Handwritten Digit Recognizer using a Neural Network

Team Members:

David Pimley: *David Pimley*

Dustin Andree: *Dustin Andree*

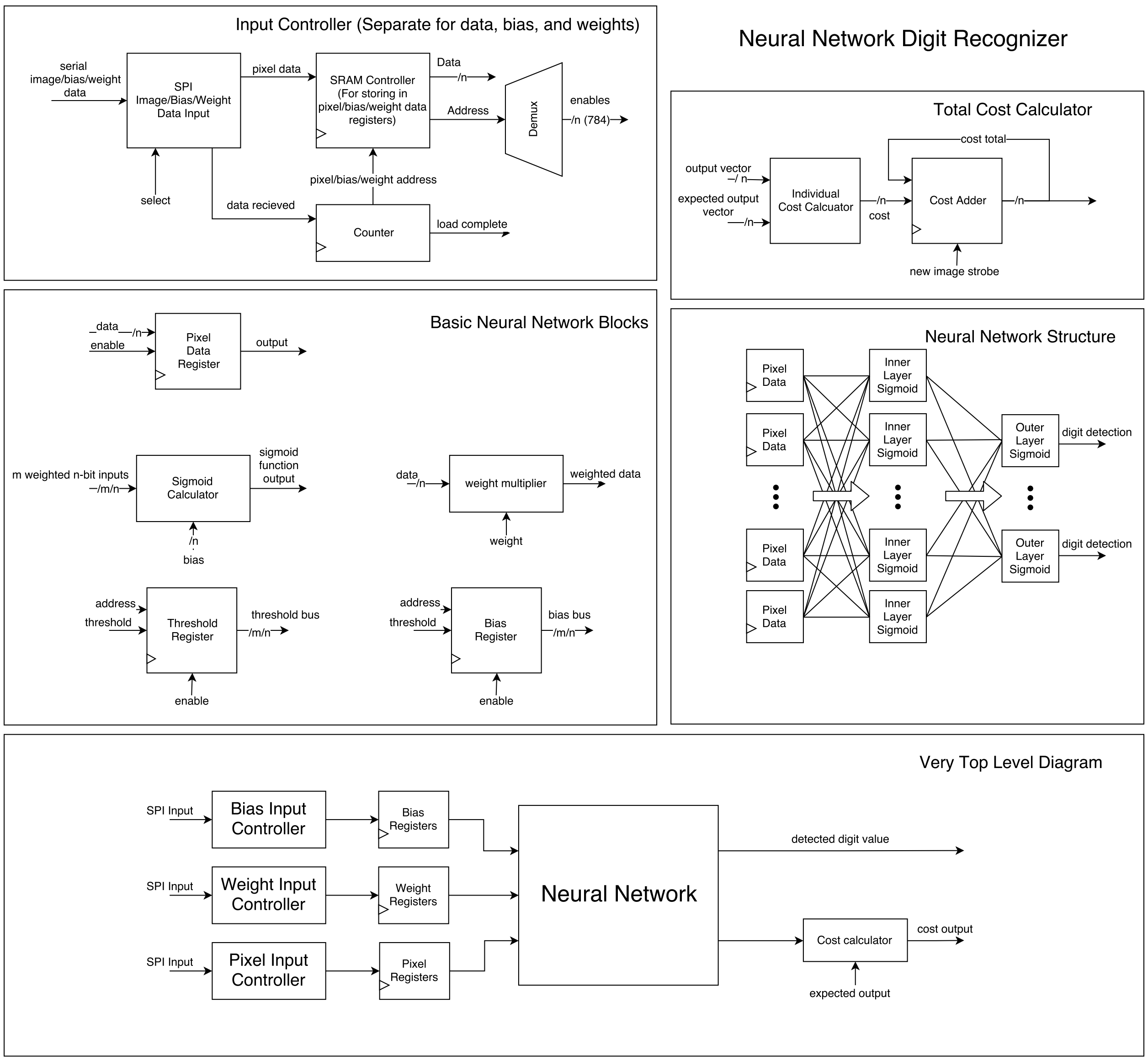
Vadim Nikiforov: *Vadim Nikiforov*

Chan Weng Yan: *Chan Weng Yan*

1 Description and Rationale

This project idea is a digit recognizer. It is designed to work with images consistent with the MNIST dataset, so it accepts 28×28 px. grayscale images, and outputs a detected digit. It accepts an image, and a list of weights and biases for the network connections. While it doesn't have an internal method for training, it outputs a cost function for each set of images, allowing for it to be connected to an external training module. Furthermore, using SPI, the network model can be reprogrammed on the fly. Digit recognition is a very useful application, as it allows for devices to interpret data ranging from checks to signs to handwritten notes. Machine learning allows for very high accuracy for digit detection, and when applied to digit recognition can handle a wide range of fonts and handwriting styles. However, due to the structure of neural networks, sequential computation such as what is found in microcontrollers is not effective for a neural network. Instead, devices such as GPUs are often used due to their capacity for parallel computation. This device uses an architecture specialized just for multiplying weights and adding inputs, allowing for much faster computation for digit recognition. While on a microcontroller this system would require tens of thousands of sequential multiplications, on an ASIC this device will have critical paths with only 3-4 multiplications necessary before the output. This is done by taking advantage of the `MapReduce` procedure, where in this case the mapping is applying weights to the biases, and the reduction is adding together all the inputs to a sigmoid, and then calculating the sigmoid output (which can be done by a lookup table.) Finally, this device allows for training to be done on separate hardware, which typically requires far more computational power than detection, and then allows the trained model to be uploaded on the fly using a standard interface (In this case, SPI) allowing for flexible operation.

2 Design Architecture


Input Controller (Separate for data, bias, and weights)
serial image/bias/weight data
SPI Image/Bias/Weight Data Input
select
pixel data
data recieved
SRAM Controller (For storing in pixel/bias/weight data registers)
pixel/bias/weight address
Counter
load complete
Data
/n
Address
Demux
enables
/n (784)
Neural Network Digit Recognizer
Total Cost Calculator
output vector
/ n
expected output vector
/n
Individual Cost Calcuator
/n
cost
Cost Adder
cost total
/n
new image strobe
Basic Neural Network Blocks
data
/n
enable
Pixel Data Register
output
m weighted n-bit inputs
/m/n
Sigmoid Calculator
sigmoid function output
/n
bias
data
/n
weight multiplier
weighted data
weight
address
threshold
Threshold Register
threshold bus
/m/n
enable
address
threshold
Bias Register
bias bus
/m/n
enable
Neural Network Structure
Pixel Data
Inner Layer Sigmoid
Outer Layer Sigmoid
digit detection
Very Top Level Diagram
SPI Input
Bias Input Controller
Bias Registers
SPI Input
Weight Input Controller
Weight Registers
SPI Input
Pixel Input Controller
Pixel Registers
Neural Network
detected digit value
Cost calculator
cost output
expected output