EasyNN-plus help

The user interface manual
# Table of Contents

EasyNN-plus help .................................................................................................................. 6

Getting Started ..................................................................................................................... 7

Data to Neural Network ....................................................................................................... 10

Grid data Formats ............................................................................................................... 12

Starting with Images ........................................................................................................... 14

Inside EasyNN-plus ............................................................................................................ 16

Samples ................................................................................................................................. 19

Projects ................................................................................................................................. 22

  Simulating a Network with Excel ......................................................................................... 25

  A Test Excel Simulation ...................................................................................................... 26

  Classifying Components from images .................................................................................. 27

  Predicting Soccer Results .................................................................................................... 28

  Predicting Horse Race Winners .......................................................................................... 29

Shortcuts .............................................................................................................................. 30

Cloning Hidden Nodes ........................................................................................................ 31

Learning Threads ................................................................................................................ 32

Main Window's elements .................................................................................................... 33

  EasyNN-plus main window .............................................................................................. 34

  Grid ................................................................................................................................. 35

  Network ......................................................................................................................... 38

  Learning Progress ......................................................................................................... 41

  Predictions ..................................................................................................................... 44

  Associations .................................................................................................................... 45
Column Values.................................................................................................................. 46
Example Errors.................................................................................................................. 47
Input Importance............................................................................................................... 48
Sensitivity .......................................................................................................................... 49
Macros & Scripts.............................................................................................................. 50
Toolbars............................................................................................................................ 63
Main ................................................................................................................................. 64
Advanced ......................................................................................................................... 68
Extras ............................................................................................................................... 70
Tools ................................................................................................................................. 71
View & zoom .................................................................................................................... 72
Reconfigure ....................................................................................................................... 76
Main Dialogs..................................................................................................................... 78
Information ....................................................................................................................... 79
Import TXT CSV file ....................................................................................................... 81
  2nd Import Dialog ......................................................................................................... 83
  Input/Output Column Dialog ....................................................................................... 85
  Example Row Type Dialog ......................................................................................... 86
Import XLS file ............................................................................................................... 87
Import Binary file .......................................................................................................... 89
Import Images ................................................................................................................ 91
Edit Grid .......................................................................................................................... 93
New Network .................................................................................................................. 95
Controls ......................................................................................................................... 98
Query ............................................................................................................................... 103
File Query.................................................................................................................. 107
Trimming ...................................................................................................................... 109
Leave Some Out .......................................................................................................... 111
Connections ................................................................................................................. 113
AutoSave ...................................................................................................................... 115
Export special files ...................................................................................................... 118
Classify Column .......................................................................................................... 121
Extend Learning .......................................................................................................... 123
Noise ............................................................................................................................. 125
Jog Weights .................................................................................................................. 127
Forget Learning .......................................................................................................... 129
Network Defaults ....................................................................................................... 131
Import Defaults ........................................................................................................... 133
Controls Defaults ....................................................................................................... 135
File Open Defaults ..................................................................................................... 137
Edit Defaults ............................................................................................................... 140
Date & Time ................................................................................................................ 142
Add to Subset ............................................................................................................. 144
Select Subset ............................................................................................................... 146
Forecast ....................................................................................................................... 148
Input Validating Rules ............................................................................................... 150
Output Validating Rules ............................................................................................. 152
Outside Limits ............................................................................................................ 155
Edit Grid Range ......................................................................................................... 157
Change Input/Output Types ...................................................................................... 159
Change Example Types.................................................................................................................. 161
Copy Input/Outputs ........................................................................................................................ 163
Copy Examples................................................................................................................................. 165
Missing Data.................................................................................................................................. 167
Image............................................................................................................................................... 168
Fragment Image............................................................................................................................... 171
Image Fragment.............................................................................................................................. 174
Set Function.................................................................................................................................... 176
Diagnostic Array.............................................................................................................................. 179
Auto Shuffle.................................................................................................................................... 181
Freezes............................................................................................................................................. 183
EasyNN-plus help

EasyNN-plus grows multi-layer neural networks from the data in a Grid. The neural network input and output layers are created to match the grid input and output columns. Hidden layers connecting to the input and output layers can then be grown to hold the optimum number of nodes. Each node contains a neuron and its connection addresses. The whole process is automatic.

The grid is produced by importing data from spreadsheet files, tab separated plain text files, comma separated files, bitmap files or binary files. The grid can also be produced manually using the EasyNN-plus editing facilities. Numeric, text, image or combinations of the data types in the grid can be used to grow the neural networks.

The neural networks learn the training data in the grid and they can use the validating data in the grid to self validate at the same time. When training finishes the neural networks can be tested using the querying data in the grid, using the interactive query facilities or using querying data in separate files.

The steps that are required to produce neural networks are automated in EasyNN-plus.

EasyNN-plus produces the simplest neural network that will learn the training data. The graphical editor can be used to produce complex networks.

The EasyNN-plus facilities are shown in this manual.

The first part of this manual covers getting started, samples and projects. Experienced users can skip those sections.
Getting Started

If you prefer an interactive guide click the Getting Started button on the EasyNN-plus Tip of the Day dialog.

This exercise is a step by step guide resulting in a neural network that is trained to produce the secondary color from pairs of the three primary colors. The primary light colors are used (red, green and blue) rather than primary pigment colors (red, yellow and blue). The guide is mainly about using the keyboard to navigate and enter data in the Grid however it is possible to navigate using the mouse if you prefer. The first few steps are described in detail. The process gets easier as you complete each stage.

Press the New toolbar button or use the File > New menu command to produce a new neural network Grid.

An empty Grid with a vertical line, a horizontal line and an underline marker will appear. The marker shows the position where a column and row will be produced.

Press the enter key and you will be asked "Create new Example row?" - answer Yes.
You will then be asked "Create new Input/Output column?" - answer Yes.
You have now created a training example with one input. 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.

Press the enter key again and you will open the Edit Grid dialog. This dialog is used to enter or edit all of the information in the Grid.
Enter "1" in the Value edit box and then tab.
Enter "Pair 1" in the Example row edit box and then tab.
The Type is already set to Training so just tab again.
Enter "Red" in the Input/Output column edit box and then press OK.
You now have an example named "Pair 1" with an input named "Red" with the value set to 1.0000

Press the right arrow key and then press enter.
You will be asked "Create new Input/Output column?" - answer Yes.

Press enter and you will open the Edit Grid dialog again.
Enter "1" in the Value edit box and then tab three times.
Enter "Green" in the Input/Output column edit box and then press OK.

Press the right arrow key and then press enter.
You will be asked "Create new Input/Output column?" - answer Yes.

Press enter and you will open the Edit Grid dialog again.
Enter "0" in the Value edit box and then tab three times.
Enter "Blue" in the Input/Output column edit box and then press OK.

Press the right arrow key and then press enter.
You will be asked "Create new Input/Output column?" - answer Yes.

Press enter and you will open the Edit Grid dialog again.
Enter "1" in the Value edit box and then tab three times.
Enter "Yellow" in the Input/Output column edit box.
Click the Output and the Bool radio buttons and then press OK.
Press the **right arrow key** and then press **enter**.
You will be asked "Create new Input/Output column?" - answer **Yes**.

Press **enter** and you will open the **Edit Grid** dialog again.
Enter "0" in the **Value** edit box and then **tab** three times.
Enter "Cyan" in the **Input/Output column** edit box.
Click the **Output** and the **Bool** radio buttons and then press **OK**.

Press the **right arrow key** and then press **enter**.
You will be asked "Create new Input/Output column?" - answer **Yes**.

Press **enter** and you will open the **Edit Grid** dialog again.
Enter "0" in the **Value** edit box and then **tab** three times.
Enter "Magenta" in the **Input/Output column** edit box.
Click the **Output** and the **Bool** radio buttons and then press **OK**.
The first training example "Pair 1" is now complete along with all the inputs and outputs.
Two more training examples need to be created called "Pair 2" and "Pair 3".
Use the **arrow** keys to move to the **Grid** location below the 1.0000 in the "Red" column.

Press the **enter** key and you will be asked "Create new Example row?" - answer **Yes**.

Press **enter** again and you will open the **Edit Grid** dialog.
In the **Input/Output column** the name "Red" and the type **Input** are already in the **Edit Grid** dialog.
Enter the **value** "0" and example row name "Pair 2" then press **OK**.
Move to each **Input/Output** column in turn and press the **enter** key.
Enter the following values followed by **OK**.
Input column "Green" enter the value "1"
Input column "Blue" enter the value "1"
Output column "Yellow" enter the value "0"
Output column "Cyan" enter the value "1"
Output column "Magenta" enter the value "0"

Repeat the process again for "Pair 3" but this time use the following values.
Input column "Red" enter the value "1"
Input column "Green" enter the value "0"
Input column "Blue" enter the value "1"
Output column "Yellow" enter the value "0"
Output column "Cyan" enter the value "0"
Output column "Magenta" enter the value "1"

You now have all the data that is required to create and train the neural network.

Press the save toolbar button or use the **File > Save** menu command to save the **Grid** before continuing.
Call it "Network". The name could be anything but this exercise assumes it will be "Network".

To create the neural network press the Grow new network toolbar button or use the **Action > New Network** menu command. If the toolbar button is not enabled make sure that you have set the first three Input/Output columns to Input and the last three Input/Output columns to Output.

This will open the "New Network" dialog.
Check **Grow layer number 1** and press **OK**.
If you get a "Generating new network will reset learning." warning message answer **Yes**.
The neural network will be produced from the data you entered into the Grid.

Answer **Yes** to the message "Do you want to set the controls?".
Check **Optimize** for both **Learning Rate** and **Momentum** and then press **OK**. Answer **Yes** if you get the “Optimizing controls will reset learning.” warning message. The controls will be set and the neural network will be ready to learn.

Answer **Yes** to the message “Do you want Network.tvq to start learning?”.

Press the View network toolbar button or use the **View > Network** menu command to see the new neural network.

Press the View learning graph toolbar button or use the **View > Graph > Learning Progress** menu command to see how the error reduced to the target level.

Press the Query network toolbar button or use the **Query > Query** menu command to open the **Query** dialog.

Press the **Add Query** button and then answer **OK** to the **Example Presets** dialog. An example named "Query" with unknown values will be generated and selected.

Select a name or value from the **Input** lists and press the **Max** button. Select another name or value from the **Input** lists and press the **Max** button. Select the last name or value from the **Input** lists and press the **Min** button. Notice that the correct secondary color in the **Output Names** list is set to "true". Select a name from the **Output** lists and press the **Seek High** button and then the **Start/Stop Cycle** button. The seek cycle will stop automatically. Notice that the correct pair of primary colors have the values set to 1.0000 in the **Inputs** list. Press the **Close** button.

Press the View grid toolbar button or use use the **View > Grid** menu command to return to the Grid view.

Notice that the example named "Query" has been added to the Grid. This example can be edited using the same **Edit** dialog you used to produce the Training examples. The Grid will be updated to show the edited values and the results.

Set any pair of primary colors to 1 and notice that the corresponding secondary color is set to "true".

You have now used all the basic functions in **EasyNN-plus**.

You should have no difficulty converting the neural network to produce secondary pigment colors from the primary pigments. You could then try to modify the pigments network to produce a "Black" output when all three primary pigment inputs are present and a "White" output when all three are absent.
**Data to Neural Network**

This procedure is a step by step guide to creating a neural network from a data file.

All you need to start is the Races.txt file. You will find a copy in the samples folder. Have a look at the contents of the file. You will see that the column titles are on the first line and the other lines start with the line number in square brackets. We will use the titles for column names and the numbers for row names. Some of the values in the race data are integers and some are boolean.

The file contains data from 370 horse races. The aim is to create a neural network that can be trained and validated using the horse race data.

After the neural network has been trained and validated it can be used to help predict the winner of other races.

Press the New toolbar button or use the File > New menu command to produce a blank grid for a new neural network.

An empty Grid with a vertical line, a horizontal line and an underline marker will appear.

Now select File > Import...

The file selection dialog will appear.

Open the file Races.txt in the samples folder.

The first Import dialog will appear with the Tab delimiter already checked.

Each line is numbered in Races.txt so that can be used as the row name. Select "Use first word(s) on each line for row names". Press OK.

The second Import dialog will appear.

Press the Set names button.

The first line and the column names will be set.

The third import dialog called Input/Output Columns will appear. The settings in this dialog are based on the value being imported but they should be checked. In particular the mode and type may not always be correct.

Column 0 will be ready to setup. The column is called Runners and has a value of 11. The column settings will be correct so press OK.

The next column will now be shown. This column is Distance with a value of 7. The settings are correct so press OK.

Column 3 is Handicap with a value of 0. The mode will change to Bool. Press OK.

Column 4 is Class and is another integer. The mode will change back to Integer. Press OK.

Column 5 and 6 are boolean so the mode will be set to Bool. Press OK for both columns.
The last column is **Win** and is also a boolean so the mode will be correctly set. The **type** is not correct and needs to be changed to **Output**. Then press **OK**.

The data will be imported and the grid columns will be set to the correct mode and type.

The **Grid** is now complete.

To create the neural network press the Grow new network toolbar button or use the **Action > New Network** menu command.

This will open the **New Network** dialog.

Check **Grow layer number 1** and press **OK**.

If you get a "Generating new network will reset learning," warning message answer **Yes**.

The neural network will be produced from the data you imported into the Grid.

Answer **Yes** to the message "Do you want to set the controls?".

This will open the **Controls** dialog.

Check **Optimize** for both **Learning Rate** and **Momentum**.

We need to change 100 of the training rows to validating so enter 100 in **Select examples at random**.

There is no point in training the neural network for more than 1000 cycles so check **Stop on cycle** and enter 1000.

Press **OK**. Answer **Yes** if you get the "Optimizing controls will reset learning," warning message.

The controls will be set and the neural network will be ready to learn.

A summary of the control settings and what is going to happen next will appear.

Answer **Yes** to the message "Do you want *filename* to start learning?". The filename will probably still be called untitled.tvq, there is no need to change it unless the file is saved.

The **AutoSave** dialog will open. Nothing needs to be done so just press **OK**.

Learning and validating will run for 1000 cycles. Use **View > Information** to see the results. The line **Validating results: ...** is all that matters. If everything has worked as expected the neural network will have correctly predicted the results of at least 64% of the races.

That is the end of this exercise.

If you want to try using the query facilities to input details of races and horses you can enter the queries directly into the Grid or use the **Query** dialog.

If you are using the full version you will be able to use a **File Query** to produce a results file for a whole days racing. Races that indicate no winners or multiple winners should be ignored.
Grid data Formats

Grid Data Types and Modes

The Grid in EasyNN-plus can contain numeric, text, image or combinations of different data formats.

**Numeric** data can be integer, real or boolean (true or false).

**Text** strings can be entered or Imported into the Grid. Text can 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. This numeric value is the sum of the ASCII code multiplied by its position in the string. For example, the word 'Dog' is \((68 \times 3) + (111 \times 2) + (103 \times 1) = 529\). The numeric value of each text string is checked when it is first calculated to make sure that it is not too close to another string. If it is found to be within 100 of any other string then 101 is added to the numeric value and it is checked again. Every text string has a unique associated numeric value.

**EasyNN-plus** neural networks use the associated numeric value for all calculations.

Words and phrases do not have any intermediate text that can be calculated. For example, 'Cog' is alphabetically intermediate to 'Cat' and 'Dog' but it does not have much to do with either animal. 'Dog' and 'Cat' will have the associated values of 529 and 713 if they are entered into the same column in that order. The intermediate values will have no associated text. **EasyNN-plus** will find the nearest associated numerical value for the Output columns of the query rows. The text is prefixed with a series of '~' characters depending on how close it is. The intermediate value 712 would give the text '~Cat', 700 would give '~Cat' and 600 would give the text '~Dog'.

The Edit > Replace facility in EasyNN-plus can replace numeric values with text but this should be used with caution. The numeric values associated with the text will not usually have a direct relationship with the numeric values that are being replaced. The series of numbers 1, 2, 3 and 4 could be replaced with the text One, Two, Three and Four. The associated numeric values would be 558, 702, 1481 and 961. After scaling the original series would be 0.0000, 0.3333, 0.6667 and 1.0000. The replaced text would be 0.0000, 0.1561, 1.000 and 0.4366 after scaling. The two series of numbers do not have a simple relationship so, in this text, should not be used to replace a simple numerical series.

**Images** can be entered or Imported into the Grid. Any bitmap image can be used in EasyNN-plus. Bitmaps of different sizes can be mixed. Other image formats (jpeg, gif etc) must be first converted to bitmap format. Images can be mixed with text and numeric data in Example rows but not in Input/Output columns. Columns can be set to image mode using the Edit dialog or while importing.

Each image in the Grid will have a name, a bitmap file and three associated numeric values. These numeric values are the Pixel Pair Code (PC), the Edge Code (EC) and the Block Code (BC). PC is derived by comparing a thousand pairs of pixels in the image. The positions of the top, bottom, left and right outer edges are used to produce EC. The position of the largest block of pixels with the same color is used to produce BC. **EasyNN-plus** must have access to the bitmap file to produce PC, EC and BC. Large bitmaps will take a few seconds to process. If **EasyNN-plus** cannot find the bitmap file a selection dialog will be opened so that the file can be located. This selection dialog is also used while a file is being imported whenever a bitmap file is needed. Once the images are in the Grid they can be used to produce networks in the same way as any other type of data. Every image Input and Output column in the Grid will map onto three nodes, a PC node, an EC node and a BC node.

Images can be fragmented into component parts using Edit > Fragment Image. Images and fragments
can be associated with the values in other Input/Output columns using **Edit > Associate Image**.
Starting with Images

Multiple images can be included in the Grid.

Images will be encoded into three separate nodes when the neural network is created. Images can also be fragmented into parts in the Grid. Each part will be encoded separately.

This procedure is a step by step guide that produces a neural network using an image in the Grid.

For this exercise you will need a bitmap image file in which you can identify a component part. For example, a nose on a picture of a face, a car on a street, a tree in a garden, any distinctive object will do. The bitmap image will be entered into a new Grid where it will then be broken into fragments. The fragments will appear in a Grid Input column. An Output column will be created that will be associated with the fragments. The values in the associated column will be set to identify the fragment containing the component. The two columns will then be used to create a neural network.

Some of the steps in this procedure will be the same as those in the Getting Started topic but they will be repeated here.

Press the New toolbar button or use the File > New menu command to produce a new neural network.

An empty Grid with a vertical line, a horizontal line and an underline marker will appear. The marker shows the position where a Grid column and row will be produced.

Press the enter key and you will be asked "Create new Example row?" - answer Yes. You will then be asked "Create new Input/Output column?" - answer Yes. You have now created a training example with one input. The example has no name and no value.

Press the enter key again and you will open the Edit Grid dialog. Type a word such as "picture" in the Value edit box. It is just a name to use for the image, it will be deleted later.

Click the Image radio button and press OK.

The Image File selection dialog will appear. Enter the location of your bitmap file or use Browse to locate it. Press OK when you have found your image.

The bitmap file will be selected, encoded and inserted into the Grid.

Double click the image in the Grid to open the Image dialog.

Press the Fragment button to open the first Fragment Image dialog. The bitmap file will already be in this dialog so just press OK to open the second dialog.

The entries in the second dialog are already set correctly so just press OK again.

The image will be fragmented into 16 parts and entered into the Grid. The first fragment will overwrite the original image which is no longer needed.

Press the right arrow key and then press enter. You will be asked "Create new Input/Output column?" - answer Yes.

Press enter and you will open the Edit Grid dialog again.

Click the Bool and Output radio buttons and press OK.
All the outputs need to be set to ‘false’ except the one that corresponds to the fragment containing the component part which is set to ‘true’. It is possible to enter the values directly using the Edit Grid dialog but in this exercise we will use an association between the two columns. This will allow the Output column values to be edited while viewing the Input column fragments.

Press the left arrow key to move back to the Grid column that contains the fragments.

Select Associate Images... from the Edit menu to open the Associate Images dialog. Enter 1 in the Input/Output column edit box and press OK.

The Image Fragment dialog can be opened either by double clicking the image fragment in the Grid or moving to the fragment and pressing enter. Open the dialog for each fragment in turn. If the fragment contains the component part that you have chosen to identify then enter 't' in the associated value edit box. If the fragment does not contain the component part enter 'f'.

Use the Action > New Network menu command to open the New Network dialog. Check Grow layer number 1 and press OK. If you get a "Generating new network will reset learning." warning message answer Yes.

Answer Yes to the message "Do you want to set the controls?".

Check Optimize for both Learning Rate and Momentum and then press OK. Answer Yes if you get the "Optimizing controls will reset learning." warning message. The controls will be set and the neural network will be ready to learn the image fragments and associations that you have entered into the Grid.

Answer Yes to the message "Do you want it to start learning?". The learning process should run to completion and stop automatically but it can be stopped by pressing the Stop toolbar button or using the Action > Stop menu command. If the learning process does not stop automatically then it is probably because the basic neural network that has been created cannot uniquely classify the fragment with your chosen component. The purpose of this exercise was only to introduce the use of images and fragments in the Grid and you have now used all the facilities. However, if you want to continue and make the neural network classify the component try using more hidden nodes. If that does not help, try two hidden layers. If it still does not work it may be that the component fragment you have chosen produces a similar value to other fragments when they are encoded.

Press the Query toolbar button or use the Query > Query menu command to open the Query dialog.

Press the Add Query button. Press OK in the Querying presets dialog. An example named "Query" will be generated and selected.

Select the Output and Start/Stop Cycle for a few seconds to check if the correct fragment is identified in the Inputs. The Output may be incorrect. It is usually possible to produce the correct Output by setting the Target Error to a lower value in the Controls dialog and then continuing Learning.

This method has been used successfully to identify the presence of cars on photographs of streets. The images were fragmented and then the associated bool value was set to true for each fragment that contained a whole side view of a car. After the network correctly identified the cars on three streets it could then identify similar car profiles on a number of other street photographs.
Inside EasyNN-plus

A **Neural Network** produced by **EasyNN-plus** has two component parts.

The component parts are the **Node** and the **Connection**. These components are replicated to make the neural network. A **Node** consists of a Neuron with positioning and connecting information. A **Connection** consists of a Weight with node addressing information.

The **Grid** used by **EasyNN-plus** also has two components. These are the Example row and the Input/Output column. These are replicated to make the grid.

All the component parts of **EasyNN-plus** are implemented as reusable classes to simplify future development. The following information is a very basic description of the classes. The true names of the variables and functions are not used.

**The Neural Network**

**Node**

**Positioning and connection**

Type:
- Input, Output or Hidden.

Number:
- Node reference number.

Layer Type:
- Input, Output or Hidden.

First In:
- Number of the first connection into this node.

Last In:
- Number of the last connection into this node.

**Neuron**

Net Input:
- Sum of all activation * weight inputs to the node + Bias.

Activation:
- \( \frac{1.0}{1.0 + e^{- \text{Net Input}}} \)

Output Node Error:
- Target - Activation

Hidden Node Error:
- Error + Delta * Weight

Delta:
Error * Activation * (1.0 - Activation)

Bias:
Bias + Delta Bias

Bias Derivative:
Bias derivative + Delta

Delta Bias:
Learning Rate * Bias Derivative + Momentum * Delta Bias

**Connection**

**Node addressing**

To:
The Node that the connection is going to.

From:
Node that the connection is coming from.

Number:
Connection reference number.

**Weight**

Type:
Variable or Fixed.

Weight:
Weight + Delta Weight.

Weight Derivative:
Weight Derivative + To: Delta * From: Activation.

Delta Weight:
Learning Rate * Weight Derivative + Momentum * Delta Weight.

**The Grid**

**Example row**

Name:
Optional Example name.

Type:
Training, Validating, Querying or Exclude.

Values:
Array of values in example row.

Scaled Values:
Array of scaled values in example row.

Forecasted Values:
Array of forecasted values in example row.
**Input/Output column**

Name:
Optional Input/Output name.

Type:
Input, Output, Serial or Exclude.

Mode:
Real, Integer, Bool, Text or Image.

Lock:
True or False.

Lowest:
Lowest value in column.

Highest:
Highest value in column.
Samples

Samples included in the samples folder.

Add.tvq
This network is trained to add two numbers together. The network has been built with three hidden nodes. The extreme values are duplicated as validating examples in order to terminate the learning process as soon as they are correct. To use this method the Controls need to be adjusted so that Stop when 100.00% of the validating examples are Correct after rounding is set.

Basket.tvq
This sample is to demonstrate the Associating facility. The examples are taken from discarded till receipts at a small general groceries and provisions store. Pairs and clusters of associated items in the shopping baskets can be found using Action > Start Associating. In this sample the associations can indicate shopping habits. Most of the important item pairs are close to each other in the store indicating the importance of layout on which items finish up in the shopping basket. The various types of soup show this association quite clearly. Some important item pairs are not close together in the store but will be used together later such as Milk and Custard Powder. A few of the associations are difficult to explain.

Cichlids.tvq
Some East African Cichlid fishes are difficult to classify because many of the characteristics that are normally measured for classification purposes lie in overlapping ranges. The network has been trained using the published descriptions of 16 different, but closely related, cichlids. Ten characteristics have been measured in each case. In this sample only one example has been used for each species. Periodic validating with the same data is used to terminate the learning process as soon as 100% correct classification is reached. It is more usual to use different validating and training data. After the network has been trained to be 100% correct on the training examples it can be used to identify other specimens.

If any cichlid experts are using EasyNN-plus the characteristics measured are SL - standard length, BD - body depth, HL - head length, PD - preorbital depth, CP - caudal length, UJL - upper jaw length, LJL - lower jaw length, LL - lateral line scales, D spines - dorsal spiny rays, D branched - dorsal branched rays. The descriptions used are taken from ‘A revision of the Haplochromis and related species (Pisces : Cichlidae) from Lake George, Uganda’ by P.H. Greenwood 1973. The Haplochromis and related genera have been revised since and the names used in this example have probably now been changed.

Diggers.tvq
This is a simple planning example. It shows how a neural network can be used to estimate the digging rate of one to four diggers under a variety of conditions. The output of the network is the number of kilograms per hour dug from a hole. The inputs produce a number of conflicts. Diggers work best when they have a spade to dig with so four diggers with one spade only perform a little better than one digger with one spade. Rocky soil is more difficult to dig but rocks weigh more than soil and we are interested in the weight rather than the volume. The amount of clay also has some impact on the digging rate but it also weighs more than soil - but not as much as rocks. Cold soil is difficult to dig. Frozen soil is very difficult to dig. On the other hand, diggers need more rest breaks when the temperature is high. If the diameter of the hole is too small then the number of diggers who can work down the hole is limited. However the diggers who are not digging must be resting so high temperature is less of a problem.

Discounts.tvq
This network is trained using a number of fixed discount breakpoints so that intermediate discount prices can be estimated.

Grocer.tvq
This demonstrates the use of the Seek buttons. A retail grocer adjusts his prices according to the
wholesale prices every morning when he collects his stock for the day. He always tries to make a reasonable profit but he knows from experience that to maximize his profit he cannot just increase his prices. When his prices reach a certain level his profit starts to decrease because his item sales begin to decrease. The neural network has learned the relationships between some prices and the resulting daily turnover and profit. Using Seek High the grocer can see which price combinations produce the maximum profit or maximum turnover.

Hometime.tvq
This network has been trained using journey times and the routes from work to home. It is a short journey through a very busy area with variable traffic. The object is to find the fastest route home at a given time. The departure and arrival times are in decimal hours, not hours and minutes. The journey has five common points including the start and end. It uses the names of the roads to indicate the route taken on each stage of the journey between adjacent common points. The example journeys cover a period of one year so the most recent traffic and road conditions influence the neural network. To find the fastest route home using Query the departure time (in decimal hours) is Set in the input list and locked by clicking its Lock in the list. Then the Seek Low facility can be used to find the earliest arrival time and thus the fastest route home for that departure time.

House prices.tvq
This is a simple example showing how EasyNN-plus can be used to estimate house prices. It has been trained using a small number of houses around the Stockport (UK) area. It uses the information commonly found in estate agents advertisements. A very small subset of items of description has been chosen for input along with the location of the house. The location inputs are postal area codes for Stockport (SK1 to SK8). Some of the training cases are incomplete (indicated by a '?' in the training example) and allowed to default. It produces quite accurate results when tested against agents data for houses in a similar area in the same year (1993).

Iris.tvq
The data in this example is from Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936), also in "Contributions To Mathematical Statistics" (John Wiley, NY, 1950). The data has been used many times before to demonstrate how neural networks (and other techniques) can be used for classification. 50 examples of three species of Iris are used to train the neural network. Two of the sets of data are not linearly separable. After training the network gets 100% correct.

Logic.tvq
All the two input logical functions are simulated using this neural network.

Parity 3.tvq
Three bit, odd parity example using validating examples to stop training as soon as 100% correct results are achieved.

Parity 4.tvq
Four bit, odd parity example trained, without using validating examples, until a low error is achieved.

Random data.tvq
This network uses random data for both training and validating. It demonstrates that a neural network can be trained using random data but shows that no amount of training will produce good validating results.

Signature.tvq
A bitmap image of 42 forged signatures has been split into individual parts using the Fragment facility. Each fragment contains one signature and has been associated with a bool in the adjacent Output column. The signatures have been assessed by the potential victim of the forgery to see which are good fakes. That is, which ones would fool him. Ten Example rows have been chosen at random and set to Querying so they will not take part in the training. These Querying examples are used to test the network
after training. The network has then been created starting with 10 hidden nodes and allowed to grow. It converges quickly when the hidden nodes are incremented to 11. The excluded column named "reference" allows the Querying results to be compared with the original values. The network gives the correct result for 9 of the 10 Querying examples. It gets fg000002 wrong.

**Species.tvq**
In this network one of the input columns and the output column is text data. It is a simple species classification network. It shows the problem that can occur when text is used in an output column because intermediate values cannot be calculated. The network requires to be trained to a very low error before it classifies the species correctly.

**XOR.tvq**
The standard "exclusive or" test that most neural networks systems use for demonstration purposes. This sample is used in the interactive Getting Started exercises.

See the [Projects](#) for more samples.
Projects

Projects included with the Samples:

Simulating a Network with Excel Stuart Moffatt

A Test Excel Simulation Marco Broccoli

Classifying Components from images Department of Mechanical Engineering, Loughborough University

Predicting Soccer Results Harry Molloy

Predicting Horse Race Winners Tom Huxley

Other Projects:

Maximising retail profit.
This project was based on the "Grocer.tvq" sample that is supplied with EasyNN-plus. The main difference is that the network was trained to identify the "special offers" that resulted in the most shoppers visiting the store and the subsequent profit. It was noted that there was no simple, direct relationship between the number of visitors and the profit.

Estimating property prices.
This is another project that started with one of the samples. The "House prices.tvq" sample is based on house prices around Stockport in England. It was modified using commercial property prices in an area of New York. Other than being trained with different measures it is the same network as the sample. Most of the room counts were replaced with floor space sizes.

Food product grading.
A producer of "traditional" oat biscuits wanted a method that could determine the best biscuits in batches of thousands. The best biscuits were those that looked to be homemade. The network used the images of freshly baked biscuits. Three states were determined by manual inspection. These were "regular", "homemade" and "irregular". The network took over a day to train and validate because the images were very complex. It could then identify biscuits that had the homemade look. The next stage in the project is to incorporate the network into a machine that will sort the biscuits into different grades for different markets. Regular biscuits go into cylindrical packs. Homemade biscuits go into flat packs and attract a higher price. Irregular biscuits are broken up and used for other products.

Electricity supply load.
Predicting the load for an electricity supply and generating company. The project was to find the time, day and conditions that resulted in high loads on an electricity supply. A neural network was trained using the load data for the previous six years sampled once every hour. This period included extremes of temperature, humidity, wind speed and sky cover. The neural network could then accurately predict the load for different weather conditions at any time.

Planning journeys.
Yet another project that started with a supplied sample, "Hometime.tvq". It is to find the route that a salesman should take to arrive at the most convenient time for the client and result in the most sales. The sales workload was also planned using a similar technique. Long journeys that resulted in few sales were given low priorities or avoided completely by referring the client to different ordering methods.

Plant growth.
This network was developed by a student in a horticultural college to find out which factors were most important to growth rate of a type of pine tree used in timber production. The factors involved were mainly related to rainfall and temperature but also included the different techniques used to encourage rapid growth.

**Horse race winners.**
This project included a neural network for every race meeting where a group of horse, jockey and trainer were expected to compete during a season. The networks were then combined to produce an imaginary race meeting in which the winners would be predictable. Future real meetings were then carefully examined to look for similarities with the model meeting. If a meeting were identified with any of the horse, jockey and trainer combinations involved in the race, a bet would be placed on that horse.

**Soccer game results.**
The results of Soccer matches are extremely difficult to predict but a neural network can do a little better than just guessing. A neural network was trained using the scores in hundreds of games, the current team positions and historical league positions.

**Identifying images of cars.**
A neural network was trained to identify cars on a series of photographs of a busy road and validated using a different set of photographs. After the validating results peaked the network was tested using a third set of photographs. All cars that were presented with a side elevation which did not overlap other vehicles were identified.

**Wound healing treatments.**
A medical physicist developed a network that was trained using the treatment and medical history of hundreds of patients to determine which treatments produced the optimum healing time.

R.J. Taylor et al: Using an artificial neural network to predict healing times and risk factors for venous leg ulcers
*Journal of Wound Care Vol.1, No.3 2002*

**Stroke rehabilitation progress.**
This network was developed by a stroke rehabilitation specialist to help find the combination of therapies that produced the best results.

Marc van Gestel

**Drug interactions.**
The side effects reported by patients involved in a stage of drug testing were used to train a network. The reported symptoms and the dosage of all the drugs being taken were used in the training. The trained network could then indicate possible drug interactions that would need further investigation.

**Multiple sclerosis symptoms and treatments.**
This project uses a network that establishes which MS symptoms are related to treatments. The inputs to the network are the treatments and the outputs are the symptom scores over the following eight days. A second network uses a much longer period. In both networks, the importance of the treatments is determined after training. Querying is used to investigate if the importance is due to a positive or a negative change in symptoms.

**Identifying liver cancer.**
The distinction of hepatocellular carcinoma (HCC) from chronic liver disease (CLD) is a significant clinical problem. In this project a network has been used to help identify tumor-specific proteomic features, and to estimate the values of the tumor-specific proteomic features in the diagnosis of HCC.

Poon et al.: Comprehensive Proteomic Profiling and HCC Detection
*Clinical Chemistry 49, No. 5, 2003*
**Who wrote this?**
A neural network was trained with hundreds of lines of text extracted from ten novels. The words per sentence, average syllables per word, average letters per word and the actual text were used for inputs. The ten authors were the outputs. After training the neural network could identify the most likely author of any given text. The long term aim of the project is to classify fictional writing styles.

**Germination rate.**
Fifty seed trays were used in which 25 seeds of a common food grain were planted. The germination rate was measured and used as the output of a neural network. The inputs were the drill depth, spacing, soil temperature, surface temperature and soil moisture by conductivity.

**Chemical analysis.**
An application in gas chromatography which predicts retention indices (the position when a chemical compound appears in a chromatogram/plot compared to other components) on the base of topological descriptors, which describe the structure and/or properties of a chemical.

**Chronic Nephropathies.**

**Reproductive intentions.**
Simulating a Network with Excel

Some people have created spreadsheets using the connection weights and node biases from the special text files that *EasyNN-plus* can produce. Some of these spreadsheets are very complex but this method is elegantly simple. The spreadsheet has been adapted to work with the *Add.tvq* sample. The Excel spreadsheet file, *Add.xls*, is included with the samples.

by *Stuart Moffatt*
A Test Excel Simulation

This spreadsheet uses a similar method to the above but it is arranged in a way to be visually similar to the network that it is simulating. The Excel spreadsheet and EasyNN-plus network file, Test1.xls and Test1.tvq, are included with the samples.

by Marco Broccoli
Classifying Components from images

This network demonstrates how a neural network can be used to classify component parts. The parts used are Long Bolts, Short Bolts, Nuts, Circlips and Washers. The component images are preprocessed to produce three inputs for the neural network. The three inputs are Area, Perimeter and Roundness. Each component part has a bool output column.

The PowerPoint presentation slides, NNDemo.ppt file is included with the samples.

See the Components.tvq sample.

by Department of Mechanical Engineering, Loughborough University
Predicting Soccer Results

Neural networks are often used in the prediction of soccer results. The methods used are many and varied but they usually depend directly on previous scores to assess the team strengths. The models produced often do just a little better than a random choice. The prediction of home and away wins using a neural network has proved easier than draw prediction but the soccer expert, almost always, does better than a neural network. Harry uses the ratings that are produced by an expert.

Harry has constructed a neural network that uses some of the RATEFORM values from http://members.aol.com/soccerslot/socrates.html

The RATEFORM home away values and the difference between them are used as inputs. The match result is used for output. The original network uses three different columns for home, draw and away. This can produce misleading validating results because the network has no knowledge that the three outputs are exclusive. So far as the network knows the three outputs have eight different possible combinations - not three. A forth output has been added with three values: 1 - home win, 2 - draw and 3 away win. The three original output columns are excluded. The network produces 53% correct results on the validating examples. A random choice would be only 33% correct.

This shows that Harry's network produces a significant advantage when picking matches.

See the formrates.tvq sample.

by Harry Molloy
Predicting Horse Race Winners

Many horse racing systems are built around the fact that over 60% of winners come from the first two horses in the betting. **First2** is an attempt to improve the punters chances of winning, by selecting those races that are likely to have a winner coming from the first two in the betting, using a trained Neural Network.

**Inputs:**

- **Runners** (number of runners)
- **Distance** (in furlongs)
- **Handicap** (race type)
- **Class** (1 for A or listed, 2 for B, 3 for C..)
- **Stake>5k** (Prize money for winner is greater than £5000)
- **Odds>2** (Odds of first horse in the betting is greater than 2/1)

**Output:**

- **Win** (Is one of the first two in the betting likely to win?)

The data for this network are for flat races run in 1998.

The network is produced with **Grow** hidden layer 1 checked.

The controls have **Optimize** and **Decay** set for both **Learning Rate** and **Momentum**. The Target error is set to 0. The Correct after rounding validating stop is set and 100 examples have been selected at random for validating.

Auto Save as been set the save every 100 cycles if the number of correct validating examples have increased.

The **First2.tvq** file can be used to predict the winners. In the 100 validating examples it correctly predicts if one of the first two horses in the betting will win in 69% of the races. This is a 9% advantage over the known results. Is that a big enough advantage to make a profit?

See the **First2.tvq** sample.

by **Tom Huxley**
Shortcuts

<table>
<thead>
<tr>
<th>Key Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ctrl</strong> J</td>
</tr>
<tr>
<td><strong>Ctrl</strong> Enter</td>
</tr>
<tr>
<td><strong>Shift</strong> Space</td>
</tr>
<tr>
<td><strong>Ctrl</strong> Space</td>
</tr>
<tr>
<td><strong>Ctrl</strong> D</td>
</tr>
<tr>
<td><strong>Ctrl</strong> C</td>
</tr>
<tr>
<td><strong>Ctrl</strong> Insert</td>
</tr>
<tr>
<td><strong>Shift</strong> Delete</td>
</tr>
<tr>
<td><strong>Ctrl</strong> X</td>
</tr>
<tr>
<td><strong>Ctrl</strong> Delete</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F</td>
</tr>
<tr>
<td><strong>Ctrl</strong> A</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F3</td>
</tr>
<tr>
<td><strong>Ctrl</strong> G</td>
</tr>
<tr>
<td><strong>Ctrl</strong> T</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F4</td>
</tr>
<tr>
<td><strong>Ctrl</strong> U</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F2</td>
</tr>
<tr>
<td><strong>Ctrl</strong> V</td>
</tr>
<tr>
<td><strong>Shift</strong> Insert</td>
</tr>
<tr>
<td><strong>Ctrl</strong> H</td>
</tr>
<tr>
<td><strong>Ctrl</strong> N</td>
</tr>
<tr>
<td><strong>Ctrl</strong> O</td>
</tr>
<tr>
<td><strong>Ctrl</strong> P</td>
</tr>
<tr>
<td><strong>Ctrl</strong> S</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F1</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F4</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F2</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F3</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F6</td>
</tr>
<tr>
<td><strong>Shift</strong> F6</td>
</tr>
<tr>
<td><strong>Ctrl</strong> F5</td>
</tr>
<tr>
<td><strong>Ctrl</strong> Q</td>
</tr>
<tr>
<td><strong>Ctrl</strong> W</td>
</tr>
</tbody>
</table>
Cloning Hidden Nodes

Cloning can help when there are problems finding a neural network configuration that will learn a complex dataset. Neural networks can be built with cloning enabled or cloning can be enabled while the network is learning. Hidden nodes are cloned during the learning process when the average error has stopped reducing. The hidden node with the greatest net input is frozen to determine if it is contributing to the learning process. If the node is contributing to the learning process it is cloned. Nodes that are not contributing are skipped and the node with the next greatest net input is tested. If no suitable node can be found a new node is created and fully connected with random low weights. It is then cloned producing an exact copy. The node that has been cloned is frozen again and its clone takes over in the learning process. The freeze level of the node that has been cloned is reduced until both the node and its clone are being used in the learning process. The learning process does not need to restart when a node is cloned.

About Clones & Cloning

1. Nodes will not be cloned if the neural network is learning and the average error is decreasing.
2. Clones may be produced during learning if cloning is enabled and the average error is not decreasing.
3. Nodes that are cloned are initially frozen.
4. The freeze level may reduce to zero very quickly and never be seen on the network display.
5. Clones are first produced in hidden layer 1, then in hidden layer 2 and then in hidden layer 3.
6. Clones produced in one hidden layer may move to other hidden layers.
7. Networks that have been created with no hidden nodes or connections can use cloning.
8. Clones can be cloned.
Learning Threads

**EasyNN-plus** neural networks can be trained using more than one learning thread. Some networks will learn much faster with multiple threads but there is no simple relationship between the learning speed and the number of threads. The optimum number of threads can only be found by experiment.

Networks using multiple threads can learn more than ten times faster than when using a single thread. The disadvantage with using multiple threads is that the final results may vary slightly.

Up to eight learning threads can be started using the threading control in the **Controls** dialog.

All **EasyNN-plus** threads run at a low priority so that other applications and the operating system do not slow down while the learning threads are running.
Main Window's elements

The main windows are selected with a single click on the view Toolbar button.
The main EasyNN-plus application window. There are nine views and six toolbars. The views and toolbars can be selected, moved to any position, switched on and off as required. The grid view is selected with all the toolbars switched on.
The Grid view shows all the Examples arranged in rows and all the Input/Outputs arranged in columns. The first column contains the Example types and names. The first row contains the Input/Output types and names. Everything on the Grid can be edited by moving to the cell containing the value and then pressing the enter key to start the Edit Grid dialog. The cell can be selected either using the arrow keys or the mouse. A single click will select the cell and a double click will start the Edit Grid dialog. A double click on the Example name cell will select the whole row and a double click on the Input/Output name cell will select the whole column. The row or the column can be deselected by pressing the Esc key.

Creating a New Grid

A new Grid is created by pressing the new toolbar button or using the File > New menu command. The new Grid will be empty except for a horizontal line, a vertical line and an underline marker that shows the current position in the Grid. New Grid rows and columns are created at the current position.

Creating the first Example row and Input/Output column

Press return and a prompt will appear that says “Create new Example row?”. Answer Yes. Another prompt will appear that says “Create new Input/Output column?”. Answer Yes again. You will now see that the Grid has one cell containing “?”, a row name containing “T:0” and a column name containing “I:0”. The “?” indicates that the cell has no value, the “T:0” indicates that it is a Training Example in row 0 and the “I:0” indicates that it is an Input in column 0. Press return again and an Edit Grid dialog box will appear that allows you to enter the cell value. Using the same dialog you can change the Input/Output column name, mode and type. The dialog can also be used to change the Example row name and type.

Creating more Input/Output columns.

Move the marker one cell to the right by pressing the right arrow or tab key. Now press the return key.
and a prompt will appear that says "Create new Input/Output column?". Answer Yes. Press return again and the Edit Grid dialog box will appear. This time the Example row will already contain a name and type. You can set the cell value, the Input/Output name and the type can be set to "I:" for input, "O:" for output, "X:" for exclude or "S:" for serial. The mode can be set to "Real", "Integer", "Bool" "Text" or "Image". Any type of Input/Output column can be inserted into the Grid using the functions on the Insert menu.

Creating more Example rows.

Move the marker one cell down by pressing the down arrow key. Now press the return key and a prompt will appear that says "Create new Example row?". Answer Yes. Press return again and the Edit Grid dialog box will appear. This time the Input/Output column will already contain a name and type. You can set the cell value, Example name and the type can be set to T: for training, V: for validating, Q: for querying or X: for exclude. Any type of Example row be inserted into the Grid using the functions on the Insert menu.

Copying Example rows and Input/Output columns.

Double click on the name to select the whole row or column. Cut will remove the selected row or column and place it on the clipboard. Copy will place a copy of the selected row or column on the clipboard. Paste will insert a copy of the clipboard before the currently selected row or column. If the clipboard contains a row then Paste will insert the row into the Grid. If the clipboard contains a column then Paste will insert a column into the Grid. The invisible Grid data, limits and defaulted values, will be regenerated after a Paste column thus any neural network that has already been generated from the Grid will be invalidated.

1. Function
   *10
   if a function is selected it is displayed in the top left corner of the grid on a green background.

2. Column names
   Diggers Spades Diam
   Input/Output column names can be entered or produced automatically. The column types can use different colors.

3. Row names
   Query
   Hole 01
   Hole 01
   Hole 02
   Hole 03
   Hole 04
   Example row names can be entered or produced automatically. The row types can use different colors.

4. Grid cells
Grid cells can contain values in real, integer, bool, text or image format. Any notes can be added to grid cells. Special key notes are used to trigger diagnostic dumps in the trace file.

The grid range is outlined in green. If a function has been set a grid range can then be set by left clicking on the top left cell followed by left clicking on the bottom right cell. The range can then be used to limit the execution of functions on the grid.
The **Network** view shows how the nodes in an **EasyNN-plus** neural network are interconnected. When this view is opened the **Network Editor** starts. 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.

**How to create a new neural network**

A new neural network can be created from the Grid by pressing the **New Network** toolbar button or selecting **Action > New Network**. This will produce the New Network dialog. This dialog allows the neural network configuration to be specified. The dialog will already contain the necessary information to generate a neural network that will be capable of learning the information in the Grid. However, the generated network may take a long time to learn and it may give poor results when tested. A better neural network can be generated by checking Grow hidden layer 1 and allowing **EasyNN-plus** to determine the optimum number of nodes and connections.

It is rarely necessary to have more than one layer of hidden nodes but **EasyNN-plus** will generate two or three hidden layers if Grow hidden layer 2 and Grow hidden layer 3 are checked.

The time that **EasyNN-plus** will spend looking for the optimum network can be controlled by setting the Growth rate variables. Every time that the period expires **EasyNN-plus** will generate a new neural network slightly different from the previous one. The best network is saved.

**How to use the Network editor**

To create a network manually using the Network Editor start with a suitable Grid and then use New Network but do not check Connect layers. This will create the optimum number of nodes but the weights between the layers will not be connected.

To connect and disconnect nodes the source and the destination of the weight connections need to be selected. This is done using the mouse. The left button selects the source and the right button selects
the destination. First left click on a node to select the source and then right click on a node to select the destination. The selected source node will now have a wide red border. The other nodes in the source layer will have a narrow red border. The destination node will now have a wide blue border. The other nodes in the destination layer will have a narrow blue border. The right click will also open a menu of functions that are used to change the network connections or add and delete nodes.

Any nodes or layers can be connected to any other nodes or layers. Feed forward, feedback and skip layer connections are possible.

The number of hidden layer nodes can be increased or decreased and hidden layers can be added if needed. The nodes will be reconnected using the connections that are held in the slave memory. These connections will be as close as possible to its previous state. It can then be edited further as required.

1. **Input Node**
   - **Diggers**
   - Input nodes are connected to the input columns in the grid.

2. **Hidden Node**
   - Hidden nodes are fully connected to input nodes, output nodes or other layers of hidden nodes.

3. **Output Node**
   - **Kilograms**
   - Output nodes are connected to the output columns in the grid.

4. **Connections**
   - The connections between the nodes are weighted. The color represents the polarity and the thickness represents the weight value. The weight values and polarity change as the network learns.
Left click menu item required.
The **Learning Progress** view shows how learning is progressing. Up to 5000 graph points are recorded. This is sufficient for over 200,000,000 learning cycles. The graph is produced by sampling these points. The horizontal axis is nonlinear to allow the whole learning progress to be displayed. As more cycles are executed the graph is squashed to the left. The scaled errors for all example rows are used. The red line is the maximum example error, the blue line is the minimum example error and the green line is the average example error. The orange line is the average validating error. The errors are also shown on the **Example errors view**.
The learning progress graph shows the maximum, average and minimum training error. The average validating error is shown if any validating examples rows are included. The graph is squashed to the left as learning progresses so that the whole learning curve is visible. Up to 200,000,000 learning cycles can be displayed on the graph.

**Vertical scale**

The default vertical scale is 0.0000 to 1.0000 but can be adjusted by clicking on the vertical scaling and zoom toolbar icons. Details of the curve can be seen by scaling and zooming the graph.

**Horizontal scale**

The horizontal scale is renumbered as learning progresses by packing to the left. The entire learning curve is visible.
Network

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input</th>
<th>Hidden 1</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>11</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Weights</td>
<td>77</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

The current counts of nodes and connection weights in the neural network. This is updated if the network is reconfigured automatically or manually.

Diagnostic Array data

<table>
<thead>
<tr>
<th>Column</th>
<th>Name</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>PPV</th>
<th>NPV</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Left</td>
<td>20</td>
<td>89</td>
<td>4</td>
<td>9</td>
<td>83.33%</td>
<td>9.10%</td>
<td>18.35%</td>
<td>69.23%</td>
<td>23.77%</td>
</tr>
<tr>
<td>12</td>
<td>Ping</td>
<td>15</td>
<td>0</td>
<td>101</td>
<td>6</td>
<td>12.93%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>5.41%</td>
<td>17.27%</td>
</tr>
<tr>
<td>13</td>
<td>Over</td>
<td>20</td>
<td>86</td>
<td>2</td>
<td>6</td>
<td>93.33%</td>
<td>6.57%</td>
<td>24.56%</td>
<td>75.00%</td>
<td>27.87%</td>
</tr>
</tbody>
</table>

A copy of the data in the Diagnostic Array. The values are updated while learning progresses.

Learning and validating

- Learning rate: 0.500000
- Momentum: 0.800000
- Maximum error: 0.008779
- Average error: 0.001182
- Minimum error: 0.000000
- Target error: 0.000000
- Validating examples: 122
  - Within 10.0% range. Score: 0
  - Correct if rounded. Score: 3
- Target score: 122 119 needed
- Validating: 2.46% 0K

The target error, learning and validating progress details. The validating and scoring results are shown when the number of validating examples is greater than 0.
The Predictions view has two boxes that show how close the predicted value for each example output is to the true value. The training and validating examples are shown in separate boxes. The position of the values on both axes are 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 the predicted values are very close to the true values then the dots will be on the diagonal line. If most of the dots are close to the diagonal line the network may be over-trained and it may not generalize well.
The Associations view. All the columns in the Grid can be checked to see if they interact with the other columns to form associations and clusters. The importance of the association for every column is determined with respect to every other column. All the columns are set to be Inputs. Each column in turn is changed to Output and a network is created. The networks are trained so that the most important Input can be determined. The Output is paired with the most important Input. The strength of the association is the importance of the most important input for the currently set output. Clusters of Columns are also formed during the Associating process.

### Associations

The associations found in Diggers.tvg

Stopped listing up to 7 associations in 7 columns. The list is finished.
The last association found was for column number 6 named "Weight".

<table>
<thead>
<tr>
<th>Associated columns</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>3  Temp</td>
<td>6  Weight</td>
</tr>
<tr>
<td>2  Diameter</td>
<td>0  Diggers</td>
</tr>
<tr>
<td>4  % Rocks</td>
<td>2  Diameter</td>
</tr>
<tr>
<td>5  % Clay</td>
<td>3  Temp</td>
</tr>
<tr>
<td>0  Diggers</td>
<td>3  Temp</td>
</tr>
<tr>
<td>6  Weight</td>
<td>0  Diggers</td>
</tr>
<tr>
<td>1  Spades</td>
<td>6  Weight</td>
</tr>
</tbody>
</table>
The **Column Values** view. All Input/Output column values are shown graphically. The left hand vertical scale shows the scaled value from 0 to 1. The right hand scale shows the real value calculated using the highest and lowest values in the column. The Equivalent integer, bool, text or image name is shown when it is available. The **blue** line is the true value, the **red** line is the running average and the **black dotted** line is the overall trend. The **lavender** line is the Risk number. This is the number of times that any forecasted value had to be adjusted before the network produced an error close to the error produced before the forecast.

Any column can be remembered using the right click menu so that two columns can be displayed and compared. The right click menu is also used to select any of the lines and switch the grid example row names on and off. Excluded rows and missing values can be shown as green **lines**. The left click is used to select and move to any grid row. The currently selected grid row is shown by the vertical dotted line.
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. The order can be switched to ascending from the smallest error by clicking on the sort order toolbar icon.
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. The inputs are shown in the descending order of importance from the most important input. The order can be switched to ascending from the lowest importance by clicking on the sort order toolbar icon.

<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></td>
</tr>
<tr>
<td>0</td>
<td>Diggers</td>
<td>6.8722</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Temp</td>
<td>5.6053</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Diameter</td>
<td>5.2092</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>% Clay</td>
<td>1.4479</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>% Rocks</td>
<td>1.0651</td>
<td></td>
</tr>
</tbody>
</table>

**Diggers.tvq** 86 cycles. Target error 0.0100 Average training error 0.009833
The first 6 of 6 Inputs in descending order.
The Sensitivity view shows how much an output changes when the inputs are changed. The inputs are all set to the median values and then each in turn is increased from the lowest value to the highest value. The change in the output is measured as each input is increased from lowest to highest to establish the sensitivity to change. Insensitive inputs are not shown. The inputs are shown in the descending order of sensitivity from the most sensitive input. The order can be switched to ascending from the least sensitive input by clicking on the sort order toolbar icon.
Macros.

Macros are recorded while EasyNN-plus is being used. Recording is started with Record on the Macro menu or Ctrl + F2. Recording is stopped using Stop recording on the Macro menu or Ctrl + F3. EasyNN-plus macros are text files with the .ens extension. Functions are fully recorded when possible but some are only partially recorded or they are nested within other functions. The right click, grid pre-processing functions are not recorded.

Script Commands can be added to macros using Commands on the macro menu. Recording can be continued after adding script commands or recording can be stopped and the commands can be played immediately. Multiple script commands are separated with a ; character. For example, if the line f; stl; fps(0,0,1,100) is entered it will be split into three separate script commands in the macro. If the macro is then played back the commands will cause the currently open network to forget learning, start learning and then stop learning after 100 cycles. Alternatively recording can be stopped and the commands played by adding three colon characters. The example would become f; stl; fps(0,0,1,100); :::

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. Recording can be continued after adding an expanded script file.

A script file can be added at the end of a macro using Add Script File on the macro menu. This will put a RunScriptFile(script file); command into the macro. When the macro is played back it will play the recorded functions and script commands. It will then run the script file and terminate the macro.

Playback is started using Play on the Macro menu, Ctrl + F4 or using Run Script File on the File menu. 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.

Scripts.
**EasyNN-plus** 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. They can also be passed to **EasyNN-plus** as a parameter.

Scripts can contain commands to import the data, build the neural network, control learning, validating and querying.

**Warning:** Script commands call low level functions. The command parameters are validated but some error checking has to be disabled so that the scripts will run. All scripts should be designed and tested very carefully. Script commands that will overwrite existing files should only be used when backup files are available.

Script files based on the currently loaded network can be produced using Generate Script File on the File menu. The generated files will contain commands to create the grid columns and set the network controls. Commands to create the grid rows, import the data and build the neural network can then be added.

Scripts can be tested by stepping through them one line at a time. Start the script with 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.

"C:\Program Files\EasyNN-plus\EasyNN.exe" "C:\Program Files\EasyNN-plus\Samples\ascript.txt"

The script imports atest.txt and uses it to create a network. The files ascript.txt and atest.txt are included in the Samples folder.

**Macro and Script Commands.**

**Comments:** Lines starting with any non alpha character can be used for comments.

**Parameters:**

String parameters are checked to be valid strings.
Integer and double parameters are checked to be within the valid range.
Bit parameters are checked to be 0 (false) or above 0 (true).
Macro parameters are produced while recording.

Macro functions can be included in Scripts but they must be tested very carefully. Script command names are shown in black, macro function names are shown in red. When a macro function has exactly the same parameters as a script command it is listed below the command. All macro functions are case sensitive. Script commands in full or as short names can be in any case.

**AddCol** Adds a column
AddRow
Adds a row

autosave
Set AutoSave
as
string File base name
intCycles per save
bitSave if lower error
bitSave if validating results improve

autosaveassociationsfile
asas
AutoSaveAssociationsFile
string File name

autosaveclustersfile
ascl
AutoSaveClustersFile
string File name

backto
Go back to label
bt
string Label
intRepeats

Warning: Only one backto or bt command is allowed in a script.

createcolumns
Creates columns
cc
int Number of columns

ChangeNetwork
Increase nodes
int Number of nodes to add to hidden layer
int Hidden layer number 1, 2 or 3
**ChangeRows** Change rows

intChange to; 0 Training, 1 Validating, 2 Querying, 3 Excluded
intFirst row to change
intNumber of rows to change

**clipcolumn** Clip column
clc

intColumn number
bitCut clipped column

**cliprow** Clip row
clr

intRow number
bitCut clipped row

**cloningoff** Cloning off
cof

**cloningon** Cloning on
con

**Close** Closes the application

**CloseDocument** Closes the document

**columntypeandmode** Set columns type and mode
ctm

**SetColumnsTypeAndMode**

intFirst column to set
intNumber of columns to set
intType; 0 Input, 1 Output, 2 Exclude, 3 Serial
intMode; 0 Real, 1 Integer, 2 Bool, 3 Text, 4 Image

**CompleteEditGrid** Complete edit grid cell

intColumn
intRow
doubleValue
string String value
intMode
intType

**createnetwork** Creates a network from the Grid
cn
intNodes in hidden layer 1
intNodes in hidden layer 2
intNodes in hidden layer 3
bitOptimize hidden layer 1
bitOptimize hidden layer 2
bitOptimize hidden layer 3

creatorows Creates rows

intNumber of rows

DeleteCol Deletes a column

intColumn
bitAsk (ignored)

DeleteRow Deletes a row

intRow
bitAsk (ignored)
bitModified
bitDuplicate
bitScale

disableresults Disable results columns

intFirst column
intNumber of columns

disablesinglestep Disable single step

dss

EditCopy Copy selection to clipboard

intColumn
intRow

EditCut Cut selection to clipboard

intColumn
intRow

EditGrid Edit grid cell

intColumn
intRow
intType
string Column name
string Row name

EditPastePaste from clipboard

bitColumn clipped
intColumn
bitRow clipped
intRow

endofscriptEnd of script

eos

enableresultsEnable results columns

intFirst column
intNumber of columns

enablesinglestepEnable single step

exitExit
x

bitPrompt to save

extendrangeExtend the training range
exr
ExtendRange

intColumn
doubleValue

forgetForget
f

bitRestart random number sequence

ForgetLearningForget

bitAsk
bitRestart
bitReset errors
bitReSeed
doubleRange
**fixedperiodstops** Set fixed period stops

*fps*

**SetFixedStops**

bitStop after seconds
doubleSeconds
bitStop after cycles
intCycles

**importbinaryfile** Import binary file

*ibf*

**SetImportBinaryFile**

string File name
intInitial bytes to skip
intBytes per column
intColumns per row

**ignoreerrors** Ignore errors

*ie*

bitCancel

**importimagefiles** Import bitmap image files

*iif*

**ImportImageFiles**

string Path to folder
string File name
intFirst column
intFirst row
bitImport all bitmaps in folder
string Prefix

**importtextfile** Import text file

*itf*

**ImportTextFile**

string File to import
bitUse first line as column names
bitUse first word(s) on line for row names
bitTab delimiters
bitComma delimiters
bitSpace delimiters
bitStop (period) delimiters
bitColon delimiters
bitSemicolon delimiters
bitInitialize Grid for text and image modes

**importxlsfile** Import XLS file
ixf
ImportXLSFile

string File to import
string Sheet name
bitset column names from first row
bitset row names from first column

learningcontrols Set the learning controls
lc
SetLearningControls

double Learning rate
double Momentum
bitset Decay learning rate
bitset Momentum
bitset Optimize learning rate
bitset Optimize Momentum

logscriptcommands Log all script commands including failures.
lsc

string Log File name

networkreconfiguration Set network reconfiguration controls
nr
SetReconfigurationControls

bitset Allow manual reconfiguration
bitset Grow hidden layer 1
bitset Grow hidden layer 2
bitset Grow hidden layer 3

NewNetwork New network from current grid

int Inputs (ignored)
int Nodes in hidden layer 1
int Outputs (ignored)
bitset Reset (ignored)

newweight New weight connection from node to node
nw

int From node
int To node

deleteweight Delete weight connection from node to node
dw

int From node
int To node
open Open a network file

string File to open

order Set the sort order
or Order

bit0 Descending, 1 Ascending

paste Paste clip

pause Pause and leave script
ps

bit0 No prompt, 1 Prompt to continue

query Query from file
q

SetFileQuery

string Query file
string Results file
bitNo row names in query file
bitNo column names in query file
bitTab delimiters
bitComma delimiters
bitSpace delimiters
bitStop (period) delimiters
bitColon delimiters
bitSemicolon delimiters
bitShow results file
bitComma delimiters in results file

refresh Refresh display when paused
rf

reset Reset AutoSave
r

bitSaved error
bitSaved validating

readbinaryfile Read binary file
rbf

string File name
intBytes to skip
intBytes to read
intRepeats

**rowtype** Set row types
rt

intFirst row to set
intNumber of rows to set
intType; 0 Training, 1 Validating, 2 Querying, 3 Exclude

**RunScriptFile** Run a script file

string Script file name
bitEdit (ignored)

**save** Save
s
string File name
bitDo not change Window title

**SetAutoSave** Set auto save

string File name
intCycles per save
bitError decrease
bitValidating increase
bitDo not ask
bitDetails on exit

**setbinarysizes** Set binary sizes
sbs

intBytes per Column
intColumns per row

**setjoglevel** Sets the jog level
sj
**SetJogLevel**

int0 None, 1 Low, 2 Medium, 3 High

**setnoisel** Sets the noise level
sn
**SetNoise**

intNoise level 1 to 10
intOn/Off 1/0
setview: Sets the view
sv
SetView

intView; 0 Blank, 1 Grid, 2 Associations, 3 Learning graph, 4 Network, 5 Errors, 6 Importance, 7 Sensitivity, 8 Predictions, 9 Column graph
bitSort; 0 Ascending, 1 Descending

Warning: Forcing inappropriate views using SetView, setview or sv in a macro may cause a crash.

Warning: When running from the command line, a script or macro that includes any SetView, setview or sv may cause a crash if the script or macro fails before it reaches the end. This is because the view does not exist during command line runs. To avoid the problem the change of view is delayed until the script has ended.

showwindow: Shows the Window
sw
int0 Hide, 1 Normal, 2 Minimize, 3 Maximize, 4 Show but no activate, 5 Show and activate

Warning: If a script or macro includes showwindow(0) or sw(0) EasyNN-plus will run hidden. It will remain hidden until another showwindow is executed with a parameter greater than zero. Scripts and macros that do not make EasyNN-plus visible may leave the process hidden and running. If the script or macro ends or fails while EasyNN-plus is hidden an attempt is made to become visible. Do not assume that this will always be successful. Some script failures cannot be detected by an hidden process. Windows Task Manager can be used to kill hidden processes.

specialfiles: Special files
sf
SpecialFiles

bitSave weights biases
bitSave all the grid
bitSave the graph points
bitSave the weights connections
string Terminal file name for immediate save
bitSave as comma delimited files (csv)
bitSave example errors
bitSave sensitivities
bitSave importances

StartAssociating: Start associating

bitSkip excluded columns
bitClear existing associations

StartForecasting: Start forecasting

intPeriod
intSteps
startlearning Start learning

stl
StartLearning

Warning: The time taken by a process between starting and stopping is recorded for use as the maximum wait time in the WaitForStop function. When the macro is played back the recorded maximum wait time is increased by a generous margin to allow for timing variations. This margin is normally sufficient for quite wide variations but on much slower systems or when the system is heavily loaded some problems may occur.

stoplearning Stop learning or associating

spl
Stop

targeterrorstops Set the target error stops
tes
SetLearningStops

bit0/1 Average error / all errors
doubleTarget error

validatingcontrols Set the validating controls
vc
SetValidatingControls

int Training cycles before first validating cycle
int Training cycles per validating cycle
int Percentage of training examples to change to validating

validatingstops Set the validating stops
vs
SetValidatingStops

double Percentage validating that must be in range or correct
doubie +/- Range percentage
bit0/1 Test in range / correct after rounding
bitStop if validating examples decreases
bitStop if target score reached
int Target score
bit Inhibit range and correct validating
bitStop if average validating error is increasing

wait Wait

w

int Milliseconds to wait
**waitForStop**
Waits for stop

**wfs**
**WaitForStop**

int Maximum milliseconds to wait
int Wait for; 14 Learning, 16 Associating, 23 Forecasting

---

1. **Notepad**

   ![test.ens](image)
   A simple macro opened in Notepad

2. **Functions**

   ```
   SetView(1, 1)
   Open(C:\EasyNN\Samples\Diggers.tvq)
   NewNetwork(6, 3, 1, 1)
   ```

   A series of recorded functions that open a file, creates a network, starts learning, waits for it to finish and then saves the trained network.
Toolbars

Most menu commands also have toolbar buttons. The toolbars can be positioned anywhere.
Standard facilities are available from the main toolbar. More extensive facilities are available using other toolbars or the menus.

1. **New**
   - Create a new document with a blank neural network Grid.

2. **Open**
   - Open an existing neural network.

3. **Save**
   - Save the active neural network document.
4 Import

Import a TXT, CSV, XLS, BMP or binary file into the neural network Grid.

5 Export

Click to open the Export special files dialog.

6 Cut

Cuts the selected row or column and puts it in the clipboard.

7 Copy

Copies the selected row or column to the clipboard.

8 Paste

Pastes the row or column to the selected grid position.

9 Undo

Undo the previous edit.

10 Redo

Redo the previous edit.

11 Check Grid

Checks that the grid is suitable to produce a network, has input and output columns and
at least two different values in each column.

12 New network
Opens the New network dialog to create a neural network from the grid.

13 Change controls
Opens the controls dialog to set the learning, validating and other controls.

14 Start learning
Starts the learning process.

15 Stop
Stops the active process.

16 Start associating
Starts the associating process.

17 Forget learning
Forget learning. The network weights are set to small random values. The random number sequence is normally started at the same point when the network forgets but it can be restarted from a point based on the current time.

18 Query
Open the query dialog and starts the query process.

19 Add Query
Click to add a query row.

Change query

Reduces, increases, minimizes or maximizes the selected query value.
Advanced facilities are available from this toolbar. Validating rules can also be edited. More advanced facilities are available using the menus.

1. Duplicate Rows
   Duplicate all example rows.

2. Delete Duplicates
   Deletes the example rows that have been duplicated.

3. Exclude Incomplete
   Excludes incomplete example rows.

4. Freeze
Opens the freezes dialog.

5 Cloning
   Enable or disable hidden node cloning.

6 Validating rules
   Edit or add validating rules.

7 AutoSave
   Opens the AutoSave dialog for setting the cycles and the files.

8 Reset AutoSave
   Resets AutoSave. The lowest error is set to 1.0, the highest score is set to 0 and the highest successful validating rows is set to 0%
Some of the full versions extra facilities are available in this toolbar.

1. **File query**
   - Query from a query file and produce a results file.

2. **Trim**
   - Open trimming dialog box.

3. **Forecast**
   - Starts forecasting of any serial grid columns.

4. **Cross Validate**
   - Cross validate the network.
1. **Missing**

Open the missing data dialog and change the actions.

2. **Tidy**

Tidy all unused texts, images, rules, notes, outputs and delays.

3. **Get support**

View FAQ or email support.
Views allow all the neural network features, learning details and input/output values to be displayed or printed.

1. **Grid view**
   - View the grid.

2. **Network view**
   - View the network.

3. **Increase width**
   - Increase the width of the network view.

4. **Decrease width**
   - Decrease the width of the network view.
5 Learning view

View the learning graph.

6 Increase height

Increase the height of the learning graph view.

7 Decrease height

Decrease the height of the learning graph view.

8 Predictions view

View the predictions.

9 Column view

View the currently selected column values.

10 Previous column

Move to previous column values view.

11 Examples view

View the example errors in order.
12 Sensitivity view
View the sensitivity of the inputs for any output.

13 Zoom in
Zoom in the current view.

14 Associations view
View the associations.

15 Next Column
Move to next column view.

16 Remember column
Remember the current column view so that it can be compared with other columns.

17 Importance view
View the importance of the inputs in order.

18 Order
Reverse the order of increasing or decreasing views.

Zoom out

Zoom out the current view.

Types & Indexes

Show row and column types and indexes.

Information

View the details of the currently open network file.
The network can be reconfigured by increasing or decreasing hidden nodes.

1. **Increase nodes 1**
   Increase the nodes in hidden layer 1.

2. **Decrease nodes 1**
   Decrease the nodes in hidden layer 1.

3. **Increase nodes 2**
   Increase the nodes in hidden layer 2.

4. **Decrease nodes 2**
   Decrease the nodes in hidden layer 2.
5. **Increase nodes 3**

Increase the nodes in hidden layer 3.

6. **Decrease nodes 3**

Decrease the nodes in hidden layer 3.
Main Dialogs

All of the facilities in EasyNN-plus can be accessed and controlled from dialogs.
Information. Details of the neural network file are displayed when the file is loaded or when Information is selected on the view menu.

1. Save

Click to save all the information details to a text file.
2 Refresh
Click to refresh the information while the neural network is learning.

3 Continue
Click to continue and close the dialog.

4 History
Click to show the file loading and saving history.
Import TXT CSV file

**Import**  Import a text or comma separated file.

**EasyNN-plus** can import a text or a csv file to create a new Grid or to add new example rows to an existing Grid.

1. **File > New** to create a new Grid or **File > Open** to add rows to an existing Grid.

2. **File > Import...**

3. Open the file that is to be imported.

4. In the first dialog check all the characters that are to be used for column delimiters. Line ends are used for row delimiters.

5. Any words before the first delimiter on each line can be used for row names. If no row names are available then **EasyNN-plus** can generate numbers for row names.

6. Press OK.
Check the characters to use as Input/Output column delimiters. The number of columns created will be one more than the number of delimiters found. If no delimiters are checked a single column will be created.

Row names
Set how to produce Example row names.

Row types
Set the example row type for all imported rows or ask for row types while importing.

OK
Click to accept all settings and close the dialog.

Cancel
Click to reject all settings and close the dialog.
The second Import dialog

1. Press Next line to step through the file until the first line to be imported is shown. Press First line to go back to the start of the file if needed.

2. Press Use line or Set names according to the instructions in the dialog.

3. Set the row types as the file is being imported.

4. Set the column types and modes when the first line is imported.

5. Press OK. The rest of the file will be imported.

**EasyNN-plus** will prompt for any image file locations it needs while the file is being imported. Warnings will be produced for any lines in the file that are not suitable for the Grid.
Click to set the names in the Grid

2 Use line
   Use line
   Click to use the current line

3 Next line
   Next line
   Click to move to the next line.

4 First line
   First line
   Click to go back to the first line

5 Cancel
   Cancel
   Click to cancel the import.
The Import Input/Output Column dialog

The dialog needs to be completed for each column or range of columns. The Mode and Type are set automatically depending on the contents of the columns. The mode is not always correct and it should be checked. Sometimes Integer mode columns will be set to Bool because the first two values appear to be boolean. Text mode columns are always initially set to Exclude.

1 **OK button**

Click to accept all settings and close the dialog.

2 **Cancel button**

Click to reject all settings and close the dialog.
The Example Row Type dialog

1. If 'Ask while importing' was set in the first import dialog the optional Example Row Type dialog needs to be completed for each row or range of rows.

1. **OK button**
   
   Click to accept all settings and close the dialog.

2. **Cancel button**
   
   Click to reject all settings and close the dialog.
Import XLS file

Import XLS file

C:\EasyNN\Samples\Test1.xls

Name of the worksheet to import: Sheet1

Ask for Example row types while importing

OK

Worksheet name

1. Worksheet name

C:\EasyNN\Samples\Test1.xls

Name of the worksheet to import: Sheet1

OK

Worksheet name

Enter the name of the worksheet to import.

OK

Import. Simple spreadsheet files produced by Excel in xls format can be imported directly into EasyNN-plus but a much greater degree of import control is available if the files are first saved as comma separated csv types. Files in xls format are limited to 256 columns and 65536 rows.

If the xls file is to be imported directly to produce a new Grid proceed as follows.

1. File > New to create a new Grid.

2. File > Import...

3. Open the xls file that is to be imported.

4. Enter the name of the sheet that is to be imported if it is different to Sheet1.

5. Answer the messages according to how you want to create the Grid Input/Output and Example names.

6. Step through all the columns setting the type and mode.
Click to accept and close the dialog.

Click to reject and close the dialog.
**EasyNN-plus** can import a binary file to create a new **Grid**.

1. **File > New** to create a new Grid.

2. **File > Import...**

3. Open the binary file that is to be imported.

**The Import dialog**

4. Enter the Sizes and first Block details.

5. Press **Next** until the file has been imported. The block details can be changed as needed.

6. Press **Finish** to complete the import.
Enter the number of bytes to import for each column and the number of columns to import for each row.

2 Block

Enter the data block details as the number of bytes to skip, then the number of bytes to read and the number of times to repeat the process.

3 Current step

Set sizes. The current process step.

4 Next

Click to import the next block.

5 Finish button

Click to import all blocks in the file.

6 Cancel

Click to stop the import and close the dialog.
**EasyNN-plus** can import a bitmap file to create a new **Grid**.

1. **File > New** to create a new Grid or **File > Open** to add images to an existing Grid.

2. **File > Import...**

3. Open the **bmp** file that is to be imported.

**The Import dialog**

4. Check the box if all images in the folder are being imported.

5. Enter the first row, first column and file name prefix for the images.

6. Press OK to start the import.
Check to import all bitmap files in the currently selected folder.

Location

The location of the currently selected folder and file.

Image distribution

The first row and column to be used for the import.

Image name

The prefix for the image names.

OK button

Press to accept the settings and close the dialog.

Cancel button

Press to close the dialog.
**Edit Grid.** Grid cells can be displayed and edited using the edit grid dialog. Entering any character will open this dialog. Other grid editing facilities are available using the edit menu.

1. **Value**
   - **Value**
     - **Value**
       - 150.0000
     - [Min: 150, Max: 350] scaled [0, 1] = 0
     - Enter the grid cell value. The value will be scaled to be from 0 to 1 using the column minimum and maximum.

2. **Example row**
   - **Example row**
     - **Hole 07**
     - **Type**
       - Training, Validating, Querying, Exclude
     - **Input/Output column**
       - Exclude column from file query results
     - **Diameter**
     - **Mode**
       - Real, Integer, Bool, Text, Image
     - **Type**
       - Input, Output, Exclude, Serial
Enter the example row name and set the type of row.

**Input/Output column**

Enter the column name. Set the column type and mode. If "Exclude column from the file query results" is set the column will still be used in the query but will not appear in the results file.

Click to complete the edit and close the dialog.
The New Network dialog is used for the creation and growth of a new neural network from the values in the Grid. During optimization a new network is created every time the period of cycles or seconds completes. The number of hidden nodes in each layer that is checked grows from minimum to maximum. Each network is checked by training it for a short period. The network that produces the lowest error is selected for further training and validating.

Frameworks can be created without connections if the Network Editor is to be used.

1. **Growth rate**
   - Growth rate
   - Change every cycles or seconds
   - A network is produced when the cycles or seconds elapses until the optimum network is found.

2. **Connect layers**
   - If checked all connections are created automatically. If not checked the optimum number of nodes will be created but not connected together. This is so that the network editor can be used to produce complex configurations.

3. **Set freezes**
   - If not checked the optimum number of nodes will be created but not connected together. This is so that the network editor can be used to produce complex configurations.
Opens the Freezes dialog. Set the node types to freeze, the initial range, the direction of freezes and the freeze decrement conditions.

4 Set trimming

Set trimming

Opens the trimming dialog to set up weight trimming, unimportant input node deleting and highest error example row excluding.

5 Set cloning

Set cloning

Sets hidden node cloning. If learning is not progressing the hidden nodes are tested. A node with a high net input that is contributing to the learning process is cloned.

6 Input layer

Input layer

- Created with 6 nodes connected to grid inputs

The number of nodes in the input layer is determined by the number of input columns in the grid.

7 Create hidden layers

Create hidden layers

Must be checked to create or grow any hidden layers.

8 Grow hidden layers

Grow hidden layers

- Grow layer number 1 2 3
- from minimum nodes 3 2 2
- to maximum nodes 9 5 5

If the layer number is checked the layer is grown from the minimum to the maximum number of nodes.

9 Output layer
The number of nodes in the output layer is determined by the number of output columns in the grid.

**OK button**

Press to accept all the settings.

**Cancel button**

Press to reject all the settings.
When a new neural network is created the controls are preset to values that will usually be suitable for the network to learn from the training data in the Grid. The training controls can often be optimized to result in much faster learning. All controls are set using the Controls dialog.

Setting the Learning Rate

The Learning Rate will be preset to 0.6 when a new neural network is created. It can be changed to any value from 0.1 to 10. Very low values will result in slow learning and values above 1.5 will often result in erratic learning or oscillations. Check "Optimize" to allow EasyNN-plus to determine the learning rate automatically by running a few learning cycles with different learning rate values. Check "Decay" to automatically reduce the learning rate during learning if erratic learning or oscillations occurs. Check "Adapt" to maintain the learning rate that produces the fastest learning.

Setting the Momentum

The Momentum will be preset to 0.8 when a new neural network is created. It can be changed to any value from 0 to 0.9. Check "Optimize" to allow EasyNN-plus to determine the momentum automatically by running a few learning cycles with different momentum values. Check "Decay" to automatically reduce the momentum during learning if oscillations occur. Check "Adapt" to maintain the momentum that produces the fastest learning.

Setting the Accelerator

The Accelerator will be preset to 0.0 when a new neural network is created. It can be changed to any
value from 0 to 3. Check "Optimize" to allow EasyNN-plus to determine the accelerator automatically. Check "Decay" to automatically reduce the accelerator during learning if oscillations occur.

**Setting the Target Error**

The Target Error will be preset to 0.01 when a new neural network is created. It can be changed to any value from 0 to 0.9 but values above 0.2 usually result in under trained networks. Learning is set to stop when the average error drops below the target error. This can be changed so that all errors must drop below the target or the predicted output is in range of the actual output.

**Setting the Validating Controls**

Validating cycles only occur when validating examples are included in the Grid. A number of training examples can be randomly changed to validating examples using the Select control or they can be entered directly into the Grid. The cycle values are preset so that 100 learning cycles occur for every validating cycle. The number of learning cycles before the first validating cycles and the number of learning cycles per validating cycle can be changed to any positive values. Learning can be set to stop when the validating target is reached. The validating target can be set to any value up to 100%. The validating test can be either when the validating results are within a specific range or when the results are correct after rounding to the nearest whole number. Validating Rules can be used for rounding to the nearest multiple. The validating range can be set to any value from 0 to 50%.

When a new neural network is created the controls are preset to values that will usually be suitable for the network to learn from the training data in the Grid. The training controls can often be optimized to result in much faster learning. All controls are set using the **Controls** dialog.

**Setting the Learning Rate**

The Learning Rate will be preset to 0.6 when a new neural network is created. It can be changed to any value from 0.1 to 10. Very low values will result in slow learning and values above 1.5 will often result in erratic learning or oscillations. Check "Optimize" to allow EasyNN-plus to determine the learning rate automatically by running a few learning cycles with different learning rate values. Check "Decay" to automatically reduce the learning rate during learning if erratic learning or oscillations occurs.

**Setting the Momentum**

The Momentum will be preset to 0.8 when a new neural network is created. It can be changed to any value from 0 to 0.9. Check "Optimize" to allow EasyNN-plus to determine the momentum automatically by running a few learning cycles with different momentum values. Check "Decay" to automatically reduce
the momentum during learning if oscillations occur.

**Setting the Accelerator**

The Accelerator will be preset to 0.0 when a new neural network is created. It can be changed to any value from 0 to 3. Check "Optimize" to allow EasyNN-plus to determine the accelerator automatically. Check "Decay" to automatically reduce the accelerator during learning if oscillations occur.

**Setting the Target Error**

The Target Error will be preset to 0.01 when a new neural network is created. It can be changed to any value from 0 to 0.9 but values above 0.2 usually result in under trained networks. Learning is set to stop when the average error drops below the target error. This can be changed so that all errors must drop below the target or the predicted output is in range of the actual output.

**Setting the Validating Controls**

Validating cycles only occur when validating examples are included in the Grid. A number of training examples can be randomly changed to validating examples using the Select control or they can be entered directly into the Grid. The cycle values are preset so that 100 learning cycles occur for every validating cycle. The number of learning cycles before the first validating cycles and the number of learning cycles per validating cycle can be changed to any positive values. Learning can be set to stop when the validating target is reached. The validating target can be set to any value up to 100%. The validating test can be either when the validating results are within a specific range or when the results are correct after rounding to the nearest whole number. Validating Rules can be used for rounding to the nearest multiple. The validating range can be set to any value from 0 to 50%.

**Network reconfiguration**

Set manual or automatic hidden layer reconfiguration.

**Validating**

Set validating cycles and selection of rows. Up to half of the training rows can be selected at random for validating.

**Slow learning**

Set learning cycle delay for slow learning demonstration.
### Stops

- When the average training error is below
- When all the training errors are below
- When all predictions are in target range of the outputs
- When the average validating error is below
- If the average validating error is increasing
- Stop when 100% of the validating examples are
  - Within 10% of desired outputs
  - Correct after rounding
- Stop if the % of validating examples decreases
- Stop if the validating score is equal or above 0
- Stop if the validating score decreases
- Stop after 20,000 seconds
- Stop on 0 cycles

Stop when **Average error is below** the target error uses the total example errors divided by the number of examples.

Stop when **All errors are below** the target error uses every example error.

Stop when all **Predictions are in range** uses the difference between the actual desired output and the output predicted by the network. This process can take a long time and the resulting network may not generalize well.

Stop when **Average validating error is below** the target error uses the total validating example errors divided by the number of validating example.

Stop when **Average validating error is increasing**. Checks for increase in the average validating error for six cycle

### Presentation

- Balanced
- Random
- Grouped

In every training cycle the presentation of training examples to the neural network is normally in the forward direction from the first example training row to the last example training row. If **Balanced** is checked the presentation is first in the forward direction and then in the reverse direction. If **Random** is checked example training rows are presented at random. If **Grouped** is pressed the example training rows are presented in groups. Every row is presented at least once and possibly three times every training cycle.

### OK button

Press to accept all settings.

### Cancel button

Press to reject all settings.
The **Query** dialog allows the neural network to be tested or interrogated. Query inputs can be increased, decreased, minimized or maximized. The effect on the outputs is shown immediately and is written back to the Grid.

1. **Example**
   - The currently selected example row.

2. **Min**
   - Click button to set the selected input to minimum value.

3. **Max**
   - Click button to set the selected input to maximum value.
Decrease
Click button to decrease the selected input value.

Increase
Click button to increase the selected input value.

Set
Click button to set the selected input value.

Value to set
The selected input can be set to any value within the column range. If Extend Range is checked the range will be increased or decreased so the value can be above or below the column range.

Locks
Click here to select and lock the input.

Inputs
Click here to select the input.

Types to Select
Check example types to select.

11 First

First

Select first example.

12 Previous

Previous

Select previous example.

13 Next

Next

Select next example.

14 Last

Last

Select last example.

15 Outputs

Outputs

<table>
<thead>
<tr>
<th>Names</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilograms</td>
<td>50</td>
</tr>
</tbody>
</table>

Click here to select output row.

16 Stop/Start

Stop/Start Cycle

Click to cycle though all inputs seeking the high or low output.

17 Show Real

Show Real

Check to show real values.

18 Extend Range
Check to extend the range when a value is above or below the currently selected input range.

Set Seek Targets
Set the high and low seek targets.

Seek High
Click to seek the selected output high value by incrementing the selected input value.

Seek Low
Click to seek the selected output low value by incrementing the selected input value.

Add Query
Press to add a new querying row.

Exclude Query
Press to exclude the selected querying row.

Save Queries
Press to save all the queries.

Close
Press to close the query dialog.
**File Query.** Neural networks can be tested using a query file and produce a results file.

Neural networks produced by **EasyNN-plus** can be tested using an external text file and create a text or CSV results file.

Neural networks can be tested from a query file as follows.

1. Open the network file that you want to use.
2. Select the **Query > File Query** or press the "Query from file" toolbar button.
3. Open the file that contains the queries.
4. Open the file for saving the results of the queries.
5. Check the delimiters to use.
6. Check the Query file names if the names are not being used in the query file.
7. Press OK.
1. **Delimiters**
   - Column delimiters
     - Check characters to use for Input/Output column delimiters
     - Tab  ✔️ Comma  Space  Semicolon
     - Colon  Stop  Other character
   - Check the Input/Output column delimiters in the query file.

2. **Names**
   - Query file names
     - ✔️ File does not contain Example row names
     - File does not contain Input/Output column names
   - Check if Example row or Input/Output column name are not included in the query file.

3. **Extrapolate**
   - Extrapolate
     - ✔️ Extend range as needed
   - Extend the range as needed to allow extrapolation beyond the training range.

4. **OK**
   - OK
   - Click to accept all settings and close the dialog.

5. **Cancel**
   - Cancel
   - Click to reject all settings and close the dialog.
Trimming.

An **EasyNN-plus** network can be trimmed while it is learning.

Trimming weights, disconnecting input nodes and excluding example rows can be enabled while the network is learning.

**Trimming** has a low weight threshold. If the weight is below the threshold for the number of cycles it will be no longer be updated during learning.

**Input** nodes can be set to disconnect if they are the least important for the number of cycles.

**Example** rows that have the greatest error for the number of cycles can be excluded.

1. **Trimming**  
   - Check to enable connection Trimming when the weight is below the level set for the number of learning cycles.

2. **Disconnect**  
   - Disconnect the input node if it is the least important for the number of cycles.
3 Exclude

Exclude the example row if it has the greatest error for the number of cycles.

4 OK

Click to accept all settings and close the dialog.

5 Cancel

Click to reject all settings and close the dialog.
**Leave Some Out (LSO)** is used to validate a network. Example rows or subsets of examples are removed from the training set and used for validating. After the optional preparation that allows Controls to be set and some row types to be excluded all but the excluded example rows are changed to training types. One subset or a random number of example rows is then selected and changed to validating types. The training rows are then used to train the network without the currently selected subset of validating rows. Training continues until a stop is hit. It is important that the controls are set with a possible stop. The simplest stop to use with LSO is a fixed number of cycles. Validating stops are not needed as LSO runs its own validating cycle. When the training stops the selected subset of validating rows is used for validation. The process is repeated until all subsets have been used once. After the process is completed a report is produced showing the validating and training results. The report also contains the **Diagnostic Array** data if it available.

**Example Rows**

1. **Split the 20 example rows**
   - The total number of example rows in the Grid after the optional preparation.

**Sequential selection**

2. **Select subsets of example rows sequentially.**
   - Select subsets of example rows for validating starting at the first example row in each subset. Every subset will contain the same number of rows.
Random selection

- Select subsets of example rows at random.
  Select each subset of rows for validating starting at a randomly selected example row. Every subset will contain the same number of rows.

Random row selection

- Select up to 2 rows at random.
  Select rows at random for validating up to the number set.

Selections

- The number of times to select subsets or random rows for validating.

Maximum rows

- The maximum number of rows to select randomly for validating.

OK

- Click to accept the settings and close the dialog.

Cancel

- Click to close the dialog.
The connections between layers can be reduced. The minimum number of connections is one per node and the maximum is that produced by fully connecting the two adjacent layers of nodes. In this neural network, the input layer has six nodes and the first hidden layer has five nodes. The fully connected neural network would have thirty connections between the input and the first hidden layer. The connections could be reduced to any number between six and thirty.

<table>
<thead>
<tr>
<th>Connections between layers</th>
<th>Connections</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer to Hidden layer 1</td>
<td>16</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>Hidden layer 1 to Hidden layer 2</td>
<td>10</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>Hidden layer 2 to Hidden layer 3</td>
<td>7</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Hidden layer 3 to Output layer</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Enter the number of connections between each adjacent pair of layers.
2  OK  
Click to accept any changes and close the dialog.

3  Cancel  
Click to close the dialog.
**AutoSave.** Files can be saved automatically while the neural network is learning.

The Auto Save dialog is produced at start of Learning or by using Options > AutoSave... Complete the Auto Save dialog. The standard Save As dialog will open once for each type of file to save. The default Auto Save file names are filename.ed.tvq for decreases in average error, filename.sc.tvq for increases in score, filename.vi.tvq for increases in the number of validating examples, filename.bu.tvq for backup files, filename.de.txt for network details and filename.as.txt for associations. The filename is replaced by the name of the main file. Auto saved tvq files can be used in the same way as normal tvq files. They can even be set to Auto Save thus producing another level of file inheritance such as filename.ed.ed.tvq or filename.ed.vi.tvq etc. If Auto Save is restarted any previously produced .ed and .vi files are only overwritten when the results are better. The Auto Save reminders can be suppressed during learning and when restarting learning by checking the "do not ask..." box.

**Cycles**

Enter the minimum number of cycles to execute between every save.
Error decreased

- [ ] if the average training error has decreased
- [x] if the average validating error has decreased

Check to save if the average training error has decreased. If there are any validating examples the average validating error can be used. Both errors use the same 'ed' file. In most networks the training error will decrease faster than the validating error.

Score increased

- [ ] if the score has increased

Check to save if score increases.

Validating increased

- [ ] if the number of validating examples within 10.00% has increased

Check to save if the validating results improve.

Backup

- [ ] backup now and repeat every ten minutes

Check to produce a backup file every minute while the neural network is learning.

Details

- [ ] save network details to Diggers.de.txt on exit

Check to save all neural network details to a text file on exit.

Do not ask

- [x] do not ask for AutoSave settings when restarting learning

Check to stop this dialog being produced when learning is restarted.

OK

Click to accept all setting and close the dialog.

Cancel
Click to reject all setting and close the dialog.
Export special files

The Grid, the neural network and the graph data can be saved to text or comma separated files for use in other applications. The dialog is opened by selecting **File > Export...**

Select **txt** for tab separated files or **csv** for comma separated files.

Click on the components that are to be saved to files. Each component is saved to a different file. The names of the files are the base file name with the extension appended.

For example, if the Grid for the "Diggers.tvq" sample is saved as text for use in a text editor the file name generated will be "Diggers.gr.txt".

On the other hand, if the Grid is saved as a comma separated file for use in a spreadsheet the file name generated will be "Diggers.gr.csv".

Associations and Clusters are saved to text files. The name of the associations file is the base file name with the extension .as.txt and the clusters file is the base file name with the extension .cl.txt.
1. **What to save**

   - **Export file names** are produced with the base file name and the selected type extensions.
   - **Base file name**: Diggers

   - **Extension**
     - 
     - Grid names, types and values: *.gr.txt, *.gr.csv
     - Connection and weight values: *.we.txt, *.we.csv
     - Node bias and weight values: *.bi.txt, *.bi.csv
     - Graph cycle and error data: *.gp.txt, *.gp.csv
     - Example row names and errors: *.ex.txt, *.ex.csv
     - Column names and sensitivity: *.se.txt, *.se.csv
     - Column names and importance: *.im.txt, *.im.csv
     - Activation values and formulae: *.ac.txt, *.ac.csv

   Check all the parts of the grid and neural network that are to be saved. Select the file type extension.

2. **When to save**

   - **When to save**
     - Save special files with the main file on ext.
     - Save special files now.

   Select when the files are to be saved.

3. **Associations & Clusters**

   - **Associations & Clusters**
     - Save Input/Output Associations now.
     - Save Input/Output Clusters now.

   Check to save Input/Output associations and clusters.

4. **OK**

   Click to accept all the settings and close the dialog.
Click to reject all the settings and close the dialog.
Classify Column. Column value ranges can be used to produce secondary boolean classification columns.

1. Classify Text as Integers

   ![Classify Text as Integers](image)

   Check to classify text strings as integer values. This can also be used to convert text strings to numeric values.

2. Lower limit

   ![Lower limit](image)

   Enter the lower range limit.

3. Upper limit

   ![Upper limit](image)

   Enter the upper range limit.

4. OK

   ![OK](image)

   Click to accept all settings and close the dialog.
Click to reject all settings and close the dialog.
Extend Learning.

The Extend Learning facility is used to fill any gaps in the range of training examples or to extend the range. The Input/Output values of the copied examples can be modified by Jitter (a small random number). Three levels of Jitter are available. Contract within range: The Jitter moves the values towards the range average. Expand out of range: The Jitter moves the values away from the range average. The values are not limited by the range. Retain current range: Copied examples that would result in a change of range are deleted.

1. **Extra examples**
   - Enter the number of extra example rows to produce by copying and jittering.

2. **Jitter Inputs**
   - Jitter all Input values of copy by a random amount.

3. **Jitter Outputs**
   - Jitter all Output values of copy by a random amount.
4 Jitter level
- High level
- Medium level
- Low level
Select the jitter level.

5 Range
- Contract within range
- Expand out of range
- Retain current range
Set the range of copied values after jittering.

6 OK
Click to accept all settings and close the dialog.

7 Cancel
Click to reject all settings and close the dialog.
Set Noise Level. Noise can be added to the all the nodes activation while the neural network is learning.

1. **On/Off switch**
   - Switch the noise On or Off.

2. **Revert**
   - Revert if the average error increases after adding Noise.
   - If the average error increases the last noise added to each node activation is removed.

3. **OK**
   - Click to accept the setting and close the dialog.

4. **Cancel**
Click to reject the setting and close the dialog.
Jog Weights. Connection weights can be jogged while the neural network is learning. If the average error increases the jogged weights can revert to their old values.

1. Jog level

Set the amount to jog the weights.

2. Auto Jog Actions

Check to jog the weights by the set amount if the error is not reducing.
3 OK

OK

Click to accept all the settings and close the dialog.

4 Cancel button

Cancel

Click to reject all the settings and close the dialog.
Forget Learning: The neural network is reinitialized. All the weights and biases are set to random values within the initial weight range.

1. **ReSeed**
   - **ReSeed random number generator**
   - If checked, the random number generator is seeded starting at a value set by the current time.

2. **Weight Range**
   - Initial weight range
   - The range of values to set the weights and biases.

3. **OK**
Click to accept all settings and close the dialog.

**Cancel**

Click to reject all settings and close the dialog.
Network Defaults. The new network dialog can be preset to use defaults. The preset settings can be changed in the New Network dialog.

1. Growth rate
   - Growth rate
   - Change every 10 cycles or 5 seconds
   - A network is produced when the cycles or seconds elapses until the optimum network is found.

2. Hidden layers
   - Hidden layers
   - Grow layer number
   - from minimum nodes
   - to maximum nodes
   - The hidden layers are grown from the minimum to the maximum number of nodes.
3 Clear
   □ Clear all Network Defaults
      Check to clear all new network defaults.

4 OK
   □ OK
      Click to accept all settings and close the dialog.

5 Cancel
   □ Cancel
      Click to reject all settings and close the dialog.
Import Defaults. The import dialog can be preset to use defaults. The preset settings can be changed in the Import dialog.

1. Column delimiters

   ![Column delimiters](image)

   Check the characters to use as Input/Output column delimiters.

2. Row names

   ![Row names](image)
Set how to produce Example row names.

3 Row types

Row types:

- Training
- Validating
- Querying
- Exclude
- Ask while importing

Set the row type for all imported rows or ask for row types while importing.

4 Column names

Column names:

- Use # numbers for column names
- Set column names from text in first line

Set how to produce Input/Output column names.

5 Clear

Clear all Import Defaults

Check to clear all import defaults.

6 OK

Click to accept all settings and close the dialog.

7 Cancel button

Cancel

Click to reject all settings and close the dialog.
**Controls Defaults.** The learning controls can be preset to use defaults. The preset settings can be changed in the [Controls dialog](#).

1. **Learning**
   - **Learning rate**: Set the learning rate and momentum manually or by using optimize. The learning rate and momentum can be set to decay during learning if the descent to the target error is irregular.

2. **Clear**
   - **Clear all Default Controls**: Check to clear all controls defaults.

3. **OK**
   - **OK**: Click to accept all settings and close the dialog.

4. **Cancel**
Click to reject all settings and close the dialog.
File Open Defaults. The most recently used file can be set to open. Validating can be run while the file is opening. The default boolean name index can be used by all files. The name index will point to the same boolean names in all files unless extra names have been added. The initial, first mouse click, cursor position can be set to any corner of the grid. List can be tidied when the file is closed. Tracing can be enabled as the opens.
The most recently used file can be set to open. Validating can be run while the file is opening. The default boolean name index can be used by all files. When the file is open, F8 steps the index to the next boolean names.

2. **First Click**

Set the cursor position when the mouse is first clicked.

3. **Tidy Lists**

All lists are tidied before the file is saved. Unused items in the lists are deleted to reduce the file size.

4. **Tracing**

Set the global trace level.

5. **Clear**

Check to clear all open file defaults.

6. **OK**

Click to accept all settings and close the dialog.
7 Cancel button

Click to reject all settings and close the dialog.
**Edit Defaults.** The editing of grid values and the selection of the next grid cell. After editing, the default movement within the grid can be upwards, downwards, towards the left, towards the right or stay in the currently selected cell.

1. **Direction to move**
   - Set the direction to move after editing the cell value.

2. **Move if change**
   - Only move if value is changed
   Move in the set direction if the value is changed.
3. **Select**
   - Check box: Select value to edit
   - If checked, fully select the value to edit.

4. **Presets**
   - Check box: Ask which presets to use
   - Open presets dialog for new example rows.

5. **OK button**
   - Button: OK
   - Click to accept all settings and close the dialog.

6. **Cancel button**
   - Button: Cancel
   - Click to reject all settings and close the dialog.
**Insert Date & Time**. The date and time can be inserted in the grid or in the Example row names or Input/Output column names.

1. **What to Insert**
   - Check to insert the date and/or time and to use a sortable format if needed. A sortable format is YYYYMMDD HHMMSS. If the sortable format is not used,
   - **Date**, **Time**, **Use sortable date and time format**

2. **Where to Insert**
   - **Grid Cell**, **Grid Row**, **Grid Column**, **Row Names**, **Column Names**

3. **Start Date and Time**
   - **Year**, **Month**, **Day**, **Hour**, **Minute**, **Second**

4. **Increment by**
   - **Days**, **Minutes**, **Hours**, **Seconds**

5. **OK**

6. **Cancel**
the current system format will be used.

2 Where to Insert

Set where to insert the date and/or time. The start position will be the current grid selection.

3 Start Date and Time

Enter the first date and/or time to be inserted.

4 Increment by

Enter the amount to increment the date and/or time.

5 OK

Click to accept all settings and close the dialog.

6 Cancel

Click to reject all settings and close the dialog.
Add to Subset. Subsets of the grid can be produced. The Input/Output names are prefixed by the subset letter.

The **Subset > Select Subset** menu command opens the **Select Subset dialog**. The subsets of columns can then be changed to inputs, outputs or excluded.

The **Subset > Add Columns to Subset** menu command opens the Add to Subset dialog. The currently selected column is the first of the group of columns to be added to a subset. Columns with names containing specified characters can be added to the other subset at the same time. The subset can be any single letter or number. Columns can be assigned to any number of subsets. The subset names and the column name are separated by \ characters.

The **Subset > Remove Column** from Subsets menu command removes the currently selected column from all subsets.
2 Containing

Add the columns with names containing [ ] to subset [ ]

Enter the characters in the Input/Output column names and the letter to use for the subset.

3 OK

Click to accept the settings and close the dialog.

4 Cancel

Click to reject the settings and close the dialog.
Select Subset. Subsets of the Input/Output columns can be set to be used as inputs, outputs or excluded.

The **Subset > Select Subset** menu command opens the Select Subset dialog. The subsets of columns can then be changed to inputs, outputs or excluded.

The **Subset > Add Columns** to Subset menu command opens the **Add to Subset dialog**. The currently selected column is the first of the group of columns to be added to a subset. Columns with names containing specified characters can be added to the other subset at the same time. The subset can be any single letter or number. Columns can be assigned to any number of subsets. The subset names and the column name are separated by \ characters.

The **Subset > Remove Column from Subsets** menu command removes the currently selected column from all subsets.
Enter the subset prefix letters.

Set Input/Output columns in subset [a] to inputs
Set Input/Output columns in subset [b] to outputs
Set Input/Output columns in subset [ ] to exclude

The other columns will be left unchanged.

2 OK
   OK
   Click to accept the settings and close the dialog.

3 Cancel
   Cancel
   Click to reject the settings and close the dialog.
**Forecast.** New values can be forecasted using the existing values in Serial columns.

Any columns can be set to Serial type by moving to the column and pressing enter to open the Grid edit dialog. The Serial type is also available on the right click menu. This type of column is the same as an Input column but is also used as an Output column during forecasting. If a Serial column is available the Forecast Action is enabled. This produces the Forecast dialog. When the dialog is completed the forecasting process begins. Forecasting can take a long time if many Serial columns are used or if the Grid has many rows. The process first creates a number of extra Engaged rows. These working locations are used while forecasting runs. They cannot be edited. Forecasting creates neural networks for testing as it runs. When forecasting has finished the extra rows will be changed to Forecast or Risk and will contain the forecasted values. The estimated risk of the forecast is the number of times the forecast values had to be adjusted before the neural network gave a similar error to that produced prior to the forecast. Forecast rows can be used in subsequent forecasts.
1. Enter the number of values to forecast.

**Example row types**

- Training
- Validating
- Forecasted
- Excluded
- Duplicated
- Querying

Check the example row types to be included in the forecast.

**Range limits**

- Allow forecasts above the current range.
- Allow forecasts below the current range.

Check to allow forecasts to go above and/or below the range of the serial Input/Output columns.

**OK**

Click to accept all settings and close the dialog.

**Cancel**

Click to reject all settings and close the dialog.
Input Validating Rules. Validating rules can be applied to input columns to modify the values used during a validating cycle. The values can be set to lowest, highest, median or average and applied to all rows. The actual values can also be multiplied by a constant.

The validating rules edit dialog is produced when the toolbar button is pressed or by using Edit > Rules menu command. The dialog is used to step through the columns and set or reset the rules. Any rules that are changed are set by moving to a different column. Rules can be reset and deleted by pressing the Reset Rules button.

1. First
   - Click to move to the first column.

2. Previous
   - Click to move to the previous column.
Next

Click to move to the next column.

Last

Click to move to the last column.

What to use

Select the value to use in the input column.

Adjuster

Enter the multiplier to adjust the value.

Reset Rules

Click to reset the rules for this input column.

Close

Click to close the dialog.
Output Validating Rules. Output column validating rules can be used to override the validating controls.

The validating stops in the Controls dialog apply to every Output Column in the same way. If more than one output is used it is possible to apply different validating rules to each column. A score can be used with each rule so that training can be stopped as soon as a certain score is reached. Rules can be defined to score points if the output value is within a range or if the value is correct after rounding it to a multiple. Both methods can be used together.

Consider predicting horse race results with a network trained from historical performance of named horses. The output will most probably be the finishing place of each horse. This may be an integer that is the actual place but it is more likely to be 3, 2, 1 for the first, second, and third. A second output could be a boolean to indicate that the horse won the race. With rules the first column could be set to score 1 if the place was correct. The second column rule could be set to score 10 if the Win was correctly predicted.
Within
Enter the % within range.

First
Click to move to the first column.

Previous
Click to move to the previous column.

Next
Click to move to the next column.

Last
Click to move to the last column.

Within range score
Enter the score if validating is within the % range.

Rounding
Enter the rounding multiple.

Correct score
Enter the score if correct after rounding to multiple.
Reset Rules

Click to reset the rules for this output column.

Close

Click to close the dialog.
Outside Limits. Columns that have values outside the limits set are excluded.

1. **Lower limit**
   - Enter the lower limit.

2. **Upper limit**
   - Enter the upper limit.

3. **Types to exclude**
   - Training
   - Validating
   - Querying
Check the example row types to exclude if outside the limits.

4 OK

Click to accept all settings and close the dialog.

5 Cancel

Click to reject all settings and close the dialog.
**Edit grid range.** The type and mode of blocks of rows and columns can be changed.

1. **First row**
   - Set type from row
   - Input/Output columns
   - Enter the number of the first row to change.

2. **Number of rows**
Enter the number of rows after the first row to change.

3 Row Type

Example rows

Rows are numbered from 0 to 18

Set type from row 6 for 0 rows to Training Validating Querying Exclude

Set the type to change the example rows to.

4 Column types, modes and results

Input/Output columns

Columns are numbered from 0 to 6

Set type from col 2 for 0 cols to Input Output Exclude

Set mode from col 2 for 0 cols to Real Integer Boolean Test Image

Set results from col 2 for 0 cols to Include Exclude

Set the type and mode to change the columns to. Set which columns are to be included or excluded from the query results file.

5 OK

Click to accept all changes and close the dialog.

6 Cancel

Click to reject all changes and close the dialog.
Change Input/Output Types. Input/Output column types can be changed from one type to another. Any columns of the type found can be changed one at a time or all the columns found can be changed at the same time.

1. **Find**
   - **Find**
   - **Find Next**
   - Set the type of Input/Output columns to find.

2. **Change to**
   - **Change to**
   - Set the type of Input/Output column to change to.

3. **Find Next**
   - **Find Next**
   - Click to find the next Input/Output column of the type set.
Change

Change
Click to change the type of the currently found Input/Output column.

Change All

Change All
Click to change the type of all the found Input/Output columns.

Close

Close
Click to close the dialog.
Change Example Types. Example row types can be changed from one type to another. Any rows of the type found can be changed one at a time or all the rows found can be changed at the same time.

1. Find
   - Check the type of example rows to find.

2. Change to
   - Training
   - Validating
   - Querying
   - Exclude
   - Duplicate
   - Forecast

3. Find Next
4. Change
5. Change All
6. Close
2. **Change to**
   - Change to
     - Training
     - Validating
     - Queuing
     - Exclude
   
   Set the type to change the example rows to.

3. **Find Next**
   - Find Next
   
   Click to find the next example row of the type set.

4. **Change**
   - Change
   
   Click to change the type of the currently found example row.

5. **Change All**
   - Change All
   
   Click to change the type of all the found example rows.

6. **Close**
   - Close
   
   Click to close the dialog.
Copy Input/Outputs. Any number of contiguous Input/Output columns can be copied. The copies can be offset vertically upwards or downwards by a number of rows.

1. **First column to copy**
   - Enter the number of the first column to copy.

2. **Columns to copy**
   - Enter the number of columns to copy.

3. **Vertical offset**
   - Enter the number of rows to offset the copied column. Positive to offset downwards, negative to offset upwards.

4. **Output types**
   - Check to set all copied columns to output types. Uncheck to keep the original types.

5. **OK**
Click to accept all changes and close the dialog.

6

Cancel

Click to reject all changes and close the dialog.
Copy Examples. Example rows can be copied and set to different types.

1. **First row to copy**
   - Enter the first row to copy.
   - [Image 13]

2. **Rows to copy**
   - Enter the number of rows to copy.
   - [Image 100]

3. **Set row types**
   - Set the copied rows type. All copies will be set to this type.
   - [Image Training, Validating, Querying, Excluded]

4. **OK**
   - Click to accept all changes and close the dialog.
Cancel

Click to reject all changes and close the dialog.
Missing data. Values based on the Input/Output column range can be used to substitute any missing data in the Grid.

1. **Missing action**
   - Set the action to use when data is missing.

2. **OK**
   - Click to accept the setting and close the dialog.

3. **Cancel**
   - Click to reject the setting and close the dialog.
Any bitmap image can be used in **EasyNN-plus**. Bitmaps of different sizes and types can be mixed. Other image formats (jpeg, gif etc) must be first converted to bitmap format. Images can be entered or imported into the **Grid**. Images can be mixed with text and numeric data in Example rows but not in Input/Output columns. Columns can be set to image mode using the **Edit dialog** or while importing.

Each image in the Grid will have a name, a bitmap file and three associated numeric values. These numeric values are the Pixel Pair Code (PC), the Edge Code (EC) and the Block Code (BC). PC is derived by comparing a thousand pairs of pixels. The positions of the top, bottom, left and right outer edges are used to produce EC. The position of the largest block of pixels that do not vary in color is used to produce BC. **EasyNN-plus** must have access to the bitmap file to produce PC, EC and BC. Large bitmaps will take a long time to process. If **EasyNN-plus** cannot find the bitmap file a selection dialog will be opened so that the file can be located. This selection dialog is also used while a file is being imported whenever a bitmap file is needed. Once the images are in the Grid they can be used to produce networks in the same way as any other type of data. Every image Input and Output column in the Grid will map onto three nodes, a PC node, a EC node and a BC node.

Images can be fragmented into component parts using **Edit > Fragment Image**. Images and fragments can be associated with the values in other Input/Output columns using **Edit > Associate Image**.
Details

The name of the image, the encoded image values and the image file name.

Association

The column associated with the image and the value in the column.

Resized image

All images are resized to the same dimensions.

Show images

Images will show in the grid cells that contain them. This is normally disabled to avoid slow grid refreshes.

Image File

Click to access or edit the image file name.
6 Grid Entry
Click to access or edit the grid entry for this image.

7 Fragment
Click to fragment the image into parts.

8 OK
Click to accept any changes and close the dialog.
Images can be fragmented into up to 10 X 10 component parts allowing a maximum of 100 fragments which can be placed in example rows. This image of the numbers 0 to 9 needs to be fragmented into the individual digits.
2 Original size

Original size (pixels)

<table>
<thead>
<tr>
<th>Width</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>28</td>
</tr>
</tbody>
</table>

The original size of the image.

3 Number of fragments

Number of fragments

| Horizontal | 11 |
| Vertical   | 1  |

The default number of fragments is 16 arrange as a 4 by 4 square. This image needs 6 horizontal and 7 vertical to fragment into 42 individual signatures.

4 Size of fragments

Size of fragments (pixels)

- Adjust to fit
- Width: 0
- Height: 0

Enter the size of each fragment or check "Adjust to fit" for an exact division of the image.

5 Fragment distribution
Enter the row number to place the first fragment.

Enter the prefix for the fragment number.

Set the example type of the fragment.

Click to accept all changes, start the fragmentation and close the dialog.

Click to close the dialog.
An individual part of an image after it has been fragmentated using **Fragment image**.

1. **Fragment**

   - **Fragment name**: fg000025
   - **Pair Code**: 325  **Edge Code**: 373  **Block Code**: 414
   - **Fragment width**: 9  **Fragment height**: 28
   - **Horizontal position**: 5  **Vertical position**: 0
   - **Image file**: C:\EasyNN\Work\bitmaps\numbers.bmp

   The fragment name, its encoded details, size, position and the original image file name.
2 Association

Fragments can be associated with another column.

3 Resized fragment

All fragments are resized to the same dimensions.

4 Image File

Click to access or change the original image file name.

5 Grid Entry

Click to access or edit the fragment grid entry.

6 OK

Click to close the dialog.
Set Function

Right click a Grid cell or a Network node to open the function menu and then click **Edit Function** to open the edit dialog. The dialog can be used to enter, select, add and delete functions.

Any function can be set up and then executed on example rows, input columns, output columns, the selected cell or a range of cells.

Functions can also be added to Nodes and Layers of nodes. They will be executed when the nodes are processed while the network is learning the training information in the grid. Some functions can decrease the number of cycles needed for the network to converge. Functions can take time to execute so the changes in learning speed are not easy to predict.

When a function is available it is displayed on the top, left corner of the grid view and on the network view. Functions that have been added to nodes are displayed above the node on the network view.

**Execute Function on the Grid**

1. Select the Grid view.

2. Right click the target grid cell to open the grid context menu. If a function is available the menu will include the function execution commands.

3. Click the execution command required.
4. The function will be tested.

5. If there are no errors, the function will be executed on the row, column, cell or range of cells.

**Add Function to a Node or Layer**

1. Select the Network view.

2. Right click a hidden node to open the network context menu. If a function is available the menu will include the function add and remove commands.

3. Click the add or remove command required.

4. The function will be added to or removed from the node or layer.

5. Functions added to nodes and layers will be executed as the network is learning.

**Function**

Functions are plain text strings and can be entered here.

**Stop on fail**

Check to stop function execution if any failure occurs.

**Add**

Press to add the function to the library.

**Next**

Press to select the next function from the library.

**Delete**

Press to delete the current function from the library.

**Close**

Press to close the dialog and make the current function available to the grid and
network.
The **Diagnostic Array** displays the validating results of the current boolean mode output column. This type of analysis is used extensively in medicine to describe the clinical performance of tests. The array data is also displayed on the **Learning Progress** graph while the network is learning or when **Leave Some Out** is running.
Press to close the Diagnostic Array
Shuffle example rows while the network is learning.

1. **Allow Shuffle check**
   - **Allow Shuffle**
   - Check to allow auto shuffle while the network is learning.

2. **Cycles**
   - Enter the number of learning cycles per shuffle.
   - 100

3. **Shuffles**
   - Enter the maximum number of shuffles while the network is learning.
   - 1000

4. **OK button**
   - Click to confirm the settings.
Click OK to accept the settings and close the dialog.

**Cancel button**

Click Cancel to reject the settings and close the dialog.
The **Freezes** dialog is used to set the node freeze levels and the way they are decremented while the network is learning.

After the initial number of training cycles are run the freeze levels are applied in the reverse order of the node importance. All nodes are considered, not just the inputs. The least important node has the lowest freeze level. The most important node has the highest freeze levels. Any nodes with their freeze level above zero is excluded from the learning process and their activation does not change. Frozen input nodes have their activation set to the median value of the input range. The freeze levels are reduced as learning progresses so that the nodes are progressively added to the learning process. When learning has finished the input importance and sensitivity measures are more accurate.

**Remove**

Check to remove the freezes that are still set from previous runs.

**Types to freeze**
Check the node types to freeze.

3. **Set freezes**
   - **Set freezes**
   - Check to allow freezes to be applied.

4. **Initial cycles**
   - **Set freezes on training cycle**
   - Enter the number of training cycles before freezes are applied.

5. **First node level**
   - **Start at freeze level**
   - Enter the freeze level to apply to the first node.

6. **Increase**
   - Then step through selected node types increasing level for each node by
   - Enter the freeze level to increase for each node.

7. **Reduce**
   - **Reduce**
   - Check **Reduce** to allow the freezes to be decremented while the network is learning.

8. **Reduce levels**
   - **Reduce all node freeze levels by**
   - Enter the freeze level reduction.

9. **Reduce cycles**
   - **every**
   - Enter the training cycles to run between every freeze level decrement.
Press OK to accept all the settings and close the dialog.

Press Cancel to reject all the settings and close the dialog.