ACKNOWLEDGEMENT

The completion of this project is a combined effort of many people, without the help of whom, the very idea would have remained a dream. The project is, therefore, indicative of kind and willful help extended by many- including professors, teachers, men from various departments (both within the college and outside) as well as friends.

We wish to place on record our sincere gratitude and indebtedness to Prof. Lavit Rawtani and Prof. R. K. Pateria, both in Department of Computer Sc. & Information Technology, MANIT Bhopal, for their valuable guidance and help throughout the tenure of the project work.

To add, we are indebted to one and all of those who have been instrumental in providing useful information to us and also those who have helped and guided us in completing the project.

We extend our sincere thanks to Dr. J. L. Rana, Head of the Dept., Department of Computer Sc. & Information Technology, for his blessings and encouragement.

We would also like to thank our friends for their precious suggestions for the betterment of the project.

Last but not the least, we would like to thank our parents for their constant encouragement and support in building us to the kind of the student we are.

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL

CERTIFICATE

This is to certify that **Archana Kannoujia, Harsha Markam, Munmun De Choudhury, Piyush Parate** and **Rupali Gupta** of B. E. Computer Sc and Engineering have successfully completed MINOR PROJECT on "**PATTERN RECOGNITION USING ARTIFICIAL NEURAL NETWORKS**", in partial fulfillment of the requirement for the Bachelor of Engineering Degree at the Maulana Azad National Institute of Technology, Bhopal, under the guidance of Prof. Lavit Rawtani and Prof. R.K. Pateria.

Prof. Lavit Rawtani
Dept. of CS & IT
MANIT
Bhopal

Prof. R. K. Pateria
Dept. of CS & IT
MANIT
Bhopal

Dr. J. L. Rana
Head Of The Dept.
Dept of CS & IT
MANIT
Bhopal

TABLE OF CONTENTS

ABSTRACT

This aim of project is to provide a means to extract the handwritten or printed data filled into a from/document and store them into the corresponding database. An OCR technology is used to get the document off the paper and on to the computer.
The overall steps involved in the project are as follows:

- ® The source materiel is scanned using an optical scanner to read in the page as a BTIMAP.
- ® The bitmap image is processed and is fragmented into characters. Each character image is obtained by using a fragmentation routine, with space as the delimiting factor between two characters.
- ® The OCR module then processes this single character image using neural networks and recognizes the character. The OCR module makes use of the Error back propagation algorithm for supervised training of multilayer feed-forward artificial neural network (ANN) for character recognition.
- ® Once the source material has been converted to text format, a routine is called which identifies the form, so that the correct database may be populated.
- ® The required data i.e. the data filled into the document, is then extracted from this recognized text and stored into corresponding database.

1 INTRODUCTION

The aim of this project is to find a means by which the database entry from handwritten forms can be automated. Consider the case where a Railway reservation form is filled and submitted. The filled in details have to be entered in the database for future verification. This job has to be done manually by the employee thus taking much of this time and energy. Won't it be easier if just just clicking one single button did all these jobs. This is exactly what this project does. All that the user has to do is scan the document and relay. The scanned document is taken and subjected to a series of processing and the required data is finally extracted and stored into the corresponding Database.

The concept of Neuron Network is used to achieve the aim mentioned above. Forms can be recognized only if the network has been trained. So, the network needs to be trained first. Once the filled in from has been recognized, it has to be identified so as to populate the correct database table. This task requires that can we recognize all the empty forms once and store it, so that the identification of the form can be performed by comparing the filled in recognized form and the empty recognized form. This project works on forms of predefined, fixed format. This extracted data is filled into the existing database, which has already been created, one for each from the project is capable of recognizing. Thus summarizing the tasks that need to be done before the project can start, we have:

- ® Creating database tables, one for each form.
- ® Training the network to recognize characters.
- ® Recognizing the empty form.

Existing and proposed OCR technology

Optical character recognition (OCR) is the translation of optically scanned bitmaps of printed or written text character into character codes, such as ASCE. This is a efficient way to turn hard-copy materials into data files that can be edited and otherwise manipulated on a computer. Perhaps you might be thinking that a number of OCR software has already been around for a while. True. But the OCR technology used in this project is based on the concept of backpropogation algorithm of neural network technology . This ensures greater accuracy thus making it more reliable.

2 LITERATURE SURVEY

2.1 INTRODUCTION TO NEURAL NETWORK

Contemporary artificial intelligence (A1) was based on the logical processing of symbols. This file has achieved great success in areas through of as high level and complex, specifically export systems. Where it has failed are task that humans think of as easy, such as computer vision (Kay; Vision) and speech. Neural network are a growing area of interest they are mathematical models which task their inpriration from observations in biological neural systems.

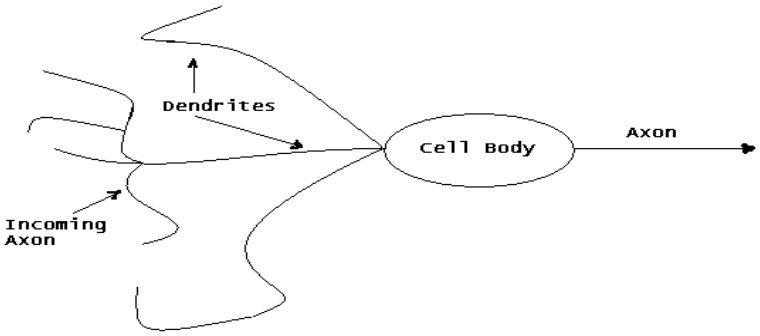


Figure [1]: Model of a simplified neuron.

Figure [1] shows the diagram of a simplified neural incoming introduction is received by the neuron at the synapse. The synapse is the connection from the axon of the neuron to the dendrite of another neuron. Information is passed from one neuron to another by a Electro-chemical transmission at the synapse. The signal is transmitted along the dendrites to the call body where some function of this summation of the electrical signal occurs. If this function is greater than a threshold the cell will fire, the cell will sent a signal (in the form a wave of ionization) along its axon to other tritons dendrites. Through this method of connection a reign is to pass information from its part of the network of neurons to another. Now it is crucial to recognise that the synapses are through to have varying efficiencies and they change the neurons lifetime, it is through these changes in efficiencies that learning occurs.

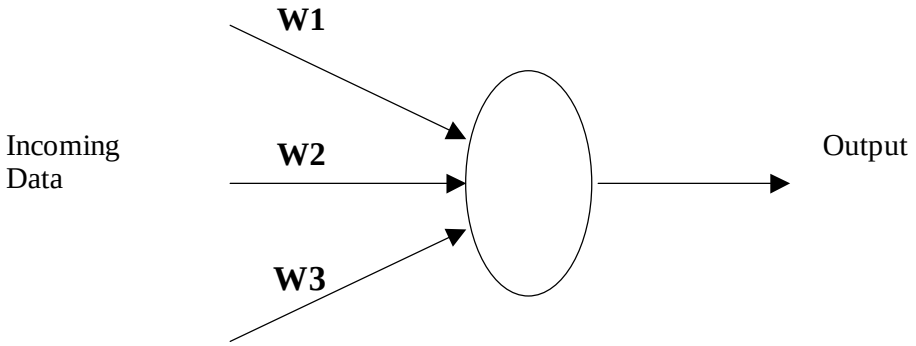


Figure [2]: An artificial neuron. The weights w, model the synaptic efficiencies. The input single is processed within the cell body and the output will be based on this processing.

Figure [2] is the simplified model we use as the basic of neuron network. The inputs to the neuron are represented by the vector, x, and the synapse efficiencies are modeled by the weight vector, w. the single output from the neuron is given by

$$y = f\left(\sum_{i=1} w_i\, x_i\right) = f\left(w.x\right) = f\left(w^T x\right)$$

The function of can represent any function, such as the identity function, threshold or a non-linearity. As can be seen the single neuron is a simple processing unit. The power of these methods comes not from a single unit, but from the network of units acting together and the learning in the network.

Learning In Neural Networks

The basic neural network function in two distinct stages, the first is the ……stage where the input activation's are fed through the network to the outputs. I be second stage is the learning where the network updates its weights, usually based on the most recent activation transfer. Before deciding on learning algorithms we need decide on the mode of learning, which may be one of three types:

® **Supervised Learning: –** in this type of learning we provide the network with input data, and the correct answer. The inputs are presented to the network activation is then propagated through the network and the output compared with the desired answer. If the answers do to agree then the weights are updated in such a manner as to ensure that the output of the network will more likely match the answer the next time the input is presented.

® **Unsupervised Learning: -**Which this method of learning we presented the network with the input, but don't provide any desired answer. The network is expected to find some structure in the data and self-organize. We could perhaps expect this structure to take the form of redundancy or clustering.

® **Reinforcement Learning: -** This method of learning is a mixture of the previews two types. The expected to organize itself in some manner to ensure a correct answer is more likely on the next presentation of the input.

2.2 BACKPROPATION ALGORITHM (ARTIFICIAL NEURAL NETWORK)

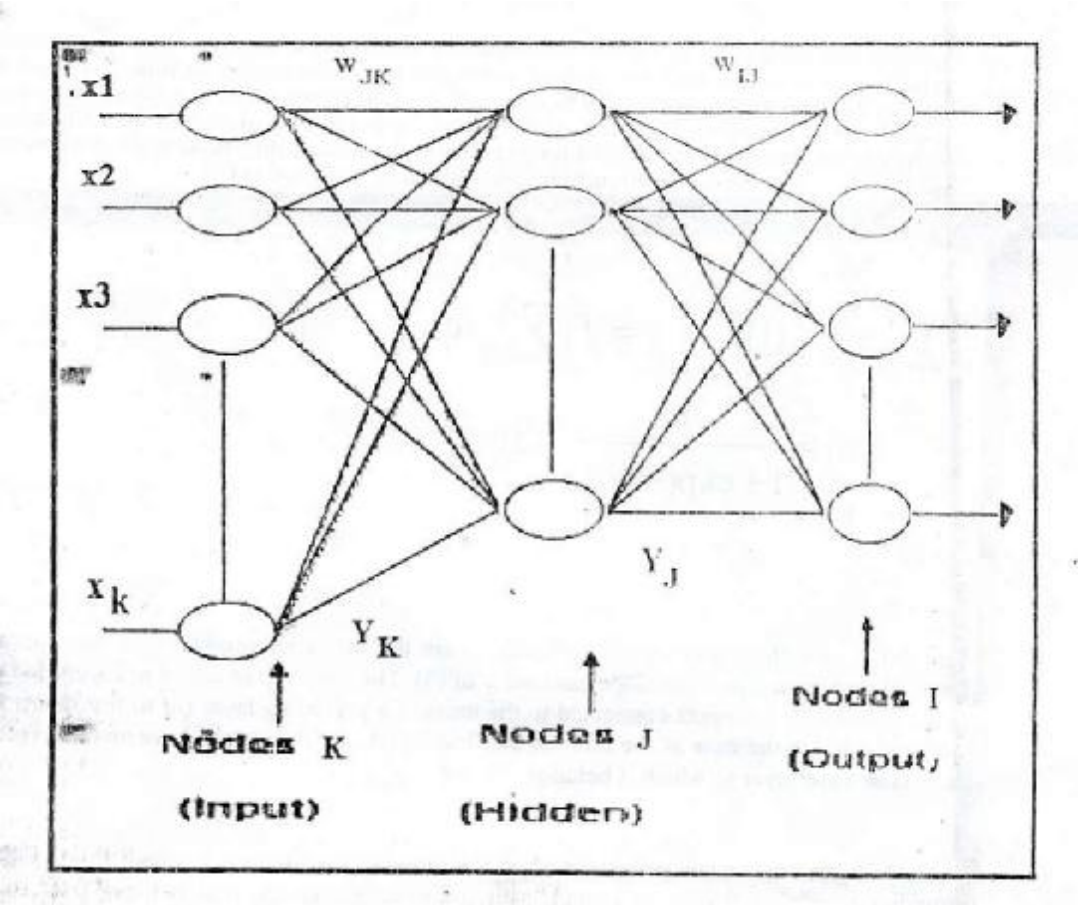
The Error-Back propagation (or simply, Backpropagation) algorithm is the most important algorithm for the supervised training of multilayer fed-forward Artificial Neural Networks (ANN). It derives its name from the fact that error signals are propagated backward through the network on a layer-by-layer basis. In this project we use an object-oriented approach selection of suitable error function or cost function, whose values are determined by the actual and desired outputs of the network, and which is also dependent on the network parameters such as the weights and the thresholds. [4] The basic idea that cost function has a particular surface over the weight space and therefore an iterative minimization process such as the gradient descent method can be used for its minimization. The method of gradient decent is based on the fact that, the gradient of a function always points in direction of maximum increase of the function then, moving to the direction of the negative gradient induces a maximal "downhill" sent that will eventually reach the minimum of the function surface over its parameter this is a rigorous and well-established technique for minimization of functions and has only been the main factor behind the success of Backpropagation. [2] A typical multi-layer forward AN is shown in Figure 1. This type of network is also known as a (MIP) the units (or nodes) of the network are nonlinear threshold units described by neurons (1) and (2) and their activation function is given by equation (3):

$$\sum_{i=0}^{N} w_i x_i \quad (1) \qquad y = f\left(\sum_{i=0}^{N} w_i x_i\right)$$

$$f(u) = \frac{1}{1 + \exp(-\alpha u)}$$

These are x1, x1……are the input signals, wl, w2…….. are the synaptic weights. U is the activation denial of the neuron and is the slope parameter of (3). The units are arranged in layers and each in a layer has all its inputs connected to the units of a preceding layer (or to the inputs from external world in the case of the units in the first layer), but it does not have any connectors units of the same layer to which it belongs.
The layers are arrayed one succeeding the other so that there is an input layer, that is those that have no inputs or outputs to the external world, are called hidden layers. Figure 1 show a MLP with only be hidden layer Back propagation neural networks are

usually fully connected. This means that each unit is connected to every output from the proceeding layer for to every input from me external world if the unit is in the first layer). Generally, the input layer is considered adjust distributor the signals from the external world and is not there are accounted as a layer.



The back propagation training consists of two passes of computation a forward pass and a backward pass [3] In the forward pass an input pattern vestal appeared to the sweats nodes of the network that is to the units in the input layer the appeal from the input layer propagate to the units in the first layer and each unit produces a output according to the equation (2). The outputs of these units are propagated to units is subsequent layers and this process continues until. Finally the signals reach the output layer where the actual response of the network to the input vector is obtained. During the forward pass the synaptic weights of the network are lived. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error signal which is propagated backward through the network against the direction of synaptic connection.

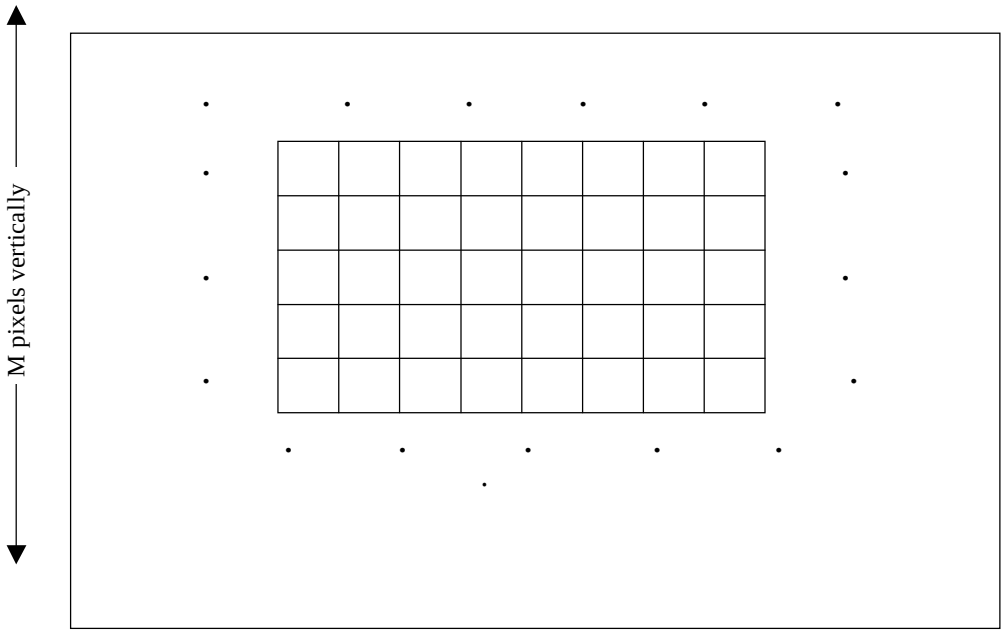
2.3 EFFICIENT INITIALIZATION-NGUYEN WIDROW

Efficient initialization can speed up the convergence process significantly even by the order of magnitude. A popular neutralization algorithm developed nyuyen and widron and used in the MAHAB Neural Network Toolbox.

2.4 BITMAP

Definition

Bitmaps are defined as a regular rectangular mesh of cells called pixels., each pixel containing a color value. They are characterized by only two parameters, the number of pixel and me information content (color depth) per pixel. There are other attributes that are applying pixels but they are derivations of these two fundamental parameters.



Note that bitmaps are always orientated horizontally and vertically. Pixels should be considers square although they may have other aspect ratios in practice in the majority of situations bitmaps are to used to represent image on the computer.

Color "depth"

Each pixel in a bitmap contains information, usually interpreted as colour information. The information content is always the same for all the pixels in a particular bitmap. The amount of color information could be whatever the application requires but there are me standard the main ones are described below.

1 bit (black and white)

This is the possible information content that can be hold for each pixel. The resulting bitmap is referred to as monochrome or black and white. The pixels then are

referred to as black pixels with a area 1 referred to as white. Note that white only two colors are possible and they could be interpreted as any two colors, 0 is mapped to one color. 1 is mapped to another color.

Specific formats

The following is a list and brief comments of some specific file format that are widely used for saving bitmaps.

Format: TIFF (Tagged Image File Format)
Platforms: Commonly supported on Mac/DOS-WINDOW/Unix.
Owner: Altus
Notes: TIFF is an international standard for storing and interchanging bitmaps between applications and hardware platforms. It is almost always supported by major applications that provide bitmap manipulation. The format consists of items called tags, which are defined by the standard. Each tag is followed a tag depended data structure. Supports most color spaces and compression methods.

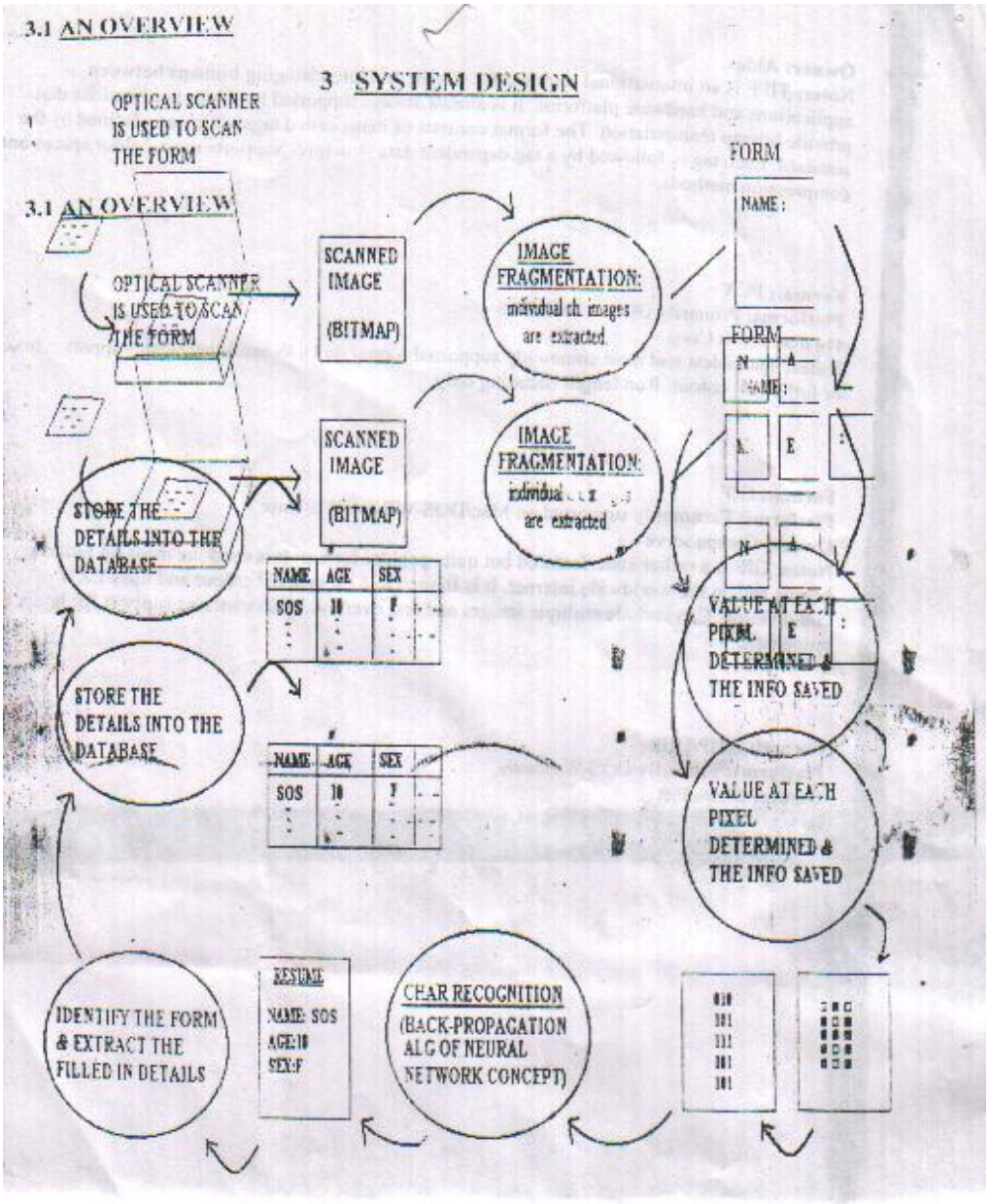
Format: PCX
Platforms: Primarily DOS-WINDOWS
Owners: Zsoft Crop
Notes: The oldest and most commonly supported formats on DOS machines can support exude or full 24-bit color. Run length encoding only.

Format: GIF
Platforms: Commonly supported on Mac/DOS-WINDOW/Unix
Owner: CompuServe
Notes: GIF is a rather underfeatured but quite popular format. It is used the most on button boards and on the worldwide Internet. It is limited to 8 bit-indexed colour and used LZW compression. Can include multiple images and text overlays. Also contains support for layers and animation.

Format: BMP/DIB
Platforms: Primarily DOS-Windows
Owner: Microsoft
Notes: Microsoft Windows format, rarely supported elsewhere. Supports 1,2,4,8,and 32 bit color images.

3. OVERVIEW

3.1 AN OVERVIEW



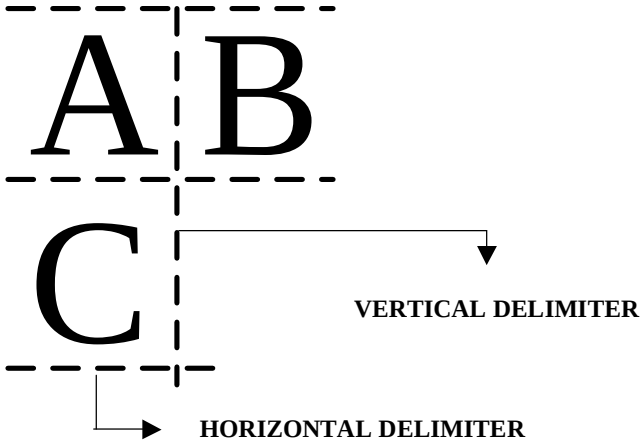
3.2 OPTICAL SCANNER

The document to be scanned is fed to the OPTICAL SCANNER. The optical scanner reads customized response forms and creates a data directly, thus avoiding the expensive and tedious hand keying of data. The output of the scanner is a BTIMAP (patter of dots) image.

3.3 FRAGMENTATION

From the output of the scanner, the bitmaps is processed individual characters are extracted the image format for further processing.
Character fragmentation is done as described below.

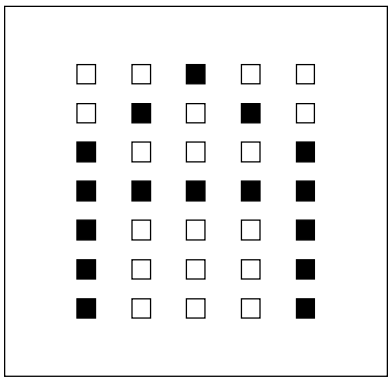
Every character is distinguished from the rest based on a delimiter.
In the case the SPACE acts as a delimiter in fig (I) the dotted lines represent the horizontal and vertical spacing (delimiter).



FIG(1)

Once a delimiter is spotted on all four sides the area within is recognized as a characters saved for further processing.

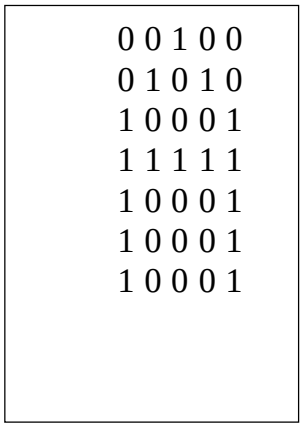
In the next stage the value at each pixel is determined. This information then saved. The bitmap image pattern of dots is shown below and the output processing is show in fig (b):



]

BITMAP
(PATTERN OF DOTS)

fig(a)



PLXEL VALUES
(0/1)
(.txt file)
fig (b)

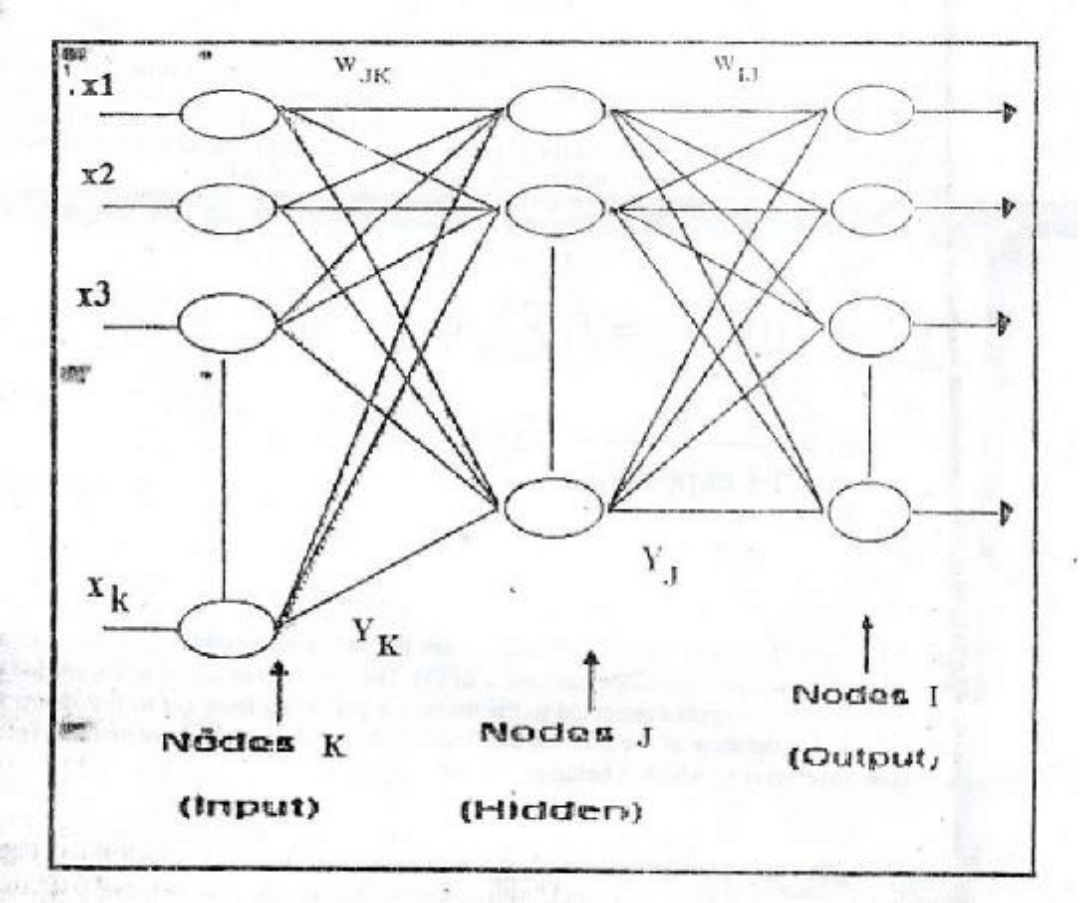
This text file containing the pixel values is saved and fed as input to the Neural Network in the next stage.

3.4 BACK-PROPAGATION ALGORITHM

3.4.1 Architecture

The Error-Backpropagation (or simply back propagation) algorithm the most algorithm for the supervised training of make feed-for and Artificial Neural Network (ANN) through the network are layer-by-layer basis.

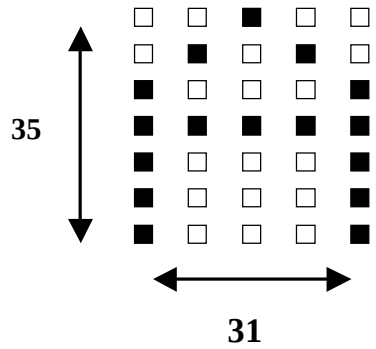
The units are arranged in layers and each unit in a layer has all is inputs connected to the units of a preceding layer (or to the inputs from the external world in the case of the units in the first layer), but it does not have any connecting to other so that there is an input layer, multiple inputs or outputs to the external world, are called hidden layer. Figure 1 shows a MLP with two hidden layer backpropagation neural networks are usually only connected. This means that each unit is connected to every output from the preceding layer or to every input from the internal world it the unit is the first layer. Generally, the input is considered as just a distributor of the signal the external world.



The diagram thus shows the basic architecture of the neuron network used it is totally four layers.

- One input layer
- Two hidden layer
- One output layer

The INPUT layer as shown in the figure has K neurons. The value of K is calculated as 35*35 1085.



INPUT LAYER K = 31*35 = 1085

The number of neurons in the HIDDEN layer is assumed to be 90 and 55.

These values are assigned arbitrarily.

Number of alphabets =26 (A, B, C……Z)
Number of digits = 9 (0, 1, ……9)
Thus we have the OUTPUT LAYER 1.

$$y = f\left(\sum_{i=1} w_i\, x_i\right) = f\left(w.x\right) = f\left(w^T x\right)$$

$$\sum_{i=0}^{N} w_i\, x_i\,(1) \qquad y = f\left(\sum_{i=0}^{N} w_i\, x_i\right)$$

$$f(u) = \frac{1}{1 + \exp(-\alpha u)}$$

These x1, x1……are the input signals, wl, w2…….. are the synaptic weights.
After all the processing (learning and testing) the network selects that output signal that gives the highest value. This is how the character is recognized.

3.5 IDENTIFY THE FORM

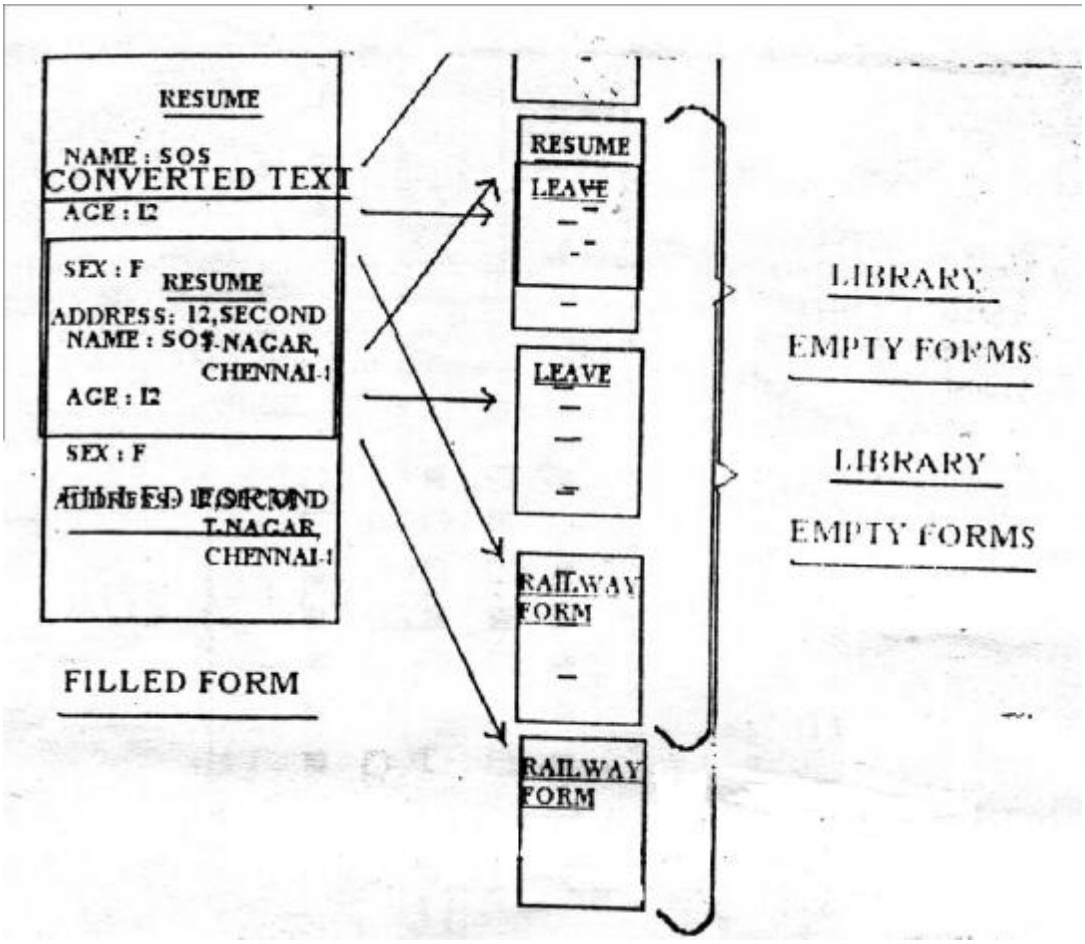


Fig (ii) **IDENTIFICATION OF THE FORM**

This form is compared with all others in the library identify what type to format i.e. Whether it is a Railway reservation form or Resume or leave application form or passport application form, etc. as shown in fig (ii). Once the form is identified it is then processed for **Data Extraction.**

During the Data Extraction process the Filled in Details are alone extracted into the database and the data that has to go into the DATABASE are alone extracted and saved which is used layer in the next stage .

3.6 DATABASE ENTRY

The data needed to be saved in the Database is saved in a text file. This data then enters the corresponding database. However every from has it's own Database. Thus the required data stored in the Database with less human effort. These Data can be later used or manipulated.

4 IMPLEMENTATION

We have used Visual C++ as the object oriented programming language in implementing the backpropagation algorithm.

Visual C++ fully supports object-oriented programming, including the four pillars of object-oriented development: encapsulation, data hiding, inheritance, and polymorphism. Encapsulation and Data Hiding. When an engineer needs to add a resistor to the device she is creating, she doesn't typically build a new one from scratch. She walks over to a bin of resistors, examines the colored bands that indicate the properties, and picks the one she needs. The resistor is a "black box" as far as the engineer is concerned--she doesn't much care how it does its work as long as it conforms to her specifications; she doesn't need to look inside the box to use it in her design.

The property of being a self-contained unit is called encapsulation. With encapsulation, we can accomplish data hiding. Data hiding is the highly valued characteristic that an object can be used without the user knowing or caring how it works internally. Just as you can use a refrigerator without knowing how the compressor works, you can use a well-designed object without knowing about its internal data members.

Similarly, when the engineer uses the resistor, she need not know anything about the internal state of the resistor. All the properties of the resistor are encapsulated in the resistor object; they are not spread out through the circuitry. It is not necessary to understand how the resistor works in order to use it effectively. Its data is hidden inside the resistor's casing.

Visual C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called classes. Once created, a well-defined class acts as a fully encapsulated entity--it is used as a whole unit. The actual inner workings of the class should be hidden. Users of a well-defined class do not need to know how the class works; they just need to know how to use it. When the engineers at Acme Motors want to build a new car, they have two choices: They can start from scratch, or they can modify an existing model. Perhaps their Star model is nearly perfect, but they'd like to add a turbocharger and a six-speed transmission. The chief engineer would prefer not to start from the ground up, but rather to say, "Let's build another Star, but let's add these additional capabilities. We'll call the new model a Quasar." A Quasar is a kind of Star, but one with new features.

Visual C++ supports the idea of reuse through inheritance. A new type, which is an extension of an existing type, can be declared. This new subclass is said to derive from the existing type and is sometimes called a derived type. The Quasar is derived from the Star and thus inherits all its qualities, but can add to them as needed. The new Quasar might respond differently than a Star does when you press down on the accelerator. The Quasar might engage fuel injection and a turbocharger, while the Star would simply let gasoline into its carburetor. A user, however, does not have to know about these differences. He can just "floor it," and the right thing will happen, depending on which car he's driving.

Visual C++ supports the idea that different objects do "the right thing" through what is called function polymorphism and class polymorphism. Poly means many, and morph means form. Polymorphism refers to the same name taking many forms.

Thus our implementation of the backpropagation algorithm involves the following directives:

- Allows the user to specify the size of each of the layers.
- Allows the use of two hidden layers.
- Be able to save and restore the state of the network.
- The recognized character is stored in a text file.

4.1 Classes and structures used

This section describes, in brief, the classes and structures used. We use a structure for the "RGB" components of the bitmap image. It has the following data items.

1 regcomp

Unsigned character blue, green, red, resv
These items store the RGB values of the bitmap.

2. Synapse

Double weight.
Each neuron in one layer is connected to all other neurons in the next layer. Double weight represents the value on each connection.

3. Neuron

Double value
Double error
Double bias

Value stores the input data each neuron. Error is calculated each neuron as the difference between the desired out and the obtained output. Bias is added to each neuron for the rapid convergence of the training process. This offsets the origin the activation function, producing an effect that is similar to adjusting the threshold of the perception neuron.

4 Layer

long stneuron
long endneuron
long stdendrite
long enddendrite
long neuoncount
long dendritecount

stneuron denotes the start neuron of each layer. For example if there are 1085 neuron in the input layer, 64 neuron in the first hidden layer, 36 neurons in the second hidden layer and 41 neurons in the output layer, thus amounting to a total of 1226 neurons in the

entire network, then the start neuron for the input layer is 0 and end neuron is 184. Similarly, stneuron for the hidden layer is 1085 and the endneuron is 1149 (1085+64) and so on. Stdendrite denotes start dendrite and end dendrite denoted end dendrite, whose values are assigned analogous to start neuron and end neuron. Neuroncount and Dendritecount represent respect the total number of neurons and dendrites present in the layer.

The class hierarchy used here is shown below:

nn
Structure Synapse Structure Neuron Structure Layer bool tranmcyles double learncoeff double learn ateincrease double learinratedcrease double sumsqerror bool running bool stopping long dendritont long dendriton long neuronont long layeront
void setinp (double data []) void processoutput () void calcuotpt () void adjustwts (double Target []) void train (double Data [], double Target []) void createnet (long input, long hidden, long hidden 2, long output) void destrounet () void stopworking () bool savenet (char *Filename) bool loadnet (char *Filename) double squish (double in Val) void nguyenwidrowinitialise () double outputlayer (long index) nn() virtual~nn()

4.2 MODES

There two mode present in the system: training mode and testing mode.
The user is allowed to operate only in the testing mode. The network is trained in the training mode by us and provided to the user.

4.2.1 Training

In this mode the system is provided with the following information:

- ® The data files and the target output patter for each data file.
- ® The values for learning rate.
- ® The number of neurons in each layer.

Based of the above information, the network is trained.

4.2.2 Testing Mode

In this mode the test data, which represents a character in the item: 0's and 1's is provided as a text file. When this file is applied to an already trained network, an output file generated which contains the output from the network.

4.3 WORKING

The document to be recognized is first scanned using a scanner and as a Bitmap file. This file is then given as input to a fragmenting routine, which extracts each character from the bitmap file. This character has to be converted to standard size 31*35 pixels, so it passed to a routine called stdsize, which converts in to standard size. This text file is given as input to the trained network, which recognizes the character and writes it into text file. The above process is repeated till all the characters to the file have been extracted.
Now, the document has been recognized. It is has to be stored in the corresponding database. The recognized file is passed on to a routine, which extracts the data filled into it, by comparing the filed form with the empty form. The empty form is already recognized and stored as a text file. Thus, the extracted data is stored into the corresponding database.

5 CODING AND TESTING

5.1 CODING

The bitmap to be converted is first passed on to the "fragment" routine. This fragments the bitmap image into individual character images.

FRAGMENT

1. The pixel values corresponding to the x and y axis the images are stored into two pointers named SPX and spy.
2. Space is used as the delimiting character to extract characters.
3. The values of spy is checked to see if they are consecutive if there is any break then it represents a space in the horizontal direction, denoting a line.
4. The values of spx are checked to see if any they are consecutive. if there is a break then it represents a space between two characters.
5. Step 4 is represented till the characters in all the lines are obtained.

Once the characters have been extracted, each character is passed on to the "find size" routine, which finds the height and width of the character.

FIND SIDE

1. The pointer spx is stored and the height value is subtracted from the least value to give the width of the character.
2 The pointer spy is sorted and the highest value is subtracted from the least value to give the height of the character.
3 The position of the bottom left most pixels is also obtained as the least value of spx and highest value of spy.

The character is then passed on the "out side "routine which converts the charted to standard size 31*35 pixels.

STDSIZE

1. The height and the width ratio is found as 34/height of the character and the width of the character respectively.
2. Scaling is performed by multiplying the x axis pixel values (spx) to width ratio of the axis pixels values (spy) by height ratio.
3. The pixel values are checked to see if there is any gap between them as a result of the scaling
   Operation if so the gaps are filled.
4. The character is stored as a text tile consisting of 1's and 0's.

This text file has to be given as input to the trained network. Whose output is a text file containing the recognized character .The input text file containing 1's and 0's is passed into the "recog" routine. Which loads the trained network and output text file.

RECOG

It has two modes

- Training mode
- Testing mode

In the training mode the network is first created the output pairs are provided to the created network. The network is trained by adjusting the weight .the trained network is saved.

1. If the net has not been already created. Create net by calling "create net " the load the created network by calling "loadnet".
2. Train the network by calling the routine "train".
3. Save the trained network by calling "savenet".

In the testing mode, the following operating are performed on the already trained network:

1. Load the network by calling "loadnet".
2. Assign the input data to each neuron in the input layer by calling "setup".
3. Calculate the output of the network by calling " process output".
4. Find the highest value in the output layer. This is the result it is written in a file.

CREATENET

1. Allocate memory dynamically for all structure (layer, neuron, synapse class nn).
2. Initialize weights by Nguyen Widrow weight initialize methods by calling " nguyenwidrowinitialise".

LOADNET

1. Open the saved network in read file.
2. Read all the details in the file into the variables.

TRAIN

1. Assign the input data to the value of the neuron by calling set input".
2. The output is calculated by calling "calcuoutpt" this performs the forward pass.
3. The weights are adjusted by calling "adjustwts". This performs the back ward pass.

SAVENET

1. Open the file where the network has to be saved in write mode.
2. Write in the following variable: save header, layer count, neuroncount, dendritecount, mtrainingeycles, learningrateincrease, learningratedecrease, and learning coefficient. The neuron count for each layer. The bias values for each neuron. Weights for each of the dendrites.

NGUYENWIDROWINITIALISE

1. Assign random weights to all the dendrites.
2. Assign random bias to all the neurons.
3. acc is a variable calculated by summing the squares of the weights.
4. acc is calculated as the square root of (1/acc)
5. The new weight is calculated as the old weight multiplied by acc 2
6. The variable acc2 is found as the summation of the new weights.
7. The new bias is calculated as the old bias- (acc22).

SETINP

Assign the input data passed to the value of each neuron in the input layer.

CALCUOUTPT

1. This is where the forward pass is done.
2. Find the summation of (X*W) input weight
3. Pass the summation along with the bias to a function which produces an output in the range of 0 to1. The routine for the activation function is called "squish".

ADJUSTWTS

1. This is where the backward pass is done.
2. For the output layer the error that is the difference between the desired output and the actual output is calculated. It is squatted and its summation is found.
3. The proportional error is found for each neuron as (error)*(value) (1-value).
4. The error for the connected neurons in the previous layer is updated by multiplying the error in the next layer with the weights connecting the two layers.
5. Once the error has been found the bias is updated. The new bias is calculated as old bias + (learning coefficient* error)
6. The weight are updated as follows New weight: (old weight)+learning coefficient* value of the neuron at left end*error of the neuron the right end.

OUTPUTLAYER

Return the value of the needed neuron.

PROCESSOUTPUT

This performs the forward pass by calling the routine "calcuoutput".

DESTROYNET

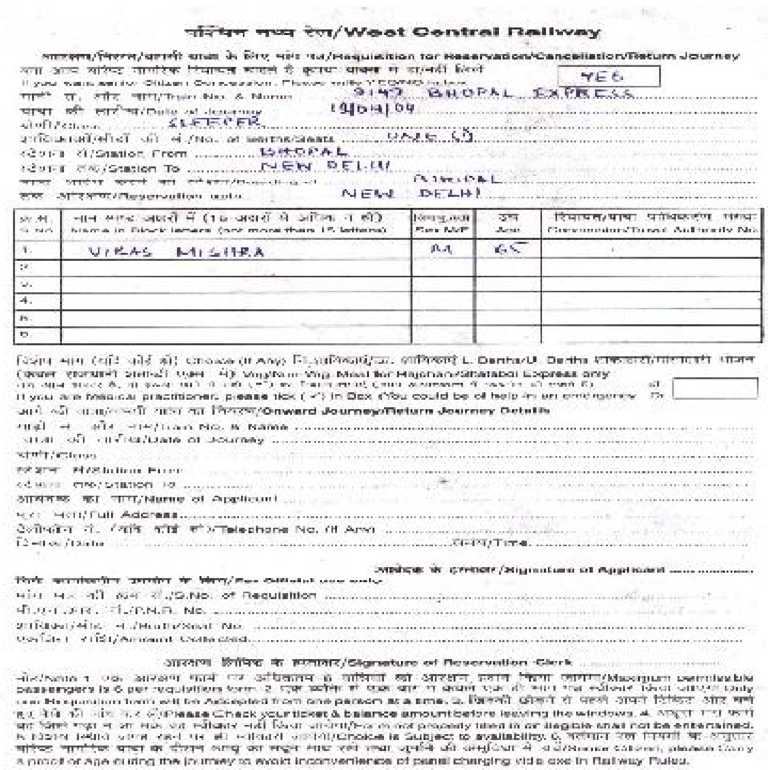
The pointer allocations are freed, using delete [] operator.

STOPWORKING

The global variable stopping is made true.

5.2 TESTING

The forms used for testing and their results are shown below:


West Central Railway

Requisition for Reservation/Cancellation/Return Journey

YES

Train No. & Name: 2143 BHOPAL EXPRESS

Date of Journey: 19/04/09

Class: SLEEPER

No. of Berths/Seats: ONE (1)

Station From: BHOPAL

Station To: NEW DELHI

BHOPAL

Reservation upto: NEW DELHI

S.No.	Name in Block letters (not more than 15 letters)	Sex M/F	Age	Concession/Travel Authority No.
1.	VIKAS MISHRA	M	65	
2.				
3.				
4.				
5.				
6.				

Choice (If Any): L. Berth/U. Berth

Veg/Non Veg Meal for Rajdhani/Shatabdi Express only

If you are Medical practitioner, please tick (✓) in Box (You could be of help in an emergency)

Onward Journey/Return Journey Details

Train No. & Name

Date of Journey

Class

Station From

Station To

Name of Applicant

Full Address

Telephone No. (If Any)

Date

Time

Signature of Applicant

For Official use only

S.No. of Requisition

PNR No.

Berth/Seat No.

Amount Collected

Signature of Reservation Clerk

Resulting text:

Senior Citizen Concession: YES
Train No: 9149 BHOPAL EXPRESS
Name of the Train: BHOPAL EXPRESS
Date of Journey: 14/04/04
Class: SLEEPER
No. of Berths or Seats: ONE (1)
Station from: BHOPAL
Station to: NEW DELHI
Boarding at: BHOPAL
Reservation upto: NEW DELHI

Name of the passenger: VIKAS MISHRA
Sex: M
Age: 65

Resulting text using an already existing OCR

Senior Citizen Concession: YES
Train No: 9149 BHOPAL EXPRESS
Name of the Train: BHOPAL EXPRESS
Date of Journey: 14/04/04
Class: SLEEPER
No. of Berths or Seats: ONE (1)
Station from: BHOPAL
Station to: NEW DELHI
Boarding at: BHOPAL
Reservation upto: NEW DELHI

Name of the passenger: VIKAS MISHRA
Sex: M
Age: 65

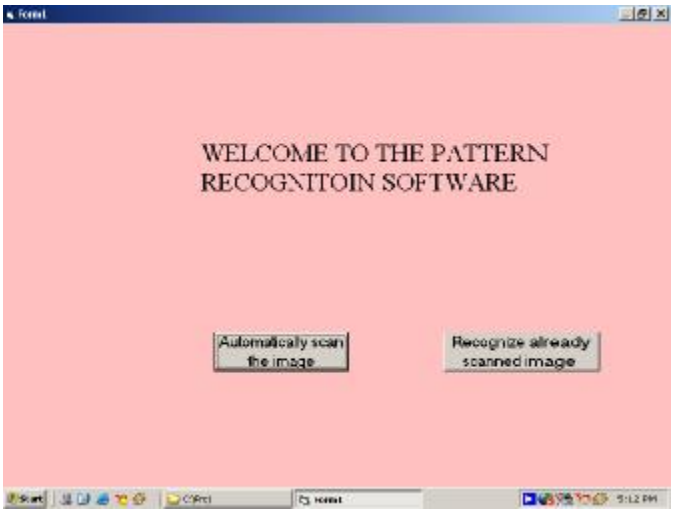
Extracted data which is to be stored in the database

YES
9149 BHOPAL EXPRESS
BHOPAL EXPRESS
14/O4/O4
SLHPER
ONE (1)
BHOPAL
NEW DELHI
BHOPAL
NEW DELHI

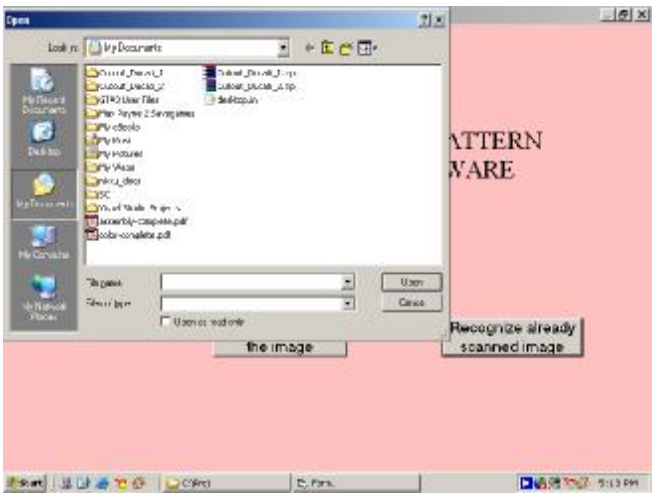
6. USER MANUAL

Initially the screen that gets displayed is shown below:
As shown in the fig it has the following buttons:
The project provides the user the following options:

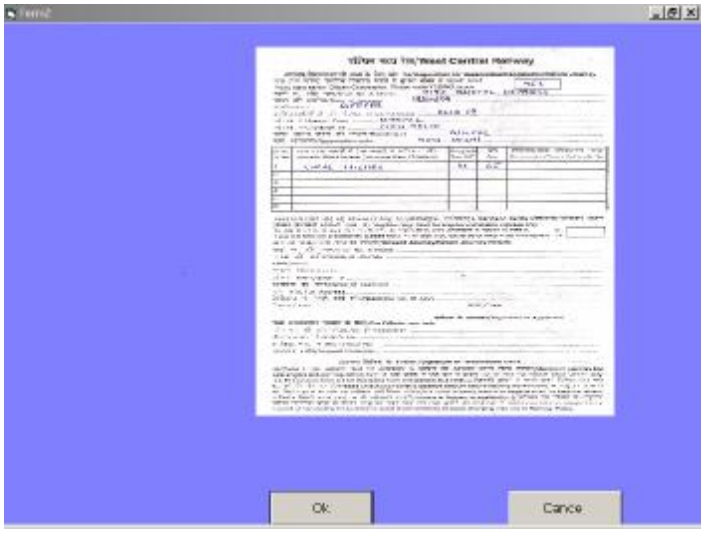
- Automatically scan the image.
- Recognize already scanned image.



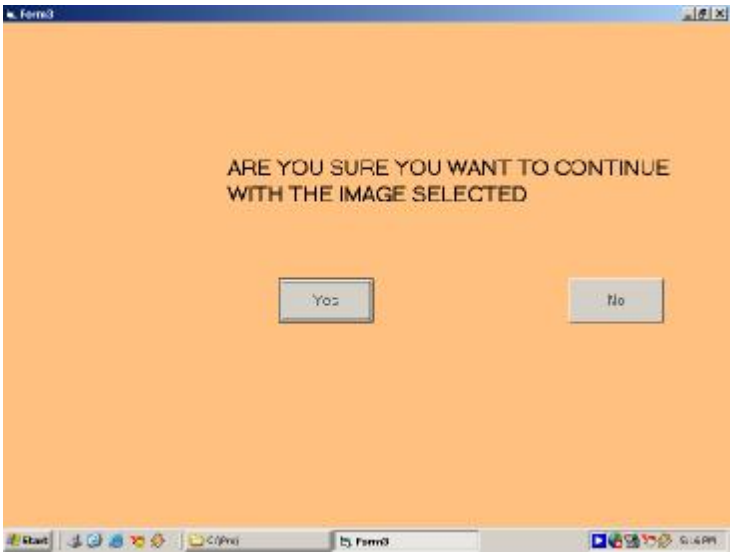
The difference between the two is that "Automatically scan the image" scans and recognizes the image fed to the scanner; and "Recognize already scanned image" derives it from a default location and then recognizes it. The user can just click this button and relax.



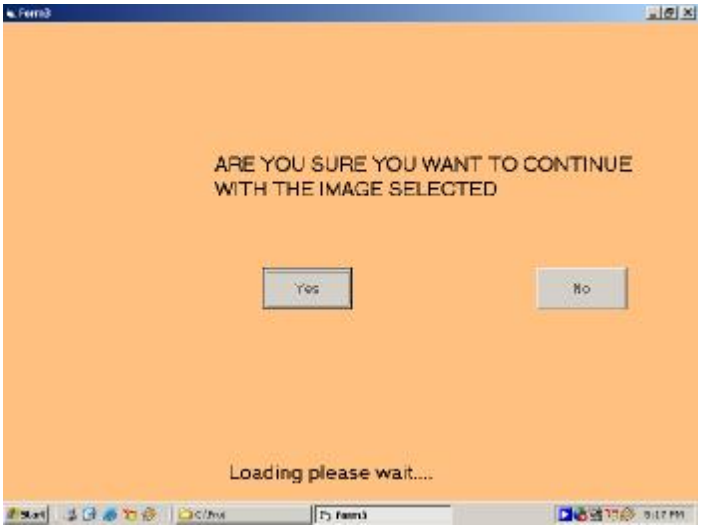
On clicking the 2nd option the following browsing window opens and the user is allowed to choose the image of his choice.



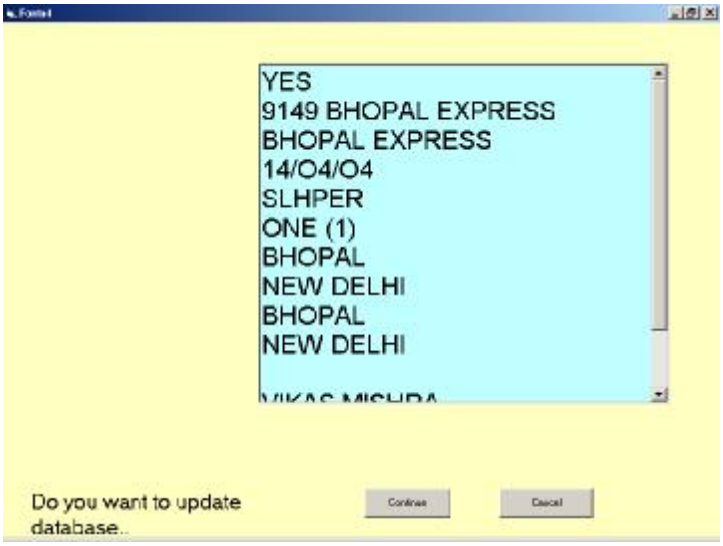
After choosing the image, it is provided as a preview to the user and a confirmation of the chosen image is asked for. If the user clicks “No” he again returns to the page wherefrom he can choose another image from the browse option.



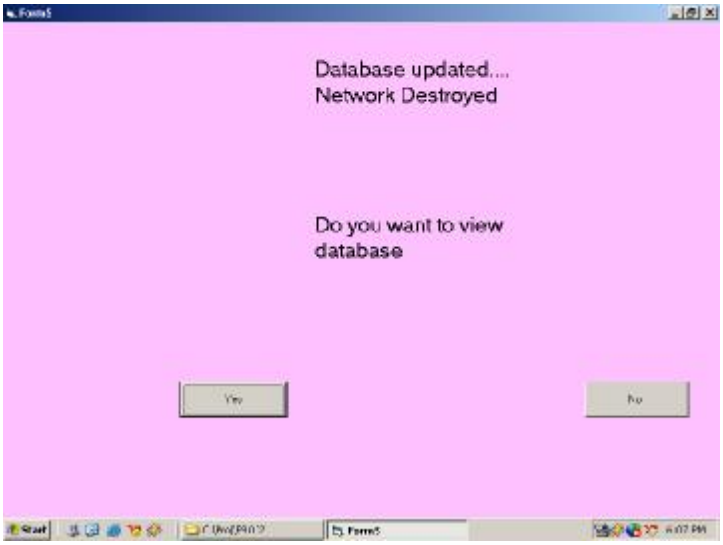
On getting the confirmation the software starts a timer and waits for the loading of the image for recognition.



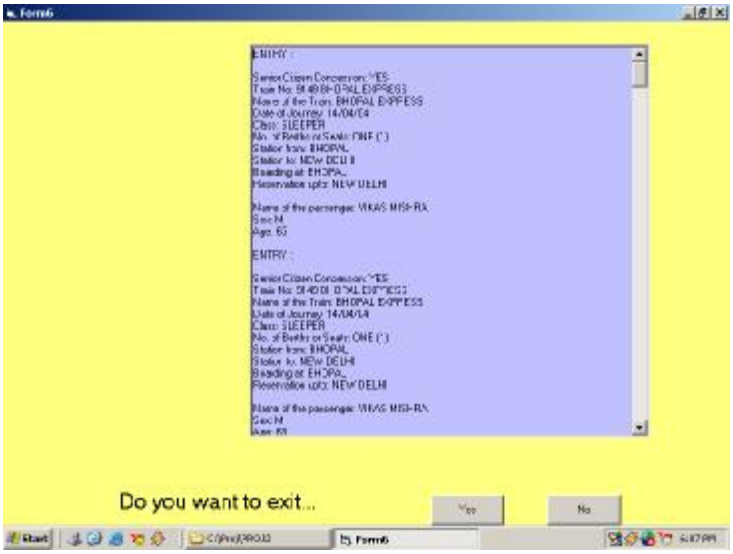
The neural network established then recognizes the handwritten characters and displays the extracted information to the user.
The user is again asked if he is interested in updating the same in the database.
If the user chooses “Cancel”, he comes out of the software.



If the user wishes to update the Database, the neural network is destroyed and it is visualized that the database has been updated.
The user id asked if he wants to view the Text Database.
If “Yes”, the Database is shown.



If the user chooses “No” he exits from the software with a warning message box without the Database getting updated.



If the user wants to exit, he can click on the “Exit” button and exit from the software.
If he chooses “No”, he is returned to the main screen wherefrom he can again choose another image and proceed for recognition.

7. CONCLUSION

This project entirely concentrates on designing an OCR technology that enables us to get the details on a document off the paper and onto the computer. The project primarily aims at getting the filled in details from any form (railway reservation form, resume, etc.) into the Database with least bit of effort. The highlight of the project is the OCR, which uses Backpropagation algorithm of the Artificial Neural Network concept. This concept ensures us greater accuracy.

In the project we use an object-oriented approach for implementing the back propagation Algorithm. Thus the end result is the Database containing the filled in details, which can be later used for verification or manipulation.

Application areas of the project:

- © Railway ticket booking.
- © Passport application processing.
- © Business
  - © Preformatted resumes.
  - © Leave application.

****

APPENDIX (Source Code)

```
//HEADER FILES

//Character Recognition

#include<windows.h>
#include<process.h>
#include<iostream.h>
#include"stafx.h"
#include"resource.h"
#include"net.h"
#include"stdio.h"
#include"conio.h"

int *findsize(int spmx[],int spmy[],int sm);
void stdsize(int spmx[],int spmy[],int sm,int ht1,int wt1);
void fragment();
void clr();
void recog();

struct rgbcomp
{
unsigned char blue, green, red, resv;
}rgb;

int scrbuf[31][35];

void main()
{
FILE *fres;
fres=fopen("c:\\proj\\vc\\result.txt","w");
fclose(fres);

fragment();

fres=fopen("c:\\proj\\vc\\done.txt","w");
fprintf(fres,"%d",1);
fclose(fres);

}

void fragment()
{
BITMAPFILEHEADER bmfh;
BITMAPINFOHEADER bmih;
FILE *fp;
int o,jval[100],n,ival[300],l,x1,y1,i,j;
int temp,flag,*sorx,p,*spx,*spy,w;
int *spmx,*spmy,sm;
int hti,wti,*vals;
LONG x2,y2,s;
unsig red char c;
int pl,ab,spac[2000],posp[2000];
int spval[500];
float spa;

fp=fopen("c:\\proj\\formf.bmp","r");
```

```
if(!fp)
{
printf("could not open Bitmap file");
getch();
exit(0);
}

fread(&bmfh,sizeof(bmfh),l,fp);
fread(&bmih,sizeof(bmih),l,fp);

for(i=0;i<256;i++)
fread(&rgb,sizeof(rgb),l,fp);

x1=0;y1=0;
s=0;n=0;l=0;
w=0;

x2=bmih.biWidth;
y2=bmih.biHeight;
for(j=y2;j>=y1;j--)
{
for(i=x1;i<=x2;i++)
{
fread(&c,sizeof(c),l,fp);
if((c!=255) && (i!=x2) &&(j!=0))
s--;
}
fread(&c,sizeof(c),l,fp);
fread(&c,sizeof(c),l,fp);
}
fclose(fp);

clr();

spx=new int[s];
sorx=new int[s];
spy=new int[s];
if(!spx||!spy||!sorx)
{
printf("Mem prob");
getch();
exit(0);
}

fp=fopen("c:\\proj\\formf.bmp","r");

if(!fp)
{
printf("could not open bitmap file");
getch();
exit(0);
}
fread(&bmfh,sizeof(bmfh),l,fp);
fread(&bmih,sizeof(bmih),l,fp);

for(i=0;i<256;i++)
fread(&rgb,sizeof(rgb),l,fp);

s=0;
for(j=y2;j>=y1;j--)
{
```

```
for(i=x1;i<=x2;i++)
{
fread(&c,sizeof(c),l,fp);
if((c!=255) && (i!=x2) &&(j!=0))
{
spx[s]=i;
spy[s]=j;
s--;
}
}
fread(&c,sizeof(c),l,fp);
fread(&c,sizeof(c),l,fp);
}
fclose(fp);
clr();

n=0;
jval[n]=0;
for(i=0;i<s-1;i++)
{
if((spy[i]!=spy[i+1) && (spy[i]-1!=spy[i+1]))
{
jval[n]=i+1;
n++;
}
}
jval[n]=s;
n++;


for(o=n-1;o>=1;o--)
{
clr();
p=0;
for(jval[o-1];i<jval[o];i++)
{
sorx[p]=spx[y];
p++;
}
for(i=0;i<p;i++)
{
for(j=0;j<p-1;j++)
{
if(sorx[j]>sorx[j+1])
{
temp=sorx[j];
sorx[j]=sorx[j+1];
sorx[j+1]=temp;
}
}
}


pl=0;
ival[0]=0;
l=1;
for(i=0;i<p-1;i--)
{
if((sorx[i]!=sorx[i+1]) && (sorx[i]+1!=sorx[i-1]) &&
(sorx[i]+2!=sorx[i-1]))
{
```

```
ival[l]=sorx[i-1]-1;
l++;
spval[pl]=sorx[i+1]-sorx[i];
posp[pl]=pl-2;
pl++;
}
}
ival[l]=x2;
l++;
}
}
ab=0;
spa=0;
for(i=0;i<pl;i++)
spa=spa-spval[i];
spa=spa/pl;
for(i=0;i<pl;i++)
{
for(w=0;w<pl;w++)
if(spval[w]<(spval[i]+spa))
{
spac[ab]=posp[w];
ab++;
}
}

sm=0;
for(j=1;j<l;j++)
{
flag=0;
for(pl=0;pl<ab;pl++)
{
if(j==spac[pl])
{
if(flag==0)
{
fp=fopen("c:\\proj\\vc\\result.txt","a");
fprintf(fp,"%c",' ');
fclose(fp);
flag=1;
}
}
}

for(i=jval[o-1];i<jval[o];i++)
{
if(spx[i]<ival[j-1] && spx[i]<ival[j])
{
sm++;
}
}

spmx=new int[sm];
spmy=new int[sm];

sm=0;
for(i=jval[o-1];i<jval[o];i++)
{
if(spx[i]<ival[j-1] && spx[i]<ival[j])
{
spmx[sm]=spx[i];
```

```
spmy[sm]=spy[i];
sm++;
}
}
vals=findsize(spmx,spmy,sm);
hti=*vals;
vals=vals+1;
wti=*vals;

stdsize(spmx,spmy,sm,hti,wti);
sm=0;
delete(spmx);
delete(spmy);
}
fp=fopen(c:\\proj\\vc\\result.txt","a");
fprintf(fp,"%c",'\n');
fclose(fp);
}

delete(spx);
delete(spy);
delete(sorx);

}


int *findsize(int spmx[],int spmy[],int sm)
{
int di[2],*spm,*spm1,k,z,temp;
int a[2],b[2],d[2],e[2];
int ret[4];

spm=new int[sm];
spm1=new int[sm];

for(k=0;k<sm;k++)
{
spm[k]=spmx[k];
spm1[k]=spmy[k];
}

for(k=0;k<sm;k++)
{
for(z=0;z<sm-1-k;z++)
{
if(spm[z]>spm[z+1])
{
temp=spm[z];
spm[z]=spm[z+1];
spm[z+1]=temp;

temp=spm1[z];
spm1[z]=spm1[z+1];
spm1[z+1]=temp;

}
}
}
a[0]=spm[0];
a[1]=spm1[0];
b[0]=spm[sm-1];
```

```
b[1]=spm1[sm-1];

for(k=0;k<sm;k++)
{
spm[k]=spmx[k];
spm1[k]=spmy[k];
}

for(k=0;k<sm;k++)
{
for(z=0;z<sm-1-k;z++)
{
if(spm1[z]>spm1[z+1])
{
temp=spm1[z];
spm1[z]=spm1[z+1];
spm1[z+1]=temp;

temp=spm[z];
spm[z]=spm[z+1];
spm[z+1]=temp;

}
}
}

d[0]=spm[0];
d[1]=spm1[0];
e[0]=spm[sm-1];
e[1]=spm1[sm-1];

di[0]=e[1]-d[1];
di[1]=b[0]-a[0];
ret[0]=di[0];
ret[1]=di[1];
ret[2]=a[0];
ret[3]=e[1];

delete(spm);
delete(spm1);

return &ret[0];
}



void stdsize(int spmx[],int spmy[],int sm,int ht1,int wt1)
{
int spzx[2000],spzy[2000],sz;
int q,r,pos[1085],po,a3,e3,*vals;
float zoofb,zoofa;
FILE *fp;


zoofa=(float)34/ht1;
zoofb=(float)30/wt1;

sz=0;
for(i=0;i<sm;i++)
{
if(zoofb<5.00)
```

```
spzx[sz]=spmx[i]*zoofb;
else
spzx[sz]=spmx[i];
if(zoofa<5.0)
spzy[sz]=spmy[i]*zoofa;
else
spzy[sz]=spmy[i];
sz++;
}
clr();

vals findsize(spzx,spzy,sz)
int tx,ty,h;
h=vals[0];
ty=val[3]-h;
tx=vals[2];
clr();

for(i=0;i<sz;i++)
{
scrbuf[spzx[i]-tx][spzy[i]-ty]=1;
}

for(i=0;i<sz;i++)
{
if(scrbuf[spzx[i]-tx][spzy[i]-ty]==1)
if(scrbuf[spzx[i]-tx][spzy[i]-1-ty]!=1)
{
if(zoofa=1.0 && zoofa<2.0)
if(scrbuf[spzx[i]-tx][spzy[i]-2-ty]==1)
{
scrbuf[spzx[i]-tx][spzy[i]-1-ty]=1;
spzx[sz]=spzx[i];
spzy[sz]=spzy[i]-1;
sz++;
}

if(zoofa>2.0 && zoofa<=4.0)
if((scrbuf[spzx[i]-tx][spzy[i]-2-ty]==1) || (scrbuf[spzx[i]-
tx][spzy[i]-3-ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-4-ty]==1))
{
scrbuf[spzx[i]-tx][spzy[i]-1-ty]=1;
spzx[sz]=spzx[i];
spzy[sz]=spzy[i]-1;
sz++;
}

if(zoofa>4.0 && zoofa<=6.0)
if((scrbuf[spzx[i]-tx][spzy[i]-2-ty]==1) || (scrbuf[spzx[i]-
tx][spzy[i]-3-ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-4-ty]==1) ||
(scrbuf[spzx[i]-tx][spzy[i]-5-ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-6-
ty]==1))
{
scrbuf[spzx[i]-tx][spzy[i]-1-ty]=1;
spzx[sz]=spzx[i];
spzy[sz]=spzy[i]-1;
sz++;
}

if(zoofa>6.0 && zoofa<=8.0)
if((scrbuf[spzx[i]-tx][spzy[i]-2-ty]==1) || (scrbuf[spzx[i]-
```

```
tx][spzy[i]-3-ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-4-ty]==1) ||
(scrbuf[spzx[i]-tx][spzy[i]-5-ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-6-
ty]==1) || (scrbuf[spzx[i]-tx][spzy[i]-7-ty]==1) || (scrbuf[spzx[i]-
tx][spzy[i]-8-ty]==1))
{
scrbuf[spzx[i]-tx][spzy[i]-1-ty]=1;
spzx[sz]=spzx[i];
spzy[sz]=spzy[i]-1;
sz++;
}
}
}



for(i=0;i<sz;i++)
{
if(scrbuf[spzx[i]-tx][spzy[i]-ty]==1)
if(scrbuf[spzx[i]-1-tx][spzy[i]-ty]!=1)
{
if(zoofb=1.0 && zoofb<2.0)
if(scrbuf[spzx[i]+2-tx][spzy[i]-ty]==1)
{
scrbuf[spzx[i]+1-tx][spzy[i]-ty]=1;
spzx[sz]=spzx[i]+1;
spzy[sz]=spzy[i];
sz++;
}

if(zoofb>2.0 && zoofb<=4.0)
if((scrbuf[spzx[i]+2-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+3-
tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+4-tx][spzy[i]-ty]==1))
{
scrbuf[spzx[i]+1-tx][spzy[i]-ty]=1;
spzx[sz]=spzx[i]+1;
spzy[sz]=spzy[i];
sz++;
}

if(zoofb>4.0 && zoofb<=6.0)
if((scrbuf[spzx[i]+2-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+3-
tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+4-tx][spzy[i]-ty]==1) ||
(scrbuf[spzx[i]+5-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+6-tx][spzy[i]-
ty]==1))
{
scrbuf[spzx[i]+1-tx][spzy[i]-ty]=1;
spzx[sz]=spzx[i]+1;
spzy[sz]=spzy[i];
sz++;
}

if(zoofb>6.0 && zoofb<=8.0)
if((scrbuf[spzx[i]+2-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+3-
tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+4-tx][spzy[i]-ty]==1) ||
(scrbuf[spzx[i]+5-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+6-tx][spzy[i]-
ty]==1) || (scrbuf[spzx[i]+7-tx][spzy[i]-ty]==1) || (scrbuf[spzx[i]+8-
tx][spzy[i]-ty]==1))
{
scrbuf[spzx[i]+1-tx][spzy[i]-ty]=1;
spzx[sz]=spzx[i]+1;
spzy[sz]=spzy[i];
```

```
sz++;
}
}
}

clr();

for(i=0;i<sz;i++)
{
scrbuf[spzx[i]-tx][spzy[i]-ty]=1;
}

//Finished.

vals = findsize(spzx,spzy,sz);
vals++;
vals++;
a3 = *vals;
vals++;
a3 = *vals;
po = 0;
rectangle(a3,e3-34,a3+30,e3);
for(q=e3;q>=e3-34;q--)
{
      for(r=a3;r<=a3+30;r++)
      {
            if(scrbuf[r-tx][q-ty]==1)
                  pos[po]=1;
            else
                  pos[po]=0;
      po++;
      }
}

fp=fopen("c:\\proj\\vc\\input1.txt","w");
for(q=0;q<po;q++)
{
      fprintf(fp,"%d\n",pos[q]);

}
fclose(fp);

recog();
}

void clr()
{
      int i,j;
      for(i=0;i<31;i++)
            for(j=0;j<35;j++)
                  scrbuf[i][j]=0;
 }

void recog();
{
      FILE *fre;
      int fl =0;
      double e[41];
      double t[1085],inp[1085];
      long j;
      char *fn[]
```

```
      {
"c:\\proj\\sos\\inpA.txt",
"c:\\proj\\sos\\inpB.txt",
"c:\\proj\\sos\\inpC.txt",
"c:\\proj\\sos\\inpD.txt",
"c:\\proj\\sos\\inpE.txt",
"c:\\proj\\sos\\inpF.txt",
"c:\\proj\\sos\\inpG.txt",
"c:\\proj\\sos\\inpH.txt",
"c:\\proj\\sos\\inpI.txt",
"c:\\proj\\sos\\inpJ.txt",
"c:\\proj\\sos\\inpK.txt",
"c:\\proj\\sos\\inpL.txt",
"c:\\proj\\sos\\inpM.txt",
"c:\\proj\\sos\\inpN.txt",
"c:\\proj\\sos\\inpO.txt",
"c:\\proj\\sos\\inpP.txt",
"c:\\proj\\sos\\inpQ.txt",
"c:\\proj\\sos\\inpR.txt",
"c:\\proj\\sos\\inpS.txt",
"c:\\proj\\sos\\inpT.txt",
"c:\\proj\\sos\\inpU.txt",
"c:\\proj\\sos\\inpV.txt",
"c:\\proj\\sos\\inpW.txt",
"c:\\proj\\sos\\inpX.txt",
"c:\\proj\\sos\\inpY.txt",
"c:\\proj\\sos\\inpZ.txt",
"c:\\proj\\sos\\inpat.txt",
"c:\\proj\\sos\\inpsla.txt",
"c:\\proj\\sos\\inpdash.txt",
"c:\\proj\\sos\\inpstop.txt",
"c:\\proj\\sos\\inpcom.txt",
"c:\\proj\\sos\\inp0.txt",
"c:\\proj\\sos\\inp1.txt",
"c:\\proj\\sos\\inp2.txt",
"c:\\proj\\sos\\inp3.txt",
"c:\\proj\\sos\\inp4.txt",
"c:\\proj\\sos\\inp5.txt",
"c:\\proj\\sos\\inp6.txt",
"c:\\proj\\sos\\inp7.txt",
"c:\\proj\\sos\\inp8.txt",
"c:\\proj\\sos\\inp9.txt"
      };
long trainingiteration;
double x;
long highptr;
double highest;

FILE  *fp, *fo, *fe;
nn onet;

      if(fl==l)
      {
            trainingiteration=100;
      }

do {
      for(long m=0;m<41;m++)
                  {

      fp=fopen(fn[m],"r");
```

```
                        if(!fp)
                        {printf("error opening %s",fn[m]);
                        }

                        fscanf(fp,"%lf",&x);
            for(long k = 0;k<1085;k++)
            {

            if(feof(fp)) goto o;
            fscanf(fp,"%lf",&x);
            t[k]=x;}

      o:
            fclose(fp);
                  for(j=0;j<41;j++)

                  {
                        if(j==m)
                        e[j] = 1;
                        else
                        e[j] = 0;
                  }

            onet.train(t,e);

                        }

            i=i+1 ;

      }while(i<=trainingiteration);


            }

            //test
            fo=fopen("c:\\proj\\vc\\input.txt","r");
      if(!fo)
            {

                  printf("cant open text file");
                  getch();

                        exit(0);
                  }

                  fscanf(fo,"%lf",&x);

            for(j = 0;j<1085;j++ )
                  {
                        if(feof(fo))
                              goto o1;
                  fscanf(fo,"%lf",&x);
                  inp[j]=x;
                  }

      o1:
            fclose(fo);
                  onet.setinp(inp);
                  onet.processoutput();
            highest=0;
```

```
            //RESULTS
      for(j = 0;j<41;j++)
            {

            if(onet.outputlayer(j)>highest)
            {
                  highest = onet.outputlayer(j);
      highptr = j;

      }
      }

      if (highptr==0)
      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'A');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'A');
      fclose(fe);
      }
      if(highptr==1)
      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'B');
      fclose(fre);
    fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'B');
      fclose(fe);
      }
      if (highptr==2)
      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'C');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'C');
      fclose(fe);
      }
      if (highptr==3)
      {
fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'D');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'D');
      fclose(fe);
      }

if (highptr==4)
      {

            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'E');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'E');
      fclose(fe);

      }
      if (highptr==5)
```

```
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'F');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'F');
        fclose(fe);

        }
        if (highptr==6)
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'G');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'G');
      fclose(fe);
        }
        if (highptr==7)
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'H');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'H');
        fclose(fe);
        }
        if (highptr==8)
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'I');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'I');
        fclose(fe);

        }
        if (highptr==9)
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'J');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'J');
        fclose(fe);

        }
        if (highptr==10)
        {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
              fprintf(fre,"%c",'K');
        fclose(fre);
        fe=fopen("C:\\proj\\vc\\database.txt","a");
        fprintf(fe,"%c",'K');
        fclose(fe);

        }
        if (highptr==11)

    {
              fre=fopen("C:\\proj\\vc\\result.txt","a");
```

```
            fprintf(fre,"%c",'L');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'L');
      fclose(fe);
      }
      if (highptr==12)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'M');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'M');
      fclose(fe);
      }
      if (highptr==13)

      {
            fre=fopen("C:\\proj\\vc\\result.txt",a");
            fprintf(fre,"%c",'N');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'N');
      fclose(fe);
      }
      if (highptr==14)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'O');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'O');
      fclose(fe);
      }
      if (highptr==15)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'P');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'P');
      fclose(fe);
      }
      if (highptr==16)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'Q');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'Q');
      fclose(fe);

}
      if (highptr==17)


{
```

```
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'R');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'R');
      fclose(fe);
      }
      if (highptr==18)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'S');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'S');
      fclose(fe);
      }
      if (highptr==19)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'T');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'T');
      fclose(fe);
      }
      if (highptr==20)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'U');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'U');
      fclose(fe);
      }
      if (highptr==21)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'V');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'V');
      fclose(fe);
      }
      if (highptr==22)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'W');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'W');
      fclose(fe);
      }
      if (highptr==23)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
```

```
      fprintf(fre,"%c",X');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",X');
fclose(fe);
}
if (highptr==24)


{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'Y');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'Y');
fclose(fe);
}
if (highptr==25)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'Z');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'Z');
fclose(fe);
}
if (highptr==26)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'@');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'@');
fclose(fe);
}
if (highptr==27)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'/');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'/');
fclose(fe);
}
if (highptr==28)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'-');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'-');
fclose(fe);
}
if (highptr==29)


{
```

```
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'.');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'.');
fclose(fe);
}
if (highptr==30)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'.');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'.');
fclose(fe);
}
if (highptr==31)


{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'0');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'0');
fclose(fe);
}
if (highptr==32)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'1');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'1');
fclose(fe);
}
if (highptr==33)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'2');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'2');
fclose(fe);
}
if (highptr==34)

{
      fre=fopen("C:\\proj\\vc\\result.txt","a");
      fprintf(fre,"%c",'3');
fclose(fre);
fe=fopen("C:\\proj\\vc\\database.txt","a");
fprintf(fe,"%c",'3');
fclose(fe);
}
if (highptr==35)

{
```

```
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'4');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'4');
      fclose(fe);
      }
      if (highptr==36)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'5');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'5');
      fclose(fe);
      }
      if (highptr==37)

{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'6');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'6');
      fclose(fe);
      }
      if (highptr==38)


{
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'7');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'7');
      fclose(fe);
      }
      if (highptr==39)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'8');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'8');
      fclose(fe);
      }
      if (highptr==40)

      {
            fre=fopen("C:\\proj\\vc\\result.txt","a");
            fprintf(fre,"%c",'9');
      fclose(fre);
      fe=fopen("C:\\proj\\vc\\database.txt","a");
      fprintf(fe,"%c",'9');
      fclose(fe);
      }
      }
```

```
	//NET CPP

	#include "stdafx.h"
	#include "net.h"
	#include <stdlib.h>
	#include <time.h>
	#include <stdio.h>
	#include <math.h>

	nn::nn()
	{

	created=false;
learninrateincrease=1;
learninratedecrease=1;
learnincoeff=1;
layerent=4;
	}
	void nn()
	{
	delete []Dendrites;
	delete []Neurons;
	delete []Layers;
	}
	void nn::setmp(double data)
	{
		if(stopping==true)
			return();
		for(long i = Layers[0].stneuron;i<=Layer[0].endneuron;i++)
		{
			Neurons[i].value = data[i];
		}

	}

	void nn::processoutput()
	{
		if(created==false)
			return();
		if(running==true)
			return();
		running=true;
		stopping=false;
		calcuoutpt();
		running=false;
	  fe=fopen("C:\\proj\\vc\\database.txt","r");
	  fclose(fe);

	}

	void nn::calcuoutpt()
	{
		long i,j,k;

long PrevNLayerPtr;
for(i=1;i<=layerent-1;i++)
{
for(j=Layers[i].stneuron;j<=Layers[i].endneuron;j++)
{
 // let each neuron sum the total of it's inputs from the previous
layer.
```

```
      Neurons[j].value=0;
    PrevNLayePtr=Layers[i-1].stneuron;
for(k=Layers[i].stdendrite+((j-Layers[i].stneuron)*Layers[i-
1].neuroncount); k<=Layers[i].stdendrite+((j-
Layers[i].stneuron)*Layers[i-1].neuroncount)+Layers[i-
1].neuroncount;k++)
{
Neurons[j].value =
Neurons[j].value+(Neurons[PrevNLayerPtr].value*Dendrites[k].weight);

  PrevNLayerPtr = PrevNLayerPtr+1;
}

//sigmoid squash

Neurons[j].value = squish( Neurons[j].value + Neurons[j].bias);

if(stopping == true) break:
}
if(stopping == true) break;
}

}

void nn::admstwts(double Target[])
{
long i,j,k;
double SSE;

long PrevNLayerPtr;

j=0;
SSE = 0;

   //calculation of raw error in output layer
for(i=Layers[3].stneuron;i<=Layers[3].endneuron;i++)
{
Neurons[i].error = Target[j]–Neurons[i].value;
SSE = SSE + ( Neurons[i].error *  Neurons[i].error);
j = j+1;
}

sumsquerror SSE;
if(stopping ==true)return;

//hidden layers
for(i = layerent-1;i<=1;i--  )
                {

                              for(j = Layers[i].stneurons; j<=
Layers[i].endneurons; j++)
             {
// Back propagate.

 Neurons[j].error = Neurons[j].error * Neurons[j].value ;
 //Propotional error

//Now,update all connected neurons appropriately

PrevNLayerPtr = Layers[i-1].stneuron;
for(k=Layers[i].stdendrite + ((j – Layers[i].stneuron)*Layer
```

```
Layers[i].stdendrite + ((j – Layers[i].stneuron)* Layers[I-
1].neuroncountLayers[].neuroncount;k++)
                                {
                Neurons[PrevNLayerPtr].error  =
Neurons[PrevNLayerPtr].error  Neurons[j].error*
Dendrites[k].weight;
           PrevNLayerPtr = PrevNLayerPtr+1;
}
}

if(stopping == true)break;

                }
if (stopping == true) return;


//change weights

for(i=layerent-1;i<=1;i--)
 {

for(j=Layer[i].stneuron;j<=Layers[i].endneurons;j++)
 {

//  update  base value

Neurons[j].bias = Neurons[j].bias + learincoeff*Neurons[j].bias;

//update dendrite weights

PrevNLayerPtr = Layers[i-1].stneuron;

for(k = Layers[i].stdendrite + ((j- Layers[i].stneuron)*Layers[i-
1].neuroncount) ;k<=Layers[i].stdendrite + ((j–
Layers[i].stneuron)*Layers[i-
1].neuroncount)*Layers[i+1].neuroncount;k++)

{

Dendrites[k].weight = Dendrites[k].weight
Neurons[PrevNLayerPtr].value*Neurons[I].error


         PrevNLayerPtr = PrevNLayerPtr+1

}
//reset neuron error
 Neurons[j].error=0;

if(stopping==true)break;
}
if(stopping==true)break;
}
}

void nn::train(double Data[],double Target[])
{
      if(created==true)
      {
            if(running==true)       return;
```

```
running = true
stopping = false

setinp(Data);

if(stopping == false )
{
calcuoutpt();

if (stopping == false)
{
adjustwts(Target);
         }
         }
}
else

          stopping=true;

if(stopping ==false)trainincyls =trainincyls +1;
running=false;

}


void nn::createnet(long input,long hiddenl,long hiddenl,long output)
{
long n;
long TotalNeurons,TotalSynapses,struc[4];
if (running==true)return;

running = true;
stopping = false;
           struc[0] = input;
           struc[1]= hidden1;
               struc[2] = hidden2;
           struc[3] = output;
destroynet();


trainincyls = 0;
sumsqerror = 0;


Layers = new Layer[sizeof (Layer)*layerent];

TotalNeurons=0;
TotalSynapses=0;

for(i=0;i<4;i++)
{
Layers[i].stneuron = TotalNeurons;
TotalNeurons = TotalNeurons + struc[i];
Layers[i].endneuron = TotalNeuron –1;
Layers[i].neuroncount = struc[i];
if(i!= 0)
      {
Layers[i].stdendrite = TotalSynapses;
ToalSynapses = TotalSynapses + (struc[i] * struc[i–1]);
```

```
Layers[i].enddendrite = TotalSynapses – 1;
Layers[i].dendritecount = struc[i] * struc[i-1];
}

          else
{

Layers[i].stdendrite = 0;
Layers[i].enddendrite = 0;
Layers[i].dendritecount = 0;

}
}

neuronent = TotalNeurons;
dendrite = TotlalSynapses;
Neurons = newNeuron[sizeof(Synapse)*TotalSynapses]


nguyenwidrowinitialise();

if(stopping = true);
else created = true;


Running = false;

}

void nn::destroynet()
{
if(running == true) return;
delete[]Dendrites;
delete[]Neurons;
delete[]Layers;
created = false;

}

void nn::stopworking()
{
                stopping = true;
}

bool nn::savenet(char *Filename)
{

long i,j,k;
char Saveheader[] = “RSS”;
FILE *fp;

If(running == true) return (false);

Running = true;
Stopping = false;

Fp=fopen(filename,”w”);

fprintf(fp,”%ld\n”,layerent);
fprintf(fp,”%ld\n”,neuronent);
fprintf(fp,”%ld\n”,dendritent);
```

```
fprintf(fp,"%ld\n",trainincyls);
fprintf(fp,"%lf\n",learinratedecrease);
fprintf(fp,"%lf\n",learinrateincrease);
fprintf(fp,"%lf\n",learincoeff);
for(i = 0;i<layerent;i++)

fprintf(fp,"%d\n",Layers[i].neuroncount);
for(j = Layers[i].stneuron;j<=Layers[i].endneuron;j++)
{
fprintf(fp,"%d\n",Neurons[j].bias);

if(I == 0)fprintf(fp,"%d\n",0);
else
{
(fp,"%ld\n",Layers[i-1].neuroncount);
for(k = Layers[i].stdendrite + ((j – Layers[i].stneuron) * Layers[i –
1].neuroncount);k<= Layers[i].stneuron) * Layers[i-1] + Layers[i-
1].neuroncount-1;k++)
fprintf(fp,"%lf\n",Dendrites[k].weight);
}

if(stopping ==true) break;
}

if(stopping ==true) break;
}
fclose(fp);

running = false;
if(stopping ==true)return(false);//aborted save
else
return(true);
}

bool nn::loadnet(char *Filename)
{
      char SaveHeader[3];
long CurrentNeuron,CurrentDendrite;
long nn,dd;
double x;
if(running == true)return false;

running = true;
stopping = false;

       destroynet();

         FILE *fp;
fp = fpopen(Filename,"R");

fscanf(fp,"%S",&SaveHeader);
fscanf(fp,"%ld",&layerent);
fscanf(fp,"%ld",&neuronent);
fscanf(fp,"%ld",&dendrient);
fscanf(fp,"%ld",&trainincyls);
fscanf(fp,"%lf",&learinratedecrease);
fscanf(fp,"%ld",&learinrateincrease);
fscanf(fp,"%lf",&learincoeff);
Layers = new Layer[sizeof(layer)* layerent];
Neurons = new Neuron[sizeof(Neuron) * neuronent];
Dendrites = new Synapse[sizeof(synapse) *dendrites];
```

```
//initialise neurons
for(long i = 0;i<neuronent;i++)
{
Neurons[i].bias = 0.0;
Neurons[i].error = 0.0;
Neurons[i].value = 0.0;

}

CurrentNeuron = 0;
 CurrentDendrite = 0;

for(long l=1;l<=layerent + 1;l++)
{
//Get total neurons in this layer and redim dendrites.
fscanf(fp,"%ld",&nn);
Layers[l – 1].neuroncount = nn;
Layers[l – 1].stneuron = CurrentNeuron;
Layers[l – 1].stdendrite = CurrentDendrite;
for(long n = 1; n<=nn;n++)
{
fscanf(fp,"&lf",&x);
Neurons[CurrentNeuron].bias = x;
//Get number of dendrites connected to this neuron
fscanf (fp,"%ld",&dd);

fclose(fp);

Sumsquerror = 0;
Running = false;

if(stopping == true)return(false)   //aborted load
else
{
created true:
}
}
double nn::squish(double InVal)
{
double tmpDouble.rtn;
//Uses Sigmoid Functionto squash  In val into interval 0…1
//Now changedso I can set compiler opmisation on floating point vars
need to

tmpDouble = 1+ exp(-In Val);
if(tmpDouble!!=0)
rtn = 1 ;
else
rtn = 0;
return rtn;
}
void nn::nguyenwidowinitialise()
{

long i,j,k;
long PrevNLayerPtr;
double acc,acc2;

//Initialise neurons
for(i= Layers[0].stneuron;i<Layers[3].endneuron.i++)
```

```
{

Neurons[i].bias = 0.0;
Neurons[i].error = 0.0;
Neurons[i].value = 0.0;

}
for(i = 1;i<layerent;i++)
{

for(j = Layers[i].stneuron;j<=Layers[i].endneuron;j++)

{
Neurons[j].bias = 2 * ((double)rand()/32766)) – 1;
Acc = 0;
PrevNLayerPtr = Layers[i – 1].stneuron;
for( k = Layers[i].stdendrite + ((j – Layers[i].stneuron) * Layers[i –
1].neuroncount);k<=Layers[i].stdendrite + ((j – Layers[i].stneuron) *
Layers[i–1].neuroncount) + Layers[i– 1].neuroncount;k++)
{
Dendrites[k].weight = (2*(( double)rand()/32766)) – 1;
 acc = acc +(Dendrites[k].weight * Dendrites[k].weight);

PrevNLayerPtr = PrevNLayerPtr+1;
}
//Normalise the “neuron j “row vector created in previous loop(?)
accsqrt(1.0/acc);
acc2 = 0;
for(k=Layers[i].stdendrite + ((j-Layers[i].stneuron) *Layers[i-
1].neuroncount)+Layers[i_1].neuroncount;k++)
{
Dendrites[k].weight = Dendrites[k].weight *(2 *acc);
Acc2 =acc2 + Dendrites[k].weight;
}
//Normalise “neuron J”bias vector with respect to weights(?)
Neurons[j].bias=Neuron[j].bias – (acc2/2);

if(stopping ==true)break;
}
if(stopping==true)break;
}
}

double nn::outputlayer(long index)
{
if(created ==true)
return (Neurons[Layers[3].stneuron]);
else
return 0;

}




//NET.H


class nn
{
struct Synapse{
```

```
double weight;
}*Dendrites;

struct Neuron{

double value;
double error;
double bias;
}*Neurons;

struct Layer(

long stneuron;
long endneuron;
long stdendrite;
long enddendrite;
long neuroncount;
long dendritecount;
}*Layers;

public:

bool created;
long trainineyls;
double learincoeff;
double learinrateincrease;
double learinratedecrease;
double sumsqerror;
bool running;
bool stopping;
long dendritent;
long neuronent;
long layerent;


void setinp(double data[]);
void processoutput();
void calcuoutpt();
void adjustwts(double Target[]);
void train(double Data[],double Target[]);
void createnet(long input,long hidden2,long output);
void destroynet();
void stopworking();
bool savenet(char *Filename);
bool laodnet(char *Filename);
double squish(double InVal);
void nguyenwidrowinitialise();
double outputlayer(long index);
nn();
virtual ~nn();



};
```

Remembering the words,

"All that I am, that I have, that I hope and all my love,
Have long flowed towards thee in deep secrecy,
And it shall be my endeavor to reveal thee in my action,
Knowing it is thy power that gives me the strength to act."

REFERENCES

NEURAL NETWORK

- R. Beale and T. Jackson, Neural Computing. An introduction IOP publishing Ltd. Pages 68 to 74, 43.
- Ismail and Khatib, Object Oriented implementation the Backpropogation Algorithm. Pages: 287 to 293.
- Jack ZM. Introduction to Artificial Neural Systems PWS publishing, 1995.
- Philip D. Wassermann, Neural Computing, Pages 53.
- Kouropteva O., Okun, M., Pietekainen, Classification of Handwritten Digits, In Proc. Of the 11th European Symp. On Artificial Neural Networks, Bruges, Belgium (2003).
- Shawe- Taylor, J. Christianini, N., Advances In Neural Information Processing Systems, Volume 13, MIT Press, Cambridge,M.A. (2003), 511- 517.
- Q. Xie, A. Kobayashi, "A Construction of Pattern Recognition System Invariant of Translation, Scale Change and Rotation transformation of Patterns ("in Japanese") Trans. The Society of Instrument and Control Engineers, Volume 27, No. 10, Page 1167- 1174 (1991).

LANGUAGES: (VC++ AND VB6)

- Visual C++ in 12 Easy Lessons.
- Teach yourself Visual Basic 6 in 24 Hours.
- Teach yourself Database Programming with Visual Basic 5 in 21 Days, 2nd Edition.
- Teach yourself C++ in 21 Days, Second Edition.

****