CHAPTER 3

INTRODUCTION TO SOFTWARE
TOOLS USED
3.1 OVERVIEW

The implementation of an artificial neural network system requires the understanding of application because some artificial neural network tools are application specific; therefore the decision about type, scope and desired output of application matters lot. There are two ways of implementing neural network. One way is to program a network i.e. to develop our own software code using programming language like visual C++ (image processing), C++ (object abstraction) and MATLAB (all application). The other way is to use a neural network simulator for creating the network. A simulator is a program that is designed to perform specific task and helps in creating, validating and querying the artificial neural network. It gives a free hand to user to decide various parameters of the network that are chosen to design, e.g., the type of input layer, output layer and hidden layer(s); the learning rate and the momentum etc. These simulators generally have user-friendly graphical user interface.

The range of simulator varies from application of system, which are aimed at education, and training, which may implement relatively simple networks and run quite slowly, to professional tool with a high degree of complexity and functionality, which can be much harder to operate. Some of these systems generally implement only very specific type of networks, whereas others offer a wide range of networks and architectures. Again the choice of simulator depends on the application, architecture of neural network supported by simulator and the type of output needed. There are many neural network simulators available. Some of them are used commercially while others are freeware. Some of the neural network simulators are

3.1.1. Papnet

It is a commercial neural network based computer program for assisted screening of Pap (cervical) smears. A Pap smear test examines cells taken from
the uterine cervix for signs of precancerous changes. A properly taken and analyzed Pap smear can detect very early precancerous changes. Papnet assisted reviews of cervical smears result in a more accurate screening process than the current practice, leading to an earlier and more effective detection of pre cancerous and cancerous cells in the cervix.

3.1.2. NNDT (Neural Network Development Tool)

It is a tool for neural network training (MS Windows). The user interface is developed with the MS Visual Basic 3.0 professional edition. DLL routines are used for most of the mathematics. Algorithms implemented are multilayer perceptron (MLP) networks of feed forward and recurrent types. The MLP networks are trained with the Levenberg-Marquardt method. NNDT includes a routine for graphical presentation of output signal, node activations, residuals and weights during run. The interface also provides facilities for examination of node activation and weights as well as modification of weights.

3.1.3. GENESIS (General Neural Simulation System)

It is a general purpose simulation platform which was developed to support the simulation of neural system ranging from complex models of single neuron to simulation of large networks made up of more abstract neuronal components. Most current GENESIS application involves realistic simulations of biological neural systems. The GENESIS and its graphical front end XODUS are written in C and runs on most UNIX platforms.

3.1.4. Stuttgart Neural Network

It is a neural network simulator developed for UNIX workstations; developed for parallel and distributed high performance system at the University of Stuttgart. The goal of the SNNS project was to create an efficient and flexible simulation environment for research and application of neural nets. The SNNS simulator consists of a simulator kernel written in C and graphical interface under X11R4 and X11R5. It includes, Backpropagation for feedforward networks (online), Backpropagation with momentum term and flat spot elimination, batch BP), Counterpropagation, Quickprop, Backpropagation 1, RProp, Generalized radial basis function (RBF), ART1, ART2, ARTMAP, Cascade Correlation,
Recurrent Cascade Correlation, Dynamic LVQ, Backpropagation through time (for recurrent networks) Quickprop through time (for recurrent network), Self Organizing maps (Kohonen maps), TDNN (time delay networks) with backpropagation, Jordan networks, Elman networks and extended hierarchical Elman network, Associative Memory.

3.1.5. PDP++

It is a neural network simulation tool written in C++. It represents the next generation of the PDP software released with the McClelland and Rumelhart “Exploration in parallel distributed processing handbook”, MIT Press, 1987. The software runs on UNIX workstations under XWindows. It has full GUI (Interviews), real-time viewer, data viewer, extendable object oriented design, CSS scripting language with source level debugger, GUI macro recording. Algorithms supported are: Feedforward and several recurrent BP, Boltzmann machine, Hopfield, Mean Field, Interactive activation and competition, continuous stochastic networks.

3.1.6. Brain

The Brain is an advanced neural network simulator. It is based on the backpropagation algorithm.

3.1.7. DartNet

It is Macintosh- based backpropagation simulator. It makes use of the Mac's graphical interface, and provides a number of tools for building, editing, training, testing and examining networks using Backpropagation Algorithm.

3.1.8 Easy Neural Network

It is a neural network simulator for the Windows operating system. The multilayered Perceptron model is implemented by it. It has a user friendly graphic user interface (GUI).

In this dissertation, for the purpose of pre processing of the collected data and for creating the Artificial Neural Networks, the following two softwares have been used:

a) MATLAB
Both of these have features and capabilities, which are unique to them. An attempt has been made to describe the features of both of these.

3.2 MATLAB 6.0

MATLAB has extensive facilities for displaying vectors and matrices as graphs. It includes high-level functions for presentation of graphics. It is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

MATLAB integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. The name MATLAB stands for matrix laboratory. Typical uses include:

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This helps in solving many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn
and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

3.2.1 Desktop Layout

When MATLAB is started, the first thing one can see is the MATLAB desktop, consisting of tools (GUIs or graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The first time when MATLAB is opened, the desktop appears with the default layout, as shown in figure 3.1. The default layout consists of three basic windows:
Command Window:
This is the main window. It is characterized by the MATLAB command prompt '>>'. All commands including those for running user-written programs are typed in this window at the MATLAB prompt.

Workspace:
This subwindow lists all the variables that one has generated so far and shows their types and size.

Command History
All commands typed on the MATLAB prompt in the command window get recorded, even across multiple sessions. One can select a command from this window and execute it in the command window. A set of commands can be selected from this window for creating an M-file.

Current Directory
This is where all the files from the current directory are listed. File navigation can be done here.

Apart from desktop, there are two more windows in MATLAB

Graphics Window:
The output of all graphics commands typed in the command window are flushed to graphics or Figure window. The user can create as many figure windows, as the system memory will allow.

Edit Window:
This is where one can write, edit, create, and save programs in files called 'M-files'. All the user-written script or function files are saved in this window.

3.2.2 File Types
MATLAB has three types of files for storing information:

- **M-files** are standard ASCII text files, with .m extension to the filename. There are two types of these files: script files and function files. All built-in functions in MATLAB are M-files. Some built-in functions are provided with source code in a readable M-file so that they can be copied and modified.
• **Mat-files** are binary data files with a .mat extension to the filename. By default, MATLAB compresses the data when saving a MAT-file, thereby reducing the storage space required. When one loads the MAT-file, MATLAB automatically uncompressed the data.

• **Mex-files** are MATLAB-callable C and Fortran programs, with a .mex extension to the filename. One can call C or Fortran subroutines from MATLAB as if they were built-in functions. MATLAB callable C and Fortran programs are referred to as MEX-files. MEX-files are dynamically linked subroutines that the MATLAB interpreter can automatically load and execute.

### 3.2.2.1 Script-files

A script file is an external file that contains a sequence of MATLAB statements. By typing the filename, one can obtain subsequent MATLAB input from the file. Script files have a filename extension of .m and are often called M-files. Scripts are the simplest kind of M-file. They are useful for automating blocks of MATLAB commands, such as computations that have to be performed repeatedly from the command line. Scripts can operate on existing data in the workspace, or they can create new data on which to operate. Although scripts do not return output arguments, any variables that they create, remain in the workspace and are available for use in further computations. In addition, scripts can produce graphical output using commands like plot. Scripts can contain any series of MATLAB statements. They require no declarations or begin/end delimiters. Like any M-file, scripts can contain comments. Any text following a percent sign (%) on a given line is comment text. Comments can appear on lines by themselves, or are appended at the end of any executable line.

### 3.2.2.2 Function Files

One can add new functions to the MATLAB vocabulary by expressing them in terms of existing functions. The existing commands and functions that compose the new function reside in a text file called an M-file. The name of an M-file begins with an alphabetic character and has a filename extension of .m. The M-file name, less its extension, is what MATLAB searches for when one tries to
use the script or function. Function file has a function definition line on the top that defines input and output explicitly.

3.2.3 Excel Link

Excel Link is a software add-in that integrates Microsoft Excel and MATLAB® in a Microsoft Windows-based computing environment. By connecting Excel and MATLAB, one can access the numerical, computational, and graphical power of MATLAB from Excel worksheet and macro programming tools. Excel Link permits exchange and synchronization of data between the two environments. The appearance of Excel link toolbar on Excel worksheet is as shown in figure 3.2.

![Figure 3.2: Excel Link toolbar on Excel Worksheet](image)

Excel Link communicates between the Excel workspace and the MATLAB workspace. It positions Excel as a front end to MATLAB. One can use Excel Link functions from an Excel worksheet or macro, and never have to leave the Excel environment. With a small number of functions to manage the link and manipulate data, Excel Link is powerful in its simplicity.
3.2.3.1 Functions

With Excel Link, Microsoft Excel becomes an easy-to-use data-storage and application-development front end for MATLAB, which is a powerful computational and graphical processor. Excel Link provides functions to manage the link and to manipulate data without leaving the Excel environment to invoke functions as worksheet cell formulas or in macros.

3.2.3.2 Link Management Functions

Excel Link provides four link management functions to initialize, start, and stop MATLAB and Excel Link. The Link Management functions are listed in table 3.1.

Any link management function can be invoked except `matlabinit` as a worksheet cell formula or in a macro. The `matlabinit` can be invoked from the Excel Tools Macro menu or in a macro subroutine. `MLAutoStart` is used to toggle automatic startup. If Excel Link is installed and configured according to the default instructions, Excel Link and MATLAB automatically start every time one starts Excel. To choose manual startup, `matlabinit` to initialize Excel Link and start MATLAB is used. `MLClose` is used to stop MATLAB without stopping Excel, and `MLOpen` or `matlabinit` is used to restart MATLAB in the same Excel session.

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>matlabinit</code></td>
<td>Initialize Excel Link and start MATLAB process.</td>
</tr>
<tr>
<td><code>MLAutoStart</code></td>
<td>Automatically start MATLAB process.</td>
</tr>
<tr>
<td><code>MLOpen</code></td>
<td>Start MATLAB process.</td>
</tr>
<tr>
<td><code>MLClose</code></td>
<td>Terminate MATLAB process.</td>
</tr>
</tbody>
</table>

Table 3.1: Link Management Functions

3.2.3.3 Data Management Functions

Excel Link provides nine data management functions to copy data between Excel and MATLAB and to execute MATLAB commands from Excel. The Data Management Functions are listed in table 3.2.

Except `MLGetVar` and `MLPutVar`, all other data management functions can be invoked as a worksheet cell formula or in a macro. `MLGetVar` and
MLPutVar can be invoked only in a macro. MLAppendMatrix, MLPutMatrix, and MLPutVar can be used to copy data from Excel to MATLAB. MLEvalString is used to execute MATLAB commands from Excel. MLDeleteMatrix is used to delete a MATLAB variable. Use matlabfcn, matlabsub, MLGetMatrix and MLGetVar to copy data from MATLAB to Excel.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>matlabfcn</td>
<td>Evaluate MATLAB command given Excel data</td>
</tr>
<tr>
<td>matlabsub</td>
<td>Evaluate MATLAB command given Excel data and designate output location.</td>
</tr>
<tr>
<td>MLAppendMatrix</td>
<td>Create or append MATLAB matrix with data from Excel worksheet.</td>
</tr>
<tr>
<td>MLDeleteMatrix</td>
<td>Delete MATLAB Matrix</td>
</tr>
<tr>
<td>MLEvalString</td>
<td>Evaluate command in MATLAB.</td>
</tr>
<tr>
<td>MLGetMatrix</td>
<td>Write contents of MATLAB matrix in Excel worksheet.</td>
</tr>
<tr>
<td>MLGetVar</td>
<td>Write contents of MATLAB matrix in Excel VBA variable.</td>
</tr>
<tr>
<td>MLPutMatrix</td>
<td>Create or overwrite MATLAB matrix with data from Excel worksheet.</td>
</tr>
<tr>
<td>MLPutVar</td>
<td>Create or overwrite MATLAB matrix with data from Excel VBA variable.</td>
</tr>
</tbody>
</table>

**Table 3.2 Data Management Functions**

MATLAB 6.0 has been used in the attempt to abstract from Bangla characters and their recognition. Hand sign recognition has also been implemented using MATLAB 6.0.

### 3.3 EASYNN PLUS 10.0

It is a neural network simulator tool for Windows operating system with a GUI (graphic user interface). It can be used to create, train, validate, test/query data for artificial neural network applications. EasyNN-plus grows multi-layer neural networks by connecting input to output layers by hidden layers. EasyNN-
plus neural networks uses, the associated numeric value for all calculations. The main components of this simulator are the Node and the Connection. A node consists of a Neuron with positioning and connecting information. A Connection consists of a Weight with node addressing information. The neural networks learn the training data in the grid. After training and validation, neural networks can be used to test grid data.

3.3.1. GRID

Numeric, text, image, or any combinations of different data formats can be imported in the grid. Text can be mixed with numeric data in example rows but not in Input/Output columns. Columns can be set to text mode using the Edit dialog or while Importing. Each text string in the Grid will have an associated numeric value.

3.3.1.1 Create a New GRID

It is created by using New toolbar button or by the File > New menu command to produce a new neural network Grid. Then, an empty Grid with a vertical line, a horizontal line, and an underline marker will appear with the marker on the position of first column and first row as illustrated in figure 3.3.

3.3.1.2 Writing Data in the GRID

New example is created by pressing the enter key on the new grid. On the prompt "New Input/Output Column" enter "Yes". It amounts to creation of a training example. The example has no name and no value. The example will be labeled "T:0" for training row '0'. The input/output column will be labeled "I:0" for input column '0'. On pressing the enter key again, Edit Grid dialog appears. This dialog is shown in figure 3.4 that is
Figure 3.3 New Grid

used to enter or edit all of the information in the Grid. It is saved by the save command of toolbar.

Figure 3.4 Grid with Edit Grid dialog box

For creating a neural network, new network toolbar button is used.
3.3.1.3 Import Data Into the GRID

EasyNN-plus can import a text or a CSV file, XLS file, image file, binary file to create a new Grid or to add new example rows to an existing Grid. Use import command, on menu of File, and use option create new Grid or File. Use open option to add rows to an existing Grid. Then use Import option and lastly open the file that is to be imported. After importing data, Grid can be viewed as in figure 3.5.

Figure 3.5 Grid

3.3.2 Grow New Network

For growing a new neural network in EasyNN-plus, Grid data should be checked first. “Check Grid” option can be accessed in EasyNN-plus by using “check grid” button on main tool bar. After successful run of “Check Grid” option, it gives message that “the Grid can be used to create a Network”. “Grow new network” option can then be accessed in EasyNN-plus from main toolbar. “Grow new network” option opens “New Network” dialog box, which is shown in figure 3.6. By default, in this dialog box, first hidden layer with 10 minimum nodes and 39 maximum nodes are created. In addition, second hidden layer and third hidden layer can be created as per requirement of application. To optimize the
results, these options can be checked one by one. However, one hidden layer is sufficient for most of the applications.

![New Network Dialog Box](image)

**Figure 3.6 New Network Dialog Box**

### 3.3.3 Setting Control Parameters

For optimizing the neural network, EasyNN-plus has many control options such as, learning rate, momentum, decay, accelerator, threading, etc. The values of these control options can be changed within certain range, which are accessed by change control option. This will open a Controls dialog. Check Optimize for both Learning Rate and Momentum. The controls will be set and a neural network will be ready to learn. The parameters of control dialog box are shown in figure 3.7.

#### 3.3.3.1 Learning Rate

Learning rate is preset to 0.6 after creating a new network. Learning rate can be changed to optimize the network to any value 0 - 0.9. It has also sub options decay and optimize. "Optimize" option allows EasyNN-Plus to determine the learning rate automatically by running a few learning cycles with different values. "Decay" option allows it to reduce the value during learning if oscillations or errors occur. This is shown in figure 3.7.
3.3.3.2 Momentum

Momentum is preset to 0.8 after creating a new network. Momentum can be changed to optimize the network to any value 0.1 – 1.0. It has also sub options decay and optimize. This is shown in figure 3.7. "Optimize" option allows EasyNN-Plus to determine the momentum automatically by running a few learning cycles with different values. "Decay" option allows it to reduce the value during learning if oscillations or errors occur.

3.3.3.3 Target Error

Target error is preset to 0.01 and can be changed to any value from 0.0 to 0.9. Learning stops when the average error goes below the target error.

3.3.4 Training

Learning is started by using option Start Learning from main tool bar. The progress of the network, while it is learning, can be observed by View Graph option of the main toolbar. The progress details and graph are updated after every five-second by default. For validation cycles, some data should be selected for validation. Validation accuracy in percentage can be viewed in "View
Learning. Learning Progress is shown in figure 3.8 with control parameters, training errors, validation error and validation accuracy. Information can be retrieved from “Query” option. The inputs in the Grid can be changed using Edit dialog box or using the Query toolbar buttons: Increase, Decrease, Maximize, and Minimize. A new Query example can be added to the Grid by pressing the Add Query button on main toolbar.

![Learning Graph](image)

### Figure 3.8 Learning Graph

#### 3.3.4.1 Network View

The Network view shows how the nodes in an EasyNN-plus neural network are interconnected. Any source node can be selected with the left mouse button and any destination node can be selected with right mouse button. The network nodes and connections can be edited to produce any configuration, which is shown in figure 3.9.
3.3.4.2 Predictions

The Predictions view has two separate boxes for training and validating example that show how close the predicted value for each example output is to the true value. The position of the values on both axes is scaled to be from a minimum of 0 to a maximum of 1. Predicted outputs for training examples get closer to the true values as training progresses. If most of the dots are close to the diagonal line, the network may be over-trained and it may not generalize well as shown in figure 3.10.

3.3.4.3 Input Importance

The Input Importance view shows the importance and the relative importance of each Input column. The Importance is the sum of the absolute weights of the connections from the input node to all the nodes in the first hidden layer. In figure 3.11, the inputs are shown in the descending order of importance from the most important input.
3.3.4.4 Example Errors - The Example Errors view shows the absolute error and the relative error produced by up to 100 examples in descending order from the greatest error. In figure 3.12, example error is shown.

**Figure 3.10 Prediction**
Diggers.tvq cycle 86. Target error 0.0100 Average training error 0.009833
The first 6 of 6 Inputs in descending order.

<table>
<thead>
<tr>
<th>Column</th>
<th>Input Name</th>
<th>Importance</th>
<th>Relative Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Spades</td>
<td>10.4185</td>
<td>100.00</td>
</tr>
<tr>
<td>0</td>
<td>Diggers</td>
<td>8.0722</td>
<td>87.22</td>
</tr>
<tr>
<td>3</td>
<td>Temp</td>
<td>6.9953</td>
<td>70.30</td>
</tr>
<tr>
<td>2</td>
<td>Diameter</td>
<td>5.2982</td>
<td>55.40</td>
</tr>
<tr>
<td>5</td>
<td>%Clay</td>
<td>1.4479</td>
<td>14.79</td>
</tr>
<tr>
<td>4</td>
<td>%Rocks</td>
<td>1.0651</td>
<td>10.65</td>
</tr>
</tbody>
</table>

Figure 3.11 Input Importance

Diggers.tvq cycle 86. Target error 0.0100 Average training error 0.009833
The first 19 of 19 Example rows in descending order. Above target Below target

<table>
<thead>
<tr>
<th>Row</th>
<th>Example</th>
<th>Normalized Error [0 - 1]</th>
<th>Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Hole 16</td>
<td>0.278508</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Hole 3</td>
<td>0.181521</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Hole 14</td>
<td>0.176890</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Hole 9</td>
<td>0.114286</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Hole 1</td>
<td>0.090928</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Hole 6</td>
<td>0.088076</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Hole 12</td>
<td>0.078632</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Hole 12</td>
<td>0.075056</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Hole 5</td>
<td>0.072995</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Hole 11</td>
<td>0.071765</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Hole 15</td>
<td>0.046713</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Hole 19</td>
<td>0.041754</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Hole 10</td>
<td>0.035046</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Hole 8</td>
<td>0.033511</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Hole 18</td>
<td>0.023018</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Hole 7</td>
<td>0.022178</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Hole 2</td>
<td>0.007587</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Hole 4</td>
<td>0.006770</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Hole 17</td>
<td>0.005404</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.12 Example Error
3.3.4.5 Sensitivity - The Sensitivity view shows how much an output changes when the inputs are changed. The change in the output is measured as each input is increased from lowest to highest to establish the network.

The sensitivity to change is shown in figure 3.13

<table>
<thead>
<tr>
<th>Column</th>
<th>Input Name</th>
<th>Change from</th>
<th>to</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Diameter</td>
<td>150</td>
<td>350</td>
<td>0.481814106</td>
</tr>
<tr>
<td>3</td>
<td>Temp</td>
<td>0</td>
<td>20</td>
<td>0.480760096</td>
</tr>
<tr>
<td>1</td>
<td>Spades</td>
<td>0</td>
<td>4</td>
<td>0.478755645</td>
</tr>
<tr>
<td>0</td>
<td>Diggers</td>
<td>1</td>
<td>4</td>
<td>0.215277940</td>
</tr>
<tr>
<td>4</td>
<td>% Rocks</td>
<td>10.0000</td>
<td>30.0000</td>
<td>0.097500508</td>
</tr>
<tr>
<td>5</td>
<td>% Clay</td>
<td>5.0000</td>
<td>30.0000</td>
<td>0.066801910</td>
</tr>
</tbody>
</table>

Figure 3.13 Sensitivity

3.3.5 Macros and Scripts

3.3.5.1 Macros - Macro recording is started with "Record" on the Macro menu or Ctrl + F2 and is stopped using "Stop Recording" on the Macro menu or Ctrl + F3. EasyNN-plus macros are text files with the .ens extension. Script Commands can be added to macros using Commands on the macro menu. Whole scripts can be added to macros using Add Expanded Script on the Macro menu. The selected script file will be expanded into its original format and then each command will be added to the macro. A script file can be added at the end of a macro using Add Script File on the macro menu. When Ctrl + F4 play the macro back or using Run Script File on the File menu, then it will play the recorded functions and script commands. It will then run the script file and terminate the macro.

Macro files can also be passed to EasyNN-plus as a parameter in the same way as script files. A macro called specscr.ens is included in the Samples folder that creates a network similar to the Species.tvq sample.
3.3.5.2 Scripts

Scripts are text files with .ens or .txt extensions and can be edited with any text editor. Script files can be loaded and run from within EasyNN-plus using Run Script File on the File menu. Scripts can contain commands to import the data, build the neural network, control learning, validating, and querying. The command parameters are validated but some error checking has to be disabled so that the scripts will run. Script files are based on the currently loaded network that can be produced using Generate Script File on the File menu. Scripts can be tested one line at a time by an “ess” command to enable single step mode.

The pause command can be used to temporarily stop and exit from a script. When the script is restarted, it will run from the line after the pause. The commands and error messages that are produced while running the script can be directed to a log file by including the “lsc” command in the script. The script will stop if the “lsc” command fails to open the log file. The name of a script file can be passed as an EasyNN-plus parameter in the command line. This test line shows how to call EasyNN-plus from Start > Run and type “C:\Program Files\EasyNN-plus\EasyNN.exe”.

3.3.7 Pruning

To access pruning option in EasyNN-plus use Options>Pruning. Check Enable connection pruning to enter the weight and cycles required to prune the connections. If a connection stays below the weight for the number of cycles then it will be pruned. Pruned connections are not updated while learning but they are not removed from the network.

3.3.7 EasyNN-Plus Toolbars

EasyNN-plus has many toolbars such as main toolbar, advanced tool bar, and View and Zoom toolbar.

EasyNN-Plus has been used for predicting the ten weeks value of Nifty, parameters contributing towards occurrence of avalanches, recognition of handwritten Gurmukhi and classification of some of diseases of the eye.