Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

SOFTWARE APPLICATION FOR OPTICAL CHARACTER RECOGNITION

EGRI, A[ngela]; STOICUTA, O[limpiu] & SIRB, V[ali] - C[hivuta]

Abstract: *Our goal was to create an original software application for optical character recognition usable in the control and supervise system for vehicle transit ofprivat and public parking, in robotic and flexible manufacturing system . For this purpose we created a multilayer perceptron neural network which is trained using the back propagation algorithm. The software application was created using the C# programming language. In the final section of this paper we present a few screenshots of the running application that show the responses of the net to various input vectors.*
Key words: *artificial neural network, multilayer perceptron, back propagation*

1. INTRODUCTION

Through this paper we set out to build a neural network that can be used for Optical Character Recognition The neurons in an artificial neural network are grouped together in layers.
There are many ways of connecting neurons in a neural network. One type of neural network is the feed forward neural network in which the output of a neuron in one layer is tied to the input of a neuron in the next layer. Probably the most used type of neural network is the Multilayer Perceptron. In this type of network the neurons perform a weighted sum of their inputs and pass this value to the transfer function to produce the output. This type of networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity.

2. SOFTWARE DESCRIPTION

The neural network used in this application has a Multilayer Perceptron architecture consisting of an input layer, a hidden layer and an output layer.

The neurons in one layer of the network are connected to every neuron in the previous layer. The input layer of the used network has 35 neurons and each of these neurons has 35 inputs.

The hidden layer has 50 neurons each with 35 inputs and the output layer has 36 neurons each with 50 inputs. Each neuron in the output layer corresponds to a letter or a digit that the network has to recognize. Each neuron in the network uses a sigmoid activation function.

The software uses two structures: Neuron structure (fig.1) and NeuronLayerstructure (fig.2) and a class Neural Net (fig.3).

The Neuron structure is used to hold each neurons data, like the number of inputs, the output and the weight for each input. The Neural Net class is the actual neural network. This class holds data about the number of layers in the network, the number of neurons in each layer and the number of inputs of each neuron. The class also had methods for training and simulating the net as well as for saving and loading the network's weights.

The constructor of the class (fig. 2) takes 3 parameters which are the number of layers, an array containing the number of neurons in each layer and an array containing the number of inputs of the neurons in each layer.

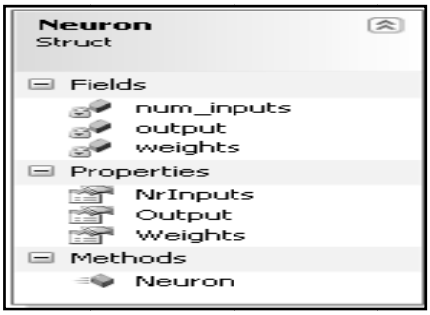

Fig. 1. Neuron structure

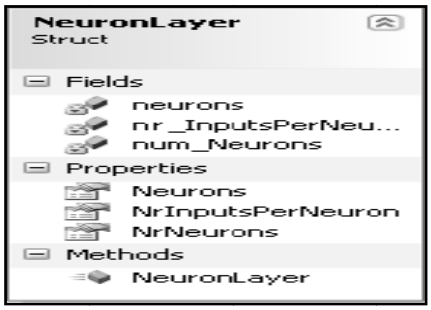

Fig. 2. Neuron Layer structure

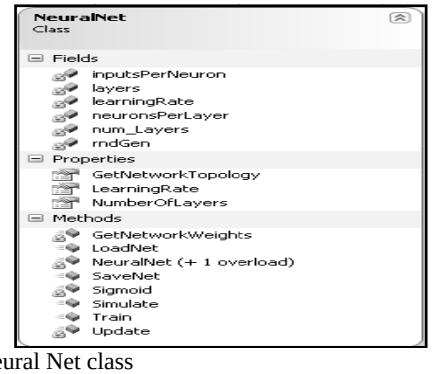

Fig. 3 Neural Net class

Using these parameters a neural network is created, a default learning rate is set and the weights of each neuron are initialised with a random value between -1.0 and 1.0.

The Neural Net class constructor has the following signature:
public NeuralNet(int nrLayers, int[ ] nrNeuronsPerLayer, int[ ] nrInputsPerNeuron)

The private GetNetworkWeights method reads the weights of each neuron from each layer and returns a jagged array containing these weights. This method is used by the Save Net

method to save the weights of the network to a file so that they can be loaded later. The Save Net method has the following signature: public void SaveNet(string filename)

The Load Net method loads the weights from the file to the current Neural Net object which calls the method. This method has the following signature: public void LoadNet(string filename)

This method reads the first line in the file to make sure that the data saved in it matches the architecture of the current neural network. If the architecture matches the method reads the file one line at a time and sets the network weights according to the data in the file. This method loops through each neuron of each layer starting with the input layer and determines the net input of each neuron:

$$\sum_i (w_{ij} \cdot y_i) + \theta_j \qquad (1)$$

This net input is then passed as an argument to the activation function to determine the output of each neuron. After execution the method will return an array of numbers which represents the output of the network. This method is used by the Train and Simulate methods based on the Update method to return the output of the network based on an input vector.

3. THE REALISED SOFWARE APPLICATION

This section presents an example of using the application. When the application starts the following form is shown (fig.4)

The main window of the application is divided into 2 areas. The left area is used for training and simulating the network and the right area is used for entering the data for the network to recognise. Before the training of the network can begin the number of epochs, learning rate and the number of digits must be entered. To train the network the user presses the Train button. When the train button is pressed a dialog window is shown in which the user can select a file that contains the training data. After the file has been selected training begins.(fig.5). After the training is finished a message box is shown telling the user that the network has been trained (fig 6 and 7). In order to simulate the network a letter or a digit must be entered in the white boxes in the right area of the main window. After the letter or digit has been entered the Simulate button can be pressed. The application will show a message box which displays what number or digit the network thinks the user has entered and the percentage with which the letter or digit was recognised. The following images show different responses of the neural net according to user input.

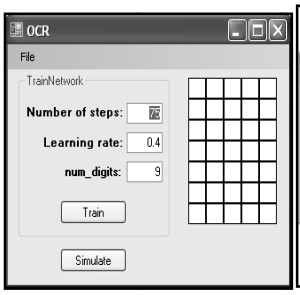

Fig. 4 Main window

Fig.5 Training selection

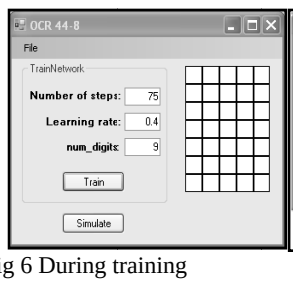

Fig 6 During training

Fig.7 Training finished

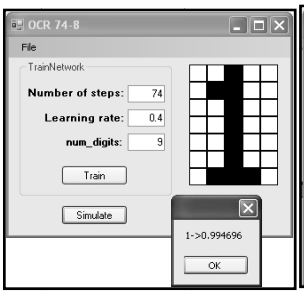

Fig.8 Recognised number :1

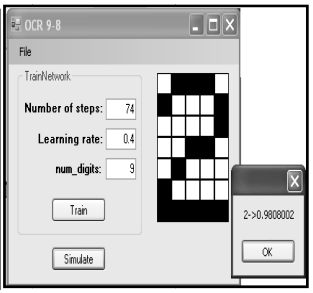

Fig.9 Recognised number :2

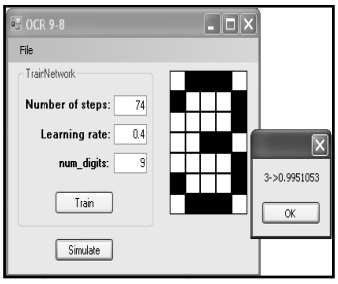

Fig.10 Recognised number :3

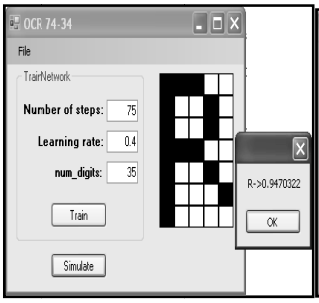

Fig.11 Recognised point :R

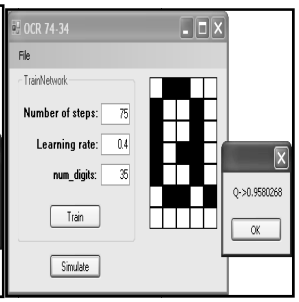

Fig.12 Recognised point :Q

4. CONCLUSION

This software application resolves with high fidelity the problem for optical character recognition. Running quickly and offer very good solutions for many domain: control and supervise system for vehicle transit of private and public parking robotic and flexible manufacturing system

5. REFERENCES

Egri, A. (2002). *Artificial Intelligence and Robotic*, Focus Petrosani, ISBN 973-8367-48-4, Romania.

Egri, A. (2002). *Flexibile Manufacturing System and Robots*, Teaching and Education Publishing House Bucharest, ISBN 973-30-2593-3, Romania.

Egri, A. & Moldovan, I. (2006). *Image processing in robotic*, *Proceedings of MicroCad2006*, Dobroka, M. (Ed.), pp. 19-24, ISBN 963-661-700-7Ö,ISBN 963-661-709-0 University of Miskolc, March 2006, Inovation and Technology Transfer Center of University of Miskolc Hungary, Miskolc.

Egri, A. & Sirb,V. C. (2007). *Modeling for Industrial Robots and Manufacturing Systems*, Anals of University of Petrosani,vol. 9, 2007 pp. 326-331, ISSN1454-8518, Universitas, Petrosani, Romania.

Sirb,V. C., Egri A., Stoicuta O. (2009). *Cim Control with Extended Kanbans System*, Anals Of The University Of Petrosani, Electrical Enginering, vol. 10.

Egri A., Sirb,V. C., Stoicuta O. (2009). *Position Control System of the Gmf360 Industrial Robot*, Anals of the University Of Petrosani, Electrical Enginering, vol. 10.