International Journal of Computer Engineering in Research Trends
Multidisciplinary, Open Access, Peer-Reviewed and fully refereed
Research Paper Volume-6, Issue-10 ,2019 Regular Edition E-ISSN: 2349-7084

A Pattern recognition model of C++ programming language using artificial neural network via Simbrain toolkit

Shallaw M. Ali [1*]

[1*] *Assistant Lecturer, Department of Computer Technology, Alqalam University College, Kirkuk, Iraq*

**Corresponding Author: shalaw.eng@alqalam.edu.iq,*

Available online at: http://www.ijcert.org

Received: 26/Oct/2019 Revised:16/Nov/2019 Accepted:25/Nov/2019 Published: 10/Dec/2019

Abstract:-

Background/Objectives: In the field of software development, the diversity of programming languages increase dramatically with the increase in their complexity. This leads both programmers and researchers to develop and investigate automated tools to distinguish these programming languages. Different efforts were conducted to achieve this task using keywords of source codes of these programming languages. Therefore, instead of using keywords classification for recognition, this work is conducted to investigate the ability to detect the pattern of a programming language characteristic by using simbrain toolkit of neural network and testing the ability of this toolkit to provide detailed analysable results.

Methods/Statistical analysis: the method of achieving these objectives is by using backpropagation neural network via Simbrain toolkit based on pattern recognition methodology.

Findings: The results show that Simbrain neural network of pattern recognition can identify and recognize the pattern of C++ programming language with high accuracy. It also shows the ability of Simbrain toolkit to represent the analysable results through percentage of certainty.

Improvements/Applications: it can be noticed from the results the ability of Simbrain toolkit to provide a useful platform for studying and analysing the complexity of backpropagation neural network model.

Keywords: Simbrain, Pattern recognition, Artificial neural network, Backpropagation neural network.

1. Introduction

In the last decade, wide range of programming languages for a variety of tasks has been created in software development field [1]. This diversity makes it difficult for new students and developers to recognize the exact programming language that been used in complex systems. Especially in the systems that require using a combination of programming languages such as C++, Java or Ruby [1]. Therefore, it would be very useful to develop a tool for identifying programming languages codes based on its pattern. One of the attempts to achieve this task is conducted by M. Robson [2] and [3] through training a neural network model to classify programming codes based on its language. According to M. Robson [2] , this classifier identify programming languages based on syntax codes in the form of words. Robson suggests using the characters patterns of the programming language instead of these keywords. Therefore, instead of using classification of keywords in the codes for different programming languages , this work aims to investigate the ability of backpropagation neural network (BNN) to identify and recognize the programming language

© 2019, IJCERT All Rights Reserved Doi: 10.22362/ijcert/2019/v6/i10/v6i1002

(C++ in this work) based on the pattern of each particular code characteristics.
This paper also aims to investigate the ability of Simbrain , a neural network simulation, to represent an analysable results.

2. Background

2.1 Pattern Recognition

Pattern recognition can be defined as a methodology of designing systems that can identify or classify patterns in complex environment [4]. It also "can be seen as a classification process" [5]. It aims to study and monitor environment for a potential pattern and make a proper decision about it [6].
According to Sharma and Kaur [7] the basic algorithm of pattern recognition can be illustrated in Figure 1.

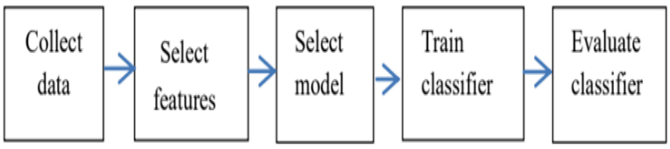


Figure 1. Algorithm of pattern recognition [7].

As shown in Figure 1, the first part of this algorithm is collecting data then selecting the features from the data to be recognized. After preparing input data, suitable recognition model is selected. This model is trained to recognize the potential pattern. Finally, the system is evaluated to check it behavior.

In addition, Based on sharma and kaur classification, the main model of pattern recognition are statistical model, syntactic model, template matching model, and Artificial neural network which is characterized for " the ability to learn complex nonlinear input- output" [7].

2.2 Artificial Neural Network

Essentially, the idea of artificial neural network (ANN) is based on the concept of how the information processes inside humans and animals brains. This concept can be oversimplified as a complex network of trillions of nerve cells interconnected with each other via pulses called action potentials [8] .

According to Steven W. S. [8] , ANN aims to mimic this process as much as possible. Which means mimicking the most important ability in human mind, which is the ability of learning. This is differs from the linear algorithm of regular machine methodology to solve problems.

In other words, ANN can be simply defined as a computer algorithm that consist of simple entities interconnect with each other to form an interaction of behaviour in response to different states of input [9]. It is structured in the form of layers each of which is consisting of a number of nodes that interconnected with each other through mathematical functions.
According to Krenker A. et al [10] the basic element of any ANN is a neuron, which is designed based on the neuron in a biological neural network. This neuron is structured as shown in Figure 2.

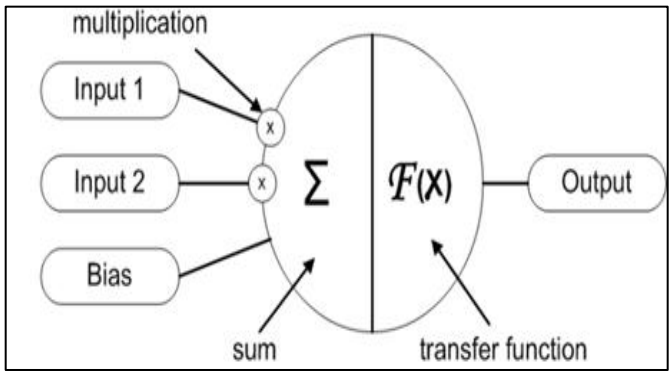


Figure 2 The structure of the artificial neuron [10]

Each neuron operates by receiving inputs, which are either the system inputs or the output from other neurons that are connected with this neuron. These inputs are weighted individually and the neuron sums it with each other and with the bias, and then the result of this summation is processed through transfer function.
In addition, According to Haykin S. [11] , the learning concept of ANN is categorized as the following:

Supervised learning: In this learning methodology of neural network , the system is trained by providing it with desired behavior (output data) for a set of specific inputs.

Unsupervised learning: This type of learning algorithm does not require providing target output. it may seems difficult to illustrate , however it can be simplified as an algorithm that aims to find a pattern in given input data , which can be used for decision making , prediction and so on [12].

Reinforcement learning: This type of learning algorithm aims on interacting with its environment "to learn to act in a way that maximizes the future rewards it receives (or minimizes the punishments) over its lifetime" [12].

2.3 Backpropagation Neural Network

Backpropagation neural network (BNN) is a one of the most popular supervised ANN, which is, as the name implies, uses the backpropagation algorithm concept for learning. It is developed in 1970 to solve the limitation of neural network (NN) algorithm, which failed to address XOR issue [13]. According to shihab K. the BNN is basically

© 2019, IJCERT All Rights Reserved

consist of a small pieces that interconnect together to solve complex issues [13]. It is a feedforward with a structure of Multi-layer, which is learning based on error back feeding [14].
According to [15] , BNN is considered as a type of learning and training algorithm rather than being a type of neural network. In order to train this network, it is required to provide the BNN with output data for specific input. After training the network, it will be ready to recognize any input pattern based on the pattern of training data [15]. Typically, the structure of a standard backpropagation network consist of three layers: input, hidden layer and output layer [16]. This structure is shown in Figure 3.

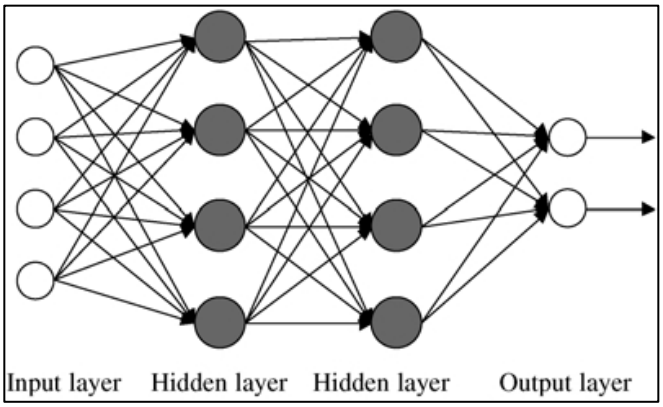


Figure 1 The structure of Backpropagation network [16]

he learning algorithm of backpropagation is essentially based on the theory of error –correction-learning concept "which uses the error function in order to modify the connection weights to gradually reduce the error [17].

In another word, the network modifies the connections between its layers' neurons by changing the weights that related to each of these neuron functions as illustrated in the previous section. The weight changing aims to minimize the differences between current output of the network and desired target output.

Many applications can be used to build and design a neural network with backpropagation methodology such as Simbrain and Matlab.

2.4 Simbrain Toolkit

Simbrain is an open source free toolkit developed to design, operate and simulate neural network methodologies in the aim of being a visual user-friendly software. It based on Java programming language and runs on Windows, Mac and Linux. According to , Simbrain is known for its ease of use . This ease of use is mainly due to its interface, which is based on graphical interaction [18]. Furthermore, simbrain also provides a set of easily learnable shortcuts [18]. It provides the ability to build analyses and study the behavior of different neural networks visually by user with any backgrounds. Simbrain can be downloaded freely from the following website address: https://www.simbrain.net/

3. Methodology

The aim of this work is to test and investigate the ability of BNN via simbrain toolkit to distinguish and recognize the pattern of C++ programming language. This is conducted by applying the algorithm of pattern recognition, which is presented by Sharma and Kaur [7]; see Figure 1. Firstly, the data are collected from the most common syntax codes of C++ programming language. Then these codes are converted into a form of binary array, from which the features of potential pattern are selected. This is follow by selecting BNN as the model used for pattern recognition. This model is then trained using the collected data. Finally, the trained BNN model is evaluated by applying Three tests. In the first test, the evaluation is conducted by applying the proposed BNN with ten arbitrary C++ syntax codes to check how it behaves. While in the second test, the system is checked using ten words and sentences to investigate its ability to recognize non-C++ pattern. Eventually, the third test check the confusion behavior of the system when it is applied with ten confusion data. In addition , another objective of this work is to check the ability of simbrain toolkit to represent analysable , measurable results . This is conducted by observing the results of each of above results numerically.

4. Collect data

The input data for this work are collected from the most used C++ programming syntax codes that distinguish C++ programming language from other languages. Instance for these codes can be as the following:

cin >> ;

cout<< ;

return 0 ;

main ()

These data then converted into a form of zer-one to represent the vectors inputs for the system. Table 1 shows how these data are formed.

© 2019, IJCERT All Rights Reserved

Table 1. Sample of input data representation

Signs\ fom	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	;	(	)	>	<	#	-	{	}	/ /
cin >>a;	1	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	1	0	0	11	0	0	0	0	0	0
#include <iostream>	11	0	0	0	1	0	0	0	11	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
Door	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Strq	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

These data then converted into a form of zer-one to represent the vectors inputs for the system. Table 1 shows how these data are formed.

In Table 1, English letters and some of C++ signs are represented in each column while the appearance and the pattern of representing the codes is performed in the rows in a form of one for the activated letter and zero for non-active letter , while the repeated letters represented as (11, 111 and so on) .

For instance the code: *cin >> a;* can be represented as *10100000100001000000001001100000*.

It can be noticed that the letters are the low case letter this is due to that in C++ there are no high case letter codes.

5.Designing the Backpropagation Neural Network

As presented previously, the proposed neural network is designed and implemented using simbrain toolkit. This is conducted by creating a neural network with 36 input neurons in in the first layer, 16 neurons in the hidden layer, and 2 neurons in the output layer.

The 36 input neurons represent the signs for C++ programming code, while the 2 neurons in output layers represent the indication part of the model that detect whether the inputs pattern is C++ programming language or not . This is shown in Figure 4 and 5.

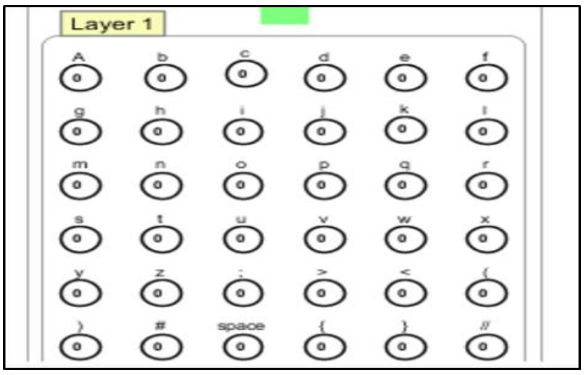


Figure 4. The first layer of the network

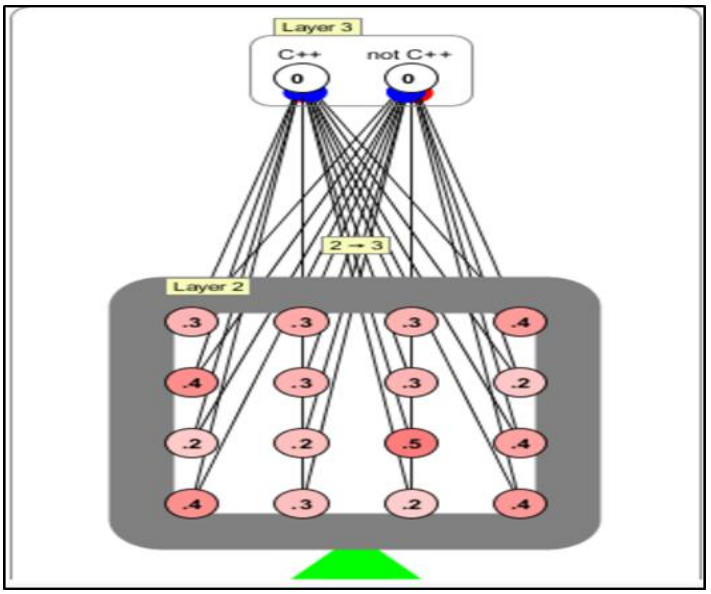


Figure 5. The second and the third layer of the neural network

It can be noticed that the Simbrain platform provides the ability to illustrate the interconnection between these layers.

6. Training the Model

The proposed simbrain backpropagation neural network (SBNN) is trained in this work by applying the network with 1500 training inputs and 1500 target outputs. As presented previously, these inputs are structured in a form of zero-one, which are imported into the system in CSV file. A sample of these inputs is shown in Figure 6.

The training inputs can be classified into two groups. First group is a data set of C++ codes, while the second group is a data set of non-C++ words pattern.

© 2019, IJCERT All Rights Reserved

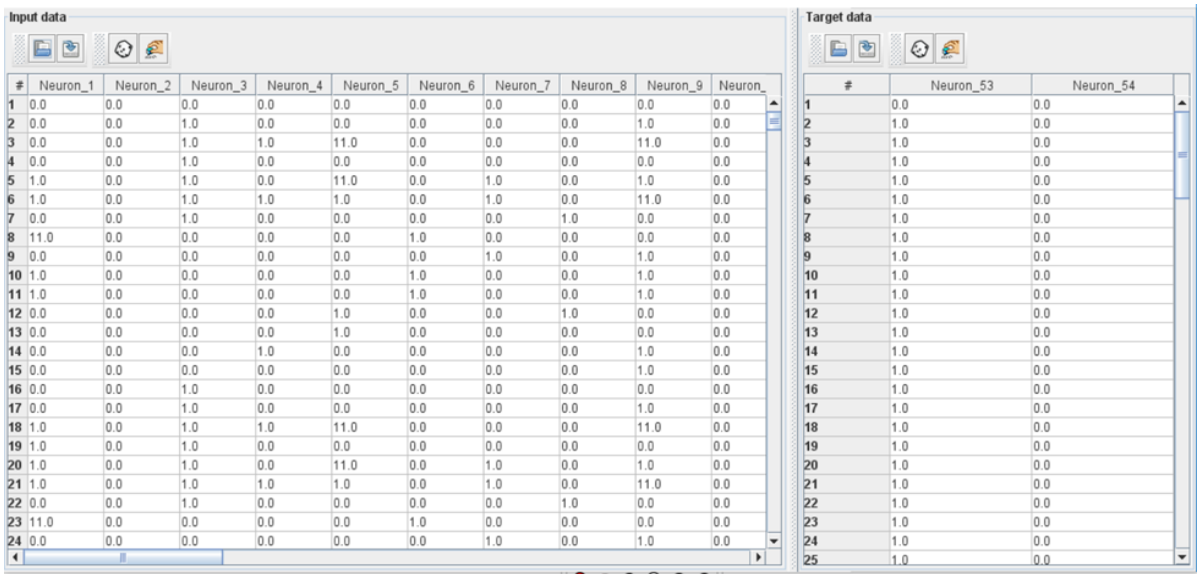


Figure 6 Sample of input and target data

The system is trained for 2898 iterations with a stable error rate of 0.0204 as shown in Figure 7.

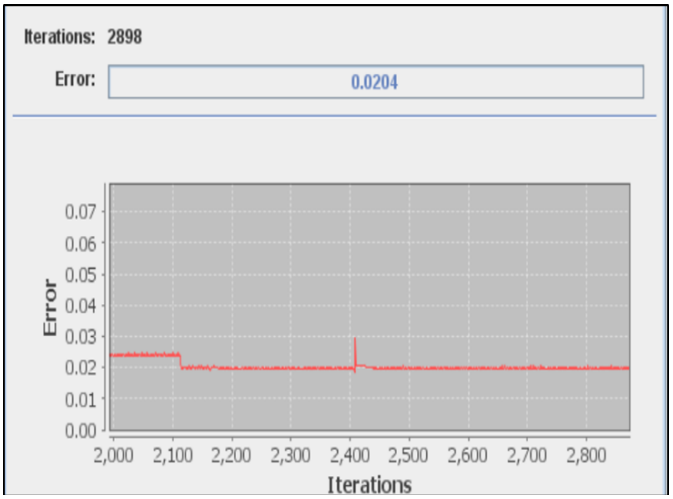


Figure 7. Error rate of the model

7. Model evaluation

The SBNN model is tested to calibrate its ability to recognize C++ programming pattern. This is conducted through three tests.

7.1 Test 1

The first test is performed by checking the SBNN model behavior after applying 100 syntax codes with a pattern of C++ programming language.

These codes are inserted into the model by exploiting the graphical interface ability of simbrain as shown in Figure 8 .

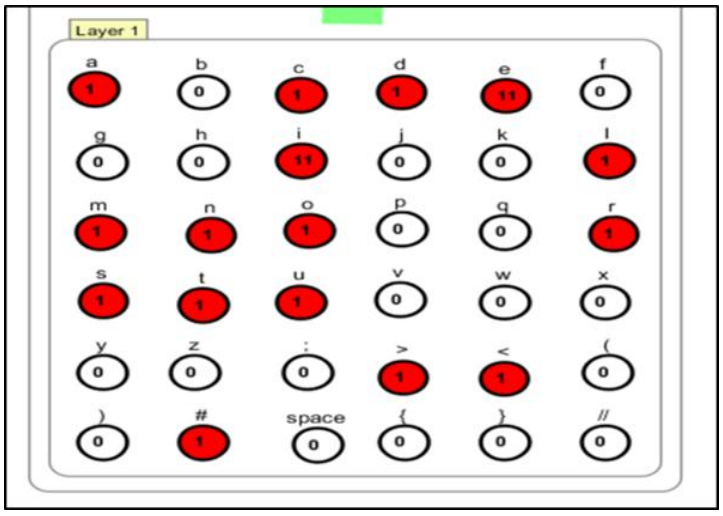


Figure 8. Sample of the input interface

© 2019, IJCERT All Rights Reserved

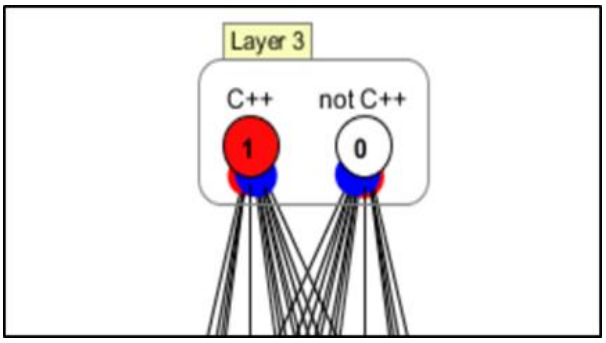


Figure 9. The sample of output interface.

The result of the first test can be seen in Table 2, which consist of three columns. The first column represents the syntax codes that are used to test the network. The second column represents the binary form of the inputs. While the last columns shows the behavior of the system in percentage of accuracy, which measures the degree of recognition. Table 2 contain as sample of 10 of these inputs.

According to Table 2, system recognition for 10 input codes were 100% for 8 inputs, while the less certainty of pattern recognition was 60% for the code // . In addition the system has the ability to recognize the code *return 0* ; by 92%.

7.2. Test 2

In the second test, the network behavior is checked with 100 arbitrary words and sentences that do not have the pattern of C++ codes as shown in Table 3 , which represent a sample of these input data.

Table 2. The result of the first test.

Syntax codes	**Input representation (Binary form)**	**MODEL behavior recognition accuracy**	
		C++	not C++
#include<iostream>	11000100011000101001110000000011110000	100% C++	0%
cin>> a;	10100000100001000000000000010011000000	100% C++	0%
int x;	00000000100001000001000101001000001000	100% C++	0%
cout<<e;	00101000000000100001100000100011010000	100% C++	0%
return 0;	00001000000010000110110000010000001000	92% C++	0%
for (; ;)	00000100000000100100000000011110000000	100% C++	0%
if ( )	00000100100000000000000000000110000000	100% C++	0%
using namespace std;	11011110101000111010011110000010000001000	100% C++	0%
char ;	10100010000000000100000000010000001000	100% C++	0%
//	00000000000000000000000000000000000001	60% C++	0%

© 2019, IJCERT All Rights Reserved

Table 3. The result of the second test.

Words and Paragraph	**Input presentation (Binary form)**	**MODEL behavior recognition accuracy**	
		C++	not C++
How are you	10001001000000110010010101000000001100	0%	100%
All on	10000000001101100000000000000001000	0%	100%
Gold	00001000010100100000000000000000000	0%	100%
Door	00010000000000110010000000000000000	0%	100%
Place	10101000000100010000000000000000000	0%	100%
good morning	00010011010001111110010000000000001000	0%	100%
London	00010000000101111000000000000000000	0%	100%
Problem	01001000000110110100000000000000000	0%	100%
Live	00001000100100000000100000000000000	0%	100%
Star	10000000000000000111000000000000000	0%	100%

7.3 Test 3

In the third test, the model behaviour is checked by using set of input data (words) with patterns that are considered confusing for the model. These inputs are ten words and expressions that are not C++ codes but they contain C++ codes pattern. For example, the word (*classroom*) is not a C++ word but it has the same pattern of C++ syntax code (*class*) in its first part. Table 4 shows as sample of the input of 10 words that are used in this test.

According to Table 4, the highest correct recognition accuracy was 100% for the words (*out, main, form, space, door*) followed by 70 % recognition accuracy for the word *using*. While the model could not clearly recognize the words *out;*, *integer;* and *avoid* which are given from 90-100% wrong recognition accuracy.

Table 4.The result of the third test

Words and Paragraph	**Input presentation (Binary form)**	**model behavior recognition accuracy**	
		C++	not C++
door;	10001001000000110010010101000000001100	2%	100%
Integer;	10000000001101100000000000000001000	90%	10%
classroom	00001000010100100000000000000000000	0%	100%
out	00010000000000110010000000000000000	0%	100%
out ;	10101000000100010000000000000000000	100%	0%
main	00010011010001111110010000000000001000	0%	100%
Form	00010000000101111000000000000000000	0%	100%
avoid	01001000000110110100000000000000000	100%	0%
using	00001000100100000000100000000000000	10%	70%
space	10000000000000000111000000000000000	0%	100%

© 2019, IJCERT All Rights Reserved

8. Analysis and Discussion

According to the results presented in previous section, the behavior of SBNN model shows high accuracy of pattern recognition especially for both Test1 and Test2 when the inputs pattern were clear for each test inputs and there were no confusion factors. Additionally, when the model tested with a set of confusing data, it recognizes six out of ten inputs with high accuracy in the range between 70-100%. This ability to measure the accuracy and certainty of the system can be noticed from the results, for instance when the pattern is more likely C++, the network gives more probability for C++ than not C++ in the output layer. The network shows this probability in the form of a range of number between 0 and 1. When 0 represent 0% while 1 represent 100%. This range and certainty depends on the quantity and the quality of the data used in the system training.

Furthermore, the results also show the high numerical and graphical ability of simbrain toolkit for analysis and study this can be seen in the behaviour of the second layer, see Figure 5. Which gives an opportunity to study the model in more details for future works.

9. Conclusion

With increasing the variety of software applications, the programming languages that are used to develop these applications increase too. Therefore, having an automated tool to distinguish these programming languages would be very useful for developers and students. One of the attempts to achieve this ability conducted by M. Robson , who developed a neural network model to classify different programming languages via the patterns of their source codes by using a sentence of these codes. M. Robson suggests performing this task using a pattern of letters in these syntax codes. Therefore this work developed a pattern recognition model using backpropagation neural network via Simbrain toolkit to recognize C++ programming language codes. This model success to identify the patterns of different inputs with high accuracy. The results also shows a high ability of Simbrain to provide numerical and graphical results for both research study and analysis. For future work, it is recommended to increase training data and perform more tests on the model. In addition, it is suggested to test the proposed pattern neural network via matlab toolkit and using the matlab model to conduct a comparative evaluation study with the current simbrain model.

10. 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.

© 2019, IJCERT All Rights Reserved

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.

[18] j. y. Zachary Tosi, "Simbrain 3.0: A flexible, visually-oriented neural network simulator," *Elsevier journal neural network*, vol. 83, pp. 1-10, 2016.

[19] W.-L. L. ,. L. L. Tung-Hsu Hou, "Intelligent remote monitoring and diagnosis of manufacturing processes using an integrated approach of neural networks and rough sets," *Journal of Intelligent Manufacturing* , vol. 14, no. 2, pp. 239-253, 2003.



Shallaw Mohammed Ali is an assistan Lecturer and academic staff member in the Departement of Copmuter Engineering Technology in college of Alqalam university . He completed his MSc degree in software Engineering form the univeristy of Hertfordshire , United Kingdom. His research interest is Agent based simulation model , Artificial neural network and Image processing.

Email: shalaw.eng@alqalam.edu.iq

© 2019, IJCERT All Rights Reserved