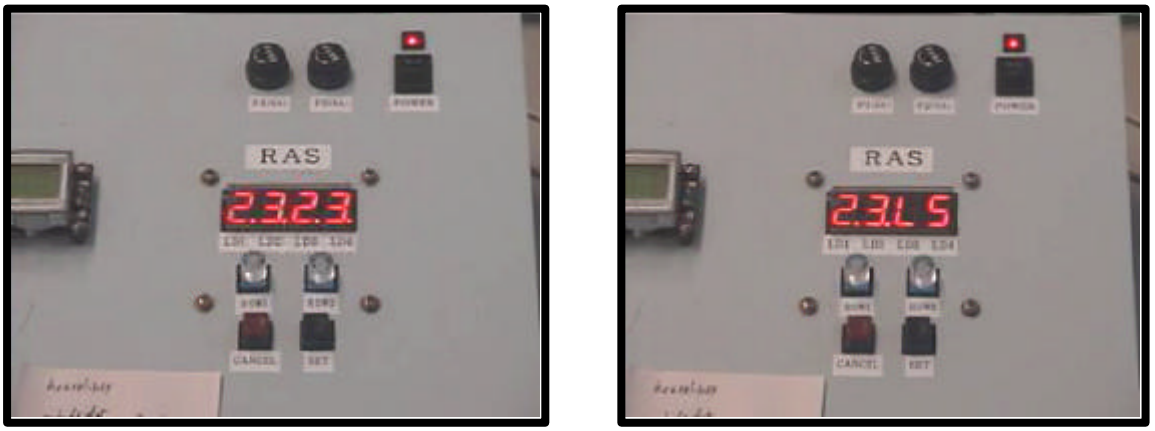


Figure 5.2 Three Operation Modes of the DSP Unit.

With 2 NN calculations and the communication program, this DSP unit can perform 3 operation modes which are i) communication mode, ii) learning mode, and iii) recognition mode. The operation mode is selected by using switches on the DSP unit as shown in Figure 5.1. First, by using communication mode, the NN initial weights, a mask set, banknote images and another parameters, are transferred and saved on the flash memory of the DSP unit. Second, learning mode, the NN error backpropagation program and learning data, which saved on the flash memory, are loaded to SRAM to execute learning operation. For this research, there are 2 types of learning that are i) the initial learning and ii) the continuous learning which will be described on section 5.3. During learning, the NN weights are renewed on flash memory several times until learning is finished. Finally, on recognition mode, the NN feedforward calculation program, final converged NN weights, and testing or real target data are also loaded to SRAM to execute recognition procedure. Moreover, during execute all modes, LED and LCD of the DSP unit, and PC screen show the DSP status, learning status, recognition results, and error. Figure 5.3 shows the DSP unit status in the communication and learning mode by LED. Details of each operation mode are presented on next sections.



(a) Communication mode (b) Learning mode

Figure 5.3 The DSP Unit Status on LED.

5.2 Data transportation

By using the communication mode, the DSP unit can be controlled via PC and its status, learning status and recognition results can be shown on a PC screen, which are comfortable for users in commercial market. Another function of the communication mode is data transferring. Data, which must be transferred to the DSP unit for learning and recognition, are

1) A mask set that used for the slab-value extraction,
2) The NN structure parameters such as learning conditions and learning rate,
3) The NN initial weights, which yield from the NN learning procedure on PC,
4) Learning banknote images
5) Testing banknote images or real banknotes.

All data are saved on the flash memory of the DSP unit for learning and recognition of the DSP unit. Figure 5.4 shows a main PC screen of the DSP unit controlling.

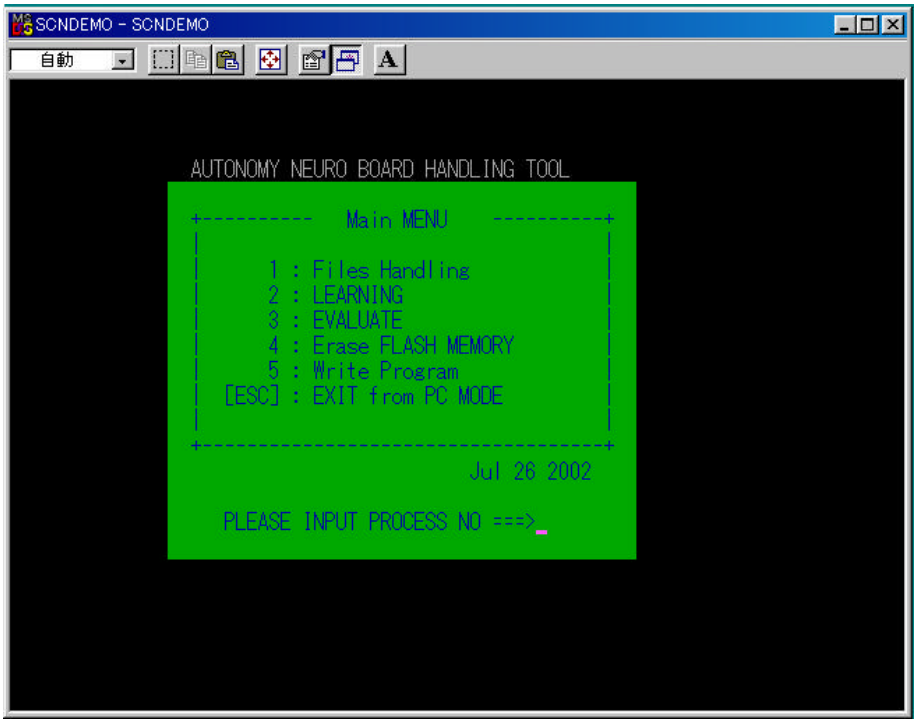


Figure 5.4 Main PC Screen for the DSP Unit Controlling.

5.3 Learning procedure

The DSP unit begins to execute the NN learning by loading data from the flash memory to SRAM which is the main memory of the DSP unit. The data that necessary for learning are i) a mask set, ii) the NN initial weights, iii) the NN structure parameters and iv) learning banknote images. At the same time, the NN calculation of the error backpropagation, which is the most important part for learning, is also loaded to the SRAM. Then, the DSP unit executes learning procedure and the initial NN weights are adjusted until reach one of learning conditions. As mentioned, there are 2 types of the NN learning by the DSP unit which are i)

the initial learning and ii) the continuous learning. For the initial learning, the NN weights are adjusted by starting from random initial weights. Another, the continuous learning, this learning is very useful when there are some modifications of target data because modified data are added to the DSP unit to adjust the NN weights again. By using the continuous learning, the DSP unit starts to adjust the NN weights by starting from the last converged NN weights from PC. Moreover, during the learning operation, the PC screen also presents the learning status by showing error graph, current error value and current iteration number as shown in Figure 5.5. After finish learning, the converged NN weights are saved on the flash memory of the DSP unit for recognition.

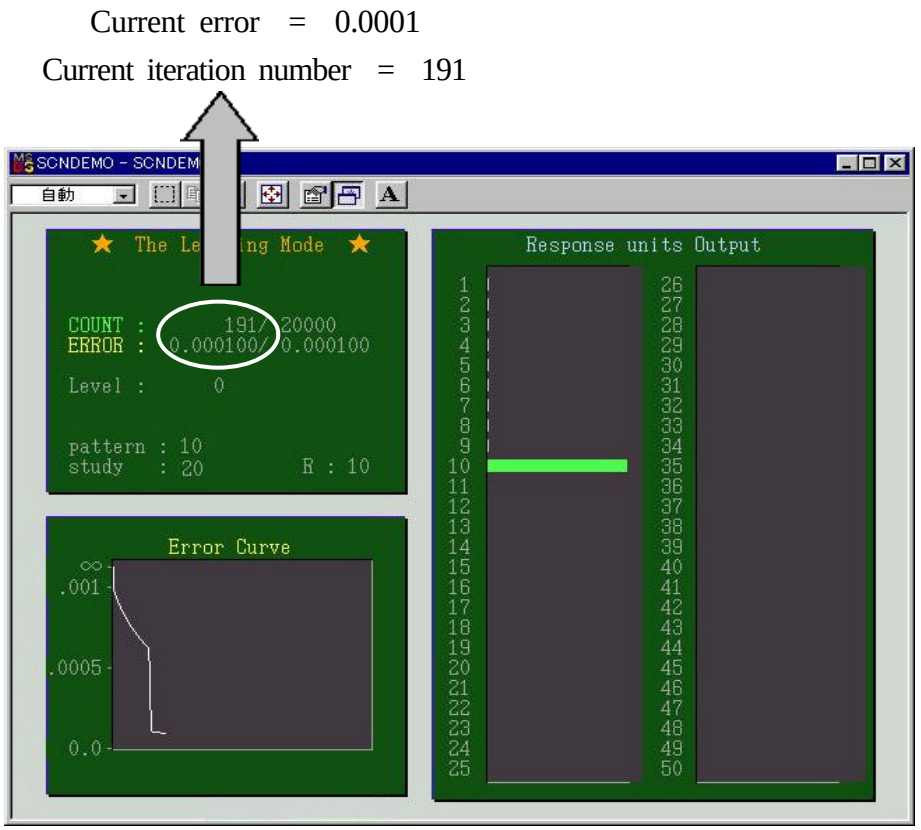


Figure 5.5 Learning Screen of the DSP Unit.

5.4 Recognition procedure

The last operation mode is the recognition mode. A mask set, the converged NN weights, banknote images for recognition, and the NN feedforward calculation that saved on the flash memory are loaded to the SRAM. Then, the DSP unit starts to perform the recognition procedure and the recognition results are shown on the PC screen (Figure 5.6). On the

recognition screen, the maximum output value of the NN and its pattern number are shown. From Figure 5.6, the maximum output value is 0.979739 and the correct pattern number of this recognition is pattern no. 1.

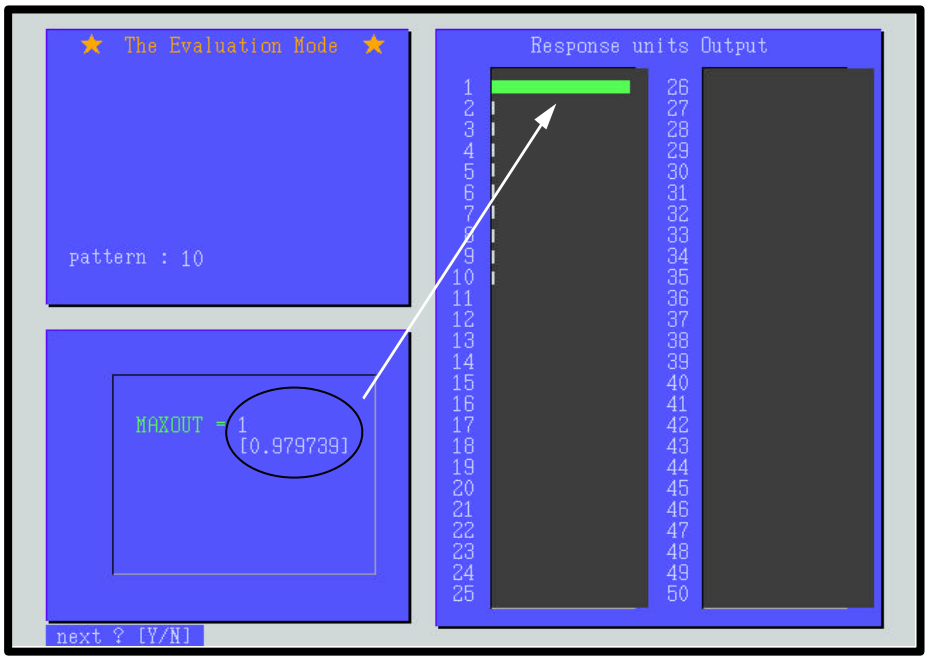


Figure 5.6 Recognition Screen of the DSP Unit.

5.5 Recognition results of the DSP unit

For the DSP unit experiment, 30 images of each banknote pattern were transferred to the DSP unit. These 30 images consist of 20 images for learning and only 10 images for evaluating the recognition ability of the DSP unit. Table 5.1 shows experimental conditions of both PC and DSP unit experiment. A mask set that applied to the DSP unit is the same as one which used in the PC experiment. There is only one difference between the DSP unit and PC experimental conditions that is amount of testing data. Although testing data amount of the DSP unit is only 10, these testing data of the DSP unit are identical to first 10 testing data of all 80 testing data for the PC experiment. Therefore, after completed the recognition process, the output responses of 10 testing data for the DSP unit experiment are similar to the output responses of first 10 testing data of the PC experiment.

Furthermore, to confirm possibility of the continuous learning on the DSP unit, both initial and continuous learning were performed and several banknotes were applied as testing

data for recognition. First, the initial NN learning, 10 learning data per banknote patterns were used for learning procedure. This learning type adjusts the NN weights from random initial weights. When the learning was completed, converged NN weights were saved on the flash memory of the DSP unit. The recognition ability of this learning is shown on Table 5.2.

Table 5.1 The Experimental Conditions of PC and the DSP Unit.

Experimental Conditions	PC	DSP Unit
Mask no. (Input node no.)	50	50
No. of banknote types (Output node no.)	10	10
Hidden node no.	30	30
Learning data / pattern	20	20
Testing data / pattern	80	10
Minimum mean square error	0.0001	0.0001
Maximum iteration no.	20000	20000
Learning algorithm	Backpropagation method	

Second, new 10 learning data of each banknote pattern were transferred to the DSP unit and then there are totally 20 learning data per banknote pattern. Next, the continuous learning starts to adjusting the NN weights from last converged NN weights of the initial learning. After learning was converged, the converged NN weights were also saved on the flash memory for recognition process. Table 5.2 shows the recognition ability of the continuous learning. By using the continuous learning, recognition ability of the system was increased about 0.19% that means the DSP unit could recognize banknote data more accurately. Because of its advantage, the continuous learning has been proposed for learning on the DSP unit.

However, the learning process of the DSP unit takes more time than the learning on PC because of the machine clock, although the error backpropagation calculation was implemented in C language on both PC and the DSP unit. PC machine clock is more than 600MHz while the DSP unit operates by only 33.3 MHz machine clock. However, if there are some data modifications, the DSP unit can perform the continuous learning by itself from the converged NN weights of the PC. From the recognition ability in Table 5.2, it seems that the recognition ability of the DSP unit is more effective than one of the PC because of the continuous learning. Thus, sometimes, the learning process is performed on PC to yield the converged NN weights and save those weights before transfer it to the DSP unit. After the DSP unit received the converged NN weights, it uses those weights for recognition process

only. By this way, it save time for the DSP unit operation.

Table 5.2 Recognition ability of Thai banknote patterns.

Banknote Type	**Recognition Ability (%)**	
	Initial Learning	**Continuous Learning**
	10 learning data	20 learning data
20 HB	100.00	100.00
20 TB	100.00	100.00
50 HB	95.00	95.56
50 TB	98.89	100.00
100 HB	98.75	98.89
100 TB	100.00	100.00
500 HB	100.00	100.00
500 TB	100.00	100.00
1000 HB	100.00	100.00
1000 TB	100.00	100.00
Average	**99.26**	**99.45**

Chapter 6: Conclusion

Thai banknotes have been applied to the NN recognition system. These banknotes were recognized by using the NN with image preprocesses and the DSP unit. In this research, the axis-symmetry mask set was applied for the slab-value extraction of all banknote images. With this axis-symmetry mask, the NN performed recognition procedure for 10 banknote patterns only. Using the DSP unit, the system can execute learning, especially the continuous learning, and recognition by itself as the banknote reader and sorter machines. Moreover, to confirm the recognition ability of the system, several banknote images were applied in both PC experiment and the DSP unit experiment.

From the experimental results, it seems that this recognition system is effective for Thai banknotes because all recognition abilities of each banknote patterns are higher than 90%. Therefore, this system will lead to Thai banknotes recognition machines in real banking system which useful for Thailand commercial system. However, this system still has problem that is the variation on the output responses. This problem may effects to the recognition ability of the system when applied in real banking machines. As discussed before, there are 3 main factors which effect to these variations, especially the mask set. Therefore, in the future, a genetic algorithm (GA) will be applied for mask position selection instead of the mask randomly selection to improve the recognition ability for Thai banknote recognition.

References

[1] Bourke, P., "Sigmoid Function", The Swinburne Center for Astrophysics and Supercomputing < URL: http://astronomy.swin.edu.au/~pbourke/analysis/sigmoid >, 2002.

[2] Hjorteland, T., "Method of Steepest Descent", < URL: http://trond.gothamnighnts.com// Thesis/node26.html >, 2002.

[3] Humphrys, M., "Single-layer Neural Networks (Perceptrons)", Dublin University < URL: http://www.compapp.dcu.ie/~humphrys/Notes/Neural/single.neural.html >, 2002.

[4] Kittipatimakorn, P., Sakulphramana, R., and Sakoobunthu, L., "Auto Tuning for Computer Numerical Control (CNC) System by Using Neural Network Theory" for Bachelor Graduation Report, Sirindhorn International Institute of Technology, Thammasat University, 2000.

[5] Kosho, B., "Neural Networks and Fuzzy System (A dynamical system approach to machine intelligence)", Prentice-Hall, 1992.

[6] Martin, T.H., Howard, B.D., and Mark, B., "Neural Network Design", PWS Publishing Company, 1996.

[7] Nishikage, T., and Takeda, F., "Axis-Symmetrical Masks Optimized by GA for Neuro-currency Recognition and Their Statistical Analysis", Proceedings of World Multi-Conference on Systemics, Cybernetics and Informatics, Vol.2, pp.308-314, 1998.

[8] Russell, C.E., and Roy, W.D., "Neural Network PC Tools", Academic Press, 1990.

[9] Sakoobunthu, L., Takeda, F., and Sato, H., "Thai Banknote Recognition Using Neural Network and Applied to the Neuro-Recognition Board", The International Workshop on Signal Processing Application and Technology, pp.107-114, 2002.

[10] Sakoobunthu, L., Takeda, F., and Sato, H., "Online Continuous Learning by DSP Unit", SICE System Integration Division Annual Conference (SI2002), Vol.3, pp. 321-322, 2002.

[11] Takeda, F., Omatu, S., and Onami, S., "Recognition System of US Dollars Using a Neural Network with Random Masks", Proceedings of the International Joint Conference on Neural Networks, Vol.2, pp. 2033-2036, 1993.

[12] Takeda, F., and Omatu, S., "A Neuro-System Technology for Banknote Recognition", Japan/USA Symposium on Flexible Automation, Vol.2, pp.1511-1516, 1994.

[13] Takeda, F., and Omatu S., "High Speed Paper Currency Recognition by Neural Networks", IEEE Trans. on Neural Networks, Vol.6, No.1, pp.73-77, 1995.

[14] Takeda, F., and Omatu, S., "A Neuro-Paper Currency Recognition Method Using

Optimized Masks by Genetic Algorithm", Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Vol.5, pp. 4367-4371, 1995.

[15] Takeda, F., Nishikage, T., and Matsumoto, Y., "Characteristic Extraction of Paper Currency using Symmetrical Masks Optimized by GA and Neuro-Recognition of Multi-National Paper Currency", Proceedings of IEEE World Congress on Computation Intelligence, Vol.1, pp.634-639, 1998.

[16] Takeda, F., Omatu, S., and Matsumoto, Y., "Development of High Speed Neuro-Recognition Board and Application for Paper Currency", The International Workshop on Signal Processing Application and Technology, pp.49-56, 1998.

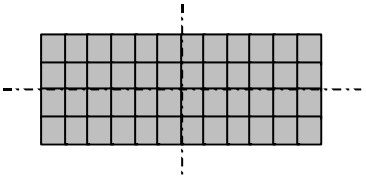
[17] Takeda, F., Nishikage, T., and Omatu, S., "Banknote Recognition by Means of Optimized Masks, Neural Network, and Genetic Algorithm", Engineering Applications of Artificial Intelligence 12, pp.175-184, 1999.

[18] Takeda, F., Nakahara, M., Ichiryuu, Y., and Uchida, H., "Automatic Neuro-Recognition Board for Paper Currency", Signal Processing Applications and Technology, pp.85-90., 2000.

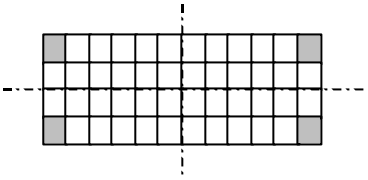
[19] Widrow, B., Winter, R.G., and Baxter, R.A., "Layered Neural Nets for Pattern Recognition", IEEE Transaction Acoustic, Speech & Signal Preprocessing, Vol.36, No.7, pp.1109-1118, 1988.

Appendix A

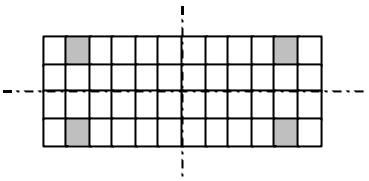
An Axis-symmetry Mask Set with 50 Mask Patterns



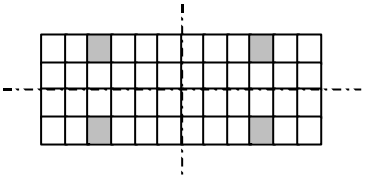
Pattern No. 1



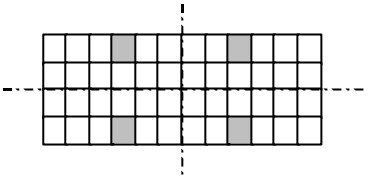
Pattern No. 2



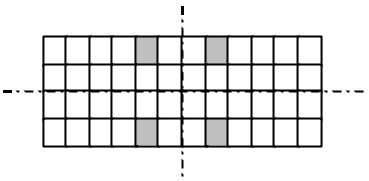
Pattern No. 3



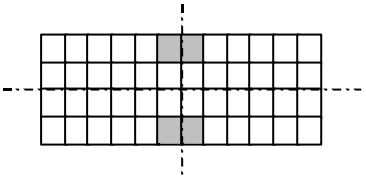
Pattern No. 4



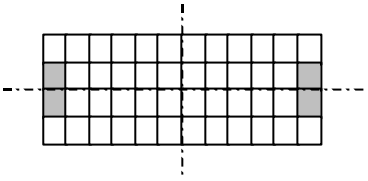
Pattern No. 5



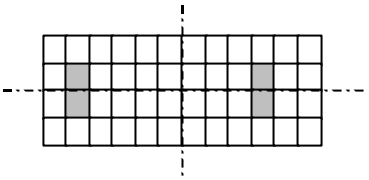
Pattern No. 6



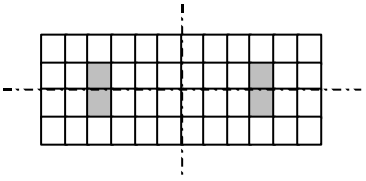
Pattern No. 7



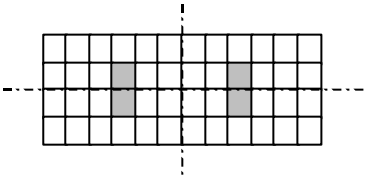
Pattern No. 8



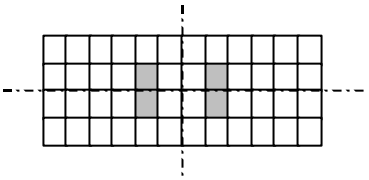
Pattern No. 9



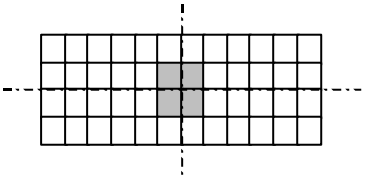
Pattern No. 10



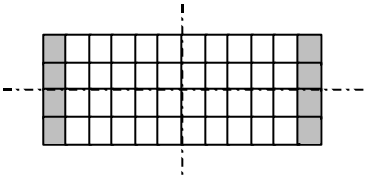
Pattern No. 11



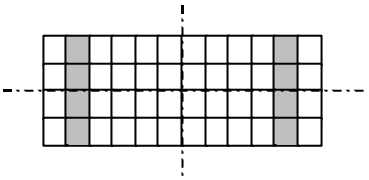
Pattern No. 12



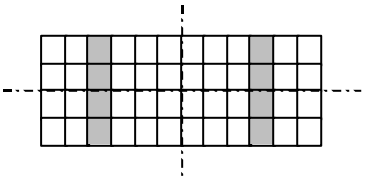
Pattern No. 13



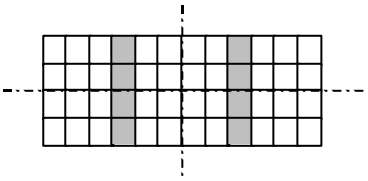
Pattern No. 14



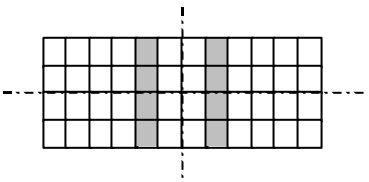
Pattern No. 15



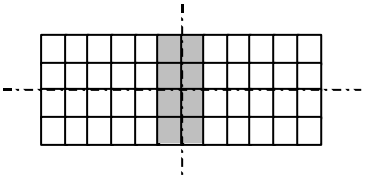
Pattern No. 16



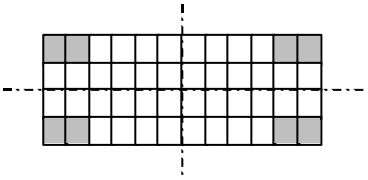
Pattern No. 17



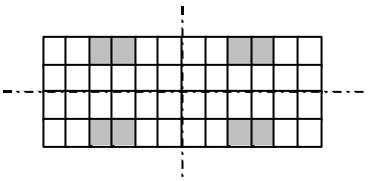
Pattern No. 18



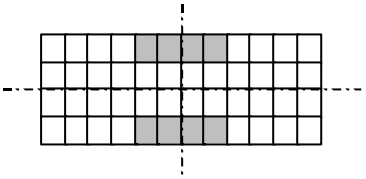
Pattern No. 19



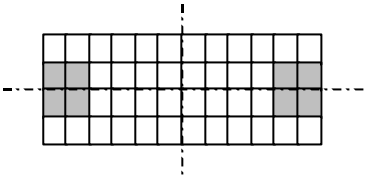
Pattern No. 20



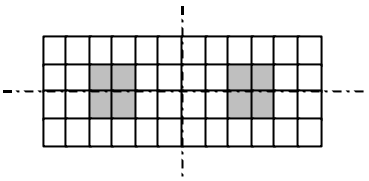
Pattern No. 21



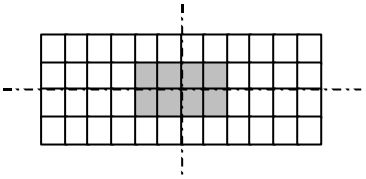
Pattern No. 22



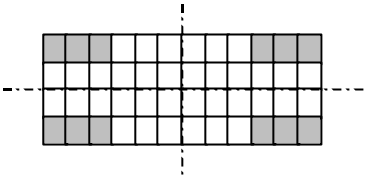
Pattern No. 23



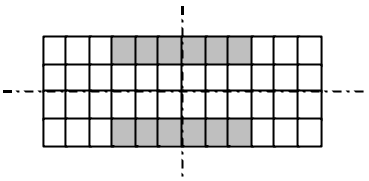
Pattern No. 24



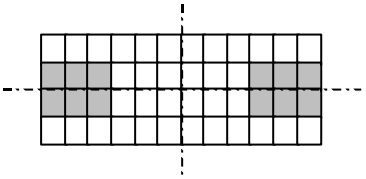
Pattern No. 25



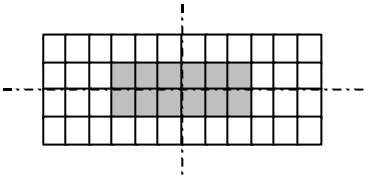
Pattern No. 26



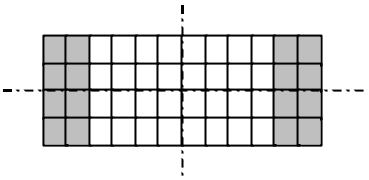
Pattern No. 27



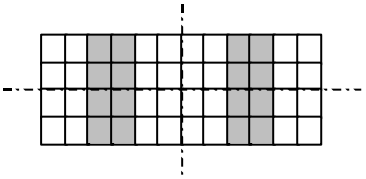
Pattern No. 28



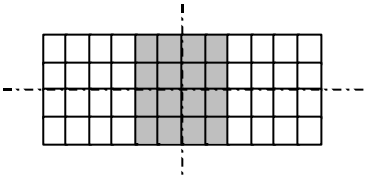
Pattern No. 29



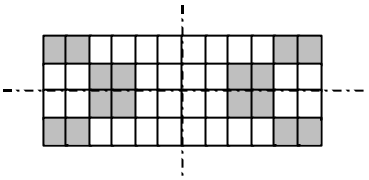
Pattern No. 30



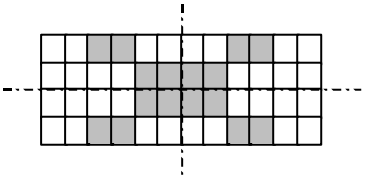
Pattern No. 31



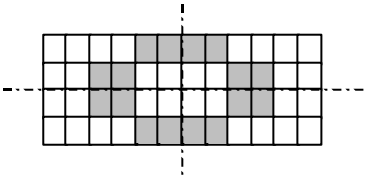
Pattern No. 32



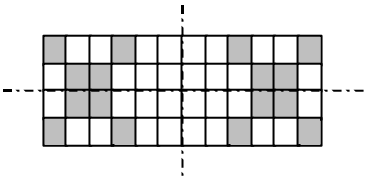
Pattern No. 33



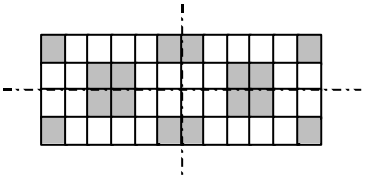
Pattern No. 34



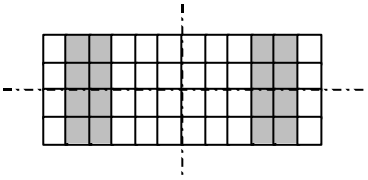
Pattern No. 35



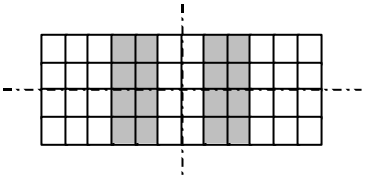
Pattern No. 36



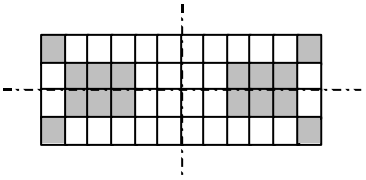
Pattern No. 37



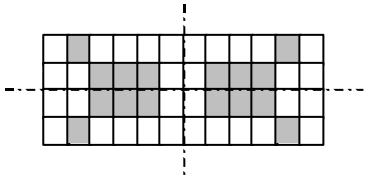
Pattern No. 38



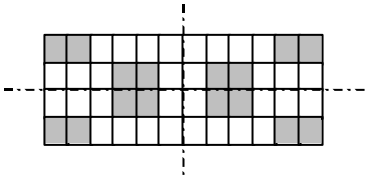
Pattern No. 39



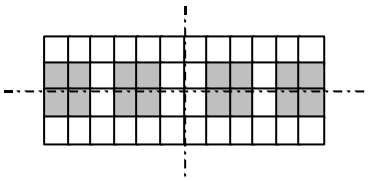
Pattern No. 40



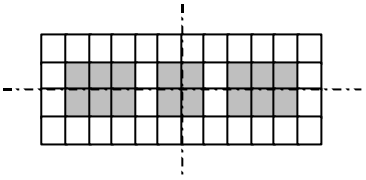
Pattern No. 41



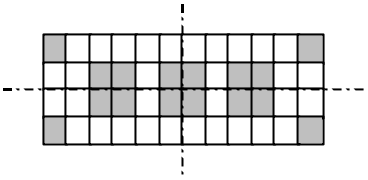
Pattern No. 42



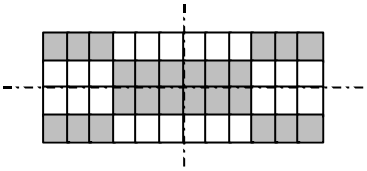
Pattern No. 43



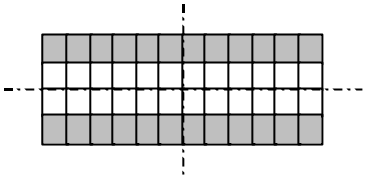
Pattern No. 44



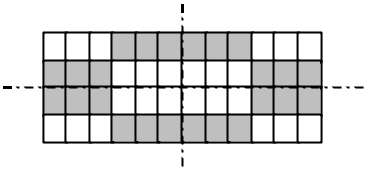
Pattern No. 45



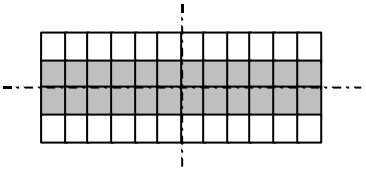
Pattern No. 46



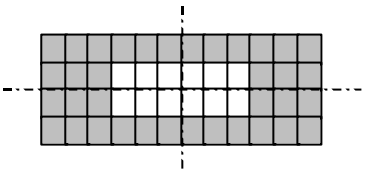
Pattern No. 47



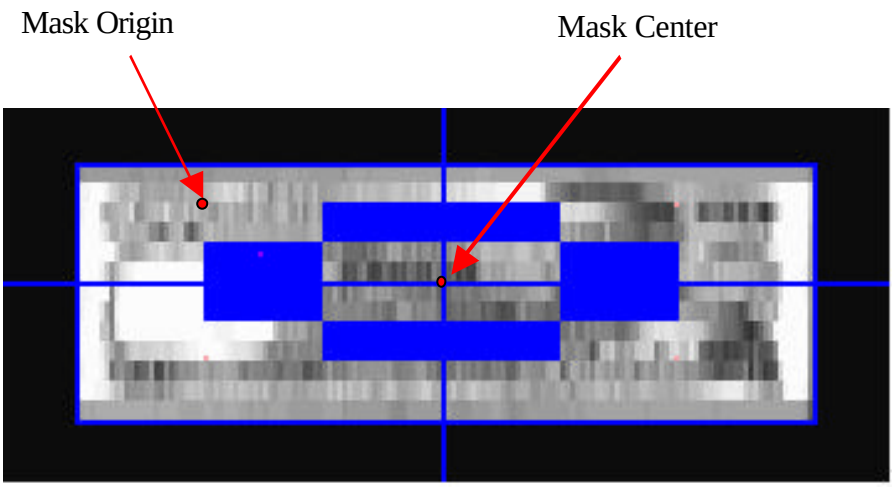
Pattern No. 48



Pattern No. 49



Pattern No. 50



Mask Pattern with (8´2) Block Size and (-48, -4) offset.

Appendix B

80 Testing Data for the 20HB Pattern

(a-1)

(a-2)

(a-3)

(a-4)

(a-5)

1^{st} - 5^{th} Testing Data of the 20HB (#6E8847225)

(b-1)

(b-2)

(b-3)

(b-4)

(b-5)

6^{th} - 10^{th} Testing Data of the 20HB (#7E7323161)

(c-1)

(c-2)

(c-3)

(c-4)

(c-5)

(c-6)

(c-7)

(c-8)

(c-9)

(c-10)

11^{th} - 20^{th} Testing Data of the 20HB

(d-1)

(d-2)

(d-3)

(d-4)

(d-5)

(d-6)

(d-7)

(d-8)

(d-9)

(d-10)

21^{st} - 30^{th} Testing Data of the 20HB

(e-1)

(e-2)

(e-3)

(e-4)

(e-5)

(e-6)

(e-7)

(e-8)

(e-9)

(e-10)

31^{st} - 40^{th} Testing Data of the 20HB

(f-1)

(f-2)

(f-3)

(f-4)

(f-5)

(f-6)

(f-7)

(f-8)

(f-9)

(f-10)

41st - 50th Testing Data of the 20HB

(g-1)

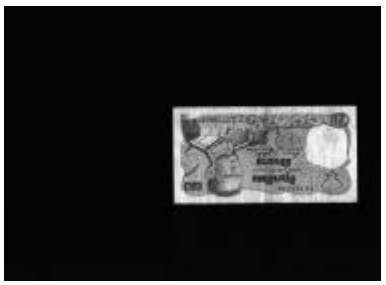
(g-2)

(g-3)

(g-4)

(g-5)

(g-6)



(g-7)

(g-8)

(g-9)

(g-10)

51st - 60th Testing Data of the 20HB

(h-1)

(h-2)

(h-3)

(h-4)

(h-5)

(h-6)

(h-7)

(h-8)

(h-9)

(h-10)

61st - 70th Testing Data of the 20HB

(i-1)

(i-2)

(i-3)

(i-4)

(i-5)

(i-6)

(i-7)

(i-8)

(i-9)

(i-10)

71st - 80th Testing Data of the 20HB

Appendix C

Scndemo Source Code for the Transformation between the Bit Map Data and the NN Data

```
 ************************************************/
 /************************************************/
 /**                                            **/
 /**          制御用ＰＣメインプログラム        **/
 /**                                            **/
 /**                作成開始                    **/
 /**      1994.01.26 by 株式会社リライアンス    **/
 /**          メニューマップ修正                **/
 /**            1994.02.03 by T.Kdn             **/
 /**                                            **/
 /************************************************/
 /**          UW－X用に変更                     **/
 /**                終了                        **/
 /**          1995.08.28      渡辺              **/
 /************************************************/
 /**                最終更新                    **/
 /**          1996.09.26    niskage             **/
 /************************************************/
 /**          ＰＤＵ－X拡張版に変更             **/
 /**                最終更新                    **/
 /**            1997.01.28    漁                **/
 /************************************************/

 /************************************************/
 /**          ２波長センサ対応                  **/
 /**                最終更新                    **/
 /**          1998.01.28      niskage           **/
 /************************************************/
```

```
 /************************************************/
 /**          スキャナデモ用                    **/
 /**                最終更新                    **/
 /**          1998.09.25      niskage           **/
 /************************************************/

 /************************************************/
 /**      Modify for request from Prof. Takeda(KUT) **/
 /**                Last modify                 **/
 /**          2000.05.15      niskage           **/
 /************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include "dspextrn.h"
#include "dsppc.h"
#include "dsppcpro.h"
#include "drvrs232.h"
#include <time.h>

/*****************************************************
名称:      main
機能:      メイン
引数:      なし
戻値:      なし
@*****************************************************/
void main( int argc, char *argv[] )
{
char        fn[14];

cmd_line( argc, argv );          /*  コマンドラインの解析   */
strcpy( fn, "SCNDEMO.cfg");

if(fast)
 baud = 38400;
else{
  if(!speed)
    baud = 19200;
}
// if ( 0 != get_config(fn) ){   /*  コンフィグレーション処理    */
// return;
// }
if(trace){
   if((logf = fopen("scndemo.log","w"))==0){
      printf("Can't create log file¥n");
       return;
    }
}

// read_init_weight();
    read_scndemo_cfg();

if(com2)
  intnum=1;
else
  intnum=0;

if(!offline){
    pc_init();                          /* RS232C 初期化  */
    get_bvinfo();                       /*  ＢＶ情報取得  */
}

//あくまでとりあえず --ここから
if(cfg.sensor_width==0){
//    cfg.sensor_width=256;
    cfg.sensor_width=216;
    cfg.recog_hight=32;
}
if(offline){
    cfg.sensor_width=216;
    cfg.recog_hight=32;
    cfg.sensor_valid1=5;
```

```
    cfg.sensor_valid2=205;
    cfg.reso_x=10;
    cfg.reso_y=40;
}
//あくまでとりあえず --ここまで

    if(!offline){
disp_menu0();                          /*  選択メニューの表示  */

while (1){
  if( _kbhit() != 0 ){
       cls();
      switch( _getch() ){
      case  '1' :                      /* file util */
    if(cfg.dsp_mode<2)
       menu1();
    break;
      case  '2' :                      /* LEARNING */
    if(((1<cfg.dsp_mode)&&(cfg.dsp_mode<6))&&(cfg.ptrn_num>0))
       menu2();
    break;
      case  '3' :                      /* BOARD MONITOR */
    if((cfg.dsp_mode==6)&&((cfg.out_sw==1)||(cfg.out_sw>10))){
       board_monitor();
   //  board_monitor(1);
}
    break;
      case  '4' :                      /*  フラッシュメモリの消去  */
     //erase_flash();
    break;
    if(cfg.dsp_mode<2)
       menu4();
    break;
      case  0x1b:                      /*  終了  */
    setvideomode(3);
    //rsstop();
    if(trace){
       fclose(logf);
    }
    return;
       default          :
    break;
    }
    disp_menu0();
    }
  }
}
else{
    disp_menu00();                     /*  選択メニューの表示  */

while (1){
  if( _kbhit() != 0 ){
       cls();
      switch( _getch() ){
      case  '1' :                      /*   file util   */
    if(cfg.dsp_mode<2)
       menu1();
    break;
      case  '2' :                      /*   DATA  Convert   */
    learning_cont(0);
    break;
       case 0x1b:                      /*  終了  */
```

```
    setvideomode(3);
    if(trace){
        fclose(logf);
    }
    return;
        default             :
    break;
    }
    disp_menu00();
    }
  }
}
if(!offline)
    rsstop();
}
```

```
/*****************************************************
名称:      disp_menu0
機能:      メインメニューの表示
引数:      なし
戻値:      なし
@*****************************************************/
void disp_menu0( void )
{
printf(    "¥x1b[0m");
cls();

printf( "¥x1b[0m"];
printf( "¥x1b[4;15H    AUTONOMY NEURO BOARD HANDLING TOOL ");
printf( "¥x1b[42;34m");
printf( "¥x1b[5;15H                                                ");
printf( "¥x1b[6;15H    +-----------      Main  MENU       ----------+    ");
printf( "¥x1b[7;15H    |                                             |    ");
printf( "¥x1b[8;15H    |          1 : Files  Handling                |    ");
printf( "¥x1b[9;15H    |          2 : LEARNING                       |    ");
printf( "¥x1b[10;15H    |          3 : EVALUATE                       |    ");
printf( "¥x1b[11;15H    |          4 : Erase  FLASH  MEMORY           |    ");
printf( "¥x1b[12;15H    |                                             |    ");
printf( "¥x1b[13;15H    |   [ESC] : EXIT  from  PC  MODE              |    ");
printf( "¥x1b[14;15H    |                                             |    ");
printf( "¥x1b[15;15H    +-------------------------------------+    ");
printf( "¥x1b[16;15H                                                ");
printf( "¥x1b[16;42H%s",__DATE__);
printf( "¥x1b[17;15H                                                ");
printf( "¥x1b[18;15H                                                ");
printf( "¥x1b[19;15H                                                ");
printf( "¥x1b[18;15H         PLEASE INPUT PROCESS NO ===>");
```

```
printf( "¥x1b[0m"];
}
```

```
/*****************************************************
名称:    disp_menu00
機能:    offline メインメニューの表示
引数:    なし
戻値:    なし
@*****************************************************/
void disp_menu00( void )
{
printf(    "¥x1b[0m");
cls();

printf( "¥x1b[0m");
printf( "¥x1b[4;15H    AUTONOMY NEURO BOARD HANDLING TOOL ");
printf( "¥x1b[42;34m");
printf( "¥x1b[5;15H                                                    ");
printf( "¥x1b[6;15H    +----------     Main MENU       ----------+    ");
printf( "¥x1b[7;15H    |                                              |    ");
printf( "¥x1b[8;15H    |          1 : Files Handling                  |    ");
printf( "¥x1b[9;15H    |         2 : DATA CONVERT                     |    ");
printf( "¥x1b[10;15H    |                  (BMP -> NEURO)              |    ");
printf( "¥x1b[11;15H    |                                              |    ");
printf( "¥x1b[12;15H    |                                              |    ");
printf( "¥x1b[13;15H    |   [ESC] : EXIT from PC MODE                  |    ");
printf( "¥x1b[14;15H    |                                              |    ")
printf( "¥x1b[15;15H    +--------------------------------------+    ");
printf( "¥x1b[16;15H                                                    ");
printf( "¥x1b[16;42H%s",__DATE__);
printf( "¥x1b[17;15H                                                    ");
printf( "¥x1b[18;15H                                                    ");
printf( "¥x1b[19;15H                                                    ");
printf( "¥x1b[18;15H       PLEASE INPUT PROCESS NO ===>");
```

```
printf( "¥x1b[0m"];
}
```

```
/*******************************************************
名称:      menu1
機能:      parameter handling
引数:      なし
戻値:      なし
@*******************************************************/
void menu1( void )
{
   disp_menu1();                          /*  メニューの表示    */
   while  (1){
     if(  _kbhit()  !=  0  ){
          cls();
         switch(  _getch()  ){
            case  '1'  :                  /*    create  weight   */
                 menu11();
                 break;
            case  '2'  :                  /*    download    */
                 menu12();
                 break;
            case  '3'  :                  /*    up    */
                 menu13();
                 break;
             case  0x1b:                  /*    終了   */
                 return;
              default:
                 break;
          }
          disp_menu1();
       }
    }
}
```

```
/*****************************************************
名称:    disp_menu1
機能:    PARAMETER Handling メニューの表示
引数:    なし
戻値:    なし
@*****************************************************/
void disp_menu1( void )
{
printf(    "¥x1b[0m"];
cls();
printf(    "¥x1b[42;34m"];
printf(    "¥x1b[5;15H                                                  ");
printf(    "¥x1b[6;15H   +---------   Parameter Handling    ----------+  ");
printf(    "¥x1b[7;15H   |                                             |  ");
printf(    "¥x1b[8;15H   |        1 : Create Initial Weight            |  ");
printf(    "¥x1b[9;15H   |                                             |  ");
printf( "¥x1b[10;15H   |        2 : Parameter Download               |  ");
printf( "¥x1b[11;15H   |                                             |  ");
printf( "¥x1b[12;15H   |        3 : Parameter Upload                 |  ");
printf( "¥x1b[13;15H   |                                             |  ");
printf( "¥x1b[14;15H   |  [ESC] : return to previous menu            |  ");
printf( "¥x1b[15;15H   |                                             |  ");
printf( "¥x1b[16;15H   +---------------------------------------+    ");
printf( "¥x1b[17;15H                                                  ");
printf( "¥x1b[18;15H                                                  ");
printf( "¥x1b[19;15H                                                  ");
printf( "¥x1b[18;15H      PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m"];
}
```

```
/*****************************************************
名称:    menu11
機能:    初期ウェイト作成（ネットワーク構成）
引数:    なし
戻値:    なし
@*****************************************************/
void menu11 (void )
{
  disp_menu11();                    /* メニューの表示   */

  while (1){
    if( _kbhit() != 0 ){
        cls();
       switch( _getch() ){
          case '1' :                /*  create weight  */
              create_weight1();
              break;
           case '2' :               /*                 */
              create_weight2();
              break;
          case '3' :                /*                 */
              create_weight3();
              break;
          case '4' :                /*                 */
              create_weight4();
              break;
           case 0x1b:               /*  終了  */
              return;
            default:
              break;
         }
```

```
            disp_menu11();
        }
    }
}
```

```
/*******************************************************
名称:      disp_menu11
機能:      Weight file 作成メニューの表示
引数:      なし
戻値:      なし
@*******************************************************/
void disp_menu11( void )
{
printf(    "¥x1b[0m");
cls();
printf(    "¥x1b[42;34m");
printf(    "¥x1b[5;15H                                                  ");
printf(    "¥x1b[6;15H    +------- Create Initial Weight File -------+    ");
printf(    "¥x1b[7;15H    |                                           |   ");
printf(    "¥x1b[8;15H    |        1 : Create Network structure       |   ");
printf(    "¥x1b[9;15H    |                                           |   ");
printf( "¥x1b[10;15H    |        2 : Switches for DSP               |   ");
printf( "¥x1b[11;15H    |                                           |
");
printf( "¥x1b[12;15H    |        3 : Conditions                     |   ");
printf( "¥x1b[13;15H    |                                           |
");
printf( "¥x1b[14;15H    |        4 : Write to File                  |   ");
printf( "¥x1b[15;15H    |                                           |
");
printf( "¥x1b[16;15H    |  [ESC] : return to previous menu          |   ");
printf( "¥x1b[17;15H    |                                           |
");
printf( "¥x1b[18;15H    +-----------------------------------------+    ");
printf( "¥x1b[19;15H                                                  ");
printf( "¥x1b[20;15H                                                  ");
```

```
printf( "¥x1b[21;15H                                                    ");
printf( "¥x1b[20;15H          PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m"];
}
/*****************************************************
名称:      menu12
機能:      ニューラルネットワークパラメータのダウンロード
引数:      なし
戻値:      なし
@*****************************************************/
void menu12 (void )
{
  int  c;

  disp_menu12();                    /*  メニューの表示    */
  while  (1){
    if(_kbhit()  !=  0){
      c  =  getch();
       cls();
       if(cfg.pdu_sw==0)            /*    pdu 拡張版でないなら  */
          if(c=='7')
             c='0';
      switch(  c  ){
        case  '1'  :          /*   フラッシュのブロック情報を表示    */
            read_nnpara();
            break;
        case  '2'  :/*    センサ補正データの書き込み    */
        case  '3'  :/*    ウエイトデータの書き込み    */
        case  '5'  :/*    スラブマスクデータの書き込み   */
            write_data(0,c);
            break;
        case  0x1b:          /*  終了  */
            return;
        case  '4'  :/*    濃淡トラップデータの書き込み   */
        case  '6'  :/*    真偽トラップデータの書き込み   */
        case  '7'  :   /*    サイズトラップデータの書き込み    */
         default:
            break;
      }
      disp_menu12();
    }
  }
}
```

```
/*****************************************************
名称:    disp_menu12
機能:    Ｎ．Ｎ．パラメータダウンロードメニューの表示
引数:    なし
戻値:    なし
@*****************************************************/
void disp_menu12( void )
{
printf(    "¥x1b[0m");
cls();
printf(    "¥x1b[42;34m");
printf(    "¥x1b[5;15H                                                  ");
printf(    "¥x1b[6;15H   +   Download Neural Network parameters +   ");
printf(    "¥x1b[7;15H   |                                          |   ");
printf(    "¥x1b[8;15H  | 1 : Display Parameters in FLASH MEMORY | ");
printf(    "¥x1b[9;15H  | 2 : Write CORRECTION data to
FLASHMEMORY   |   ");
printf( "¥x1b[10;15H  | 3 : Write WEIGHT data to FLASH MEMORY | ");
printf( "¥x1b[11;15H  | 5 : Write SLAB MASK data to FLASH MEMORY
|   ");
printf( "¥x1b[12;15H  |                                          |   ");
printf( "¥x1b[13;15H  |  [ESC] : return to previous menu          |   ");
printf( "¥x1b[14;15H  |                                          |   ");
printf( "¥x1b[15;15H   +------------------------------------------+   ");
printf( "¥x1b[16;15H                                                  ");
printf( "¥x1b[17;15H                                                  ");
printf( "¥x1b[18;15H                                                  ");
printf( "¥x1b[17;15H      PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m");

}
```

```
/*****************************************************
名称:    menu13
機能:    Ｎ．Ｎ．パラメータアップロード
引数:    なし
戻値:    なし
@*****************************************************/
void menu13( void )
{
  UCHAR  *bip;
  int  sw,used_blk1,used_blk2;

  bip  =  work+4096;          /*  作業領域のセット  */

/*  フラッシュメモリブロック情報取得  */
/*
if( 0!=cmd_read_blk_info( cfg.fl_block, blk, bip ) ){
printf("¥n*** flash memory data read error!! ***");
printf("¥npush any key to return to previous menu ");
getch();
return;
}*/
if(get_wr_blk(0,&used_blk1,&used_blk2)!=0){
    printf("¥n*** Can't get used block information ***");
    printf("¥npush any key to return to previous menu");
    getch();
    return;
}
disp_menu13();
while(1){
  if(_kbhit() != 0){
      cls();
```

```
    sw = _getch();
     if(cfg.pdu_sw==0)                  /* pdu 拡張版でないなら */
        if(sw=='7')
           sw='0';

    switch( sw ){
       case '1' :   /*   フラッシュメモリダンプ   */
           dump_flblk(used_blk1,used_blk2);
           break;
       case '3' :
           upload_data((sw-1),used_blk2);
           break;
       case '2' :   /*   パラメータのアップロード   */
    // case '4' :
       case '5' :
    // case '6' :
    // case '7' :
           upload_data((sw-1),used_blk1);
           break;
       case 0x1b: /*   終了   */
           return;
        default:
           break;
        }
        disp_menu13();
     }
  }
}
```

```
/*****************************************************
名称:     disp_menu13
機能:     N．N．パラメータアップロードメニューの表示
引数:     なし
戻値:     なし
@*****************************************************/
void disp_menu13( void )
{
printf( "¥x1b[0m");
cls();
printf( "¥x1b[42;34m");
printf( "¥x1b[5;15H                                                    ");
printf( "¥x1b[6;15H   +---  Upload Neural Network parameters  ---+ ");
printf(   "¥x1b[7;15H   |                                          |   ");
printf(   "¥x1b[8;15H   |   1 : Dump FLASH MEMORY                  |   ");
printf(   "¥x1b[9;15H   |   2 : Upload Sensor Correction DATA      |   ");
printf( "¥x1b[10;15H   |   3 : Upload NN's weight DATA            |   ");
printf( "¥x1b[11;15H   |   5 : Upload Slab-Mask DATA              |   ");
printf( "¥x1b[12;15H   |                                          |   ");
printf( "¥x1b[13;15H   | [ESC] : return to previous menu          |   ");
printf( "¥x1b[14;15H   |                                          |   ");
printf( "¥x1b[15;15H   +------------------------------------------+   ");
printf( "¥x1b[16;15H                                                    ");
printf( "¥x1b[17;15H                                                    ");
printf( "¥x1b[18;15H                                                    ");
printf( "¥x1b[17;15H      PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m"];

}
```

```
/*****************************************************
名称:      menu2
機能:      learning
引数:      なし
戻値:      なし
@*****************************************************/
void menu2( void )
{
   disp_menu2();                        /* メニューの表示    */

   while  (1){
     if( _kbhit() != 0 ){
          cls();
         switch( _getch() ){
               case '1':                /* online    */
//                learning_single(0);
                  menu21();
                  break;
              case  '2':
                  menu22();                         /* offline    */
                  break;
              case  0x1b:                           /* 終了  */
                  return;
               default:
                  break;
          }
          disp_menu2();
       }
    }
}
```

```
/*****************************************************
名称:      disp_menu2
機能:      学習メニューの表示
引数:      なし
戻値:      なし
@*****************************************************/
void disp_menu2( void )
{
printf( "¥x1b[0m");
cls();
printf( "¥x1b[42;34m");
printf( "¥x1b[5;15H                                                    ");
printf( "¥x1b[6;15H    +-------- LEARNING on DSP BOARD ---------+ ");
printf(    "¥x1b[7;15H   |                                          |   ");
printf(    "¥x1b[8;15H   |          1 : ONLINE                      |   ");
printf(    "¥x1b[9;15H   |                                          |   ");
printf( "¥x1b[10;15H   |          2 : OFFLINE                     |   ");
printf( "¥x1b[11;15H   |                                          |   ");
printf( "¥x1b[12;15H   | [ESC] : return to previous menu           |   ");
printf( "¥x1b[13;15H   |                                          |   ");
printf( "¥x1b[14;15H    +-----------------------------------------+    ");
printf( "¥x1b[15;15H                                                    ");
printf( "¥x1b[16;15H                                                    ");
printf( "¥x1b[17;15H                                                    ");
printf( "¥x1b[16;15H       PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m"];

}
```

```
/*****************************************************
名称:     menu21
機能:     ONLINE 学習
引数:     なし
戻値:     なし
@*****************************************************/
void menu21( void )
{
  disp_menu21();              /*  メニューの表示    */

  while  (1){
    if( _kbhit() != 0 ){
        cls();
       switch(  _getch() ){
          case  '1' :          /*    ビットマップ    */
               learning_single(0);
               break;
          case  '2' :          /*    ニューロデータ    */
               learning_single(1);
               break;
          case  '3' :          /*    Not  Serial    */
               learning_single(2);
               break;
          case  '4' :          /*    SLAB    */
               learning_single(3);
               break;
           case  0x1b:         /*    終了 */
               return;
            default:
               break;
       }
                                                    disp_menu21();
                                                 }
                                              }
                                           }
```

```
/*****************************************************
名称:     disp_menu21
機能:     学習メニューの表示
引数:     なし
戻値:     なし
@*****************************************************/
void disp_menu21( void )
{
printf(    "¥x1b[0m"];
cls();
printf(    "¥x1b[42;34m"];
printf(    "¥x1b[5;15H                                                  ");
printf(    "¥x1b[6;15H +--- ONLINE LEARNING on DSP BOARD----+ ");
printf(    "¥x1b[7;15H   |                                              | ");
printf(    "¥x1b[8;15H   |       1 : BITMAP  DATA                       |   ");
printf(    "¥x1b[9;15H   |                                              |   ");
printf( "¥x1b[10;15H   |        2 : NEURO  FORMATED  DATA      |   ");
printf( "¥x1b[11;15H   |                                              |   ");
printf( "¥x1b[12;15H   |        3 : NOT  VIA-SERIAL  PORT          |   ");
printf( "¥x1b[13;15H   |                                              |   ");
printf( "¥x1b[14;15H   |        4 : SLAB  DATA                        |   ");
printf( "¥x1b[15;15H   |                                              |   ");
printf( "¥x1b[16;15H   | [ESC] : return  to  previous  menu            |   ");
printf( "¥x1b[17;15H   |                                              |   ");
printf( "¥x1b[18;15H    +-------------------------------------------+    ");
printf( "¥x1b[19;15H                                                  ");
printf( "¥x1b[20;15H                                                  ");
printf( "¥x1b[21;15H                                                  ");
printf( "¥x1b[20;15H       PLEASE  INPUT  PROCESS  NO  ===>");
printf( "¥x1b[0m"];
}
```

```
/*****************************************************
名称:     menu22
機能:     連続学習
引数:     なし
戻値:     なし
@*****************************************************/
void menu22( void )
{
  disp_menu22();                    /*  メニューの表示  */

  while  (1){
    if( _kbhit() != 0 ){
      cls();
     switch( _getch() ){
        case  '1' :               /*    ビットマップ    */
            learning_cont(0);
            break;
        case  '2' :               /*    ニューロデータ    */
            learning_cont(1);
            break;
        case  '3' :               /*    Not  Serial    */
            learning_cont(2);
            break;
        case  '4' :               /*  スラブデータ   */
            learning_cont(3);
            break;
         case  0x1b:              /*  終了   */
            return;
         default  :
            break;
          }
```

```
            disp_menu22();
        }
    }
}
```

```
/*******************************************************
名称:      disp_menu22
機能:      連続学習メニューの表示
引数:      なし
戻値:      なし
@*******************************************************/
void disp_menu22( void )
{
printf(    "¥x1b[0m");
cls();
printf(    "¥x1b[42;34m");
printf(    "¥x1b[5;15H                                                  ");
printf( "¥x1b[6;15H +----- OFFLINE  LEARNING on DSP BOARD -----+ ");
printf(    "¥x1b[7;15H   |                                          |   ");
printf(    "¥x1b[8;15H   |          1 : BITMAP  DATA                |   ");
printf(    "¥x1b[9;15H   |                                          |   ");
printf( "¥x1b[10;15H   |         2 : NEURO  FORMATED  DATA    |   ");
printf( "¥x1b[11;15H   |                                          |   ");
printf( "¥x1b[12;15H   |          3 : NOT  VIA  SERIAL  PORT      |   ");
printf( "¥x1b[13;15H   |                                          |   ");
printf( "¥x1b[14;15H   |          4 : SLAB  DATA                  |   ");
printf( "¥x1b[15;15H   |                                          |   ");
printf( "¥x1b[16;15H   | [ESC] : return  to  previous  menu       |   ");
printf( "¥x1b[17;15H   |                                          |   ");
printf( "¥x1b[18;15H    +-----------------------------------------+    ");
printf( "¥x1b[19;15H                                                  ");
printf( "¥x1b[20;15H                                                  ");
printf( "¥x1b[21;15H                                                  ");
printf( "¥x1b[20;15H       PLEASE  INPUT  PROCESS  NO  ===>");
printf( "¥x1b[0m");
}
```

```
/*****************************************************
名称:     menu4
機能:     フラッシュメモリ消去
引数:     なし
戻値:     なし
@*****************************************************/
void menu4( void )
{
  disp_menu4();                         /* メニューの表示   */

  while (1){
    if( _kbhit() != 0 ){
        cls();
       switch( _getch() ){
          case '1' :            /*   全消去   */
               erase_flash();
               break;
          case '2' :            /*   セクタ消去   */
               erase_sector();
               break;
          case 0x1b :           /* 終了  */
               return;
           default :
               break;
        }
        disp_menu4();
     }
  }
}
```

```
/*****************************************************
名称:     disp_menu4
機能:     フラッシュメモリ消去メニューの表示
引数:     なし
戻値:     なし
@*****************************************************/
void disp_menu4( void )
{
printf(    "¥x1b[0m");
cls();
printf(    "¥x1b[42;34m");
printf(    "¥x1b[5;15H                                                  ");
printf(    "¥x1b[6;15H    +-------- ERASE FLASH MEMORY -------+ ");
printf(    "¥x1b[7;15H    |                                    |    ");
printf(    "¥x1b[8;15H    |          1 : ALL BLOCK ERASE       |    ");
printf(    "¥x1b[9;15H    |                                    |    ");
printf( "¥x1b[10;15H    |          2 : SECTOR ERASE          |    ");
printf( "¥x1b[11;15H    |                                    |    ");
printf( "¥x1b[12;15H    | [ESC] : return to previous menu    |    ");
printf( "¥x1b[13;15H    |                                    |    ");
printf( "¥x1b[14;15H    +------------------------------------+    ");
printf( "¥x1b[15;15H                                                  ");
printf( "¥x1b[16;15H                                                  ");
printf( "¥x1b[17;15H                                                  ");
printf( "¥x1b[16;15H       PLEASE INPUT PROCESS NO ===>");
printf( "¥x1b[0m");

}
```

```
/*****************************************************
名称:     cmd_line
機能:     コマンドラインを解析する（debug スイッチのチェック）
引数:     int          argc     コマンドラインの単語数
引数:     char     *argv     コマンドライン各文字列を示すポインタ
配列先頭アドレス
戻値:     なし
@*****************************************************/
void cmd_line( int argc, char *argv[] )
{
   int    i;

  trace = offline = nodisp = fast = step = com2 = speed = 0;
  for ( i=1; argc>1; argc--,i++ ){        /* 引数をすべて処理する */
     if ( strcmp(argv[i],"trace")==0     /* 比較 */
          || strcmp(argv[i],"TRACE")==0)
        trace = 1;                       /* デバッグスイッチＯＮ */
     if ( strcmp(argv[i],"offline")==0
          || strcmp(argv[i],"OFFLINE")==0)
        offline = 1;

     if ( strcmp(argv[i],"nodisp")==0
          || strcmp(argv[i],"NODISP")==0)
        nodisp = 1;

     if ( strcmp(argv[i],"fast")==0
          || strcmp(argv[i],"FAST")==0)
        fast = 1;

     if ( strcmp(argv[i],"step")==0
          || strcmp(argv[i],"STEP")==0)
        step = 1;

     if ( strcmp(argv[i],"com2")==0
          || strcmp(argv[i],"COM2")==0)
        com2 = 1;

     if ( strcmp(argv[i],"speed")==0
          || strcmp(argv[i],"SPEED")==0){
       baud=atoi(argv[i+1]);
        speed = 1;
      }
    }
}
```

```
/*****************************************************
名称:     get_bvinfo
機能:     ＢＶハードウェア情報を得る
引数:     なし
戻値:     なし
@*****************************************************/
void get_bvinfo( void )
{
  UCHAR  *dp;
  union  f_tag  fl_data;
    if ( 0 != cmd_get_bvinfo( work ) ){
        printf("¥n***  can't  get   bv-hardware parameters!! ***");
        printf("¥npush any key ");
        rsstop();
        getch();
        exit(1);
     }

/*  パラメータセット（全て int 型）  */
cfg.sensor_width =  work[ 0]*1000+work[ 1]*100+work[ 2]*10+work[ 3];
                        /*  センサの画素数    */
cfg.sensor_valid1 =  work[ 4]*1000+work[ 5]*100+work[ 6]*10+work[ 7];
                        /*   センサ有効ﾁｬﾈﾙ番号(L)    */
cfg.sensor_valid2 = work[ 8]*1000+work[ 9]*100+work[10]*10+work[11];
                        /*  センサ有効ﾁｬﾈﾙ番号(H) */
cfg.reso_x   =    work[24]*1000+work[25]*100+work[26]*10+work[27];
                        /*  センサ分解能Ｘ    */
cfg.reso_y   =    work[28]*1000+work[29]*100+work[30]*10+work[31];
                        /*  センサ分解能Ｙ    */
cfg.sns_mon_area  =   work[32]*1000+work[33]*100+work[34]*10
                        +work[35];   /*  センサモニタ域の画素数    */
cfg.recog_hight   =    work[36]*1000+work[37]*100+work[38]*10
                        +work[39];        /*  画像ﾊﾞｯﾌｧのライン数    */
cfg.fl_block    =     work[40]*1000+work[41]*100+work[42]*10
                        +work[43];        /*  ﾌﾗｯｼｭﾒﾓﾘのブロック数   */
if(rlngth!=56){
   cfg.out_sw = work[44]*1000+work[45]*100+work[46]*10+work[47];
                    /*  出力スイッチ   */

   cfg.dsp_mode   =    work[48]*1000+work[49]*100+work[50]*10
                        +work[51];        /*     DSP 動作モード  */;


//dp=work+52;
/   /if(c8lix(dp,&fl_data.long_value)!=0){
//       if(trace){
//          fprintf(logf,"LRN : CONVERT ERROR¥n");
//          }
//       }
   fl_data.long_value   =   work[52]*0x10000000+work[53]*0x1000000
                            +work[54]*0x100000+work[55]*0x10000
                            +work[56]*0x1000+work[57]*0x100
                            +work[58]*0x10+work[59];
   cfg.lrn_cnst   =   fl_data.float_value;
//dp=work+60;
//   if(c8lix(dp,&fl_data.long_value)!=0){
//       if(trace){
//          fprintf(logf,"MOMENTUM : CONVERT ERROR¥n");
//       }
//   }
     fl_data.long_value   =   work[60]*0x10000000+work[61]*0x1000000
                              +work[62]*0x100000+work[63]*0x10000
                              +work[64]*0x1000+work[65]*0x100
```

```
                                +work[66]*0x10+work[67];
   cfg.mmn_cnst   =   fl_data.float_value;
   dp=work+68;
//   if(c8lix(dp,&fl_data.long_value)!=0){
//      if(trace){
//         fprintf(logf,"VBR : CONVERT ERROR¥n");
//      }
//   }

   fl_data.long_value   =   work[68]*0x10000000+work[69]*0x1000000
                            +work[70]*0x100000+work[71]*0x10000
                            +work[72]*0x1000+work[73]*0x100
                            +work[74]*0x10+work[75];
   cfg.vbr_cnst   =   fl_data.float_value;

cfg.ptrn_num =     work[76]*1000+work[77]*100+work[78]*10+work[79];
                   /* 学習パターン数   */

cfg.lrn_num =     work[80]*1000+work[81]*100+work[82]*10+work[83];
                   /* 提示枚数   */

//dp=work+84;
//   if(c8lix(dp,&fl_data.long_value)!=0){
//      if(trace){
//         fprintf(logf,"FINAL ERROR : CONVERT ERROR¥n");
//       }
//   }
 fl_data.long_value = work[84]*0x10000000+work[85]*0x1000000
                      +work[86]*0x100000+work[87]*0x10000
                      +work[88]*0x1000+work[89]*0x100
                      +work[90]*0x10+work[91];
 cfg.final_err = fl_data.float_value;

 cfg.max_itteration =
work[92]*0x1000+work[93]*0x100+work[94]*0x10
                          +work[95];  /* 最大終学習回数 */

if(trace){
   fprintf(logf,"LRN = %f¥n",cfg.lrn_cnst);
   fprintf(logf,"MOMENTUM = %f¥n",cfg.mmn_cnst);
   fprintf(logf,"VBR = %f¥n",cfg.vbr_cnst);
   fprintf(logf,"FINAL ERROR = %f¥n¥n",cfg.final_err);
}
}
}
```

```
/*****************************************************
名称:      read_scndemo_cfg
機能:      定義(初期値)ファイルを読み込む
引数:      なし
戻値:      なし
@*****************************************************/
void       read_scndemo_cfg(void){
FILE *fp;
//   int i;
int instr[64];

fp=fopen("scndemo.cfg","r");
/*  先頭 3 行はコメント  */
fscanf(fp,"%s",instr);
fscanf(fp,"%s",instr);
fscanf(fp,"%s",instr);
/*  ネットワーク ID */
fscanf(fp,"%s %d",instr,&whdr.nn_id[0]);
fscanf(fp,"%s %d",instr,&whdr.nn_id[1]);
fscanf(fp,"%s %d",instr,&whdr.nn_id[2]);
fscanf(fp,"%s %d",instr,&whdr.nn_id[3]);
/*  層数  */
fscanf(fp,"%s %d",instr,&whdr.nn_layer);
/*  学習パターン数  */
fscanf(fp,"%s %d",instr,&whdr.num_of_lrn_ptrn);
/*  提示枚数  */
fscanf(fp,"%s %d",instr,&whdr.num_of_1ptrn);
/*  マスク ID */
fscanf(fp,"%s %ld",instr,&whdr.mask_id);
/*  各層素子数  */
fscanf(fp,"%s %ld",instr,&whdr.cell_l);
fscanf(fp,"%s %ld",instr,&whdr.cell_m);
/*  目標誤差  */
fscanf(fp,"%s %f",instr,&whdr.final_err);
/*  最大学習回数  */
fscanf(fp,"%s %ld",instr,&whdr.max_lrncnt);
/*  学習パラメータ  */
fscanf(fp,"%s %f",instr,&whdr.lrn_cnst);
fscanf(fp,"%s %f",instr,&whdr.mmnt_cnst);
fscanf(fp,"%s %f",instr,&whdr.vbr_cnst);
/*  温度勾配  */
fscanf(fp,"%s %f",instr,&whdr.slope);
/*  動作モード  */
fscanf(fp,"%s %ld",instr,&whdr.input_dtype);
fscanf(fp,"%s %ld",instr,&whdr.input_sw);
fscanf(fp,"%s %ld",instr,&whdr.output_sw);
fscanf(fp,"%s %ld",instr,&whdr.lrn_sw);
fscanf(fp,"%s %ld",instr,&whdr.dsp_interval);
/*  モジュールスイッチ  */
/*   fscanf(fp,"%s %ld",instr,&whdr.mask_sw);  */
fscanf(fp,"%s %ld",instr,&whdr.mkslab_sw);
fscanf(fp,"%s %ld",instr,&whdr.rtslab_sw);
/*  書き込み条件  */
fscanf(fp,"%s %d",instr,&whdr.write_w[0]);
fscanf(fp,"%s %d",instr,&whdr.write_w[1]);
/*  表示条件  */
fscanf(fp,"%s %f",instr,&whdr.evl_l[0]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[1]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[2]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[3]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[4]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[5]);
```

```
fscanf(fp,"%s %f",instr,&whdr.evl_l[6]);
fscanf(fp,"%s %f",instr,&whdr.evl_l[7]);
/* 切り出し位置 */
fscanf(fp,"%s %d",instr,&sline);
fscanf(fp,"%s %d",instr,&eline);
fscanf(fp,"%s %d",instr,&scolumn);
fscanf(fp,"%s %d",instr,&ecolumn);
/* ユーティリティの場所 */
fscanf(fp,"%s %s",instr,&bmppath);

/* フォントマップ作成用ファイル */
//   fscanf(fp,"%s  %s",instr,fontmap);

fclose(fp);

cfg.mask_id=(int)whdr.mask_id;
cfg.ptrn_num=(int)whdr.num_of_lrn_ptrn;
cfg.lrn_num=(int)whdr.num_of_1ptrn;
cfg.final_err=(float)whdr.final_err;
cfg.max_itteration=(int)whdr.max_lrncnt;
cfg.out_sw = (int)whdr.output_sw;

}
```