NOVATEUR PUBLICATIONS
JournalNX- A Multidisciplinary Peer Reviewed Journal
ISSN No: 2581 - 4230
VOLUME 8, ISSUE 4, Apr. -2022

PATTERN RECOGNITION OF PROGRAMING LANGUAGE WITH THE IMPLEMENTATION OF A BINARY FORMATTING

Shallaw Mohammed Ali
Department of Computer Engineering Technologies,
Al-Qalam University College, Kirkuk, Iraq
Shalaw.eng@alqalam.edu.iq

ABSTRACT:

In the current era of digital information and technology, the importance of developing software applications that enable the access and the use these information technologies raise dramatically. This requires the raise of various type of programming languages which makes it difficult to distinguish these different languages for the beginner learner and developer. Consequently, the ability of detecting and recognizing each programming language is very useful in both the fields of learning and developing of software applications and programs. Therefore, many studies were conducted to investigate the ability of using pattern recognition models of Artificial neural networks to distinguish between different syntax codes from several programming languages. These endeavors were performed using alphabetic presentation of the syntax codes of each programming language. So, in this paper, we propose a method of converting and reformatting the alphabetic format of input data into a binary presentation of the targeted programming syntax codes. This is conducted and tested by using a backpropagation neural network BPNN algorithm via KNIME toolkit.

In the results, the BPNN recognition of the proposed Binary reformatting of the alphabetical syntax codes shows the ability of distinguishing the targeted set of programming language with a high percentage of accuracy. These results will be of interest to researcher community of studying technical learning materials and the applications of recognition models.

Keywords: Binary format, Pattern recognition, Artificial neural network, Backpropagation neural network.

I. INTRODUCTION:

In the past decade, a wide variety of programming languages have been created for a variety of tasks in the field of software development [1]. This increases the confusion for the new learners and developers to distinguish this wide range of programming languages, especially when developing complex software systems [1]. This challenge increases the need for an automated way to recognize syntax source codes of several programming languages based on the common patterns in each of these languages.

Many studies were conducted to achieve this aim via recognition model of Artificial neural network ANN. Both [2] and [3], used a backpropagation neural network methods to distinguish the patterns of the alphabetical grammar in several programming language. In addition, it was recommended by [2] to target the characteristic of these codes instead of their alphabetical grammar patterns for more accurate recognition.

Therefore, in this paper we aim to use form a characteristic pattern of the programming syntax codes and using this pattern in recognition process via neural network algorithms. This is conducted by reformatting the alphabetical presentation of the syntax code

of any targeted programming language of a binary formation of (0,1).

II. BACKGROUND:

A. Pattern Recognition

The methodology of identifying or classifying patterns of any complex environment can be described as Pattern recognition [4] which also known as a process of classification [5]. Studying and monitoring systems to extract potential behaviour and forming decision about it, is one aim of the pattern recognition [6]. Figure 1 shows a basic description flowchart for the pattern recognition [7].

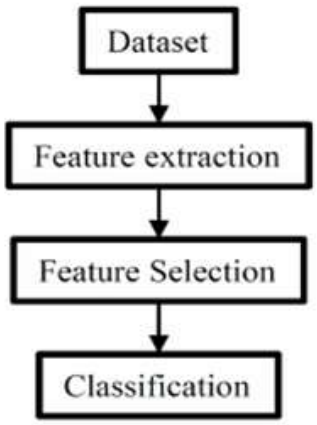

Figure 1: Flowchart of pattern recognition [7]

Accordingly, the collection and the gathering of the targeted datasets comes as the first step of pattern recognition. This is followed by the feature selection of these datasets. Then the extracted features then selected with feature selection methods. This is end with the classification and recognition of the patterns in these features as shown if figure 1.

The models of Pattern recognition can be classified in to three basic types of algorithms (Artificial Neural Network ANN algorithms, Syntactic algorithms, statistical algorithms)[7] from which the ANN model is known for its ability of learning complex algorithms.

B. Artificial Neural Network:

The concept of the artificial neural network (ANN) can be described as the attempt of mimicking the information process in human brain cells which consist trillion of cell and nerves that communicates with each other's electrically via pulses [8].

Accordingly, the ANN concept can solve challenging algorithms which are more complex than the linear algorithm problems.

In addition, according to [9], the ANN also considered as a computer algorithm embedded with elements that form interaction behaviour by interconnecting with each other in reflect to different inputs scenarios.

The structure of ANN can be illustrated as a layer of nodes that uses a mathematical function to interconnect with each other's. Accordingly, the fundamental element of the ANN structure design is the neuron. The neuron in ANN is developed and formed based on the biological neural network of animal and human brain [10]. Figure 2 shows the structure of neuron.

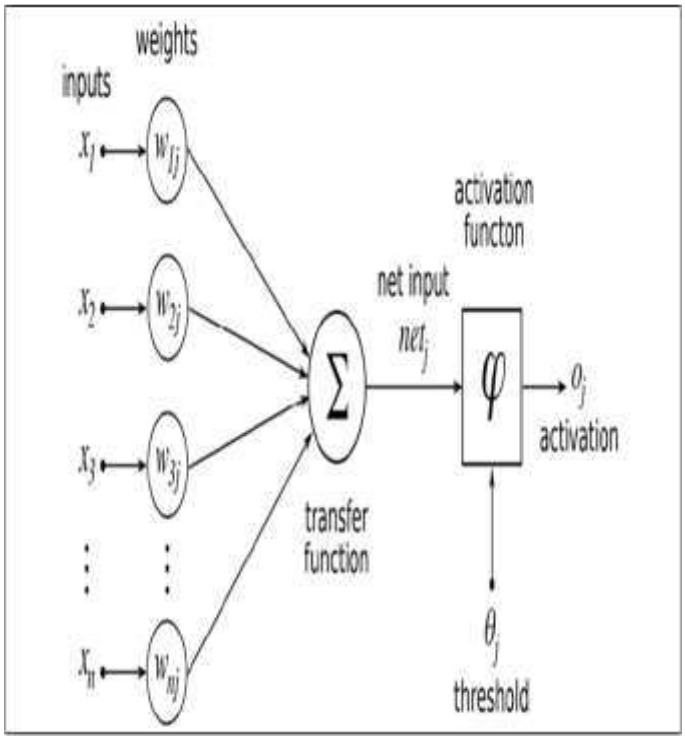

Figure 2: Design of Artificial neuron [10]

The operation of each neuron is implemented by receiving several lines of input. These inputs can be either from the system itself or from other neurons that they connected to.

For each individual neuron, these inputs are weighted separately with the bias and summed up. Then, as shown in Figure 2, this summation task is performed in the specific function known as a transfer function.

Based on the concept of learning, the ANN is categorized in to three types of learning [11] as following:

- **Supervised learning**: In this type of ANN learning, the training process is performed by feeding the neural network with a target behaviour of data output for specific set of inputs. In this type of learning the targeted results should have predefined in advanced.

- **Unsupervised learning**: The targeted results in this type of neural network are not required for the learning process. As the learning algorithm aims to form and extract a behavioural pattern from a set of inserted data. The behavioural pattern then used for the forecasting and non-linear problem decision [12].

- **Reinforcement learning**: the learning process in the Reinforcement ANN is performed through interacting with the system environment. So, the system aims to maximize future reward in learning process[12].

C. Backporbagation neural network (BPNN):

Backpropagation neural network (**BPNN**) is considered as one of most popular supervised ANN models that exploits the concept of backpropagation algorithm in the learning process.

The concept of BNN model was designed in 1970 to address the issues and limitations of the normal neural network (NN) to solve XOR problems[13].

This model is structured as a multi-layer feedforward model that learns from back feeding errors[14]. It's more considered as a learning algorithm rather being a method or approach of neural network [15].

The training process of BNN is performed by feeding the network with two main training parameters, one as a set of inputs and another as a set of targets. This model uses the input datasets to form a network of interconnection based on the best path of interaction led to the target. This new interconnection network can be then used for any similar input [15].

Consequently, BBN consists of the following layers learning: input, hideen and output layer [16] as shown in Figure 3.

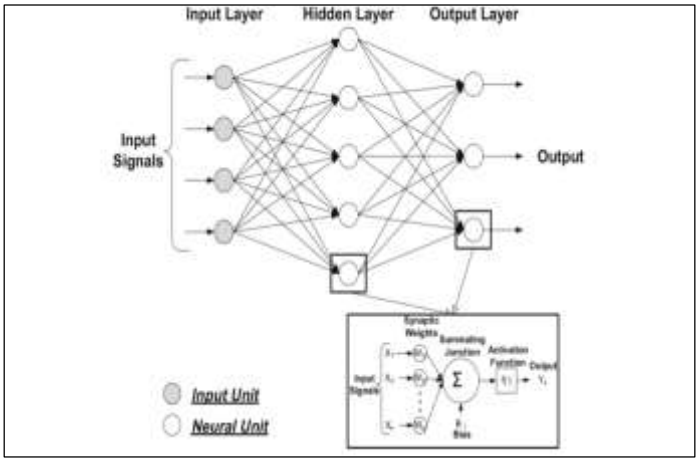

Figure 3: The structure of Backpropagation network [16]

According to [17], the learning concept of BBN exploits the error-correction theory in which the error function is exploited to enhance weights of interconnections in the network in the aim of minimizing the error. In simple term, the weights of each neuron's function is changed repeatedly to modifies the layers' interconnection. Weight tuning works to reduce the distance between the current solution in compared to the desired one.

There are different tools that support the implementation of BPNN methodology such as Python, Matlab and Simbrain.

Source code/syntax	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	;	(	)	>	<	#	space	{	}	/ /
int R;	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
#include <iostream>	1 1	0	0	0	1	0	0	0	1 1	0	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Print(y)	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0
read x	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
cin>>	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1 1	0	0	0	0	0	0

int R;

#include<iostream>

print(y)

main ( )

cin>>

TABLE 1: BINARY FORMAT OF DATA INPUT

III. METHODOLOGY

This study aims to test and investigate the ability of models based on the BPNN methodology to identify and distinguish the pattern of any programming language, such as Java and C++ language. The pattern recognition algorithm which, proposed by Sharma and Kaur [7] shown in Figure 1, is used to conduct this.

This starts with gathering the input data from most common syntax codes for the programming language that aimed for recognition. These syntax codes then reformed int set of binary matrix. From this matrix, the BNN model can select the feature of potential pattern. The selected BNN model then trained with these binary data. The performance of the BNN implementation by Simbrain is evaluate through three investigation tests. In the first test, the BPNN is feeded with the 100 input sentences from common syntax in programming languages to check its ability to distinguish them. Then, in the second, the BPNN model is tested with syntax code input data mixed with normal English sentence for 200 input line of words. Finally, confusion data are used to test the confusion behaviour of the BPNN model.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

IV. DATA GATHERING AND COLLECTION

In this paper, the data used in the testing process are gathered and collected from most used syntax codes form there programming languages (C++, Java and python). These syntax codes distinguish these programming languages from English language.

The following syntax codes are example of the codes in different programming language.

Then the data inputs are converted into a 0-1 format based on a keyboard of letters and signs as shown in Table 1. This forms a vector of inputs for testing the proposed model.

Table 1 shows how the how the letters from (java, C++ and python) programming language are converted in to 0-1 format. Table 1 illustrates the conversion and reformation process for alphabetical form into binary.

In which, the letters in each syntax or word represented by 0 or 1. As 1 represents the letter that is exist in the syntax code while 0 represent the not-using case. In addition, the repeated letters are denoted by repeated 1 for each. As a result, the syntax code of print(y) in python

language is represented as the following (00010000100000010101000010011000000).

V. IMPLEMENTATION:

As illustrated in the previous sections, the implementation task of this work is conducted via KNIME toolkit. This toolkit is an open-source node-based graphical interface tool for implementing various data mining algorithms.

This is conducted by setting a propagation neural network with 36 input neurons, 16 neurons hidden layers and 2 neurons as output layers. This is conducted in KNIME toolkit as a node of pre-designed algorithm for recognition learner and predictor as shown in Figure 4 below.

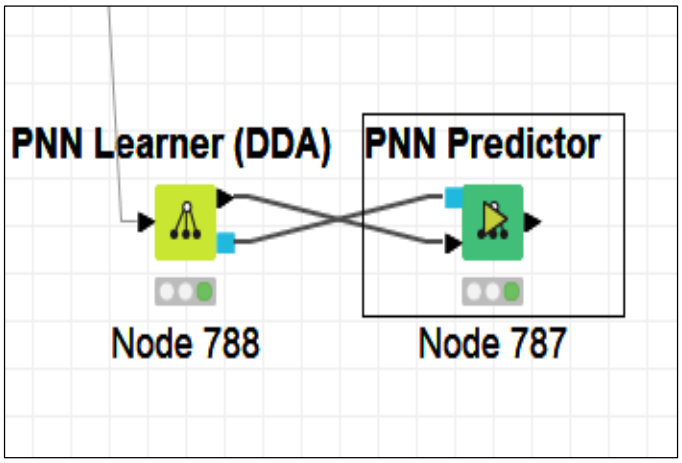

Figure 4: Recognition implementation in KNIME

A. Traninig the BPNN model:

In this paper, the BPNN model is trained with 1000 inputs data as training data and 200 for testing data.

As described in the previous section, these inputs are reformatted in a binary form which are imported into the KNIME as a .XLSX format as shown in Figure 5.

Figure 6 shows two groups of training data, first with syntax codes of programming languages and another with a natural language.

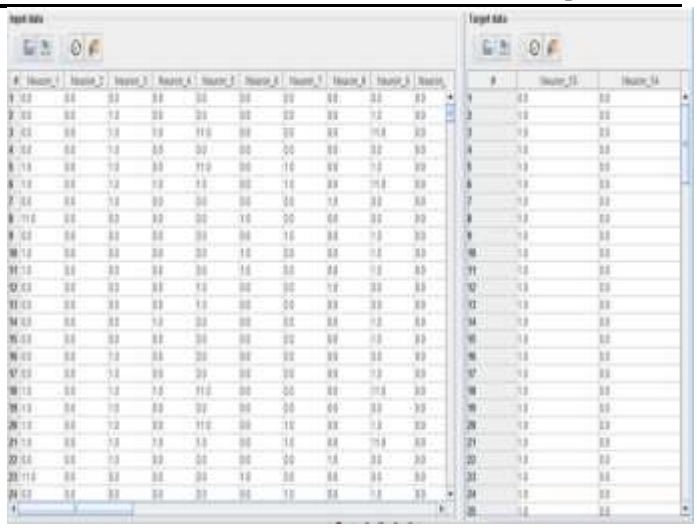

Figure 5: Training and Targeted data (sample)

Figure 6 shows training details of the prediction process in the system.

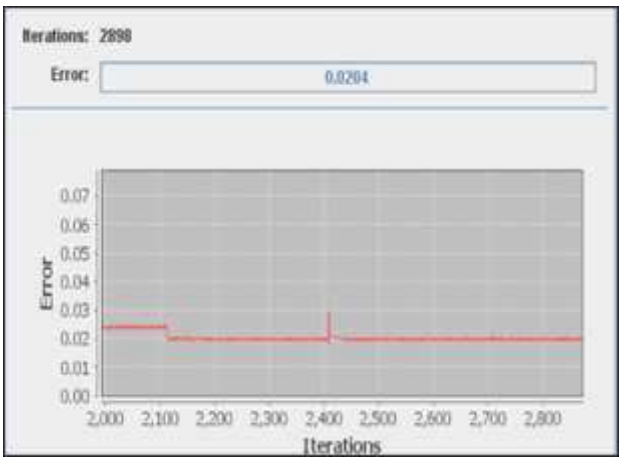

Figure 6: Training detail of the PPN model

B. Tests and Results:

The evaluation process is conducted in this paper to calibrate the ability of the BPNN model to distinguish the pattern of programming languages such as (Java, C++ and python). This evaluation is performed through three main tests.

Test 1:

In the first test, the BPNN model is evaluated with 200 syntax codes from (Python, C++ and Java) languages in its binary format. A sample results of the evaluation is shown in Table 2. This table consists of three main parts that labelled as the Syntax code, Binary format

and Recognition accuracy. In which Syntax code represents that input data used for testing and followed by a column for its Binary format. While the Recognition accuracy column shows the percentage accuracy of recognizing the pattern of each data input whether it is Programming Languages (**PL**) or non-PL syntax codes.

The results in Figure7 shows that the BPNN able to recognize 8 input codes with 100% of accuracy and 98% to recognize if(). While for input(y) the accuracy was 77%.

Syntax codes	(Binary format)	Recognition Accuracy	
		PL	Non - PL
#include<iostream>	11000100011000101001110000000011110000	100%	0%
cin>> a;	10100000100001000000000001001000000	100%	0%
int x;	00000000100001000001000100100001000	100%	0%
cout<<e;	00101000000000100001100000100011101000	100%	0%
return 0;	00001000000010000110110000010000001000	100%	0%
for (; ;)	00000100000000100100000001111000000000	100%	0%
if ( )	00000100100000000000000000110000000	98%	2%
using namespace std;	11011110101000111010011110000010000001000	100%	0%
char ;	10100010000000001000000001000001000	100%	0%
iput (y)	00000000100100101000100000001100000000	77%	23%

Figure 7: Sample of the first test result

The overall results of all 200 inputs used in this test show that the accuracy of recognizing this type of data recorded around 97%.

Test 2.

In this test, the BPNN model is evaluated with 200 words and sentences from non-programming language (English language). These input data do not have patterns for programming language. Table 2 shows a sample of the results in this test.

TABLE 2: SAMPLE OF THE SECOND TEST RESULTS.

Words and Sentence	Recognition accuracy	
	PL	Non-PL
What's your name?	0%	100%
All right	0%	100%
Golden	0%	100%
Deep	1%	99%
Portion	0%	100%
Great	0%	100%
Light	2%	98%
Problem	0%	100%
Leave	0%	100%
Saturday	0%	100%

The performance of the BPNN method shows a high accuracy of recognition as shown in the sample results in Table 2. In which, it can be noticed the system clearly recognized the non-programming language expressions with around 98% to 100% of accuracy.

Test 3

In the third test, we use set of English language phrases and words in evaluating the performance of BPNN model through using binary format. These words considered confusing to the BPNN model as it contains expressions that are similar to the programming language syntax codes. For example, the expressions of out and using are using widely both in normal and in programming languages which makes it difficult for recognition test. Table 3 shows a sample of 10 of these words and expression.

TABLE 3: SAMPLE OF THE THIRD TEST RESULTS.

Words and Paragraph	Recognition accuracy	
	PL	Non- PL
door;	0%	100%
Integer	24%	76%
classroom	0%	100%
out	0%	100%
out ;	50%	50%
main	0%	100%
Form	0%	100%
avoid	50%	50%
using	27%	73%
space	0%	100%

Table 3 shows that the BPNN mode recognized the expressions of (out, main, space, door) as non-PL expression. While the system shows confusion and less uncertainty in distinguishing the expressions of (avoid and out;) by giving 50% accuracy of both. In other hand, it gives more certainty in recognizing both (using and integer) expressions with the accuracy of around 75%.

As mentioned previously, this is conducted for 100 inputs, in which we calculated the overall performance of BPNN in recognizing 100 confusing inputs. Accordingly, the BPNN show overall performance of 85% of accuracy.

VI. ANALYSIS AND DISCUSSION

Based on the results presented previously in section 3, the pattern recognition model via binary format shows high accuracy in distinguishing (C++, Python and Java) programming language. This is shown in both Test1 and Test2 when there are low confusion factors in the data in which the BPNN model shows a performance between 99% to 100% of accuracy. These confusion factors are those words and expressions which are used similarly in both programming language and natural languages. While in the case of using data with more of these noisy factors, the performance of the system still shows a high accuracy between 75-100% in categorizing the patterns of programming language and non-programming language syntax.

It's worth to mention that this range and certainty affected also by quantity and the quality of the data used in the recognition process.

These shows that the Binary format in recognition process highly assist the recognition process by filtering the characteristics of each data input. This can be exploited more in extracting the patterns of sentences and a whole programming languages and syntax codes. In addition, this implementation of binary converting can also be used in the pattern recognition applications of natural languages.

VII. CONCLUSION AND FUTURE WORK

In the current area of information technology, the use of software applications for different purpose in the daily life activities are dramatically increased. This is conducted by a various types of programming languages to develop these applications. Consequently, the ability of distinguishing all these programming languages for the new learner and fresh developers become more challenging and difficult. Therefore, the necessity of having an automated method to recognise each syntax code is very important to assist in learning and implementing programming languages. Accordingly, M Robson [2] proposed a neural network to distinguish and classify several programming languages. He conducts this by extracting the sentence patterns of the syntax code of each programming language. He also recommended to conduct the recognition process through targeting letters patterns instead of the whole sentence. Therefore, we conduct this work by targeting the syntax cod pattern of each letter through converting it into a binary format. This is performed by using backpropagation algorithm of neural network. The implementation of the binary formatting in the BPNN model shows a high accuracy of 95% to 99% in recognizing programming and non-programming language patterns.

For the future work, we intend to implement this model for entire line of codes for different programming languages. We also recommend the Binary formatting furtherly in more recognition tasks. This work will of interest for the community of research in the ANN area of study and implementations.

REFERENCES:

[1] A. B. philip Mayer, "An Empirical Analysis of the Utilization of Multiple," in 19th International Conference on Evaluation and Assessment in Software , April 2015.

[2] R. Montenegro, "Source Code Classification Using Deep Learning," 30 8 2016. [Online]. Available: http://blog.aylien.com/source-code-classification-using-deep-learning/. [Accessed 7 8 2019].

[3] M. S. J. D. G. S. Jyotiska Nath Khasnabish, "Detecting Programming Language from Source Code Using Bayesian Learning Techniques," Springer International Publishing Switzerland 2014, p. 513–522, 2014.

[4] C. Sargur N. Srihari, "pattern recognition," London, Chapman , 1993, pp. 1034-1041.

[5] E. SALIBA, "An overview of Pattern Recognition," University of Burgundy, Antonine University, pp. 1-7, 2014.

[6] D. B. ,. T.-h. K. Jayanta Kumar Basu, "Use of Artificial Neural Network in Pattern Recognition," International Journal of Software Engineering and I International Journal of Software Engineering and Its Applications, vol. 4, no. 2, pp. 23-34, 2010.

[7] M. K. Priyanka Sharma, "Classification in Pattern Recognition: A Review," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 4, pp. 298-306, 2013.

[8] S. W. Smith, "Chapter 26: Neural Networks (and more!)," in The Scientist and Engineer's Guide to digital signal processing, california, California Technical Pub; 1st edition (1997), 1997, p. 626.

[9] K. Gurney, An introduction to neural networks, london , new york: UCL Press, 1997.

[10] J. B. ,. A. K. Andrej Krenker, "Introduction to the Artificial Neural Networks," in Artifital neural network, In Tech, 2011, p. 362.

[11] S. Haykin, Neural Networks and Learning Machines Third Edition, Hamilton, Ontario, Canada: pearson, 2008.

[12] Z. Ghahramani, "Unsupervised Learning," Springer-Verlag Berlin Heidelberg, pp. 72-112, 2004.

[13] K. Shihab, "A Backpropagation Neural Network for Computer Network Security," Journal of Computer Science , vol. 2, no. 9, pp. 710-715, 2006.

[14] J.-h. C. J.-y. S. F. H. Jing Li, "Brief Introduction of Back Propagation (BP) Neural," springer, vol. 2, pp. 553-558, 2012.

[15] K. T. RashmiAmardeep, "Training Feed forward Neural Network With Backpropogation Algorithm," International Journal Of Engineering And Computer Science, vol. 6, no. 1, pp. 19860-19866, 2017.

[16] K. O. ,. S. A. M. N. Mutasem Alsmadi, "Back Propagation Algorithm : The Best Algorithm Among the Multi-layer Perceptron Algorithm," International Journal of Computer Science and Network Security, vol. 9, no. 4, pp. 378-383, 2009.

[17] Y. Z. Alaeldin Suliman, "A Review on Back-Propagation Neural Networks in the Application of Remote Sensing Image Classification," Journal of Earth Science and Engineering, vol. 5, pp. 52-65, 2015.