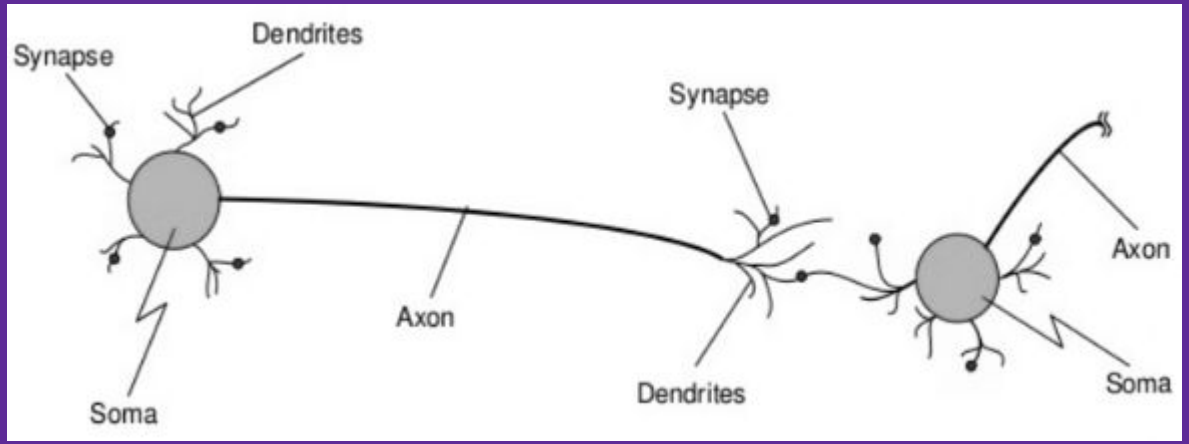
Handwritten Digits Recognition Using Neural Network

Aakash Sarang, Sassoun Gostantian

Project Advisor: Prof. Morris E Jones
Electrical Engineering Department, San Jose State University, San Jose, California 95192

Artificial Neural Network

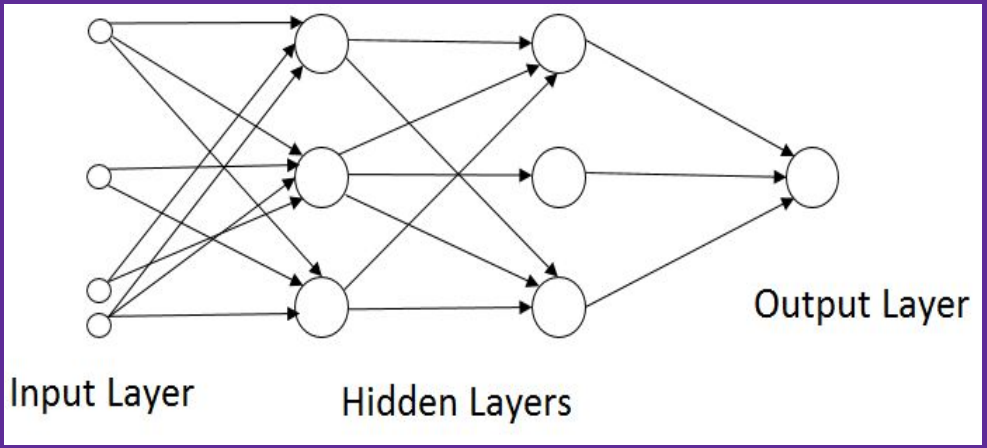
- Artificial Neural Networks are inspired from the brain neurons of brains.
- Neural Networks can be used to perform complicated computational tasks such as identifying patterns or trends which can be used for image recognition and medical diagnostics.
- The exploitation of ability of adaptive learning in neural networks exhibits the possibility of using them for complicated real time applications



Biological Neuron	Artificial Neuron
Soma	Node
Dendrites	Input
Axon	Output
Synapses	Weight

Feedforward ANN

- The network has three layers namely one input layer, two hidden layers and one last output layer.
- Every layer is connected to other layers by weighted connections and each layer forwards the calculated value of signal in neuron to the neurons in the following layer.
- The signal flows from left to right i.e. input to output layer which is called feedforward structure.



MNIST Database

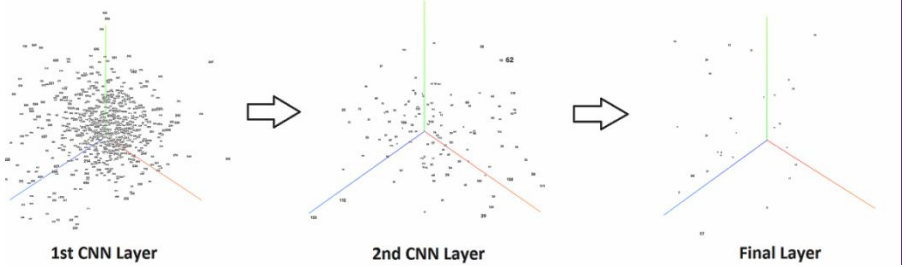
- MNIST stands for Mixed National Institute of Standards and Technology Database.
- MNIST database is basically a huge set of images of handwritten digits gathered from various sources and formatted resized and maintained in specific format to be used for training and evaluation of machine learning systems.
- It is useful because of its readiness and ease to be used with python.

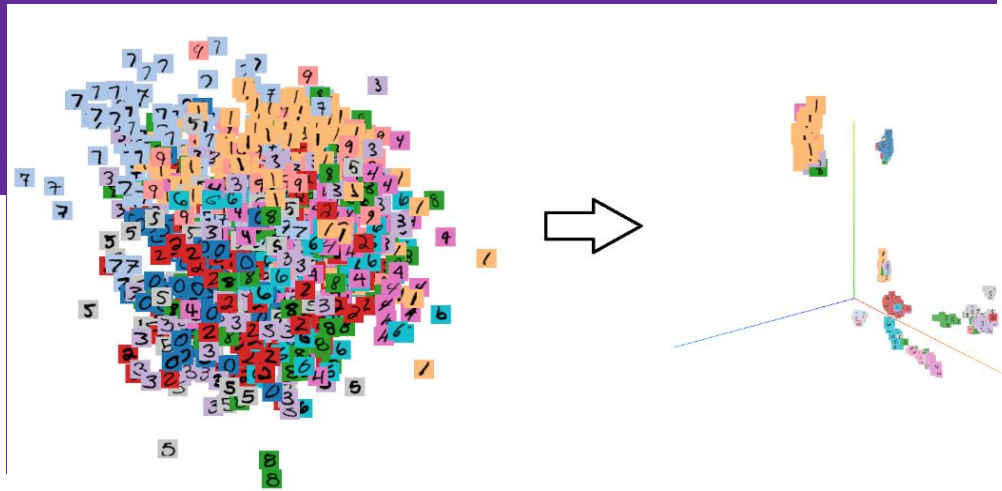
- It contains about 60,000 training images and 10,000 testing images of handwritten digits each black and white colored and formatted to 28x28 pixel size.

Some of the sample images from MNIST database.

Tensorflow

- Tensorflow can be considered as a neural network library.
- It is used to import the images from MNIST database to train and test the model that was built in python.
- It is also used to observe the performance of the design network and thereby modify it to improve accuracy of the design.
- The graphs show how ANN in tensorflow classifies the input samples into classes.





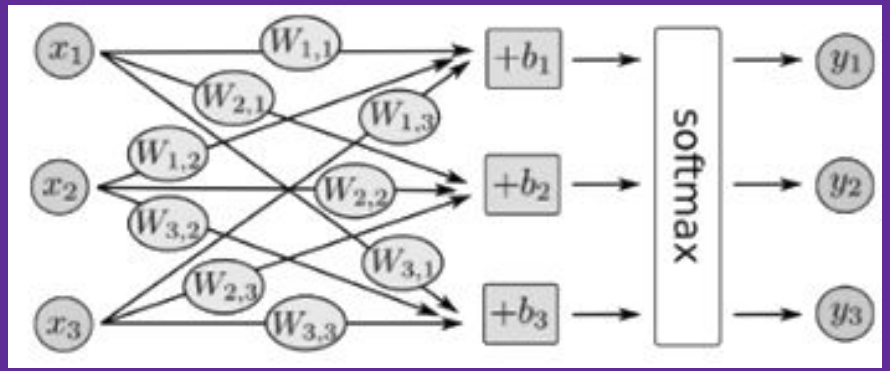
Algorithm

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = Soft\max \begin{pmatrix} W_{1,1} \cdot x_1 + W_{1,2} \cdot x_2 + W_{1,3} \cdot x_3 + b_1 \\ W_{2,1} \cdot x_1 + W_{2,2} \cdot x_2 + W_{2,3} \cdot x_3 + b_2 \\ W_{3,1} \cdot x_1 + W_{3,2} \cdot x_2 + W_{3,3} \cdot x_3 + b_3 \end{pmatrix} = Soft\max \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

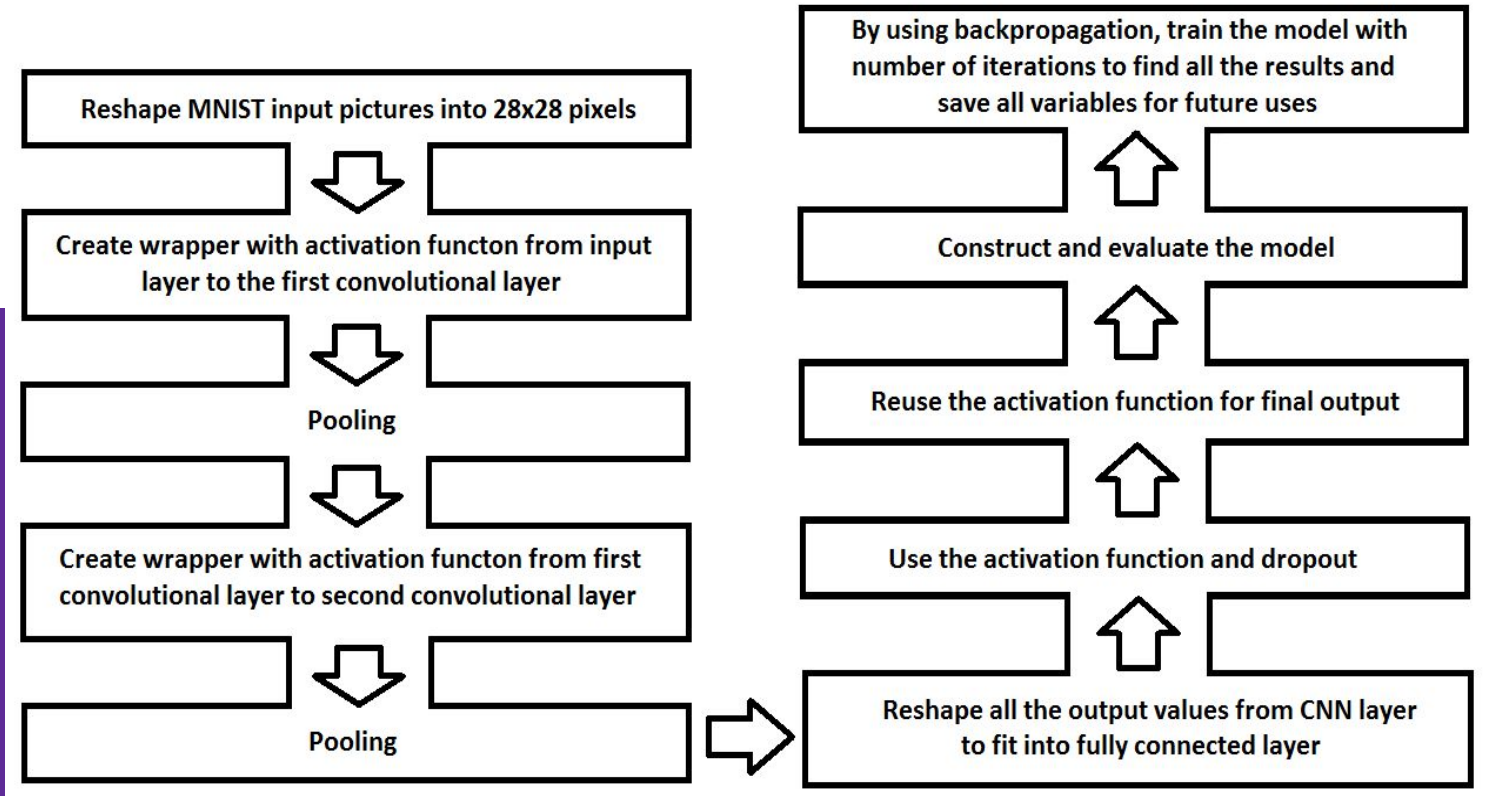
By using Softmax regression with matrix multiplication, the activation function can be found:

Y = Softmax (W.x + b)

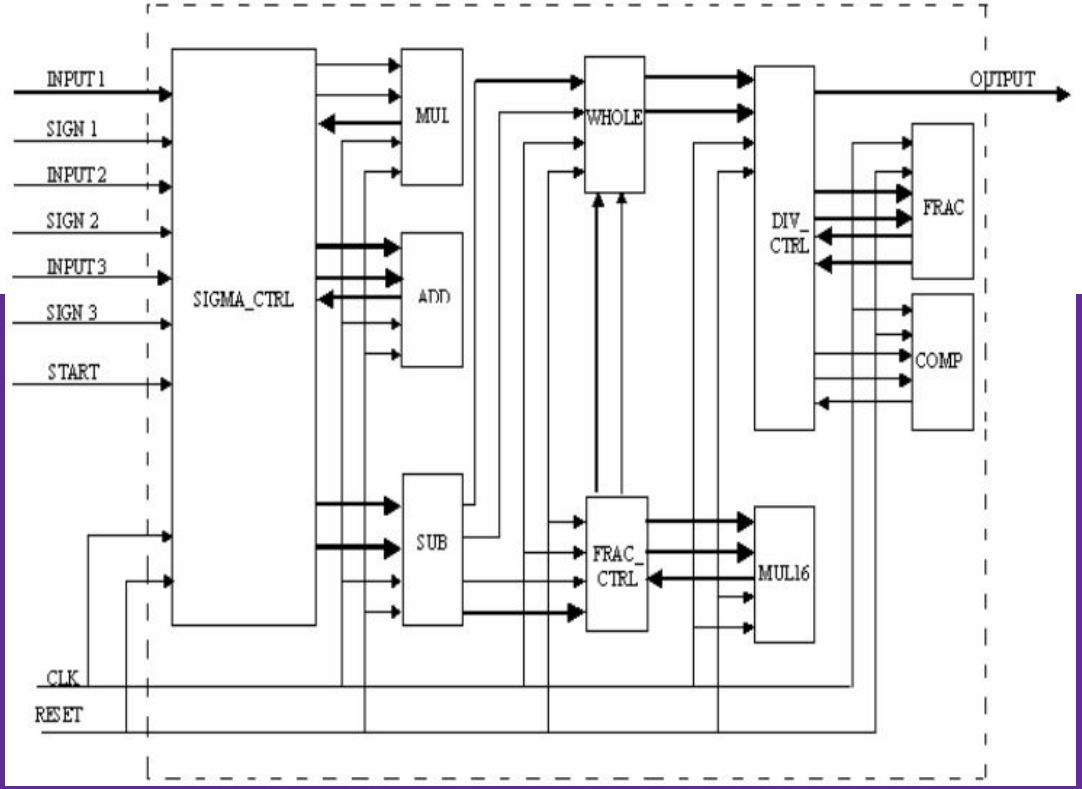
Where W is the weight of the connection between neurons, x is the inputs, b is bias and Y is the output from the neuron.



The State Diagram of the Model by using Tensorflow features



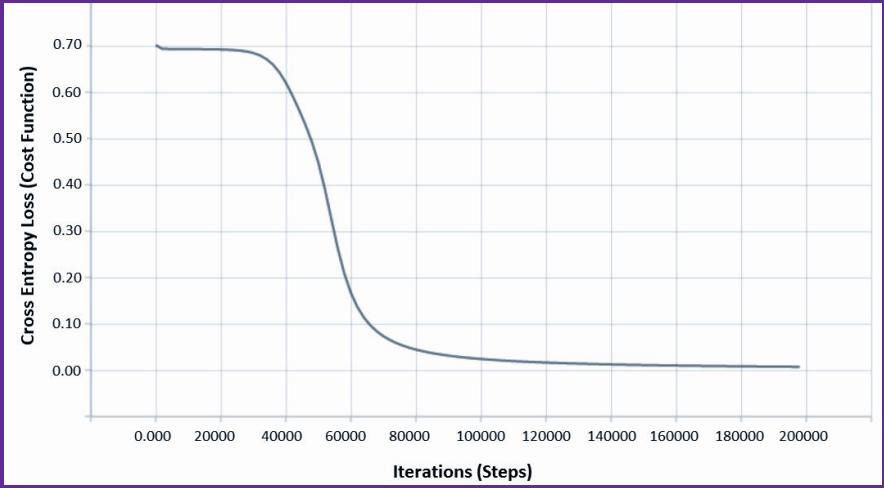
Hardware Implementation and FPGA selection



- The block diagram of the verilog design is shown in the figure along with all the major blocks.
- This block diagram explains the process involved in each neuron which is the basic building block of the design.

Results and Observations

- The screenshot shown in next slide is from the command prompt after running the python code.
- As can be seen the entropy loss efficiency results can be pulled out from it to plot them on graphs for performance analysis.

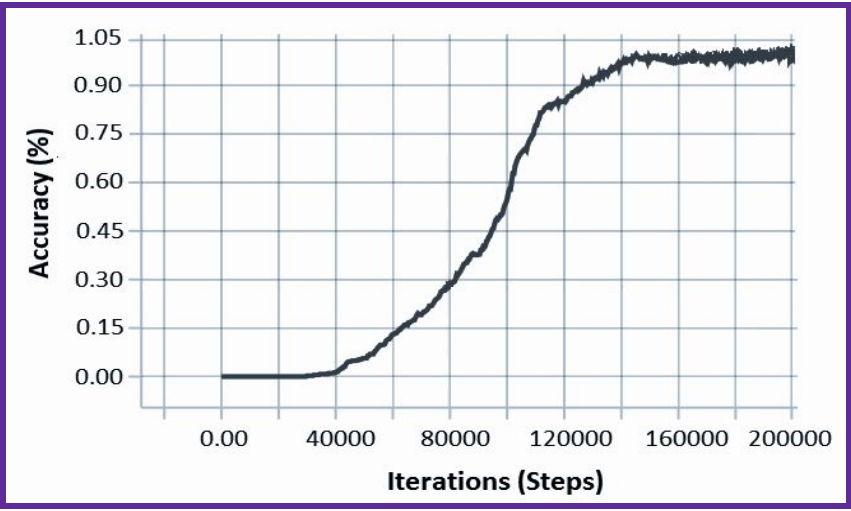


- As can be seen from the figure, the entropy loss i.e. the errors in prediction reduces with increase in the number of iterations used to train the model.

```
Number of Steps: 170240, Cross Entropy Loss: 289.208740, Total Accuracy: 0.95312

Number of Steps: 171520, Cross Entropy Loss: 136.190536, Total Accuracy: 0.97656

Number of Steps: 172800, Cross Entropy Loss: 88.445633, Total Accuracy: 0.96875
Number of Steps: 174080, Cross Entropy Loss: 91.565063, Total Accuracy: 0.95312
Number of Steps: 175360, Cross Entropy Loss: 166.442352, Total Accuracy: 0.96875

Number of Steps: 176640, Cross Entropy Loss: 146.195801, Total Accuracy: 0.96875

Number of Steps: 177920, Cross Entropy Loss: 341.528961, Total Accuracy: 0.96094

Number of Steps: 179200, Cross Entropy Loss: 34.507210, Total Accuracy: 0.98438
Number of Steps: 180480, Cross Entropy Loss: 113.395409, Total Accuracy: 0.96875

Number of Steps: 181760, Cross Entropy Loss: 39.158493, Total Accuracy: 0.99219
Number of Steps: 183040, Cross Entropy Loss: 174.324249, Total Accuracy: 0.95312

Number of Steps: 184320, Cross Entropy Loss: 109.138550, Total Accuracy: 0.96094

Number of Steps: 185600, Cross Entropy Loss: 51.924332, Total Accuracy: 0.99219
Number of Steps: 186880, Cross Entropy Loss: 194.501450, Total Accuracy: 0.96875

Number of Steps: 188160, Cross Entropy Loss: 18.838676, Total Accuracy: 0.99219
Number of Steps: 189440, Cross Entropy Loss: 95.550507, Total Accuracy: 0.97656
Number of Steps: 190720, Cross Entropy Loss: 24.949722, Total Accuracy: 0.98438
Number of Steps: 192000, Cross Entropy Loss: 107.345337, Total Accuracy: 0.98438

Number of Steps: 193280, Cross Entropy Loss: 143.922913, Total Accuracy: 0.99219

Number of Steps: 194560, Cross Entropy Loss: 120.081161, Total Accuracy: 0.95312

Number of Steps: 195840, Cross Entropy Loss: 41.068249, Total Accuracy: 0.97656
Number of Steps: 197120, Cross Entropy Loss: 68.289284, Total Accuracy: 0.98438
Number of Steps: 198400, Cross Entropy Loss: 141.722534, Total Accuracy: 0.97656

Number of Steps: 199680, Cross Entropy Loss: 83.544533, Total Accuracy: 0.98438
It has finished all the iterations
```

Results and Observations

- The performance analysis of different types of neural networks was performed using python.
- The highest accuracy was achieved using convolutional neural network with four layers which was 98.4%.
- Multi layer neural network had accuracy results between 80-90 percent depending upon the number of iterations.
- Single layer network was tested to have accuracy of 81.3 percent.



- As can be seen the accuracy increases with increase in number of iterations.
- For example the accuracy rises from 85.5% to 91.9% by increasing number of iterations from 2000 to 20,000.

Future Developments

- Matlab can be used for formatting the input images for faster operation.
- Implement camera and touchscreen to capture live images for real time operation using softwares such as OpenCV.
- The project can be extended from numbers to alphabets and then to signs for handwriting recognition of all kind which can be used in real life applications to do automatic calculations.
- The whole system can be implemented on FPGA to verify the real time operation of the design which could not be done within available time period.

Thank You.

For more information, full code, and full report,
Please go to: https://github.com/sassoun

Any Questions?