Handwriting Recognition

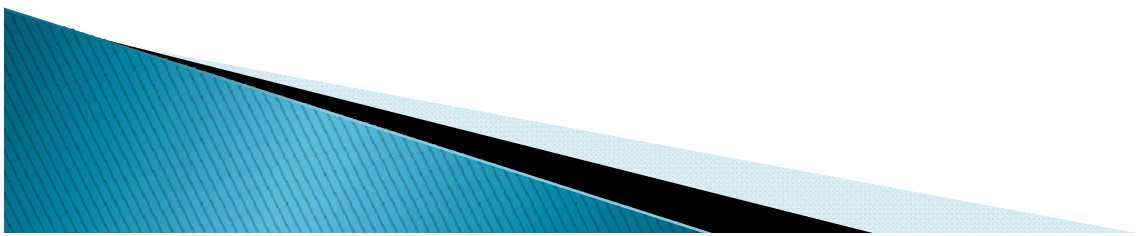
Using Neural Networks

Matt Nitzken

ECE614 – Neural Networks

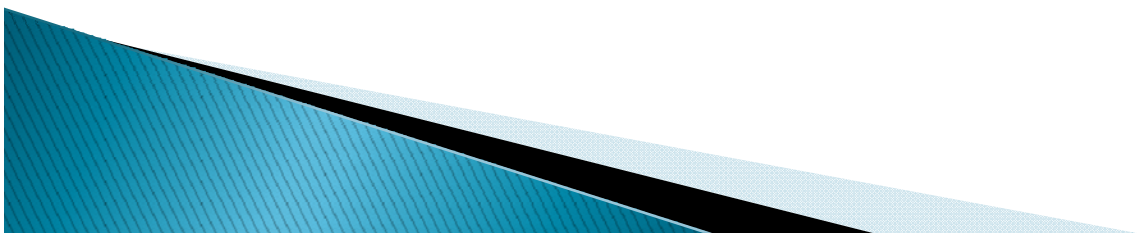
Definition

- Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices.
- The image of the written text may be sensed "off line" from a piece of paper by optical scanning (optical character recognition) or intelligent word recognition.
- By extracting features from written characters a neural network can be designed to parse these features and distinguish numbers and letters.



System Layout

- Template Documents – used for easily accepting training/testing inputs
- Image and Character segmentation algorithms
- Conversion of segmented images into neural network inputs
- Training and Testing of the network



Document Templates

- The training documents were divided into grids.
- This allowed the system to more easily located letters for training and testing. Reduced the complexity of trying to discern gaps between strings of characters.
- Numbers 0-9 were used for testing.

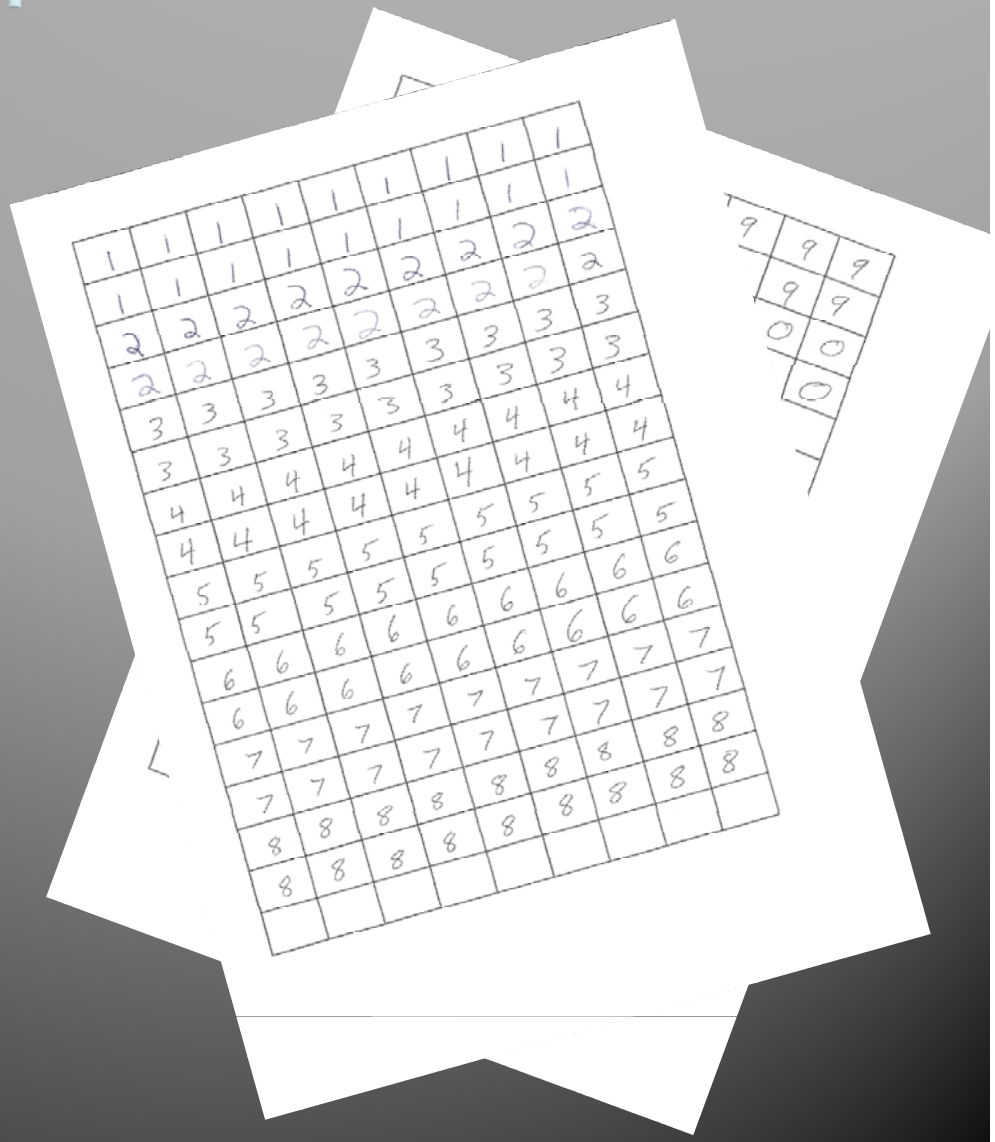


Image Segmentation

- Due to the usage of the grid segmentation was easier than reading “off-line” characters.
- Initially the images were globally thresholded to convert them to a binary black and white format that was easier for the computer to read.
- Begin by roughly finding the locations of grid lines.

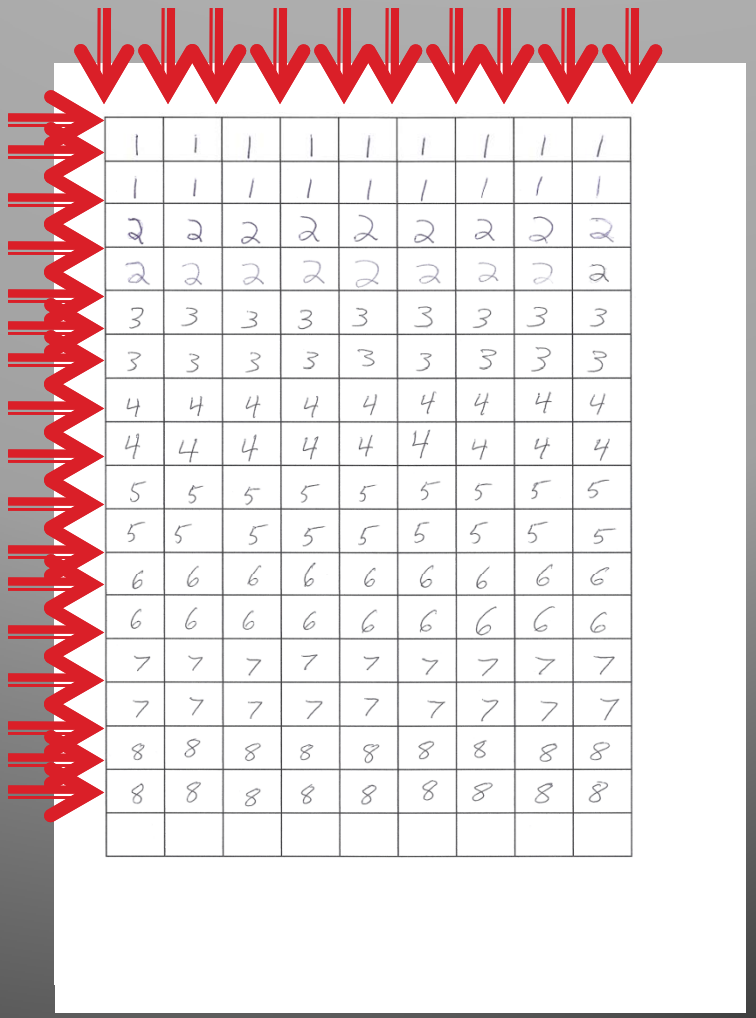


Image Segmentation

- Use these grids to begin moving around in the image.
- A search starts near a grid intersection. The algorithm then begins looking for pixels.
- Once it finds the first pixel it expands out from this pixel to encompass a "letter".
- If the object is too small (i.e. not enough pixels near one another) it ignores it.

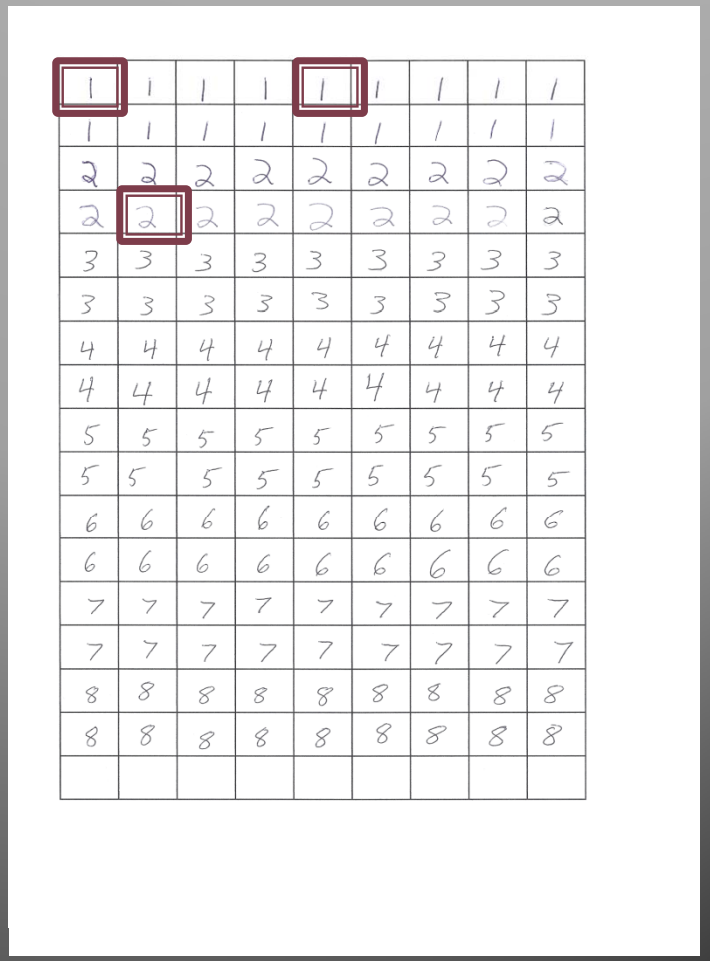


Image Segmentation

- Use these grids to begin moving around in the image.
- A search starts near a grid intersection. The algorithm then begins looking for pixels.
- Once it finds the first pixel it expands out from this pixel to encompass a “letter”.
- If the object is too small (i.e. not enough pixels near one another) it ignores it.

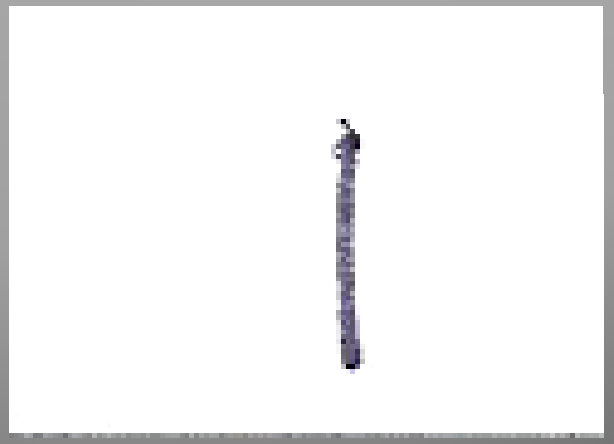
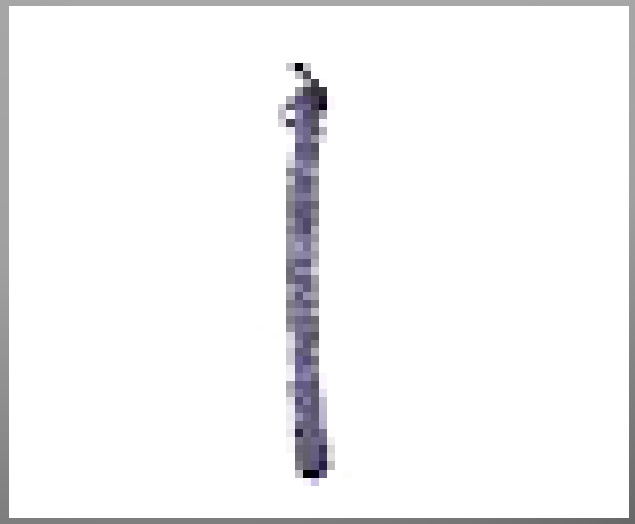


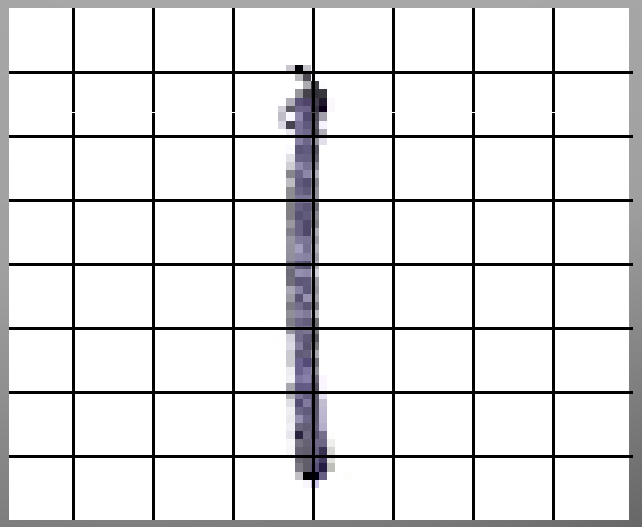
Image Segmentation

- To create a uniform network input the detected object is then centered inside a window.
- To accomplish this the centroid of the letter is calculated an a window is calculated surrounding the centroid.
- Any residue around the letter is then clipped away if it is part of the boundary grids.



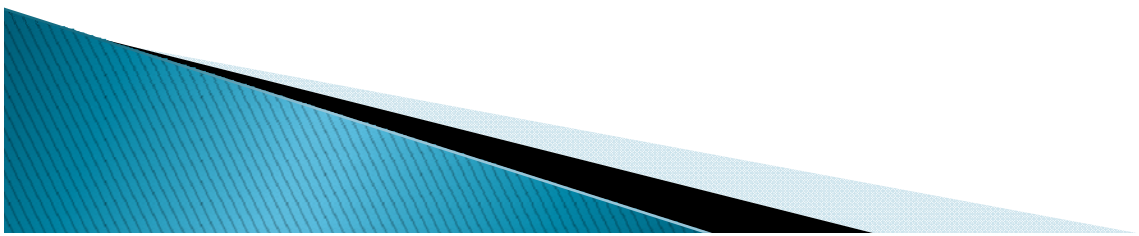
Creating a Neural Network Input

- The resulting letter is scaled into an 88x88 pixel box using an Affine Transform.
- The letter is then broken into 121 smaller boxes each consisting of 8x8 pixels.
- The distribution of pixels in each box is then calculated into a single value creating an 11x11 array of values where each cell is representative of the surrounding pixels.
- This array is then reshaped into a 1x121 array.



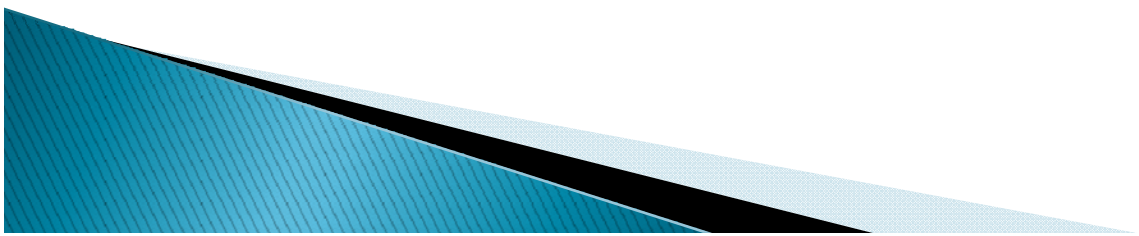
Creating a Neural Network Input

- For each letter detected a 1x121 array is calculated. This constitutes the neural network input for a specific object.
- These are then all placed into one larger training array.
- For a training array consisting of 200 objects (numbers in this case) the training array would be of size 200x121.



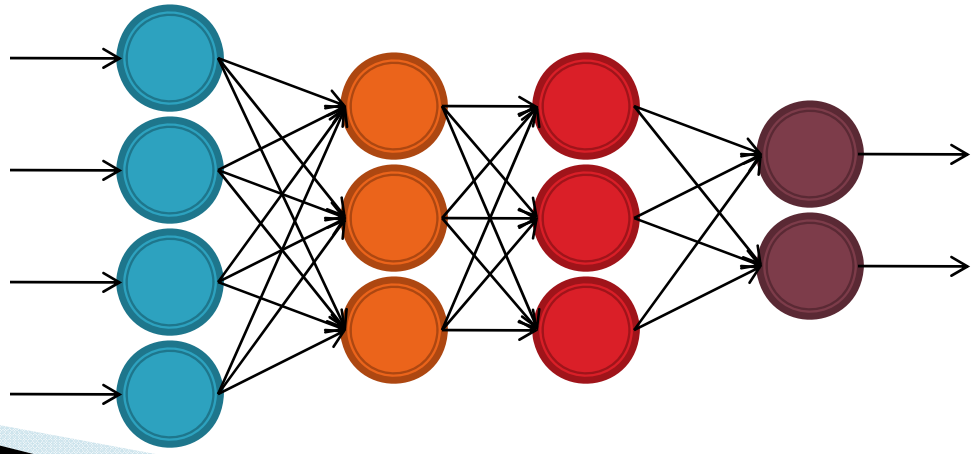
Creating a Neural Network Input

- I wrote the network to utilize a binary output.
- The network would output a 6-bit response (e.g. for the letter 1 the output is 000001).
- Each letter was given a definitive value for training purposes.
- The separate set of test data was also given the correct value for validation purposes.



The Neural Network Classifier

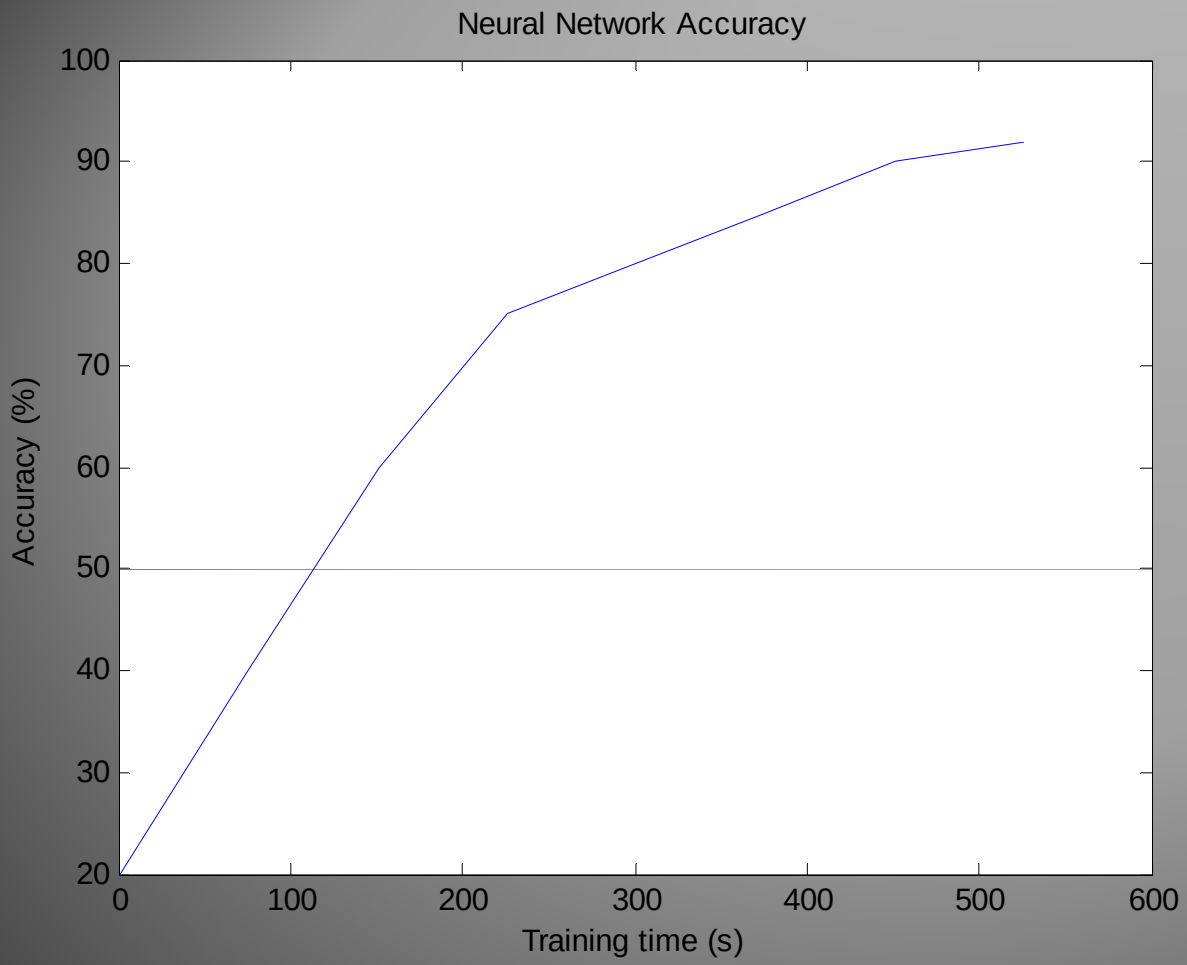
- I constructed a neural network with the following specifications:
  - Input Layer – 121 Input Nodes
  - First Hidden Layer – 28 Hidden Nodes
  - Second Hidden Layer – 28 Hidden Nodes
  - Ouput Layer – 6 Input Nodes
- The network had 3 weight matrices and a combined total of 569,184 different weights.
- The initial weights were seeded from a random distribution of data as I found this gave the best result.
- I selected to use continuous uni-polar neurons throughout the entire network.
- For the final solution the ouputs were rounded to the closest value of either 0 or 1 giving the final binary output of the network.
- A uniform bias node of -1 was used.



Testing the Network

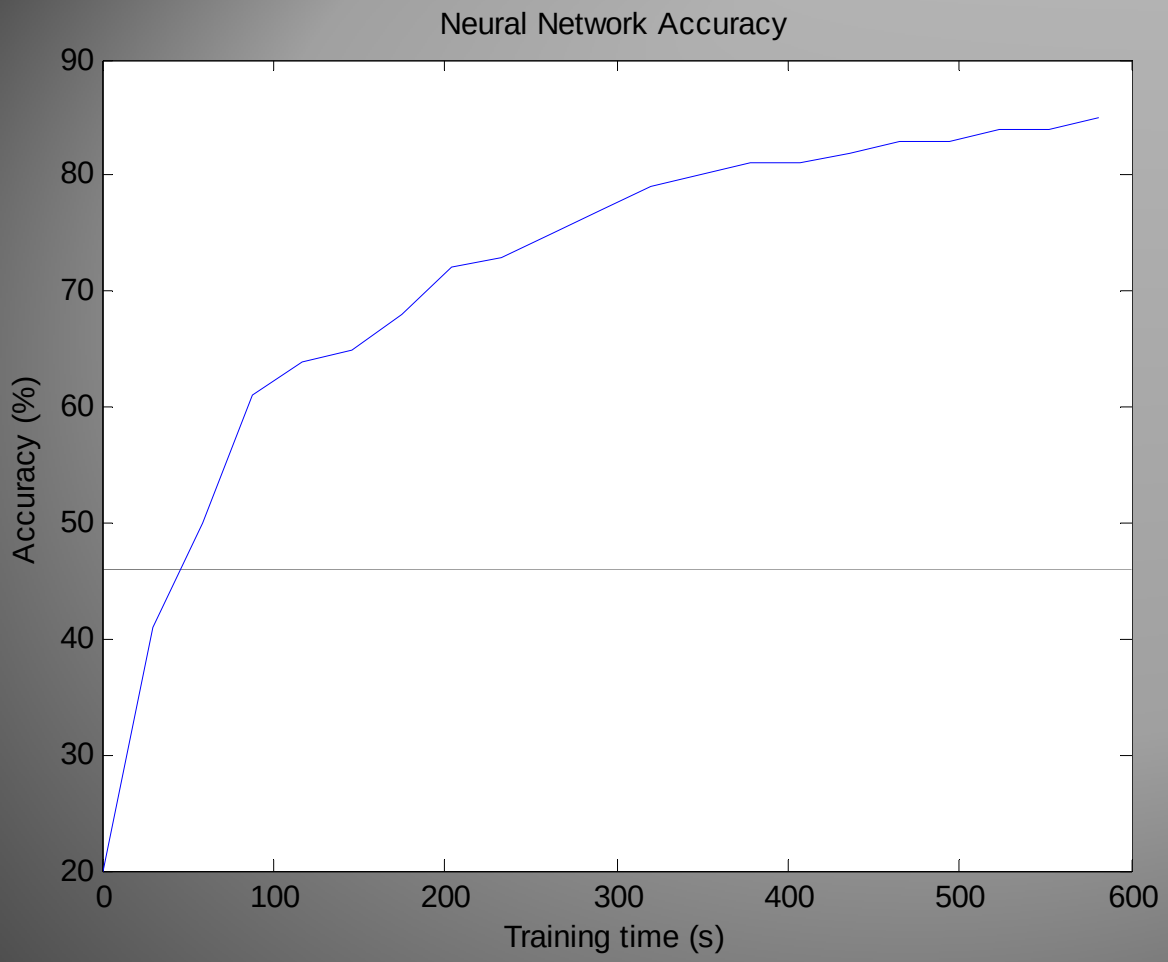
- The network was trained and the accuracy was examined at numerous points between 1s and 10m of testing.
- The error was accuracy was determined by the percentage of numbers it correctly identified.
- After reading the image and translating the image to a number string this was compared against the known true number string.

- I used 2 test populations as well.
  - Initially I wrote the data onto sheets and used only my data.
  - I then had several other people fill out the training data sheets and each create their own random test data set.
  - These samples were all combined into one large population that the network was required to analyze.



Network Accuracy (Single Population)

There were not any significant individual problem areas when examining only an individual subjects writing.



Network Accuracy (Full Test Population)

The most problematic values were 8 and 9 with over 70% of errors being these two numbers. Looking at the original data it appears that several test data subjects make 8s and 9s that look almost identical making it hard for the network to differentiate.

Network Results

- Columns 1 through 19
- 1 7 5 1 4 2 5 6 8 4 3 6 2 1 7 8 3 9 2
- 1 3 2 1 4 2 2 6 4 4 7 6 2 0 7 8 3 9 2

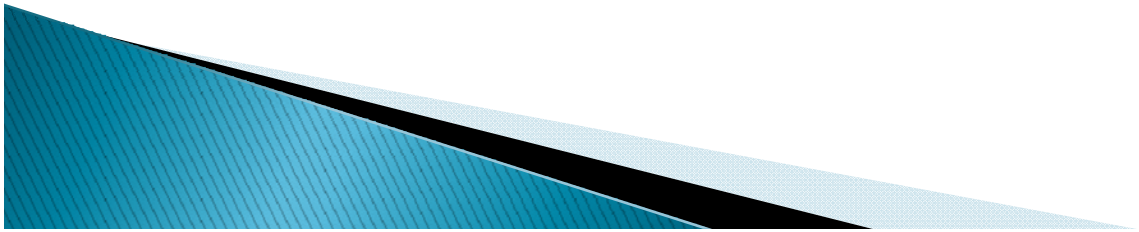
- Columns 20 through 38
- 5 1 3 5 7 2 9 4 0 8 0 9 3 6 7 1 0 5 4
- 7 1 3 7 7 2 0 4 2 9 0 9 3 6 7 1 0 4 4

- Columns 39 through 57
- 8 7 9 9 0 3 9 1 2 5 0 2 3 1 2 7 6 8 3
- 8 7 8 0 0 7 9 1 2 5 0 7 2 1 2 7 6 8 3

- Columns 58 through 76
- 6 2 5 6 3 9 1 5 2 3 7 1 8 0 5 3 4 5 6
- 6 2 5 6 3 9 1 5 0 3 7 5 8 0 5 3 4 5 6

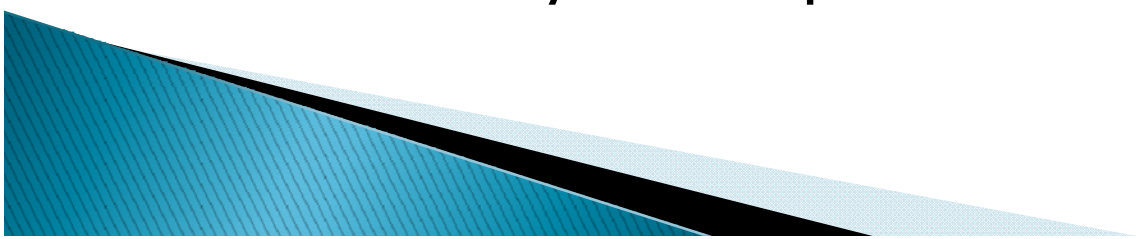
- Columns 77 through 95
- 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
- 7 2 5 0 1 2 3 4 5 6 7 8 5 0 1 2 3 4 5

- Columns 96 through 108
- 7 6 9 8 0 8 7 6 5 4 3 6 2
- 7 6 9 2 0 8 7 6 5 4 3 6 2



Conclusions

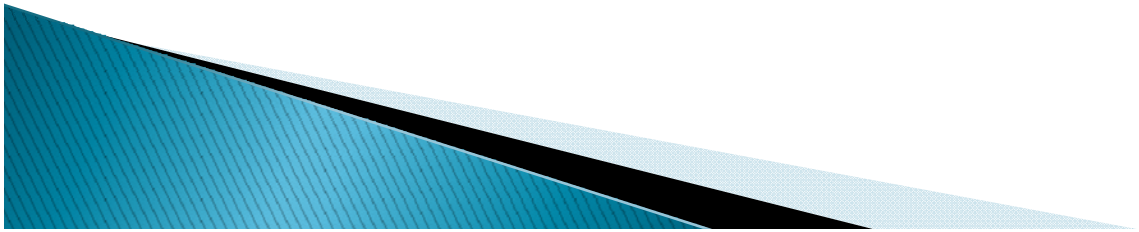
- The network shows promise at successfully identifying numbers correctly using this technique.
- While I did not test it directly, the network could easily be trained to identify numbers, letters or even symbols with no modifications.
- With relatively low training times the network can achieve a high accuracy making it a good technique for handwriting recognition.
- This type of network can also easily be modified and updated using future data to further refine its accuracy for a specific subject.



References

- Bishop, C.M., “Neural Networks for Pattern Recognition",Oxford University Press., 1995.
- Kantardzic, M., “Data Mining: Concepts, Models, Methods, and Algorithms”, Wiley-IEEE Press, 2002.
- Witten, I.H., “Data Mining: Practical Machine Learning Tools and Techniques, Second Edition”, Morgan Kaufmann, 2005.
- Zurada, J.M., “Introduction to Artificial Neural Systems”, Pws Pub. Co., 1992.

- All code used in this project is original and unique. No outside software or reference code was used.



Thank You

- Questions?