ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

Automatic Pseudocode to Source Code Translation Using Neural Network Technique

Assistant Professor Dr. safwan Omer Hasson, MSc. Student Fatima Mohammed Rafie Younis
University of Mosul / College of Computer Sciences and Mathematical / Software engineering Department

Abstract*— *The development is taking place in the software engineering, especially in the field of development there is a need to use artificial intelligence techniques for building software automatically converts pseudo code to a source code by write it in a particular language based on neural network techniques. In this research, three techniques are used of neural networks to select the best neural network for converting pseudo code to source code written in Matlab version (R2013a). Depended on training and test time the best neural network type for convert pseudo code to source code is a cascade back propagation.

***Index Terms*—pseudo code, Back-propagation neural network, Cascade-forward back propagation neural network, Radial basis function networks, and source code.**

I. INTRODUCTION

Design phase is one phase's software development life cycle refers to the set of activities that includes specifying an algorithm for each program component the design activity that follows is limited to specifying an algorithm for this program [1]. An algorithm is a step-by-step procedure for solving a problem or accomplishing some end, especially by a computer. A good algorithm must [1]:

1. list the activities that need to be carried out
2. List those activities in the proper order.

There are two methods of algorithm description: pseudo code and flowcharts [2]. Pseudo code derived from 'pseudo' which means imitation and 'code' means instruction [3]. Pseudo code is a compact and informal high-level description of a computer programming algorithm that uses the structural conventions of a programming language, but is intended for human rather than machine reading [4]. Program Design Language (PDL), also called structured English or pseudo code [5]. The after design phase in software development life cycle is implementation phase is to develop a program that runs correctly on a computer the design phase provided a solution in the form of a pseudo code algorithm, the implementation phase Translate an algorithm into a programming language[1].

II. ADVANTAGE AND LIMITATIONS OF PSEUDOCODE

There are several Advantage and Limitations of pseudo code can be below [3].

A. Advantage

1. Its language independent nature helps the programmer to express the design in plain natural language.
2. Based on the logic of a problem it can be designed without concerning the syntax or any rule.
3. It can be easily translated into any programming language.
4. It is compact in nature and can be easily modify.

B. Limitations [3]:

1. It is unable to provide the visual presentation of the program logic.
2. It has not any standard format or syntax of writing.
3. It cannot be compiled or executed.

III. PREVIOUS STUDIES

Suvam mukherjee, Tamal Chakrabarti, 2011, suggested tried to present a translation process where the user presents a pseudo code as an input, and the output is an implementation of the pseudo code in a specific programming language. The pseudo code has to be written in XML use ALGOSmart 60 to translate this XML pseudo code into C and Java programs [6].

Anthi Karatrantou, Chris Panagiotakopoulos, 2008, suggested presents a pilot study which investigated the way prospective primary school teachers handled the process of converting an algorithm - pseudo code to a program while working with the programming environment of the Robolab programming tool of Lego Mind storms. Conclude the Lego Mindstorms environment helped and motivated them to compose the algorithm expressing it with a pseudo code in every step, and to convert it into a program in a simple and easy way [7].

Stuart Garner, 2007, suggested concerns the development of a simple tool that helps students create pseudo code and convert to the Visual BASIC code. This tool has been used and evaluated in an introductory programming unit of study. The results suggest that the tool was easy for students to use and that it helped support their learning [8].

Anne L. Olsen, 2005, suggested describe an approach that emphasizes the use of pseudo code in the introductory Computer Science, this approach is to teach students how to first develop a pseudo code representation of a solution to a problem and then create the code from that pseudo code. The

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

describe this use of pseudo code in CSCI 207 as a tool to teach students problem solving skills [9].

IV. NEURAL NETWORKS

This is mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Such a model has three simple sets of rules: multiplication, summation and activation [10]. Fig. (1) Shows the function of a single neuron [11].

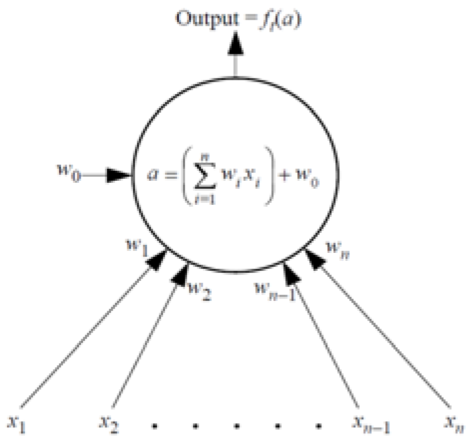

Fig. (1) A single neuron [11]

there are several neural network can be used to convert pseudo code to program in matlab languages, which is explained in sections (A), (B), and (C).

A. Back-propagation neural network

Back propagation use supervised training algorithm for multi layer network, the input and target output has been prepared for the training process. A data error in output layer is counted using network output and target output. The data errors then back propagated to the hidden layer, resulting in weight change for the synapses heading to the hidden layer back propagation network consists of two phases, feed forward phase and backward phase [12], the Fig. (2) Shows simple three layer back propagation neural network [13].

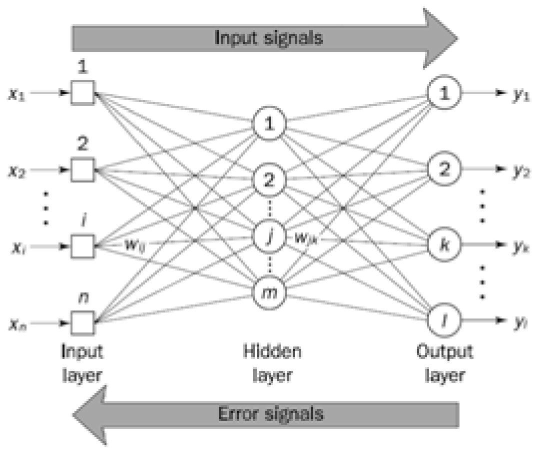

Fig. (2) Back propagation network structure [13].

The back-propagation training algorithm are shown below [13]:

Step 1: Initialization

Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range.

Step 2: Activation

Activate the back-propagation neural network by applying inputs $x1(p), x2(p), \dots, xn(p)$ and desired outputs $y_{d,1}(p), y_{d,2}(p), \dots, y_{d,n}(p)$.

(a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = \text{sigmoid}([\textstyle\sum_{i=1}^{n} x_i(p) * w_{ij} - \theta_j] \qquad (1)$$

where n is the number of inputs of neuron j in the hidden layer, and sigmoid is the sigmoid activation function.

(b) Calculate the actual outputs of the neurons in the output layer

$$y_k(p) = \text{sigmoid}[\textstyle\sum_{j=1}^{m} x_{jk}(p) * w_{jk}(p) - \theta_k]$$

(2)

where m is the number of inputs of neuron k in the output layer.

Step 3: Weight training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

(a)

Calculate the error gradient for the neurons in the output layer:

$$\delta_{k(p)} = y_k(p) * [1 - y_k(p)] * e_k(p) \qquad (3)$$

Where

$$e_k(p) = y_{d,k}(p) - y_k(p) \qquad (4)$$

Calculate the weight corrections:

$$\Delta w_{jk}(p) = \beta \times \Delta w_{jk}(p-1) + \alpha * y_j(p) * \delta_k(p) \qquad (5)$$

Update the weights at the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \qquad (6)$$

(b)

Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = y_j(p) * [1 - y_j(p)] * \textstyle\sum_{k=1}^{l} \delta_k(p) * w_{jk}(p) \qquad (7)$$

Calculate the weight corrections:

$$\Delta w_{ij}(p) = \beta \times w_{ij}(p-1) + \alpha * x_i(p) * \delta_j(p) \qquad (8)$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \qquad (9)$$

where β is a positive number ($0 <= \beta < 1$) called the momentum constant.

Typically, the momentum constant is set to 0.95.

Step 4: Iteration

Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.

B. Cascade-forward back propagation neural network

Cascade-forward back propagation neural network is similar to BPNN but it includes a weight connection from the input layer to each layer and from each layer to the successive layers. The fig. (3) shows a three-layer network has connections from layer 1 to layer 2, layer 2 to layer 3, and

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

layer 1 to layer 3, In the cascading neural network, there are three types of nodes: input nodes, inner nodes and output nodes, The three-layer network also has connections from the input to all three layers. The additional connections might improve the speed at which the network learns the desired relationship [14].

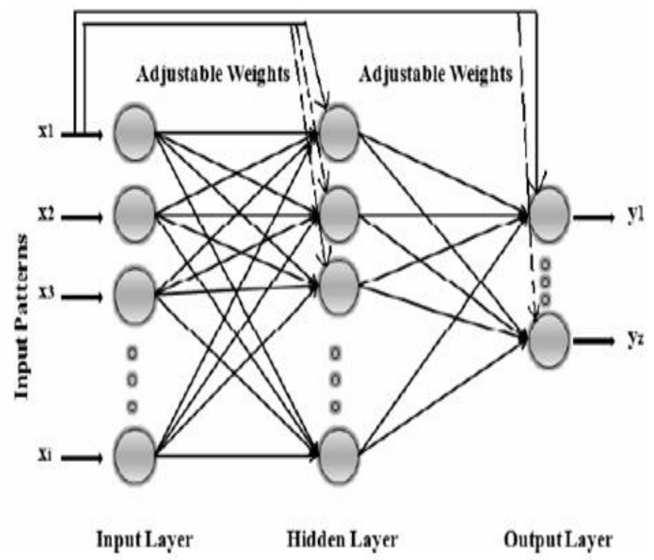

Fig. (3) Cascade-forward neural network [14].

C. Radial basis function networks

RBF network includes three basic layers, whose structure is shown in Fig. (4). each layer has a completely different role [15].

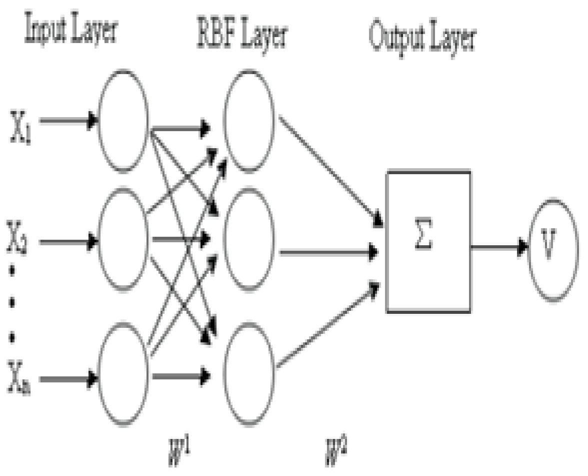

Fig. (4) Radial basis function neural network structure [15].

The input neurons do not perform any processing. The neurons in the output layer produce the weighted sum of their inputs, which is usually passed through a linear transfer function, the neurons in the hidden layer, sometimes called the *prototype* layer, behave differently.

For an input vector (*x*1, *x*2, …*xn*), a neuron *i* in the hidden layer produces an output, *yi*, given by as mentioned in [11]:

$$y_i = f_r(r_i) \tag{10}$$

$$r_i = \sqrt{\sum_{j=1}^{n} (x_j - w_{ij})^2} \tag{11}$$

where *wij* are the weights on the inputs to neuron *i*, and *fr* is a symmetrical function known as the *radial basis function* (RBF). The most commonly used RBF is a Gaussian function:

$$f_r(r_i) = exp\left[\frac{-r_i^2}{2\sigma_i^2}\right] \tag{12}$$

Where $\sigma\, i$ is the standard deviation of a distribution described by the function. Each neuron, *i*, in the hidden layer has its own separate value for σi [11].

During unsupervised learning, the network adjusts the weights — more correctly called centers in an RBF network — so that each point (wi1, wi2,…win) represents the center of a cluster of data points in pattern space [11].

Training of an RBFN consists of two phases [16]:

(1) Adjusting the RBF of the hidden neurons by applying a statistical clustering method; this represents an unsupervised learning phase.

(2) Applying gradient descent (e.g., the back propagation algorithm) or a linear regression algorithm for adjusting the second layer of connections; this is a supervised learning phase.

During training, the following parameters of the RBFN are adjusted [16]:

- The n-dimensional position of the centers ci of the RBFi (particular intermediate node i). This can be achieved by using the k-means clustering algorithm the algorithm finds k (number of hidden nodes) cluster centers which minimize the average distance between the training examples and the nearest centers.
- The deviation scaling parameter σi for every RBFi; it is defined by using average distance to the nearest m-cluster centers:

$$\sigma_i = \left(\left(\sum_{p=1,m} abs(c_i - c_{ip})\right)/m\right)^{\frac{1}{2}} \tag{12}$$

Where cip is the center of the pth cluster near to the cluster i.

- The weights of the second layer connections.

  The recall procedure finds through the functions RBFi how close an input vector x' is to the centers ci and then propagates these values to the output layer.

V.PROPOSED ALGORITHM

Conventions that must be followed when writing the pseudo code:

1. Write keywords of the pseudo code in lower case.
2. Each statement in pseudo code should express just one action for the computer.
3. Variable, constant and function names may be more than one word then they are joined with an underscore.
4. After and before the any word in pseudo code text has only one space or spatial character.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

5. Summations and counters must be initialized to zero, and other variables that require initial values must also be initialized.
6. When write the function in the another text not the test call function and store the function same the name function in pseudo code.

The steps of training neural network are explained below:

1. The definition of a matrix containing binary numbers, each number represents a keyword in pseudo code.
2. Initialize weights (random values), set learning rate to 0.001, and error rate to 0.001.
3. Broadcast each binary numbers of the matrix to input layer.
4. Applied training of the neural network to get the output throw the hidden layer using equations 1 and 2.
5. Compare the actual output of the neural network with target, show table 1, if match then stop; otherwise adjusting the weights using equations 3 to 9 and continue the learning of the neural network through steps 4 and 5 again.
6. Store the weights.

The training algorithm of Cascade neural network is same to back propagation neural network but the different the input confectioned to all successive layers. The algorithm of the Radial basis function networks are same steps but different equations 10 to 12 used between input layer to hidden layer. The steps translate the pseudo code to source code as show below:

1. Open text file for reading (pseudo code).
2. Read one line at each time and split the line into keywords.
3. Check each keyword if exist in the array then take the keyword binary number else the keyword write into output text (program) directly.
4. Load the store weights and apply neural network algorithm to get the output using equation 1 to 2.
5. Print the output of neural network into output file (program).
6. Repeat steps 2 to step 6 until read each line into text file.

The fig. (5) is shown the training of the back propagation neural network:

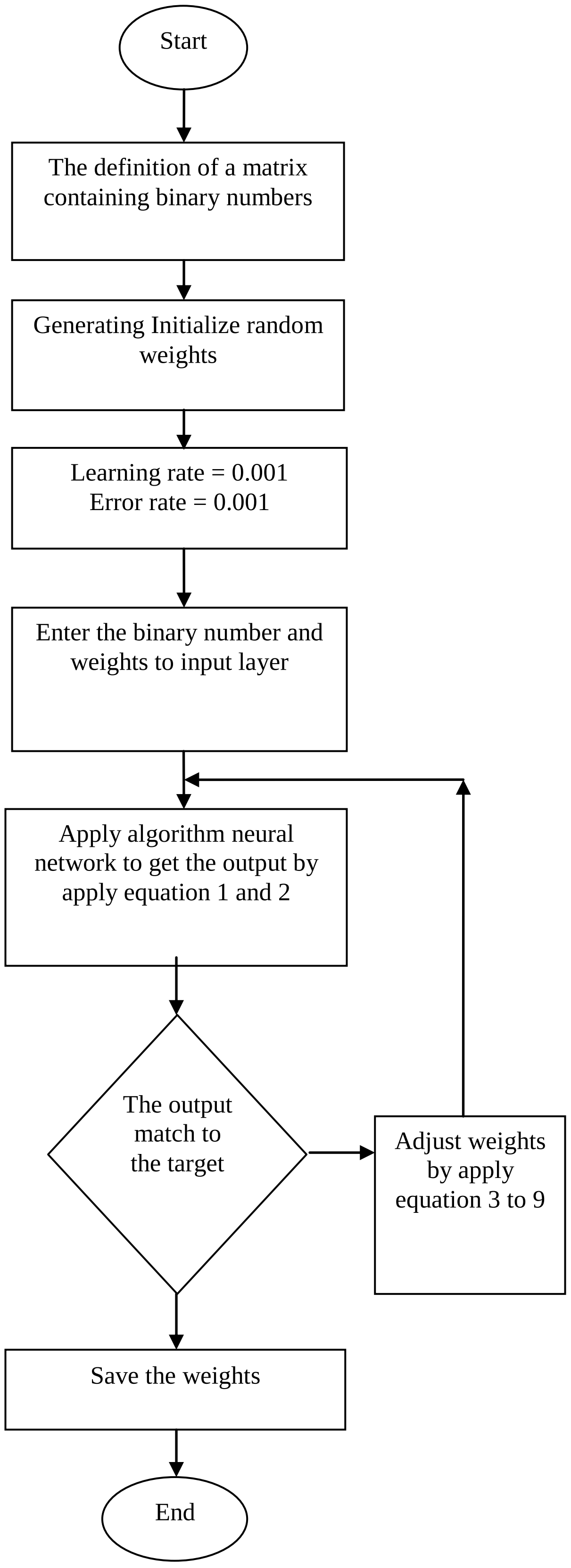
Start
The definition of a matrix containing binary numbers
Generating Initialize random weights
Learning rate = 0.001
Error rate = 0.001
Enter the binary number and weights to input layer
Apply algorithm neural network to get the output by apply equation 1 and 2
The output match to the target
Adjust weights by apply equation 3 to 9
Save the weights
End

Fig. (5) Flowchart the back propagation neural network training for proposed system

When the fig. (6) Explain the process for converting pseudo code to source code:

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

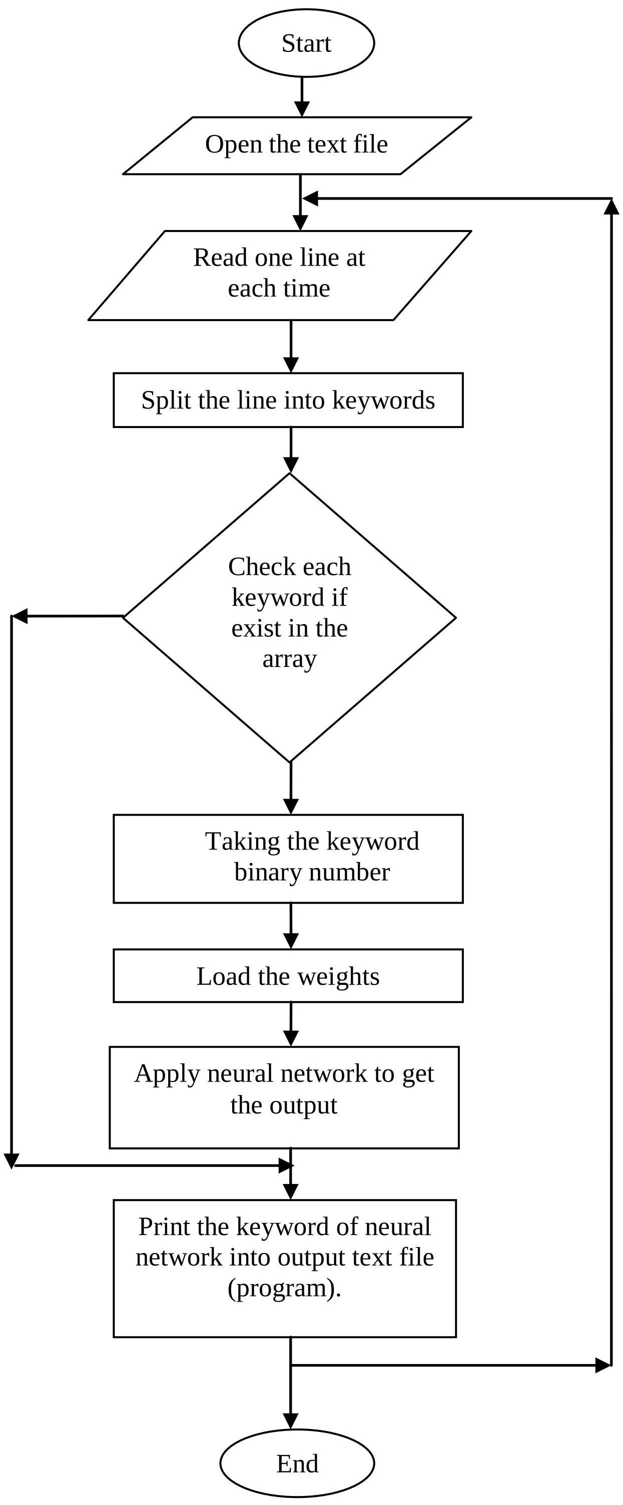
Start
Open the text file
Read one line at each time
Split the line into keywords
Check each keyword if exist in the array
Taking the keyword binary number
Load the weights
Apply neural network to get the output
Print the keyword of neural network into output text file (program).
End

Fig. (6) Flowchart for applying back propagation neural network to convert pseudo code into source code

Table 1. Training pairs of neural networks (a) input (pseudo code) and (b) target (matlab program language).

(a) Input	(b) Target
1. Selection statement	
i. select case, case where, select,	switch
or case of.	
case else, others, or default.	otherwise
case close, end switch, end select,	end
end case, end if, or endif.	
case where text expression	
case value1	switch variable
Block1	case 1
case value2:	any statement;
Block2	case 2
case value3	any statement;
Block3	case 3
default	any statement;
Block4	otherwise
Endcase	any statement;
	end
ii. if / then*	
if condition then	
true block	if b~=0
end if	any statement;
if condition then	end
true block	if a<0
else	any statement;
false block	else
endif	any statement;
if condition1 then	end
true block	if c==1
elseif condition2	any statement;
true block	elseif c<1
else	any statement;
false block	else
endif	any statement;
	end
2. Loop statement	
i. dowhile or do while	while
enddo, end do, end while,	end
endwhile, endloop, end loop,	
next, endfor, or end for.	
dowhile condition	while variable>10
Block	any statement;
enddo	end
ii. do for, dofor, loop,	for
loop for, do forever, or doforever.	
to, downto, ; , down to, or step	
dofor i= start to finish step	:
number	
Block	for variable =1:2:10
Endfor	any statement;
	end
3. Sequence statement	
i. write, output ,or print:	disp
write variable	disp(variable);
write " any statement "	disp(' any statement');
ii. read or enter:	input('')
read variable	variable =input('');
iii. procedure, method, or	function
subprogram.	
mothed sum (variable1,	function sum (variable1,

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

variable2)
variable3 := variable1+ variable2
write variable3
mothed sum (variable1, variable2)
variable3 := variable1+ variable2
return variable3
	variable2)
	variable3 := variable1+ variable2;
	disp (variable3)
	function variable3 =sum(variable1, variable2)
	variable3 := variable1+ variable2;

iv. comment or // // any statement	% % any statement

v. :=, is assigned or <-
variable:= value
set or initialize
set variable to value
	=
	variable= value;
	=
	variable = value

vi. not or negation: not variable	~ ~ variable
vii. <> or not equal variable1 <> variable 2	~= variable1~= variable 2
viii. equal to variable1 equal variable 2	== variable1== variable 2
ix. or: variable1> variable 2 or variable 2 < variable3	|| variable1> variable 2 || variable 2 < variable3
x. And: variable1> variable 2 and variable 2 < variable3	&& variable1> variable 2 && variable 2 < variable3
xi. exclusive-or: variable3 := variable1 exclusive-or variable 2	xor variable3=xor (variable1 , variable 2);
xii. exponential, or **: variable3 := variable1 exponential variable 2	^ variable3= variable1 ^ variable 2;
xiii. remainder, integer modulo, integer remainder, or modulus: variable3 := variable1 exponential variable 2	rem variable3= rem (variable1 , variable 2);
xiv. add or addition variable3 := variable1 add variable 2	+ variable3 := variable1 + variable 2;
xv. sub or subtraction: variable3 := variable1 sub variable 2	- variable3 := variable1 - variable 2;
xvi. div, division, or divide: variable3 := variable1 div variable 2	/ variable3 := variable1 / variable 2;
xvii. multiply, mult, or multiplication: variable3 := variable1 multiply variable 2	* variable3 := variable1 *

xviii. greater than
variable3 := variable1 greater than variable 2
	variable 2;
	
	>
	variable3 := variable1 >variable 2;

xix. greater than or equal to variable3 := variable1 greater than or equal to variable 2	>= variable3 := variable1 >= variable 2;
xx. less than variable3 := variable1 less than variable 2	< variable3 := variable1< variable 2;
xxi. less than or equal to variable3 := variable1 less than or equal to variable 2	<= variable3 := variable1 <=variable 2;
xxii exit	break

*the if/then not enters to the neural network because the same input and output.

The following example is shown the result of proposed algorithm when applying neural network techniques to convert pseudocode into matlab program automatically:

Pseudocode

```
read b,c
do for i=1 to 5
if b greater than i then
b is assigned b remainder c
print "this is the b" b
else
b<-b add 5
print b
endif
end for
```

source code

```
clc
clear all
close all
b=input('');
c=input('');
for i=1 : 5
if b > i
b = rem(b,c);
disp('this is the b');
disp(b);
else
b=b + 5;
disp(b);
end
end
```

VI. CONCLUSION

Depending the on training and testing time the best type of neural network for converting pseudo code to program is a cascade back propagation neural network. As shown table (2).

The algorithm of neural networks are shared in the same input layer (5 nodes), one hidden layer (21 nodes), the output layer (5 nodes), and the same error rate is 0.001. but the Back propagation and Cascade back propagation is used sigmoid activation functions to get the actual output, while the Radial basis function used the linear activation functions.

Table (2): The training and test time of neural network algorithms

Neural network algorithm	Train time (seconds)	Time recall (seconds)
Back propagation	2.5	0.0156
Cascade Back propagation	2.4	0.0156
matlab Radial basis function	4.2	0.4368

REFERENCES

[1] Mercer Rick, "Introduction to Computer Science", Citeseer, pp.4.

[2] Methods of Algorithm Description ", Second Edition, Board of Studies NSW, pp.6, March 1995.

[3] Prof. Kakati Mahanta Anjana, Prof. Kr. Deka Jatindra, and Prof. Goswami Diganta, " Master of Computer Applications COMPUTER PROGRAMMING USING C", Registrar on behalf of the Krishna Kanta Handiqui State Open University, pp.(14,15), July 2011.

[4] Prof. Gabriele Edward, "Journal of Research Administration", the Society of Research Administrators International, pp. (16), Volume XLI, Number 3, 2010.

[5] Roger S. pressman, Ph.D., "Software engineering a practitioners approach", 7th edition, McGraw-Hill Company, pp.301, 2010.

[6] Mukherjee Suvam and Chakrabarti Tamal, "Automatic Algorithm Specification To Source Code Translation Mukherjee", Indian Journal Of Computer Science And Engineering (Ijcse), Vol. 2 No. 2 Apr-May 2011.

[7] Karatrantou Anthi and Panagiotakopoulos Chris, "Algorithm, Pseudo-Code and Lego Mind storms Programming", Proceedings of International Conference on Simulation and Programming for Autonomous Robots/Teaching with Robotics: Didactic Approaches and Experiences, 2008.

[8] Garner Stuart, "A program design tool to help novices learn programming", ICT: Providing choices for learners and learning. Proceedings ascilite Singapore, 2007.

[9] Olsen Anne L, "Using pseudo code to teach problem solving", Journal of Computing Sciences in Colleges, December 2005.

[10] Krenker Andrej, Bester Janez, and Kos Andrej, "ARTIFICIAL NEURALNETWORKS-METHODOLOGICAL ADVANCES AND BIOMEDICAL APPLICATIONS Introduction to the Artificial Neural Networks", InTech, pp.1, 2011.

[11] Hopgood Adrian A., "Intelligent Systems for Engineers and Scientists", 2nd ed, CRC Press, pp.(210,229-231), 2001.

[12] Yeremia Hendy, Yuwono Niko Adrianus, Raymond Pius, and Budiharto Widodo, "GENETIC ALGORITHM AND NEURAL NETWORK FOR OPTICAL CHARACTER RECOGNITION", JCS, pp.(7), 2013.

[13] Negnevitsky Michael," Artificial Intelligence A Guide to Intelligent Systems ", Second Edition, , pp.(176,185), 2005.

[14] AL-Allaf Omaima NA and AbdAlKader Shahlla A, "NONLINEAR AUTOREGRESSIVE NEURAL NETWORK FOR ESTIMATION SOIL TEMPERATURE: A COMPARISON OF DIFFERENT OPTIMIZATION NEURAL NETWORK ALGORITHMS", UbiCC Journal, pp.45, 2011.

[15] Chen Dao-jiong and Zhao, Peng, "Study of the fault diagnosis method based on RBF neural network", Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on, IEEE, pp. (4350), 2011.

[16] Kasabov Nikola K., "Foundations of neural networks, fuzzy systems, and knowledge engineering ", Marcel Alencar, 2nd printing, pp.284, 1998.