Initial Report

Timothy Baalman

ECEN 4303

Part 1 What I Learned (Research):

(3441) But what is a neural network? | Chapter 1, Deep learning - YouTube

Python Programming Tutorials

Multilayer perceptron (MLP) - Introduction to neural networks | Coursera

1. Introduction to Artificial Neural Networks - Neural networks and deep learning [Book] (oreilly.com)

PyTorch Prerequisites - Syllabus for Neural Network Programming Course - deeplizard

Neural networks and deep learning

PyTorch documentation — PyTorch 1.9.0 documentation

Verilog (chipverify.com)

Python classes were a little confusing at first, but once I realized that doing self.variable was the same as setting up public or private variables in a C++ class it made way more sense. And how self refers to that instance of the class. I also found out that Python classes are weird since it's so old and classes were added in as an afterthought.

I also learned about Pytorch Tensors and their many operations like how a batch of images is typically stored in a Tenosr, and how to go about accessing each pixel in the rank 4 Tensor where axis 0 is the batch index (or image we are accessing), axis 1 is the color channel index we want (gray only has 1 RGB has 3), axis 2 is the height, and axis 3 is the width. There is also the flattening operation which takes the color channel, the height, and width and puts them into one axis.

Since we are using the MNIST dataset I also learned about it where it has 60,000 images to train on and 10,000 images to test with. Along, with each image being grayscale and 28x28 pixels. Where the class index relates to the number value or label since there are 10 classes from 0-9.

How we inherit the Module class to allow us to use the forward method for forward propagation and the Module's other built-in functionality for keeping track of our Tensor data. Like how we can access all layers weights with the network.parameters() or network.named_parameters() functions, or individually with network.layer.weight. Also, how Linear layers are essentially just matric multiplication.

Also, how a basic neuron where we have multiple input values from the tensor getting multiplied by the weight of that single tensor value. Then after all the inputted values have been multiplied by their corresponding weight, we sum them all together, then run through an activation function to determine if the neuron is active.

Finally, I have investigated Verilog adders, multipliers, RAM, and how to go about implementing functions like relu and argmax.

Part 2 The Plan:

1) **Training the Network with PyTorch:**
    i. Setup Network to train on using an input layer, a single hidden linear layer, and an output linear layer. (Done)
    ii. Implement Training on the GPU to allow for faster training (Done)
    iii. Figure out Collecting the weights and bias from the best (as in the most accurate with the least loss) run when I have the training loop vary the batch_size, learning_rate, and shuffle. (9/18/21)

2) **Output weights from Network into Verilog code:**
    i. Loop through the collected weights/bias and use python to write them to a .sv or a .v file for the Verilog modules to use, and possibly a JSON file which has the parameters used along with the weights. (9/20/21)

3) **Setup Verilog hardware:**
    i. Create a basic neuron using the adders and multipliers as a template for later use. (9/27/21)

4) **Utilize inputted weights in hardware:**
    i. Use python to loop through the save weights and bias and create the interconnections and setup of each neuron (corresponding Verilog modules). (10/10/21)

5) **Process images into tensors:**
    i. Use python to grab selections of test images from the MNIST test dataset and process them into tensors. Saving each image into it's own .v or .sv file. Where the outputs of these image modules will be can be interchanged as the input for the network previously created. (11/17/21)

6) **Additional Layers**
    i. If I get everything set up and in working order with just the one hidden linear layer, I would like to add another hidden linear layer and try to implement a convolution layer too. (TBD)