Journal of Intelligent Learning Systems and Applications, 2014, 6, 94-112
Published Online May 2014 in SciRes. http://www.scirp.org/journal/jilsa
http://dx.doi.org/10.4236/jilsa.2014.62008

Scientific Research

A Comparison of Neural Classifiers for Graffiti Recognition

A. H. Al-Fatlawi[1], S. H. Ling[1], H. K. Lam[2]

[1]Centre for Health Technologies, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia
[2]Division of Engineering, King's College London, London, UK
Email: Ali.H.Al-Fatlawi@student.uts.edu.au, Steve.Ling@uts.edu.au, hak-keung.lam@kcl.ac.uk

Received 22 March 2014; revised 22 April 2014; accepted 29 April 2014

Abstract

Technological advances and the enormous flood of papers have motivated many researchers and companies to innovate new technologies. In particular, handwriting recognition is a very useful technology to support applications like electronic books (eBooks), post code readers (that sort mails in post offices), and some bank applications. This paper proposes three systems to discriminate handwritten graffiti digits (0 to 9) and some commands with different architectures and abilities. It introduces three classifiers, namely single neural network (SNN) classifier, parallel neural networks (PNN) classifier and tree-structured neural network (TSNN) classifier. The three classifiers have been designed through adopting feed forward neural networks. In order to optimize the network parameters (connection weights), the back-propagation algorithm has been used. Several architectures are applied and examined to present a comparative study about these three systems from different perspectives. The research focuses on examining their accuracy, flexibility and scalability. The paper presents an analytical study about the impacts of three factors on the accuracy of the systems and behavior of the neural networks in terms of the number of the hidden neurons, the model of the activation functions and the learning rate. Therefore, future directions have been considered significantly in this paper through designing particularly flexible systems that allow adding many more classes in the future without retraining the current neural networks.

Keywords

Handwriting Recognition, Neural Networks, Network Structures

1. Introduction

Emergence networks mimic the biological nervous system unleash generations of inventions and discoveries in

the artificial intelligence field. These networks have been introduced by McCulloch and Pitts, and they are called neural networks. Neural network's function is based on the principle of extracting the uniqueness of patterns through trained machines to understand the extracted knowledge. Indeed, they gain their experiences from collected samples for known classes (patterns). The quick development of these networks promotes a concept of the pattern recognition by proposing intelligent systems such as handwriting recognition, speech recognition and face recognition.

In particular, problem of handwriting recognition has been considered significantly during the last decades in the academic and industrial fields by employing types of direct matching. The performance of this recognition has been paying strong attention through developing several schemes and algorithms to learn the machines. In the light of that development, David Shepard invented the first modern OCR's version to read texts in 1951. After few years, this innovation has followed by originating a prototype machine to read upper case characters with speed of a character per minutes [1]. In the same way, many companies, such as IBM, have continued in developing reading systems to challenge the problems of character recognition [2]. The competition between the realized systems concentrated on improving accuracy and speed of the intelligent machine. Later, [3] introduced a simple system to recognize handwritten numerals by using geometrical and local measurements. Following that, [4] acknowledged that handwriting recognizer in general needs four steps to classify the patterns: extract features, applied a relaxation process, arranged statistical data in dataset and finally classified the patterns through a tree classifier. This was an important research that introduces a procedure to classify handwritten characters [5]. In like manner, [6] and [7] made a comprehensive study about using mathematical morphology in the character recognition. The first study discussed the basic operations of erosion and dilation and presented a system to recognize six handwritten digits. For this purpose, [8] developed a novel method to recognize the cursive and degraded texts by using the technology of OCR. Parts of symbols (primitives) are detected to interpret the symbols with this method [9]. The study involves mathematical morphology operations to perform the recognition process.

The popularity of this technology has been increasing substantially during the last decade due to growing in the demand of fast recognition for handwritten documents such as reading addresses in post offices. The process of classifying handwritten characters is described as a complicated and difficult task [10] due to the variety in their styles and sizes [11]. In addition, the similarity between some digits is another difficulty in the recognition problem, such as the similarity between the digits 4 and 9 in their shapes. Several methodologies have been proposed to study this problem from different perspectives, but the neural network was the base for most of these methodologies; hence they are different in the system structures and the learning techniques.

This research aims to achieve four objectives. Firstly, the main contribution of this paper is to design a recognition system to separate the handwritten graffiti digits: 0 - 9 and the commands: "space", "backspace" and "carriage". For this purpose, a conventional classifier has been created with one neural network. This classifier is called single neural network (SNN) classifier. It is built as one feed-forward multilayer neural network with one hidden layer. This network has sixteen neurons in the output layer to classify the sixteen classes. The introduced system should be designed to perform the recognition process efficiently and accurately. Because of the importance of the future developments for this research, they will be considered strongly. Therefore, the second objective is to create another classifier that is designed as a combination of parallel neural networks. The structure of the parallel neural networks (PNN) classifier grants the system high flexibility to allow adding extra classes without affecting the trained networks. This classifier is provided to have high flexibility and efficiency in recognizing the classes. However, its scalability in handling too many extra classes is limited because with too many classes, the structures of this classifier will be complicated. Therefore, the tree-structured neural network (TSNN) classifier is the third objective in the research. This classifier has high efficiency in handling any number of extra classes in the future with keeping the current networks without any change. It is constructed as a tree of subclassifiers that represent groups of classes. The tree-structured neural network (TSNN) classifier is the highly arranged system that discriminates the classes through two classification phases.

As a result, the three classifiers are proposed to meet different purposes. The first classifier (SNN) is presented as a solution to recognize the digits: 0 - 9 and some commands while the second classifiers (PNN) and the third classifier (TSNN) are introduced as a solution for the current stage as well as the future developments. The paper shows the relationship between flexibility and complexity of the classifiers to do the same function, in addition to the speed of that system. The future developments have been considered significantly during the design process by building the three versions with different structures and characteristics. Therefore, the future exten-

sion will be paid full attention by increasing flexibility of the system in classifying more classes (patterns) without needing to retrain it with the current classes. In addition, the scalability of the classes has been considered by designing efficient systems in handling too many classes. The fourth achievement is introducing a study about the training procedure for the neural networks and analyzing impacts of some factors on the learning process. This study has been made by analyzing the first classifier because it has a simple structure with one neural network. It uses the back-propagation algorithm to perform the learning process and gets the best parameters to recognize handwriting graffiti digits. Different structures and configurations will be discussed and examined to show the role of some factors in changing the performance and the accuracy of the networks.

This paper is organized as follows. Section 2 analyzes the system requirements and explains its structure. The architecture of the three proposed classifiers will be presented in this section. Section 3 and Section 4 illustrate the training and the validation processes for each of the proposed classifiers. The training procedure has been investigated intensively to study some impacts and factors that can affect the process by using the back-propagation algorithm. It also presented simulation and experimental results on interpreting handwritten digits and commands. Different configurations for the neural networks of three versions have been applied and tested to exam the classifier accuracies and properties. Section 5 gives a discussion for the network designs and the results. Finally, a conclusion will be drawn in Section 6, in addition to the future works.

2. Methods

The recognition process starts from scanning the image that captures the text, then extract the features of that character and ends at the classification and the determination processes, as shown in **Figure 1**. This research adopts the shown operations to design a reliable and flexible system that recognizes the digits 0 - 9 and some commands. It focuses on the classification networks and their configurations. The neural network is the core of the recognition system. Hence, this network should have enough learning (training) to achieve an optimal design for the recognition function. The neural network can be arranged in different ways depending on the system requirements and the nature of the input data (linear or nonlinear).

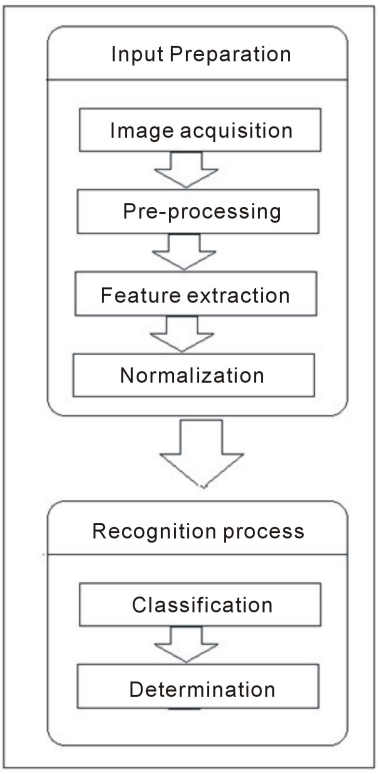

Figure 1. The handwriting recognition process.

2.1. Input Preparation

The image acquisition is the first step in the recognition process because the written digits or commands should be captured by an image. It refers to the process of getting a scanned image (digital image) for the pattern in gray level. The pre-processing operations are imperative to prepare that image, such as noise removing, edge removing, and dilation & filling. This image is represented by a set of too many points (pixels), but only some of these points denote its features. As a consequence, the digital image is not fully representative because it includes too many values and most of these values are not beneficial. Only some of these values are important for the classifier to distinguish between the patterns (the digits). In this direction, compressing the high dimensional data into low dimensional features is important to increase the efficiency of the system, but this should be done without discarding too many features. Thus, these features represent what the neural network needs for the recognition process because they indicate the uniqueness of each pattern. Therefore, the objective of extracting these features is to characterize the input patterns (the graffiti digits), then to classify them by the neural networks. The feature extraction is crucial for the neural network applications that provide important tools for the classification and the recognition.

The performance of the method that extracts features of the input can affect the accuracy and quality of the whole system. Therefore, it is important to select the method that can meet the requirements of the system and its functions. In this regard, several techniques have been developed to extract the features of the input data. One of the general-purpose methods is the procedure that proposed by Ling *et al*. [12] [13]. In practice, this system is designed to recognize the digits: 0 to 9 and the three commands: space, backspace and carriage. The features of these patterns are extracted based on the coordinates: x and y in the writing area with limited size ($X_{\max}$, $Y_{\max}$) starting from (0, 0), as shown in **Figure 2**. Ten sampled points have been taken as features. According to [12] [13], each graffiti pattern will be divided into a distance segment represented by ten points. Firstly, the five points (X_i, Y_i), i = 1, 3, 5, 7 and 9 are converted to five numbers (σ_i) by using ($\sigma_i = X_i \times X_{\max} + Y_i$) while the others are transferred by the formula ($\sigma_i = Y_i \times Y_{\max} + X_i$). Consequently, this operation result in ten numbers for each pattern, as can be seen from **Figure 2**. By means, we have ten inputs for the next level (the neural network). **Table 1** lists these sixteen patterns.

The ten features describe each pattern that will be classified by the trained neural network. However, they cannot feed the network directly (as inputs to this network) because they are usually big values and the system is nonlinear. In the nonlinear system, the transfer functions of the neurons in the hidden layers and mostly in the output layer are either hyperbolic tangent or sigmoid functions which include an exponential function. Accordingly, these inputs require to be normalized within a specific range before feeding them into the system. The values of the features must keep the asymptotic performance of the neural network. In addition, normalization of the input can help the neural network to process the input with an equal attention. In order to give an enough comparison between the input patterns, the input will be normalized with the same symmetric range: −1 to 1. In this regard, [14] proposes some methods to standardize the input data within the range (−1, 1). The system uses one of these methods, as shown in the Equation (1).

$$x_{norm} = 2 * \left[\frac{\left(x - x_{\min}\right)}{x_{\max} - x_{\min}} \right] - 1, \tag{1}$$

where x is the input, x_{mim} is the minimum value in the input vector and $x_{\max}$ is the maximum value in the input vector.

2.2. The Proposed Classifiers

The performance of the system and the structure of the neural network is significantly influenced by representing of the input. As described previously, the outputs of the feature extraction stage are ten numbers that will feed the input layer of the neural network, as shown in **Figure 3**. Therefore, this layer includes ten units which receive the input without any computational function.

Another principle issue is to investigate whether the network should have a single layer (no hidden layer) or multilayer (has one or more hidden layers). It is obvious from many researches and studies that the single layer perceptron can solve only linear problems. In contrast, the multilayer perceptron is designed for nonlinear decision regions, “but it is liable to get stuck in a local minimum which may be inferior to the global minimum” [15]. Therefore, it is important to consider using a multilayer perceptron in designing this system because the features

Figure 2. Example for input preparation for digit "3" and its coordinates.

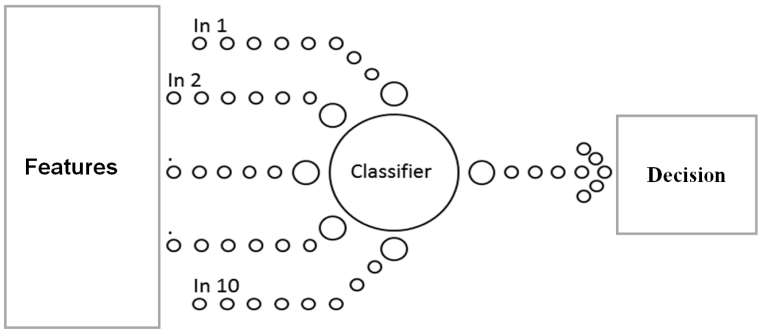

Figure 3. General diagram for the classifiers.

Table 1. Handwritten digits and commands.

Sq.	Digits or commands	Handwritten form	Sq.	Digits or commands	Handwritten form
1	0 (left)		9	6	
2	0 (right)		10	7	
3	1		11	8 (a)	
4	2		12	8 (b)	
5	3		13	9	
6	4		14	Space	
7	5 (a)		15	Backspace	
8	5 (b)		16	Carriage return	

of the handwritten digits are known as a nonlinear domain. As explained so far, the system is designed to distinguish between sixteen patterns. Therefore, the system has 16 outputs (one for each pattern), then the determiner will shows the final result. This paper introduces three types of classifiers (that are designed based on neural network topology): single neural network (SNN) classifier (which has only one network with 16 outputs), parallel neural networks (PNN) classifier (which has a network for each pattern) and tree-structured neural network (TSNN) classifiers (two consecutive groups of parallel networks). Each of these classifiers has ten inputs and sixteen outputs (which are specified by the determiner to measure the decision of the system), but they have different architectures. All networks of these classifiers have been built as multilayer feed-forward neural network with one hidden layer. Therefore, the neurons of each network are organized in layers and connected to pass the signal in one direction only (from the input to the output). This paper examines the structure and design of each of the three classifiers to get an optimal solution and introduces an analytical study.

2.2.1. Single Neural Network (SNN) Classifier

The single neural network classifier is a conventional classifier designed as one multilayer feed-forward neural

network with multi-input and multi-output (as shown in **Figure 4**). Firstly, the feature extraction stage provides the system ten numbers (points) for each digit. These numbers are considered as inputs for the neural network. Therefore, the input layer of the neural network has ten nodes that receive the features and pass them to the next layer via connection weights to be processed in the hidden and output layers. In the other words, the input units do not include any activation function because it is employed to pass the signal only without any modification. Consequently, the signal flows from the input layer to the hidden layer, then to the output layer. The hidden layer is the most important stage in the multilayer networks because most of the data processing is happening in this layer. Many researchers have introduced different techniques to determine the number of the hidden neurons, but the most popular method is trial and error.

The system is designed to classify sixteen classes. Thus, the neural network has 16 output neurons; each neuron refers to a class. Therefore, the output vector indicates the similarity between the results of the input graffiti pattern and the standard pattern (that is used to train the system). Hence, there is a binary target for each pattern which contains sixteen digits. The target digits can be either –1 or +1; where +1 means that this input belongs to the pattern and –1 means the versus. **Table 2** shows the target vectors for all of the sixteen classes. It is clear from the table that only one element of the output vector should be positive and the rest are negatives.

Figure 4 illustrates the neural network of this classifier. It is obvious from this figure that the neural network has ten input neurons, sixteen outputs and a certain number of the hidden neurons. The number of the hidden neurons will be specified in the training level. The outputs of this network are calculated according to the following equations.

$$y_j = f\left(b^1 + \sum_{i=1}^{10}\left(w_{ji}^1 \cdot x_i\right)\right),\ \ i = 1,2,\cdots,10,\quad j = 1,2,\cdots,n, \tag{2}$$

$$z_k = f\left(b^2 + \sum_{j=1}^{n}\left(w_{kj}^2 \cdot y_i\right)\right),\ \ k = 1,2,\cdots,16 \tag{3}$$

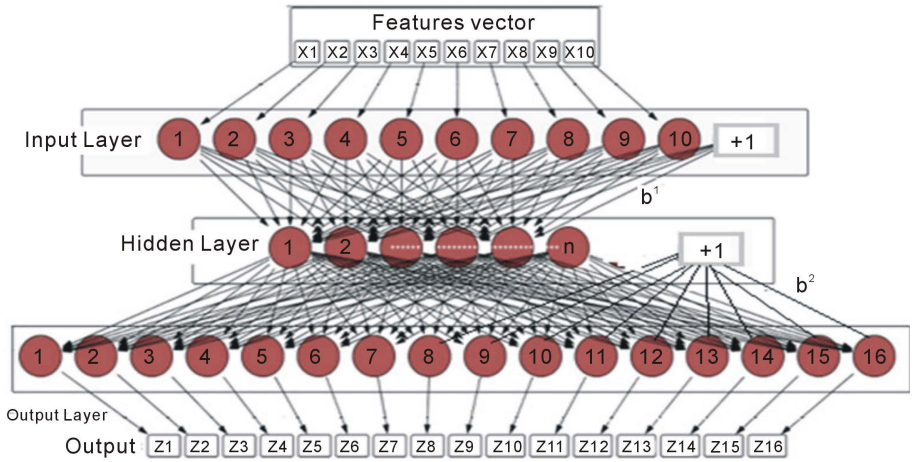

Figure 4. The neural network of the SNN classifier.

Table 2. The target vectors for the SNN classifier.

Digits/ Commands	The target vector	Digits/ Commands	The target vector
0 (left)	[1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	6	[−1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1]
0 (right)	[−1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	7	[−1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1]
1	[−1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	8 (a)	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1]
2	[−1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	8 (b)	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1]
3	[−1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	9	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1]
4	[−1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	Space	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1 −1]
5 (a)	[−1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1 −1]	Backspace	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1 −1]
5 (b)	[−1 −1 −1 −1 −1 −1 −1 1 −1 −1 −1 −1 −1 −1 −1 −1]	Carriage return	[−1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 −1 1]

where:

$f(\cdot)$ is the transfer function of the neurons,

b^1 and b^2 are the weights of the bias in the input layer and hidden layer respectively,

w^1 is the vector of the connection weights between the input layer and the hidden layers,

w^2 is the vector of the connection weights between the hidden layer and the output layers,

n is the number of hidden node,

y is the output of the hidden layer,

x is the input vector,

z is the output of the network.

The sixteen bits of the neural network enter the last stage of recognition that is called determination. This determiner receives these inputs and assesses them to show the system's decision by applying Equation (4). The determiner finds the maximum value of the 16 outputs to display the output.

$$\text{Decision} = \text{index}\left(\max_{i=1\to 16}\left(z_i\right)\right) \tag{4}$$

where z_i is the output vector of the classifier.

2.2.2. Parallel Neural Networks (PNN) Classifier

The parallel neural networks classifier is an arrangement of neural networks connected to the same input. Instead of using one neural network with multi-output, this classifier adopts multiple neural networks with one output for each network. Each class (digit) has its own neural network that recognizes it. Therefore, sixteen neural networks are required to recognize the sixteen patterns, as illustrated in **Figure 5(a)**. More classes can be added at any time without any change in the networks that are designed previously. So, this classifier is more flexible than the first one and this is the main of its features.

Figure 5(a) shows the structure of the parallel neural networks (PNN) classifier. Each network in this system has multi-inputs, which receive the features of the digits, and one output that represents the decision of the network (as a number between −1 and 1). Therefore, the desired output of each network is one bit either 1 (for the corresponding pattern) or −1 (for not corresponding pattern). For example, the digit "0" (a) is represented by the sixteen bits that are shown in **Table 3**. A feed forward neural network with multi-layers has been designed for each pattern. Consequently, the system includes 16 feed-forward networks. The experiences demonstrated that using the same structure of the neural networks for all classes can simplify the system's design and keeping the same accuracy. However, each network has its own parameters (the connection weights) which will be adjusted during the training process. Therefore, each of these networks will be trained individually to recognize a specific pattern only. **Figure 5(b)** illustrates one of these neural networks and demonstrates the flow of the signal in one direction from the input layer to the hidden layer, then to the output layer. The output of this network is calculated by using Equation (4). It is clear from **Figure 5(a)** shown that each network has only one output. The determiner of this classifier measures the system decision by checking the output of each network. Only one of these sixteen networks produces a positive output and the rest 15 network have negative outputs.

Finally, the determiner is working as an interpreter to present the final decision of the system. The determiner of the previous classifier can be used for this classifier without any change. Therefore, this classifier adopts Equation (4) in making the final decision. This classifier has solved the problem of the future extension because it allows adding extra classes without retraining the networks. Only the determiner will be updated to consider the new classes. However, this classifier has a complex structure when it handles too many classes. Therefore, this can be indicated as a drawback of this classifier.

2.2.3. Tree-Structured Neural Network (TSNN) Classifier

Handwriting recognition, in general, is a complex process because it deals with too many classes (letters, digits and commands). The variation in the writing styles increases the complexity of the recognition system by increasing number of the classes. This number of classes needs high arranged classifier that divides these classes into groups of sub-classifiers. Therefore, TSNN network classifier can be a good solution for this problem. Besides its ability in handling too many classes, it will be very flexible in accepting new extra classes in the future.

The tree-structured neural network classifier is a combination of organized neural networks. It has two classification stages (two phases of networks) that are connected serially. The first phase is working as a selector to divide the patterns into groups, as can be seen from **Figure 6**. It activates the corresponding sub-classifier to in-

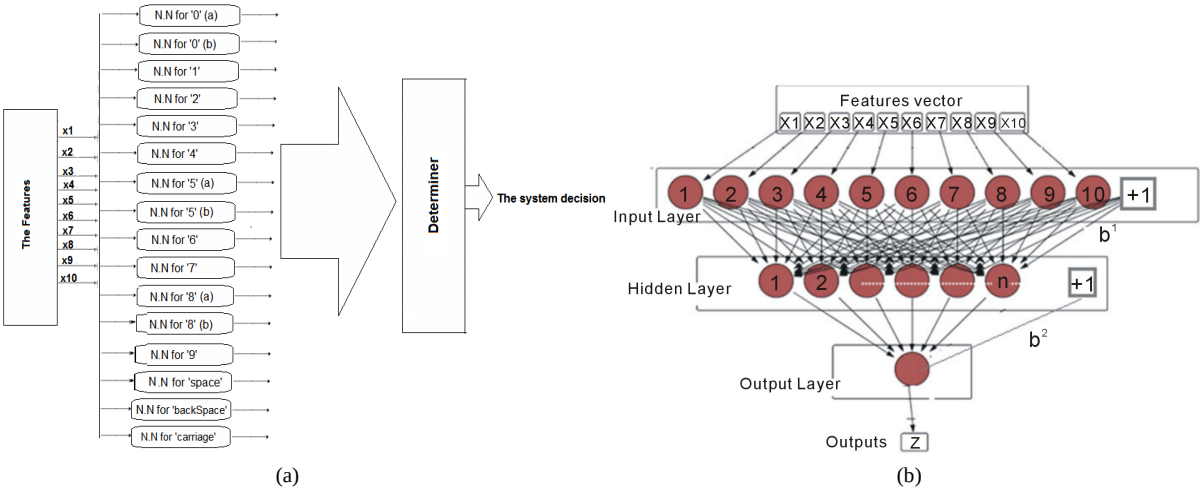

Figure 5. (a) Block diagram of PNN classifier; (b) one of neural networks of PNN classifier.

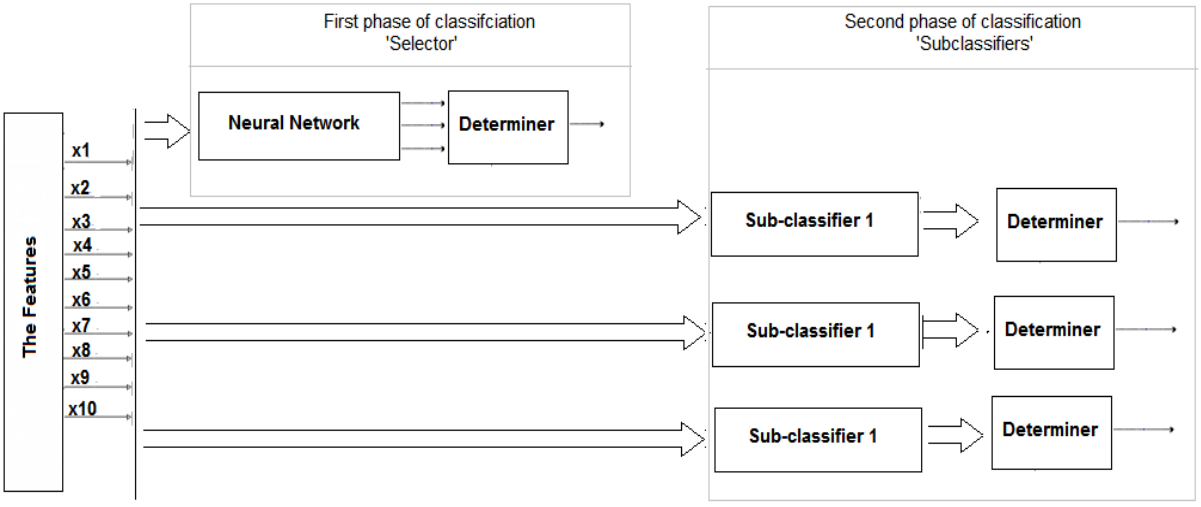

Figure 6. The block diagram of TSNN classifier.

Table 3. The networks' outputs of the system for the digit "0" (a).

Net 1	Net 2	Net 3	Net 4	Net 5	Net 6	Net 7	Net 8	Net 9	Net 10	Net 11	Net 12	Net 13	Net 14	Net 15	Net 16
1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1

itiate the second classification phase. Therefore, only one sub-classifier will be activated based on the decision of the first phase.

Obviously, in the handwriting recognition system, the patterns can be either digits, commands or letters. This system is designed to recognize the digits 0 to 9 and some commands (space, backspace and carriage). The future development of this system can expand the classifier to recognize letters, in addition to the digits and commands. Therefore, it is important to consider this direction in the current structure. In this regard, this classifier is very flexible and reliable in supporting many more classes to be recognized without any need to retrain the sub-classifier of the second phase. However, the network of the first phase requires new training with any extra class or group of classes (sub-classifier). The sub-classifiers of the second phase are designed as three groups of parallel neural networks connected to the same inputs, as shown in **Figure 6**. The first group has 13 parallel neural networks to recognize the digits from 0 to 9 while the second group includes 3 neural networks for the three commands. One of these networks is shown in **Figure 5(b)**. However, the third group will not be designed

at the current stage, and this can be the first future direction for this research.

Although the first classification phase is simple and has only one neural network (as shown in **Figure 7**), it is no less important than the second phase. The second phase classifier can use the same design and configuration of the neural networks in the previous proposed classifier (PNN classifier). **Table 4** demonstrates the role of the first classification phase while the second phase classification is already explained in **Table 2**.

Finally, the selector classifier (first phase) needs one determiner to calibrate the output of the neural network based on Equation (5).

$$\text{Select} = \text{index}\left(\max_{i=1\to S}\left(z_i\right)\right) \tag{5}$$

where z_i is the output vector of the selector, and s is the number of the sub-classifiers.

Furthermore, a determiner is required for each sub-classifier in the second phase to identify the final decision, as the below:

$$\text{Decision} = \text{index}\left(\max_{i=1\to n}\left(z_i\right)\right) \tag{6}$$

where z_i is the output vector of the selector, and$n$ is the number of the classes (output neurons) for the sub-classifier.

3. Pre-training and Initialization

3.1. Learning Samples

Collecting the training samples are the first step in the training process. Considering too many samples (that are written manually by different people), can increase the ability of the system in overcoming the problem of the input variation. As explained previously, the vectors of these characters are not fully representative and include too much not useful values. Consequently, the feature extraction process is very important to prepare expressive information. The dataset consists of ten normalized features for each digit or command with the desired output that defines the corresponding input. The dataset of the collected samples are divided into three groups: a training set, a validation set and a testing set. **Table 5** demonstrates these sets and number of the samples in each set. These samples can be arranged in a linear sequence (as they are taken naturally) or in a randomized sequence. With some experiences, I have found that the randomized sequence is much better than the linear way, and it is preferable because it brings diversity in the token data. Consequently, the random sequence for the data can support and accelerate the learning and the validation processes.

Because the used transfer function is either hyperbolic tangent function or sigmoid function, the input data should be normalized before feeding the neural networks. In this regard, the below equation has been used to normalize the input in the range (−1, 1).

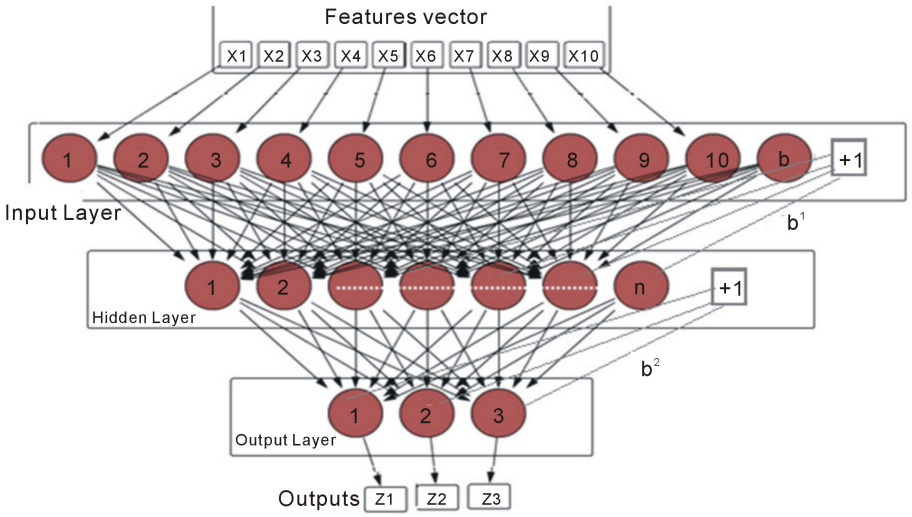

Figure 7. The neural network of the selector.

Table 4. The target of the selector and the corresponding sub-classifiers.

The output vector ([z1 z2 z3])	The activated sub-classifier
[1 −1 −1]	Sub-classifier 1 (digits)
[−1 1 −1]	Sub-classifier 2 (commands)
[−1 −1 1]	Sub-classifier 2 (letters)

Table 5. The percentages of the three sets of the data.

Set	Number of the samples	The percentage
Training set	1250 samples	60%
Validation set	624 samples	20%
Test set	624 samples	20%
Total	2080 samples	

$$x_{norm}^{j} = 2 * \left[\frac{\left(x^{j} - x_{\min} \right)}{x_{\max} - x_{\min}} \right] - 1 \tag{7}$$

where, x^j refers to each element in the input vector, $x_{\min}$ is the minimum value in the input vector and $x_{\max}$ is the maximum value in the input vector.

3.2. Initial Weights

The networks of the three classifiers are designed as multilayer feed-forward neural networks that are trained by the back-propagation algorithm. This algorithm is widely used in learning neural networks because of its accuracy and versatility. However, it is slow training method and sometimes taking too long time to get proper connection weights because it works based on steepest descent technique [16]. Therefore, several studies and researchers have developed methods to improve the speed of the training process. In this respect, they develop techniques and procedures to set the initial weights within a specific range instead of random values. Practically, the speed of the training process is measured by the number of the required iterations to achieve the final weights that can minimize the error in the output signal to the lowest applicable value. Researches have presented many methods to generate initial weights within a certain range to accelerate the training process and affect the speed of convergence.

Equation (8) is one of the useful techniques to initiate the connection weights. This method is proposed by [14]. Based on this formula, the initial weights depend on the number of the input units and the hidden neurons.

$$\text{randz}(i, j) = (i * \theta) - \frac{(\beta)}{j} - \frac{(\mu * j)/i}{c} \tag{8}$$

$$\theta = e^{(0.9)} \tag{9}$$

$$\beta = \theta - 1 \tag{10}$$

$$\mu = \beta^{0.075} \tag{11}$$

where, $i = 1, 2 \ldots n_1$, n_1 refers to the number of the input units, $j = 1, 2 \ldots n_2$, n_2 refers to the number of hidden neurons, and c is constant (In this paper, $c = 10$).

3.3. The Hidden Layer

As stated previously, the networks of the proposed classifiers have been built as feed forward multilayer neural networks. These networks are created with no more than one hidden layer because the problems that are solved by a multi-layer perceptron with multi hidden layers can also be solved with only one hidden layer [17]. In addition, [15] points out, “Experience has shown that training time increases substantially for networks with multiple hidden layers”. Measuring the number of the neurons in the hidden layer is one of the most important considerations and controversial issues. In spite of the importance of these units in processing the nonlinear data, there is

no efficient and direct method or formula to calculate their number in all networks because each system has his owned behavior. The number of these neurons has a significant impact on the performance of the system. With many hidden neurons, the network will have more complex decision surfaces, so we expect better classification accuracy. However, with redundant hidden neurons, the network may have too much modelling power that it can model the idiosyncrasies of the training data, undermining its performance on testing data [15]. Zurada [18] presents a general formula to determine the minimum number of hidden neurons, *M*, that are required to build multilayer neural networks, as the below:

$$M = 1 + \frac{k(k+1)}{2} \tag{12}$$

$$M = 2^{J^*} \tag{13}$$

$$J^* = J - 1 \tag{14}$$

where *K* is the number of the output neurons and *J* represents the number of the necessary neurons in the hidden layer. However, this is only the minimum number and may not represent the best number. So, there are some techniques can be used to determine the required number, such as genetic algorithm or trial and error. These systems use the trial and error to solve this issue.

4. Results

4.1. Single Neural Network (SNN) Classifier

4.1.1. Training

In the training process, this classifier will be provided pairs of training data (input and target vectors). The collected samples are paired with the target vectors which indicate the corresponding pattern. Then, the back-propagation algorithm is used to train the network. The role of the teacher (trainer) is very important to learn the classifier because it is a supervised learning rule. All of the uncertain issues in the architecture will be solved at this stage, such as the number of the hidden neurons, the learning rate and the model of the activation function. After solving these problems, the network will be ready for the training and the validation processes.

The significance of finding the proper number of the hidden neurons in the classification process has been taken to be the first consideration in the training process. As explained previously, the number of the hidden neurons can be specified by trial and error. **Figure 8** demonstrates the root mean square error curves for different designs (different number of neurons in the hidden layer). It is clear from these curves that the three cases 30, 41 and 48 are the best values and they are predicted to propose the best design for the network. They are selected based on the root mean square error for the network and the speed of the learning process to achieve the final parameters. However, it is difficult to decide what the best choice from the three designs is. Therefore, the testing stage will answer this question and show the right decision because they are chosen to support the network to provide a system with high accuracy in the recognition operation. The curve of **Figure 8** is calculated as accumulated mean square errors, as shown in Equation (15). The figure concludes the effect of the hidden neurons on the outputs of the neural network and the convergence between the practical and the desired outputs. It illustrates the strong impact of these neurons on performance and speed of the network and its error.

$$E_{r.m.s} = \frac{1}{PK}\sqrt{\sum_{P=1}^{P}\sum_{K=1}^{K}\left(d_{pk} - z_{pk}\right)^2} \tag{15}$$

where *P* is the number of the patterns, *K* is the number of the outputs, d_{pk} is the desired output and $\mathbf{z_{pk}}$ is the actual output.

The aim of the training process is to update the connection weights of neural networks. It adopts the shape of the error curve to achieve the highest applicable accuracy. The value of the learning rate is an important factor to be considered during the training process. If the learning rate is too small, the training process will be very slow while when it becomes too high, the training will overshoot the downward slope which can lead to an oscillation in the output of the network. In between these, there are suitable values that can provide a good response. Many factors can affect this rate, but there is no reliable formula or direct method to consider these factors in determining the optimal learning rate. Therefore, different values for this rate will be examined (from 0.00001 to 10) to record their impacts on the speed of learning and quality of the training, as shown in **Figure 9**. Therefore, it

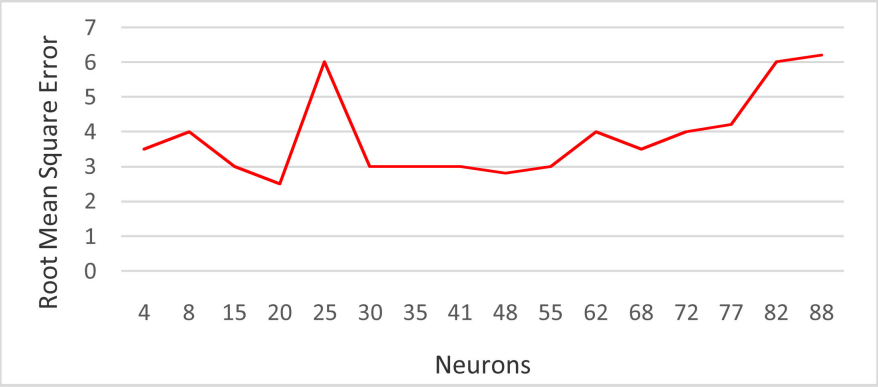

Figure 8. The final root mean square error of different hidden neurons.

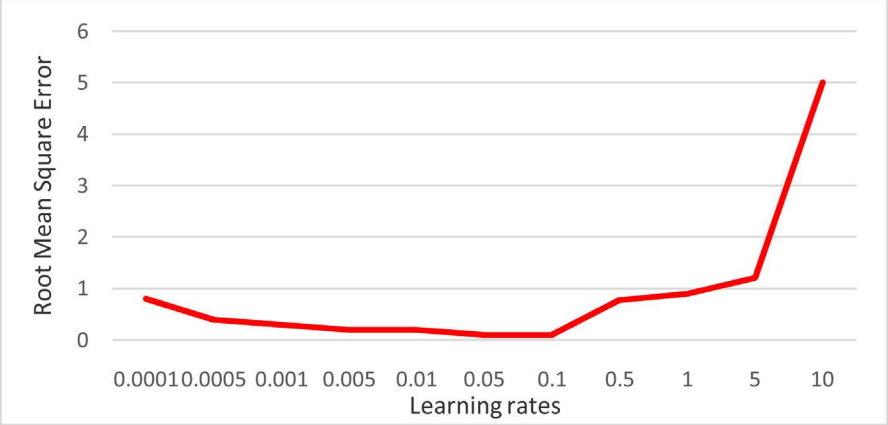

Figure 9. The root mean square error with different learning rates.

illustrates an approximate image about the best learning rate for the network. The proper rate can be selected by examining sixteen different values. Some values offer good shapes for the error curve and any one of them can be considered to be the proper one. Thus, the rate of 0.05 presents a stable shape with the lowest error and acceptable speed. Therefore, it is selected to be the learning rate of the neural network.

The back-propagation rule has two activities in two directions: a forward and backward. During the classification process, the signal is passed in a forward direction by feeding the input data from the input units to the hidden neurons through weighted connection between them. Other weighted connections carry information from the hidden layer to the output layer. This information will be processed by the activation functions of the hidden and output neurons. On the other hand, the training procedure is propagated in a backward manner by computing the error of the output neurons then updating the weights of the connection using the delta rule method (the back-propagation). The training data are divided into two sets: a training set and a validation set. Although the training is the main process that learns the network and defines its parameter (weights), it is maybe not enough to assign the final parameter of the network without validation. Therefore, the validation process is an important operation to validate the learning procedure and verify the stopping point (the necessary number of the training iteration). However, there is no adjusting or updating for the weights at this process.

As indicated before, the structure of the system also will be examined during the learning process to set some of the uncertain parameters such as the number of the hidden neurons and model of the activation functions for the hidden and output neurons. So far, there are three possibilities for the hidden neurons: 30, 41 and 48 neurons and for two activation functions: the bipolar sigmoid function and the tangent function, as shown in **Table 6**. One of the objectives of this research is to analyze all factors that contribute in affecting the accuracy of the system and its reliability. Consequently, all of these cases will be considered and examined in this paper through the training, the validating and the testing to present a comparison between different designs for the neural networks to show the

Table 6. Characteristics of the neural network of SNN classifier.

Model of the network	Feed-forward neural network
Number of the neural network	1
Number of the input units	10
Number of the hidden neurons	Case 1: 30 neurons Case 2: 41 neurons Case 3: 48 neurons
Number of the output neurons	16
Learning rate	0.05
The learning algorithm	Supervised algorithm: Back-propagation

impact of these factors on performance of the classifiers. Then, the final design of these classifiers will be proposed after the testing.

4.1.2. Testing

Table 7 records the test results of the single neural network (SNN) classifier. These results indicate the following points:

1) The number of the hidden neurons has a significant impact on the network's performance. Although the three cases: 30, 41 and 48 hidden neurons present similar behavior during the stage of determining the number of the hidden neurons; each of them shows different accuracies and behaviors. Firstly, thirty hidden neurons were not enough to provide an optimal design because the network presents low accuracy. Therefore, the network needs more neurons in the hidden layer to enhance its performance. However, increasing the number to 48 neurons reduces the accuracy and speed of the network. Finally, 41 hidden neurons support the network to achieve the highest accuracy. Therefore, beyond a certain number of neurons in the hidden layer, the training process can be too long and complex and also the network may not achieve the desired behavior.

2) In spite of the similarity between the two activation functions: tangent and bipolar sigmoid function, the two networks have a difference in their accuracies. According to the results of the test, the tangent function presents higher recognition abilities for the graffiti digits.

Finally, the test results indicate that the overall accuracy is 95%. Therefore, it meets the minimum requirements. However, it may not accurate enough to provide a reliable recognition system.

4.2. Parallel Neural Networks (PNN) Classifier

4.2.1. Training

The back-propagation algorithm is used to train all of the sixteen neural networks individually to specify the parameters of each network. The training procedure that is followed in the first classifier will be applied in training the networks of this classifier. The number of the hidden neurons and the learning rate have been examined with each network and chosen by trial and error (as described in Section 4.1.1).

In this paper, I discussed and explained the impacts of all of the learning factors (such as the hidden neurons, the learning rate, the activation function and updating the connection weights) intensively in the first classifier. This is because the first classifier has a single neural; thus, the impact of these factors can be shown clearly. Therefore, I focus on analyzing and studying the performance of the classifier instead of re-discussing the training procedure that is already presented. As a result, the final properties of the neural networks of this system are demonstrated in **Table 8**.

4.2.2. Testing

The accuracy of the system has been increased with PNN classifier compared with previous classifier (SNN), as shown in **Table 9**. The overall accuracy of SNN classifier was 95% while PNN classifier presents an efficient system with high accuracy and flexibility. It is clear from **Figure 10** and **Table 9** that the overall accuracy of this system exceeds 98%, but there is some confusion in recognizing the two digits: 4 and 9 because of the similarity in their shapes and coordinates. Although this similarity results in reducing their accuracy to about 95%, but this is still in the acceptable range. In the worst case, the classifier satisfied the minimum requirements and presented 98% overall accuracy for all digits. The flexibility is another important advantage for this classifier.

Table 7. The test results of the SNN classifier.

Activation function	Number of the hidden neurons	Accuracy	Number of iterations	Root mean square error
Tangent	30	91%	675	1.2
	41	95%	700	2
	48	89%	1500	0.2
Bipolar sigmoid	30	90%	1100	0.35
	41	93%	1000	0.32
	48	88%	1300	0.5

Table 8. The test results of the SNN classifier.

Model of the network	Feed-forward neural network
Number of the neural network	10
Number of the hidden neurons	35
Number of the output neurons	1
Learning rate	0.05
The learning algorithm	Supervised algorithm: Back-propagation
The activation function	Tangent function

Table 9. The test results of the PNN classifier.

Sq.	The network	Accuracy %	Number of the iterations	The root mean square error
1	0 (left)	99%	1700	0.002
2	0 (right)	99%	3550	0.0005
3	1	99%	3000	0.003
4	2	96%	4000	0.005
5	3	98.5%	500	0.0019
6	4	95.2%	4300	0.0004
7	5 (a)	98.7%	2500	0.0002
8	5 (b)	99.1%	3000	0.00001
9	6	98.5%	4200	0.0006
10	7	98.5%	3000	0.0003
11	8 (a)	98.5%	200	0.003
12	8 (b)	99.5%	3300	0.0005
13	9	96%	3400	0.0002
14	Space	99.5%	2650	0.0001
15	Backspace	99.5%	1300	0.0001
16	Carriage return	98.5%	2500	0.0005
Overall accuracy	98%	3000	0.001	

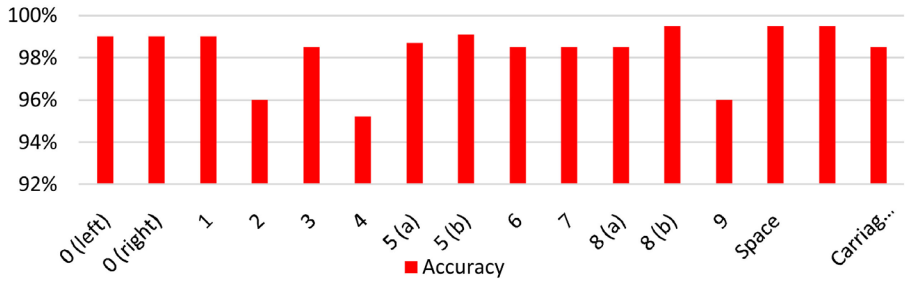

Figure 10. The test results for each network in the PNN classifier.

With this classifier, extra classes can be added any time without affecting the trained networks. Therefore, the future development can be done easily by adding new trained neural networks to the system. However, with too many classes, the efficiency of the system will be decreased and the complexity will increase.

4.3. Tree-Structured Neural Network (TSNN) Classifier

4.3.1. Training

The tree-structured neural network (TSNN) classifier is a neural network based classifier. It adopts two phases of classification through highly arranged construction. This results in very flexible and reliable system to recognize handwritten digits. This classifier has been introduced and paid attention to be extended to classify too many extra classes. Therefore, this can be described as a part of a bigger project to build an intelligent system to recognize the handwritten digits and letters. The system should be able to achieve high accuracy and reliability. The tree-structured neural network classifier works on the principle of dividing the classifier into sub-classifiers that can classify groups of classes. The first classifier phase has been built as a feed-forward multilayer neural network with one hidden layer and three neurons in the output layer. Several configurations have been applied and examined to get the final structure that is shown in **Table 10**.

The input units of this network receive the features of the pattern and pass them into the hidden layer then to the output layer via connection weights. The computations and the signal processing are made in the hidden and output layers. This network is designed to classify the patterns into three groups: digits, commands and letters (as explained previously in Section 2.2.3). Therefore, the network has three outputs which refer to these classes. The target vectors of this network are shown in **Table 4**. It can be seen from this table that the first output presents 1 if the input pattern is one of the digits 1 - 9 while the second output becomes 1 only with the commands. Lastly, the third output is considered to recognize the letters (A to Z), but this part is not implemented at the current stage and it will be one of the future works for this research.

The training and validation processes of the first phase are illustrated by the curve of the root mean square error that is shown in **Figure 11**. It is trained based on the networks' configuration that is indicated in **Table 10** by using the back-propagation algorithm.

The second phase of classification includes three sub-classifiers. The first sub-classifier has 13 parallel neural networks that recognize the digits 0 - 9 while the second sub-classifier contains three parallel networks to recognize the three commands: space, backspace and carriage. These two sub-classifiers use the neural networks that are already trained in the previous classifier (PNN) while the last sub-classifier will be the future direction of this research to recognize the letters: A-Z. **Table 11** illustrates the structures of these networks. The back-propagation algorithm has been used to train all neural networks of this classifier.

Table 10. Properties of the neural networks of the TSNN classifier.

Model of the network	Feed-forward neural network
Number of the input units	10
Number of the hidden neurons	25
Number of the output neurons	3
The learning algorithm	Supervised algorithm (Back-propagation)
Learning rate	0.05
The activation function	Tangent function

Table 11. The properties of each of neural networks in the second phase of TSNN.

Model of the network	Feed-forward neural network
Number of the input units	10
Number of the hidden neurons	35
Number of the output neurons	1
The learning algorithm	Supervised algorithm (Back-propagation)
Learning rate	0.05
The activation function	Tangent function

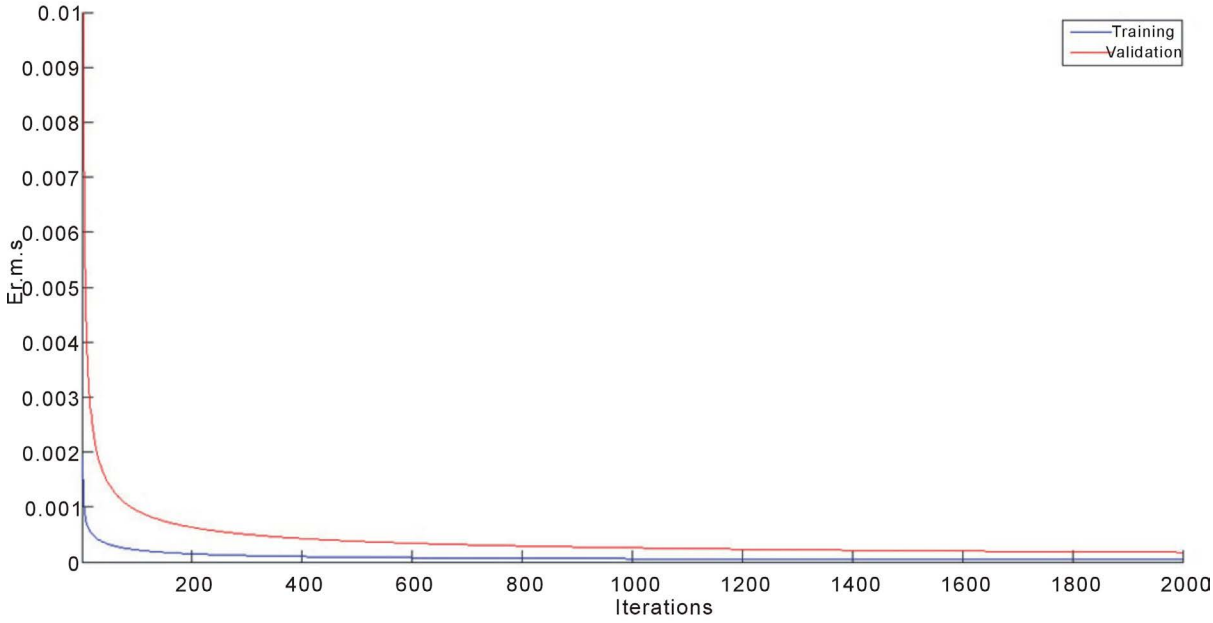

Figure 11. The root mean square error curves for the training and validation process.

4.3.1. Testing

Using the tree-structured neural network classifier has brought a recognition system with very high accuracy and flexibility. Because of the highly arranged construction of this classifier, the signal undergoes two stages of classifications. It can be noted in **Table 12** that this arrangement results in higher accuracy and lower confusion in the recognition process. The first classification task classifies the pattern to be one of the three groups: digits, commands or letters. It is implemented in a multilayer neural network and achieved 100% accuracy with this simple design. Although the first classification phase is only initial operation, it is the most important task in the recognition process because it will direct the signal to enter the correct sub-classifier. Therefore, achieving this accuracy supports the whole classifier in its function. Secondly, the current design of the system has three sub-classifiers in its second classification phase. As detailed previously in Chapter 3, the second phase has sub-classifiers which are trained to classify specific patterns. Consequently, the three sub-classifiers of the system have different parameters that are set to perform the function that is designed for. The accuracy of the first sub-classifier is very high and satisfactory in recognizing the digits: 0 to 9. The same results have been recorded for the second sub-classifier that is designed to recognize the three commands: space, backspace and carriage. Such classifier (with 99% accuracy) is very reliable in establishing a recognition system. In addition, TSNN classifier is very flexible with the future developments because other classes or sub-classifiers can be added without affecting the current sub-classifiers. Only the first classification phase will be updated to consider a new group (or groups) of patterns. Moreover, the tree of the classification can be extended by dividing the sub-classifiers into sub-groups. Therefore, TSNN classifier is very flexible with any type of extension and development.

5. Discussion

This paper proposed three different classifiers, namely single neural network (SNN) classifier, parallel neural networks (PNN) classifier and tree-structured neural network (TSNN) classifier. The aim of introducing the first classifier, despite its low flexibility (compared with the second and third classifiers), is to make an analytical study about the impacts of some of the architectural elements of the neural network on the performance of that network and the learning procedure. Specifically, it discussed the role of the hidden neurons, the activation function and the learning rate. All of the neural networks of the three presented classifiers have been designed as feed-forward networks with multilayer. The difference between the proposed classifiers is their structures. Firstly, the first classifier has a simple and conventional design with the lower number of units and quicker processing comparatively with the second and third classifiers. However, it has low flexibility with future developments because adding an additional class requires retraining the whole network. In the other words, it is a static classifier and

Table 12. The test results of the TSNN.

The classification phase	The sub-classifier	Accuracy	Number of the training iterations	The mean square error
First Phase		100%	1700	0.0002
	Sub-classifier1	99%	3500	0.001
Second Phase	Sub-classifier2	99.5%	2000	0.0003
	Sub-classifier3	NA	NA	NA

the network is designed to recognize predefined classes only, and adding any extra class causes to replace the trained network. For example, this system is designed and trained to recognize ten digits (from 0 to 9) and some characters. As a future direction, it is predicted to extend this system to recognize all of the alphabetic letters and additional commands. This network is not flexible for any extension and development.

The second classifier and the third classifier were very accurate and flexible. The test results prove that the two classifiers are reliable in recognizing the handwritten graffiti digits. The third classifier (TSNN) overcomes the second classifier (PNN) in its ability in classifying too many classes with lower complexity and higher accuracy and flexibility. **Table 13** shows a comparison between the three classifiers based on the following factors:

1) Average accuracy: it refers to the accuracy of the system in recognizing the patterns.
2) Complexity: it is the number of the neural networks and neurons of each network.
3) Recognition scale: it means the ability of the system in handling large scale of classes.
4) Flexibility: it refers to the adaptation of the system with the future developments and its ability in classifying extra classes without affecting the trained networks.
5) Recognition speed: it is the required time to classify one pattern.
6) Recognition rate: it is the percentage of the pattern that are recognized successfully.
7) Error rate: it is the percentage wrong recognitions.

Table 14 indicates the differences between architectures of the three classifiers. Although all of the three classifiers are created based on a neural network topology, they are designed with different approaches. SNN has one neural network with multi-inputs and multi-outputs. The input units receive the ten features of the pattern to classify that pattern to be one of sixteen classes. Therefore, this neural network has sixteen neurons in the output layer. However, the neural networks of PNN and the sub-classifiers of TSNN have only one neuron in theiroutput layer. Although the two classifiers (PNN and TSNN) have many neural networks, they still not very complex because these networks have only one neuron in the output layer. The number of the hidden neurons was one of uncertain issues because there is no direct method to determine the optimal number. Therefore, this problem has been solved during the training stage by trial and error.

6. Conclusions

Ability of neural networks is well known in solving complex problems through many decision-making techniques. They provide powerful technical tools to learn the machines in classifying patterns. In this paper, the problem of handwritten digit recognition has been investigated by introducing three neural based classifiers. All of the three proposed classifiers use multilayer feed-forward neural networks but with different structures. The three proposed classifiers have been studied and tested in the paper to examine their characteristics and capabilities in recognizing handwritten digits. The study focuses on measuring their accuracy, flexibility and scalability. They have achieved high accuracy and reliability but with a slight difference. The main concern of the research was not only reliability of the system for the current stage (handwriting digits recognition) but also on its capacity for the future developments and extensions to recognize more classes without affecting the trained networks. In this regard, the second and the third classifiers provide efficient systems with very high flexibility. They can be developed to include extra classes with keeping the current trained networks. Although both of the two classifiers: PNN classifier and TSNN classifier offer impressive accuracy and flexibility, TSNN classifier is much better than the PNN classifier in its scalability in handling too many extra classes because PNN is limited in its capacity. The well-arranged structure of the tree-structured neural network (TSNN) classifier grants it durability and competence in separating too many classes for the future developments. Therefore, this classifier is the main achievement of this research because it has high flexibility, high scalability and very high accuracy (over 99%).

Another important achievement in this research is introducing a comprehensive study about the impacts of some factors on the training process and the performance of neural networks by employing the first classifier to

Table 13. Comparison between characteristics of the three classifiers.

The classifier	Average accuracy	Complexity	Flexibility	Recognition scale	Speed	Recognition rate	Error rate
SNN	Acceptable	Low	low	Medium	Very High	95%	5%
PNN	High	Medium	High	High	High	98%	2%
TS	Very High	Medium	Very High	Very High	Acceptable	99%	1%

Table 14. Architecture of the three classifiers.

The model		Number of N.N	Number of determiners	Classification phases	Number of the hidden neurons	Number of the output neurons	Execution time
SNN		1	1	1	41	16	0.001
PNN		16	1	1	35 in each N.N	1 in each N.N	0.005
TSNN	1'st phase	1	2	2	25	3	0.012
	2'nd phase	16			35 in each N.N	1 in each N.N	

describe the study. The first classifier has been chosen because it has single neural network, then it is easier to explain the impacts. The paper focuses on the effect of the three factors: the number of the hidden neurons, the activation function and the learning rate because they are uncertain issues and there is no direct method to specify them. It can be concluded from the test results that they influence behavior of the network and its error rate. Thus, considering them and well choosing of their values can improve the classification process significantly and ensure correct learning. Different probabilities and several configurations have been considered and applied to show their effects on the results through many curves and graphs.

Finally, it can be concluded from this paper that the performance and the quality of any handwriting recognition system can be measured by two operators. The first operator is how well the structures of the classifier are organized to process the current problem efficiently and to be flexible with the future developments. Secondly, how accurately the parameters and units of the neural network have been optimized and employed in supporting the network's function and handling the great variety of the hand printed digits or letters.

This research results in creating a robust and reliable system to cope the variability of the handwriting process; thus, it provides successful methods for the pattern recognition. The final analysis asserts that the tree-structured neural network (TSNN) classifier is designed not only to be successful in handling the current problem (handwritten graffiti digit recognition), but it also originated to establish a bigger research about recognizing handwritten letters and digits. Therefore, the future extensions have been considered strongly in this research through building a flexible system that is able to handle too many more classes with high efficiency. Thus, the first future direction is developing the tree-structured neural network classifier to recognize the letters (A-Z).

This research builds the basic architecture of this work by creating a flexible classifier. This development for the research is concluded in retraining the selector (the first phase) of the tree-structured neural network classifier with the new patterns and designing the third sub-classifier to recognize the new classes (the letters). The first and second sub-classifiers will not be affected by these processes. Therefore, the current system can be described as the first part of a bigger system. Although the new part will be designed and trained based on outcomes and results of this paper, it needs more attention and investigation to cope the great variety of the hand printed letters. It is recommended to use a hybrid system (Neuro-Fuzzy) in classifying these letters to handle the strong non-linearity of these patterns. Moreover, all of the analyzed techniques that are described in the paper could be applied to recognize digits that are written in other languages. This direction can be another step toward creating a robust system for the electronic books (eBooks) and post offices. As a result, there is no limit for the future directions of this research because there are too many languages to consider and each of them has its characters and styles.

References

[1] Srihari, S.N. and Lam, S.W. (1995) Character Recognition.

[2] Jin, L.W., Chan, K. and Xu, B.Z. (1995) Off-Line Chinese Handwriting Recognition Using Multi-Stage Neural Network Architecture. *IEEE International Conference on Neural Networks*, Perth, 27 November 1995-1 December 1995, 3083-3088.

[3] Gu, L.X., Tanaka, N.K., Kaneko, T. and Haralick, R.M. (1999) The Extraction of Characters from Cover Images Using

Mathematical Morphology. *Systems and Computers in Japan*, **29**, 33-42.

[4] Suen, C.Y., Nadal, C., Legault, R., Mai, T.A. and Lam, L. (1992) Computer Recognition of Unconstrained Handwritten Numerals. *The Proceedings of the IEEE*, **80**, 1162-1180.

[5] Zhang, P. (2006) Reliable Recognition of Handwritten Digits Using a Cascade Ensemble Classifier System and Hybrid Features. Doctor of Philosophy, The Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec.

[6] Kanungo, T. and Haralick, R.M. (1990) Character Recognition Using Mathematical Morphology. *Proceeding of the United States Postal Service Advanced Technology Conference*, Washington DC, November 1990, 973-986.

[7] Gu, L.X., Tanaka, N. and Kaneko, T. (1996) The Extraction of Characters from Scene Image Using Mathematical Morphology IAPR Workshop on Machine Vision Applications. Tokyo.

[8] Badr, A.-B. and Haralick, R.M. (1994) Symbol Recognition without Prior Segmentation. 303-314.

[9] Kumar, V.V., Srikrishna, A., Babu, B.R. and Mani, M.R. (2010) Classification and Recognition of Handwritten Digits by Using Mathematical Morphology. *Indian Academy of Sciences*, **35**, 419-426.

[10] Li, X. and Yeung, D.Y. (1997) On-Line Handwritten Alphanumerics Character Recognition Using Domimnal Points h Stmkes. *Rttem Recognition*, **30**, 3144.

[11] Casey, R. (1970) Moment Nomurliwtion of Handprinted Charoclers. *IBM Joumd of Research Development*, **10**, 548-557.

[12] Ling, S.H., Lam, H.K. and Leung, F.H.F. (2007) Input-Dependent Neural Network Trained by Improved Genetic Algorithm and Its Application. S*oft Computing*, **11**, 1033-1052. http://dx.doi.org/10.1007/s00500-007-0151-5

[13] Ling, S.H. (2010) A New Neural Network Structure: Node-to-Node-Link Neural Network. *Journal of Intelligence Learning Systems and Application*, **2**, 1-11. http://dx.doi.org/10.4236/jilsa.2010.21001

[14] Tayfur, G. (2012) *Soft Computing in Water Resources Engineering*: *Artificial Neural Networks, Fuzzy Logic and Genetic Algorithms*. WIT Press.

[15] Tebelskis, J. (1995) Speech Recognition Using Neural Networks. PhD Thesis, Carnegie Mellon, Pittsburgh.

[16] Yam, J.Y.F. and Chow, T. (2000) A Weight Initialization Method for Improving Training Speed in Feed forward Neural Network. *Elsevier Science*, **30**, 219-232.

[17] Cybenko, G. (1989) Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, **2**, 303-314. http://dx.doi.org/10.s1007/BF02551274

[18] Zurada, J.M. (1992) Introduction to Artificial Neural Systems. West Publishing Company, St. Paul.