Handwritten Digit Recognizer using Neural Networks

Phase 1 Project Proposal
TA: Anirudh
Wednesday | 11:30

David Pimley
Chan Weng Yan
Dustin Andree
Vadim Nikiforov

ECE 33700
2/12/2018

Purdue University
Computer Engineering

1 Executive Summary

This project idea is a digit recognizer. It is designed to work with images consistent with the MNIST dataset, so it accepts 14×14 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 it will be possible to run calculations in parallel, and implement pipelining for addition and multiplication. 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.

To complete this design of a digit recognizer, the following will be needed:

- SST39LF200A (×16) Multi Purpose Flash Datasheet
- Reference Standard Cell Simulation Library for Mapped Design Verification
- Reference Standard Cell Technology Library for Final Design Layout Verification
- Verilog HDL Simulation and Design Synthesis Tool Chain

The following design proposal will include:

- A top level system usage diagram for an example configuration/usage of the digit recognizer circuit.
- The standards/algorithms to be implemented by the digit recognizer circuit.
- The design pinout of the neural network, and input output signals with their associated access codes.
- The specific design architecture to be utilized by the digit recognition circuit.

2 Design Specifications

2.1 System Usage

2.1.1 System Usage Diagram

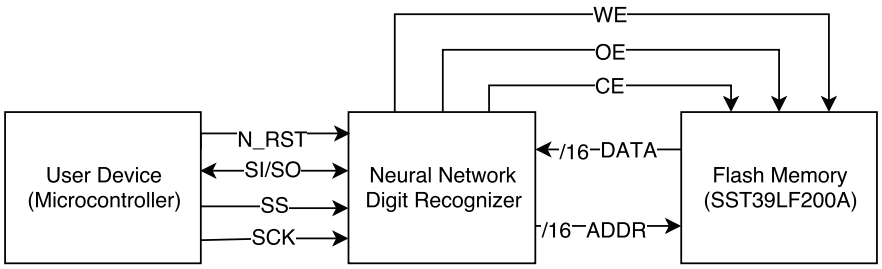

Figure 1: Example system usage diagram of neural network digit recognizer

This design is intended to accelerate the recognition of digits by implementing a neural network in an ASIC, allowing for parallel computation. However, the functionality can only be useful when interfaced with a general purpose computing device. The application depicted in the system usage diagram shows the recognizer interfacing with a microcontroller via SPI. An additional necessary module is flash memory. This is used to store the weights and biases for the neural network, and is used to prevent the need to reupload the model every time. If the model needs to be changed, then it can be uploaded directly to the flash memory using an external chip. The microcontroller in this example would be used as a control device. While the majority of the work being done is accomplished within the Neural Network Block, the control functionality comes from the user device performing the following tasks:

1. If necessary, reset the registers (everything but the Flash Memory)
2. Using the SS, the SPI register will choose which operation to be done. The four operations are:
   (a) Let the digit recognizer know that image data will be sent over MOSI.
   (b) Ask the digit recognizer for the result of the previous operation.
   (c) Ask the digit recognizer for the cost of the previous operation.
   (d) Let the digit recognizer know that the weights and bias will be changed in which case the registers will need to be flushed.
   (e) Let the digit recognizer know the expected label of each image (needed for cost output)

2.1.2 Implemented Standards and Algorithms

SPI:

- Synchronous serial communication between the user device and the neural network digit recognizer.
- MOSI/MISO compatible with external device (such as a microcontroller).

Flash Memory:

- Permanent storage memory for bias/weight data.
- Communicates with a 16 bit address bus, a 16 bit data bus, and read,write, and chip enable lines.

Pipelining:

- "Pre-Processing", once values from a register are used, those values are updated with the next value to be utilized, if a calculation requires multiple cycles.
- Allows the same arithmetic unit to be used for multiple calculations at once.

ALU:

- Fixed Point Addition
  - 16 bit signed addition used for bias addition to each weighted pixel.
- Fixed Point Multiplication
  - Extension of Fixed Point Addition, used to multiply matrices in parallel within the digit recognizer to speed up processing.
- Sigmoid Function
  - Used for the output of the sigmoid neuron registers, calculated using a lookup table.

Signals:

Command Code	Description
`1'b000`	New image data is being input over MOSI
`1'b001`	New expected label is being input over MOSI
`1'b010`	Output the cost of the last operation
`1'b011`	Output the result of the last operation
`1'b100`	The weights/biases have changed, flush registers

Table 1: Device instruction bits for SPI

2.2 Pinout

Signal Name	Direction	# Bits	Description
PWR	power	N/A	System Power
GND	ground	N/A	System
CLK	in	1	System Clock
N_RST	in	1	Active low reset signal

Table 2: Miscillaneous Pinouts

Signal Name	Direction	# Bits	Description
MISO	out	1	Master in slave out signal
MOSI	in	1	Master out slave in signal
SCK	in	1	SPI clock
SS	out	1	Slave select Signal

Table 3: User device pinouts

Signal Name	Direction	# Bits	Description
WE	out	1	Write Enable Signal
OE	out	1	Output Enable Signal
CE	out	1	Chip Enable Signal
DATA	in	16	Bias/weight data from the external flash memory
ADDR	out	16	Address for flash memory

Table 4: Flash memory pinouts

The signals specified in Table 2 represent any signals necessary for the chip to be powered and clocked. These are not labeled in the design architecture. Next, the pins in Table 3 are the signals that are exposed to any device that will use the neural network for detecting digits. Finally, the pins in Table 4 are the signals that are necessary for the neural network to interface with its flash memory.

3 Design Architecture

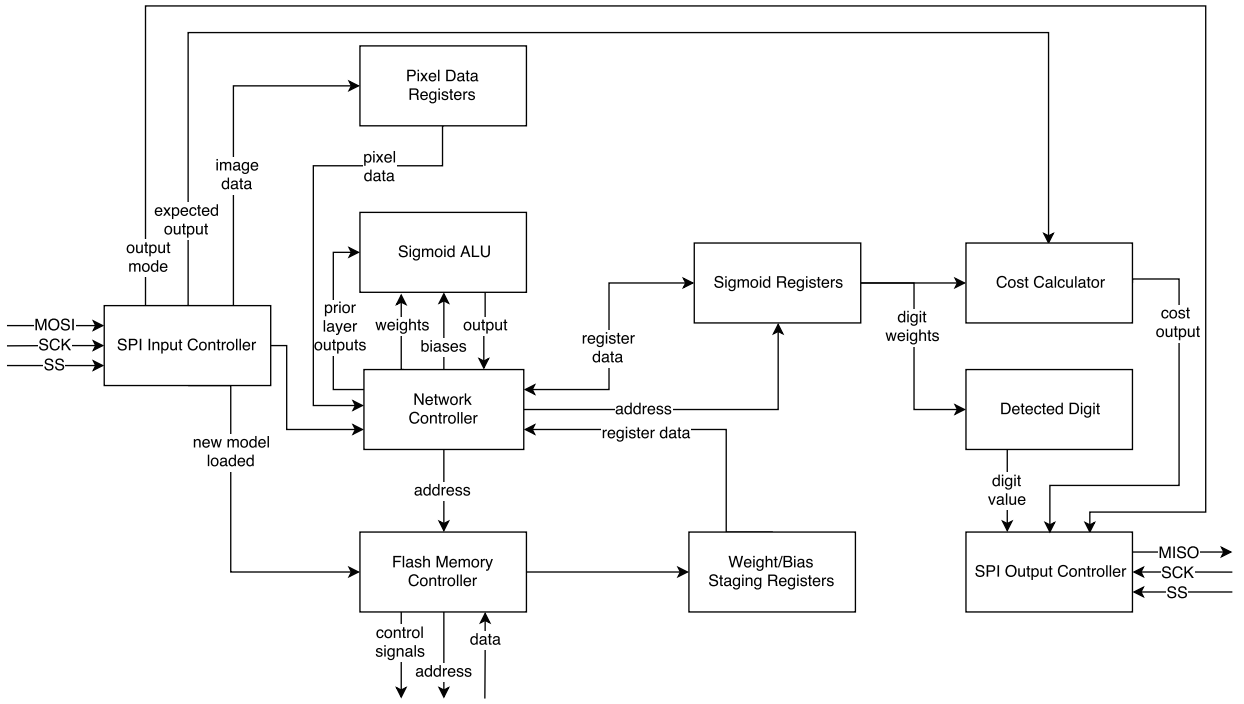

Figure 2: Digit recognition circuit design architecture

The design architecture of the digit recognition circuit is shown in Figure 2. All input first is fed to the SPI Input Controller where the data is filtered into one of a few different blocks. Each data destination block is determined by the input code. The input codes are laid out in Section 2.1.2. For the standard operating mode (output), image data is fed into the input controller and sent to a staging register used specifically for holding the data of each image. The network controller uses this data as well as data from the weight/bias staging registers (loaded from flash memory) to be input into the Sigmoid ALU to calculate probabilities for each number, *i.e.* 0-9. Depending on if the microcontroller asks for the cost analysis of the previous operation, the cost can also be output on the MISO line used for outputting the detected digit.