User's Manual for the Handwritten Recognizer

By

Maeda Hanafi

CSC400 01

Prof. Daponte

November 30, 2011

Table of Contents

1. Introduction

    This program attempts to recognize user's handwritten character. Before the network can be used to recognize characters, it has to be trained. The user presents a list of handwritten characters as training data for the feed forward neural network, along with the English capital letter or number to recognize. The network learns using back propagation training. The program train using either all of the given training data (unfiltered list) or a shorter list of the training data (filtered list). Note that only the filtered process requires the user to present the letter to recognize before the network is trained.

    This user manual will serve as guidance regarding how to install the software, how to use it, and troubleshooting. The appendix contains a brief introduction to handwritten recognition using feed forward neural network with back propagation learning.

2. Installing the software

    The software works on all platforms as long as Java Runtime Environment (JRE) is installed. The JRE allows java programs to run. It must at least be version 6 or above. In order to install the software, the disk must be inserted, and the zip file inside must be unzipped to local disk. After unzipping the file, the program can be run by clicking on the jar file under the /dist folder.

3. Using the software

    -What it does

    The purpose of this software is to allow the user to present training data to the system, train the neural network, and finally recognize a letter. The user can present the data by loading data from the sample.dat file (and save training data to it). Note that the file only stores 7 by 6 down sampled training data. If the user wishes to create his or her own training data, one must draw a character on the canvas and inform the software to associate that particular character with the letter or number it is. The training data must be a set of English alphabet or single digit number (0 to 9). The network is feed forward and trains using back propagation. The network also deals with only down sampled data. The user has the choice of whether to train the network using all of the training data presented (unfiltered) or a portion of the training data that is similar to the character the user wants to recognize (filtered).

- Interface Elements

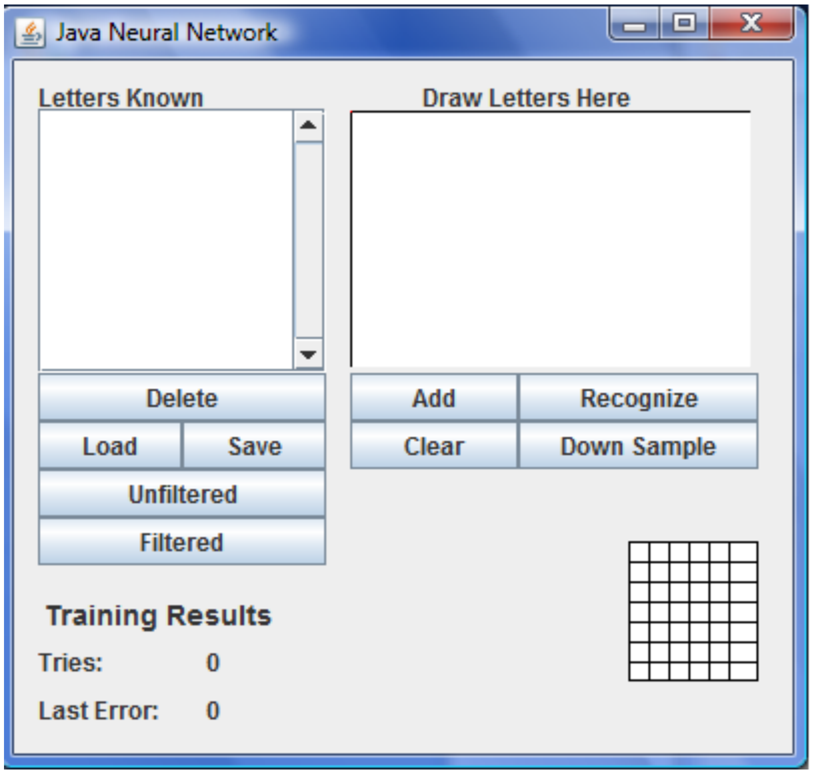


Figure 3.1 A screenshot of the software when it first starts

Shown in Figure 3.1 is a screenshot of the software when it is run. The box under the “Letters Known” text lists the characters the system can recognize. When the user is training using the filter, only the characters used to train the network will be listed. Otherwise, if the user is training using all of the training data (unfiltered), the whole list of training data shown in the box.

Beside the box is the drawing canvas that allows the user to draw a character. The canvas is used to get training data input, it is also used to get the character that the user wishes to recognize.

The 7 by 6 grid below it displays the down sampled image of the letter drawn in the canvas. The "Down sample" button allows the user to down sample the drawing from the canvas, which would then be displayed in the 7 by 6 grid.

The "Load" and "Save" buttons are used to load and save the training input data and corresponding ideal outputs to a text file named sample.dat. Alternatively, if the user wishes to create his or her own training data, the "Add" and "Delete" button are used. The "Add" button will prompt the user what the corresponding ideal output is. (The data input would come from the drawing canvas.)

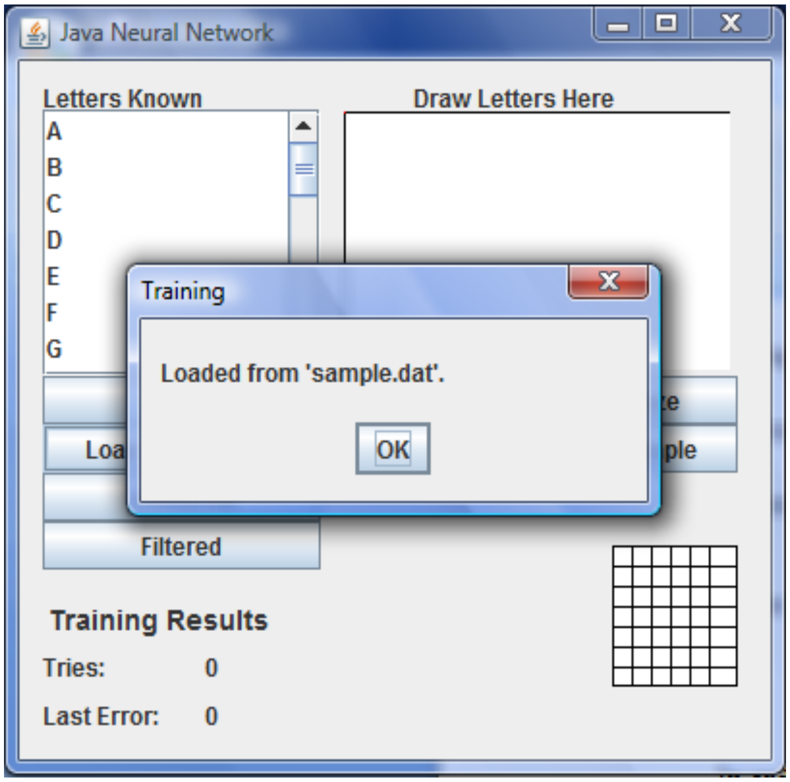
The "Filtered" and "Unfiltered" buttons are used to train the network. The "Filtered" button filters the list of known letters based on the similarity of the known letter with the letter to recognize. Note that the letter to recognize must be drawn on the canvas before clicking the "Filtered" button. The training process starts right after the system finishing displaying the filtered list in the "Letter Known" box. The filtered process and the training take about 30 seconds, and the system recognizes a letter, assuming that it is listed in the filtered list, with an accuracy of 80%.

On the other hand, the "Unfiltered" button trains the network using all of the training data. The training process takes about 45 min and the system recognizes with a 60% accuracy.

On the bottom corner of the program, information regarding training is displayed. The "Tries" text refers to the number of training epochs (iterations) the network has gone through. The "Last Error" refers to the error of the network at that particular epoch. As the training process is running, these two values are constantly updated. The training process is done when the "Last Error" is 0.99999.

- How to use it

There are two phases when using this software: training and testing (recognizing). In order to train, the user can load the file containing input data and ideal outputs (sample.dat). If the user wants to import a file containing training data, then the load button would load the file.



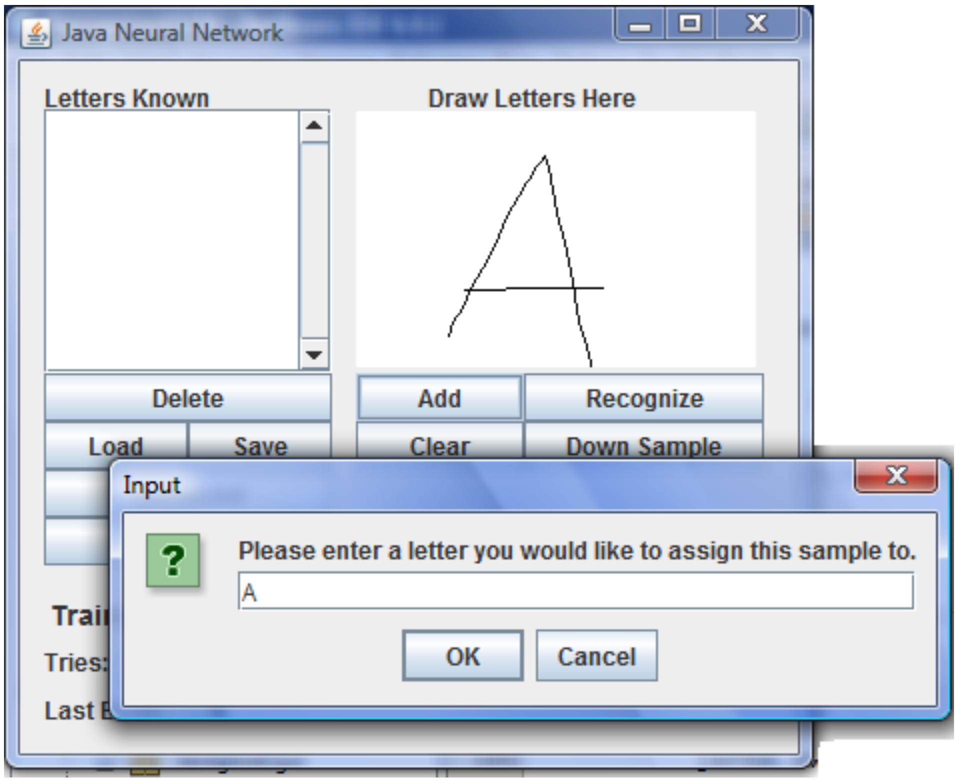
Training

Loaded from 'sample.dat'.

OK

Figure 3.2 Using the “Load” button to retrieve training data

If the user wants to create a new training data, then the user would have to draw a character in the canvas that is under the “Draw Here” text. Afterwards, the “Add” button is used to add training data. Consequently, the program will prompt the user what the letter should be, as shown in the figure below. To delete a known letter, the user must first select a character from the “Known Letters” list and click on the “Delete” button. The save button would save the training data, and the delete button would delete the selected known letter. The user can also save the down sampled input data and ideal outputs to sample.dat using the “Save” button.



Please enter a letter you would like to assign this sample to.

A

Figure 3.3 Using the "Add" button to get training data

Once the user is done creating training data or importing it, the user can click on the "Begin Training" button to start training the network. The training results will appear in the bottom left corner, which displays the number of tries and error information. Afterwards, the user can draw a letter to recognize and click the "Down Sample" button to down sample the letter. The down sampled image of the letter is shown in the grids below the drawing canvas. If the user wants to recognize it, he or she must click the "Recognize" button. A flow chart is shown in Figure 3.4.

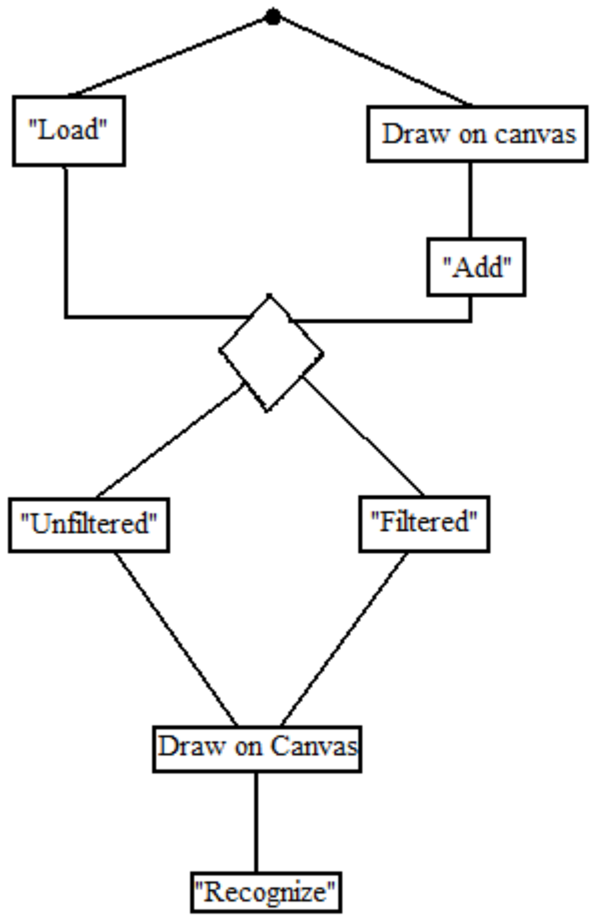
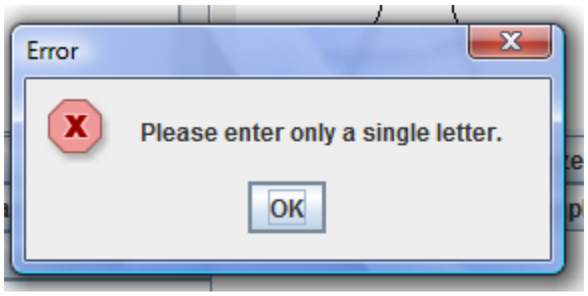


Figure 3.4 A Flowchart of how to use the program

4. Troubleshooting

   - Adding training data

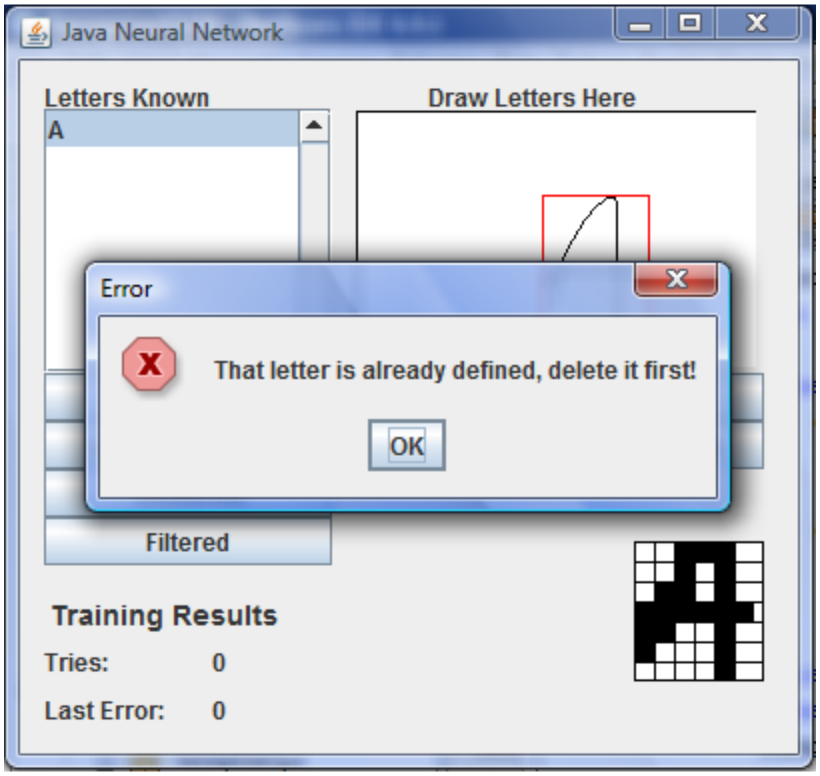
When attempting to add a character to the training data set, be sure that the character one enters is a single character. Otherwise, the program will not associate the character to the drawing and add it to the list of known letters, as shown below.



Please enter only a single letter.

Figure 4.1 Entering in non-single characters

Another thing to be caution about is to be sure the character user enters is not on the list of known letters. Otherwise, the system will ask user to delete that particular letter, as shown in Figure 4.2.

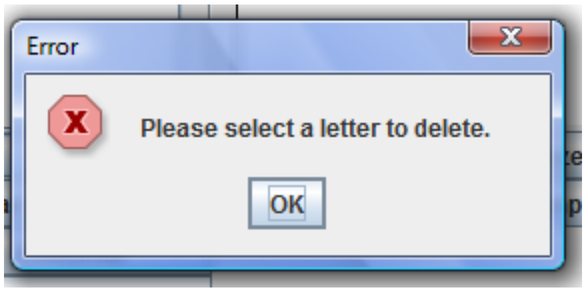


That letter is already defined, delete it first!

Figure 4.2 Adding in a letter that is already on the list

- Deleting a letter

If the user can't delete a letter properly, as shown in Figure 4.3, it may be because the user hasn't selected a letter to delete.

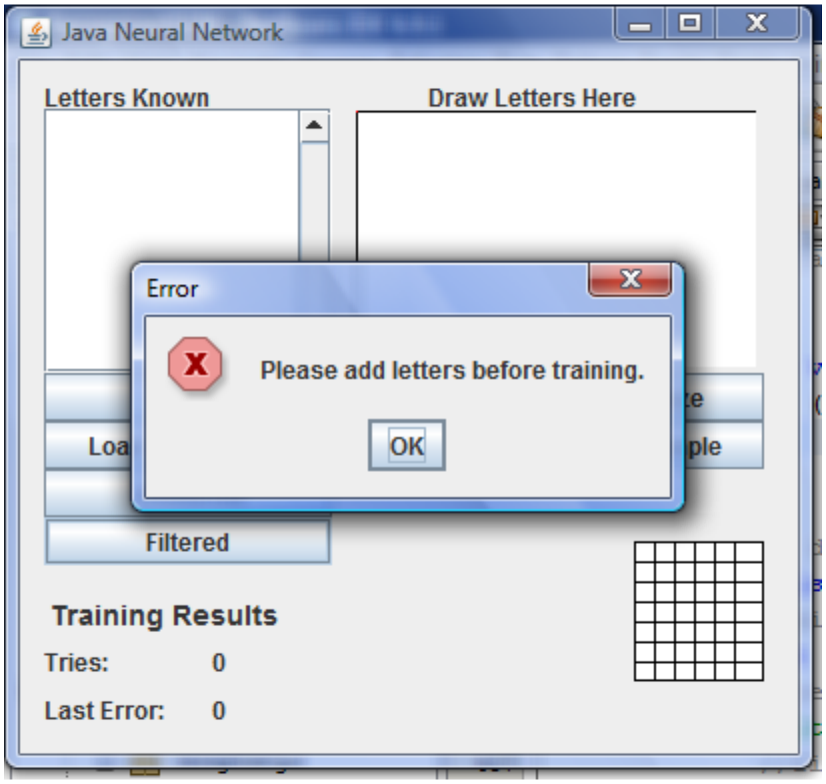


Please select a letter to delete.

Figure 4.3 Attempting to delete a letter that isn't selected

- Training

Since back propagation is a supervised kind of training, data must be provided before trying to train. When attempting to train the network, be sure that there are some letters in the list. Otherwise, the system will not begin training and ask user to add letters.



Please add letters before training.

Training Results
Tries: 0
Last Error: 0

Figure 4.4 Training with the training data

If the user did not successfully filter (the list doesn't become shorter), then the letter to recognize must be drawn before training. In other words, before clicking on "Filtered", draw the handwritten letter onto the canvas.

- Loading training data

If the program doesn't work after attempting to load from sample.dat, restart the program, and be sure that sample.dat contains information with the format in the following:

[single letter]:[down sampled image],

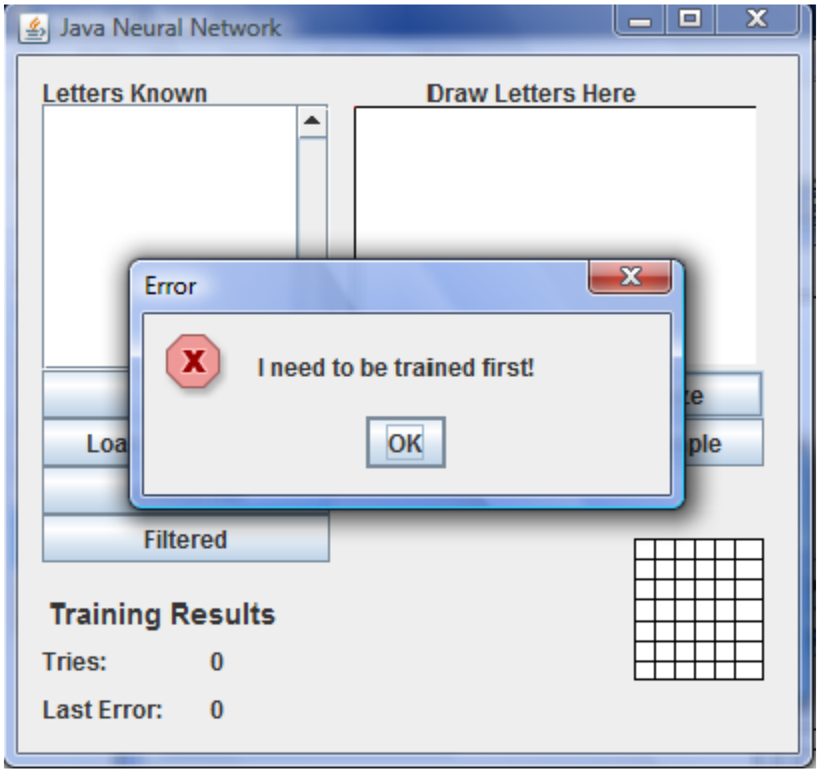
where single letter is the letter that is associated with the following down sampled image and down sampled image is a string of 0's and 1's representing a 6 by 7 grid. An example of the letter A is shown in the following:

A:001100011100010100010100111111100010100010

Note that there are no spaces.

- Recognizing a letter

If the following message is shown, as illustrated in Figure 4.5, then it means the network isn't trained yet. The "Recognize" button can only be used after the network is trained.



I need to be trained first!

Figure 4.5 Clicking on "Recognize" without training the network

5. Appendix

- Neural Networks and Handwritten recognition

The basic building block of a neural network is the neuron. A neuron contains a threshold and a weight. When an input goes to a neuron, the neuron's weight is amplified. If the amplified weight is greater than the threshold, then the neuron fires a one, else a zero. A neural network makes decisions based on groups of neurons. In a feed forward network, there is an input layer of neurons, hidden layers, and an output layer. The firing of the neuron is one way. In other words, neurons aren't fully connected. Also, in order for a network to learn, the weights must be recalculated, and one way to recalculate them is to use back propagation rules. In back propagation learning, an error between the anticipated and actual outputs is calculated using back propagation formulas. This error is used to change the network's weights, from the output layer to the input layer.

Because of the ability of a neural network to train and make decisions, we can supply training data and train the network. The network will be supplied with a set of inputs and a set of ideal outputs. . In other words, the user provides a set of image inputs (containing a letter) and ideal outputs image that are used to train the network. The program trains the network using the set of inputs so that its outputs are the same as the anticipated outputs. In the process, the network's weights will be recalculated using back propagation learning formulas, until the network's outputs are correct. Once they are correct, we may test it by presenting an image containing a letter.