Carl G. Britt III
ECE 692 – Project 1
9/12/2017

Task 1 & 2:

TensorFlow Portion

Figures 1-3 are used to compare performance across multiple neural network architectures. Other important important parameters for this setup is as follows: the epoch number was set to 100, learning rate at 1E-3, and the batch size was fixed at 50 samples. Figures 4-7 are used to study the scalar hyperparameters for a single neural network architecture. Throughout this analysis, The convention as that each layer is fully connected and is separated by an underscore, such that "784_30_10" describes a one hidden layer fully connected neural network with a 30 neuron hidden layer, and 784 neuron input layer and 10 neuron output layer. For any architecture, the code is always put through a softmax activation function in the final output layer for classification, as it was observed that this obtained the best accuracy. Furthermore each neuron was initialized using a uniform sampling from -1 to 1.

According to Figure 1, the simplest structure, 784_10, had the best accuracy. Furthermore, across all structures, the sigmoid activation function required the less number of epochs to train. Note that I would not trust relu's results for 784_30_10 within this analysis, as accuracy of around 10% usually denotes a bug.

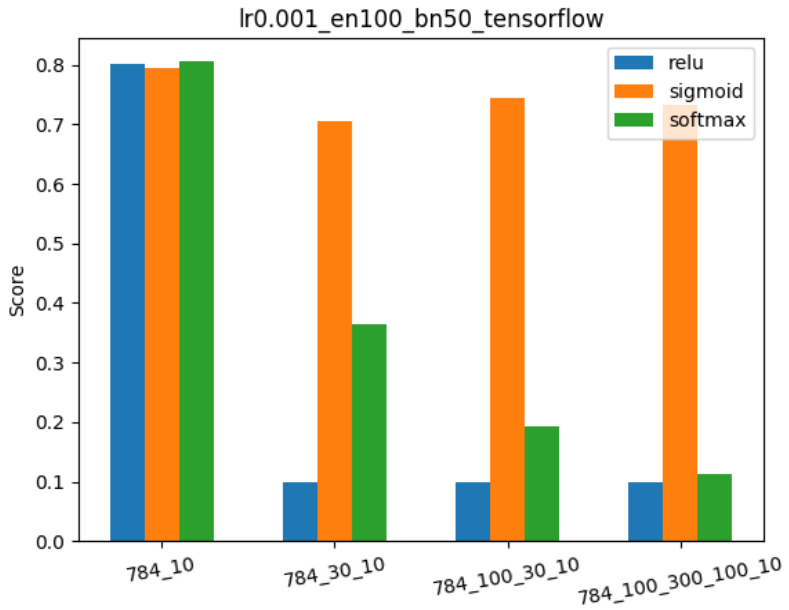

Figure 1: TF architecture accuracy comparison

According to Figure 2, the more complicated architectures take longer to train. Furthermore, the softmax activation function takes longer the more complicated the architecture. Figure 3, the evaluation time as a function of network architecture, looks very similar to Figure 2, as expected.

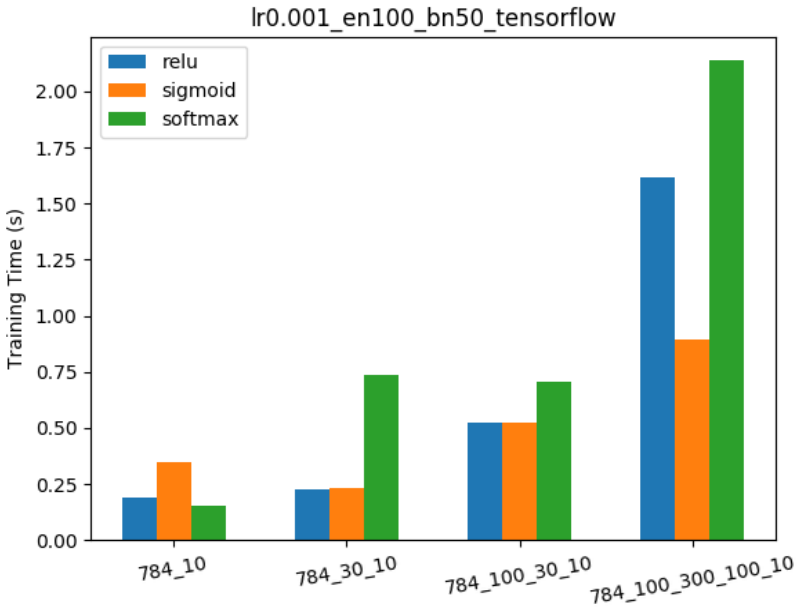

Figure 2: TF architecture training time comparison

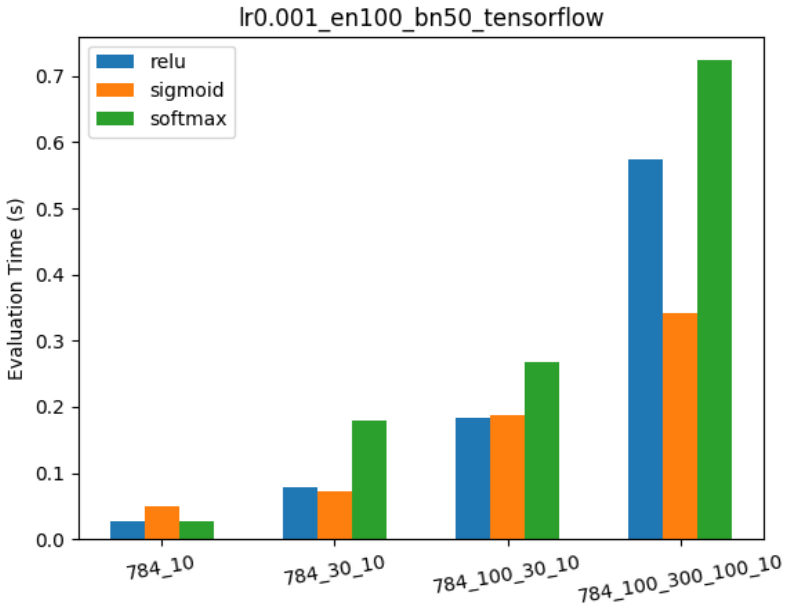

Figure 3: TF architecture evaluation time comparison

For Figures 4-7, the scalar hyperparameters were varied while using the 784_30_10 neural network using the sigmoid activation function. Figure 4 reveals that while the number of epochs increased, the better the accuracy. This matches expectation as the epoch number is an indicator of how much training the neural network has received.

As a general trend, the greater the batch number, the better the accuracy. This is not always true, as shown when the epoch number is 200, but it is a general observation. This is also generally true for bigger batch sizes as they have a better estimate on the gradient of the objective function, therefore altering the weights in a more favourable direction. Lower batch sizes tend to have erratic changes in the gradient, causing the weights to alter in extreme directions on its journey towards the cost functions minimum.

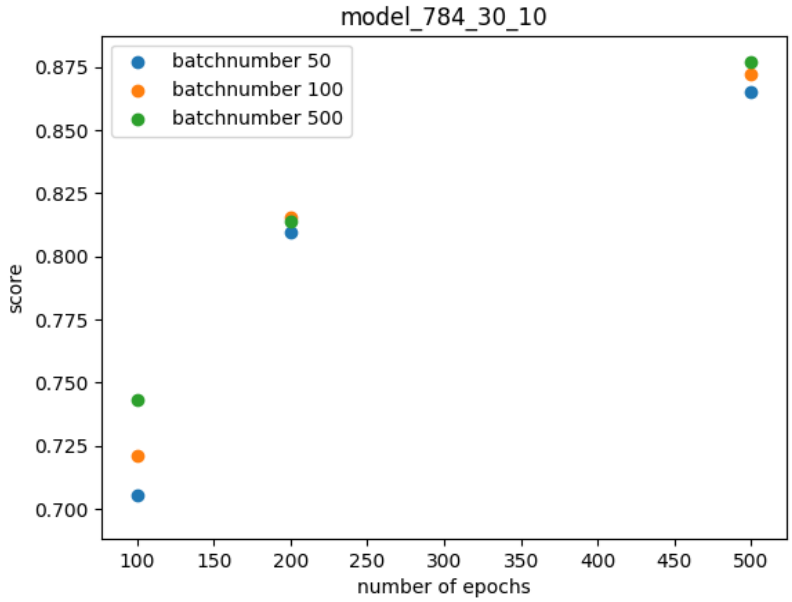

Figure 4: TF epoch accuracy comparison

Figure 5 shows the training time for different epochs and batch numbers. As batch numbers increase, the training time increases, as the gradient calculation is more intense (includes more samples).

Figure 7 reveals that the score is independent of the learning rate for this particular case. However, the score is dependent on the batch number. As the batch number increases, the score tends to increase. This is due to the better estimate on the cost functions gradient brining the network closer to convergence for a fixed epoch number. Also note that the lower learning rates tend to yield slightly better accuracy.

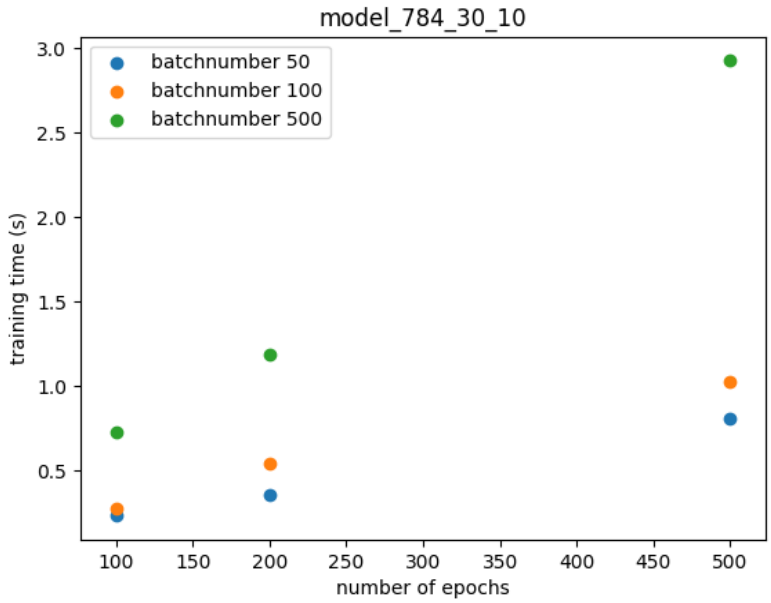

Figure 5: TF epoch training time comparison

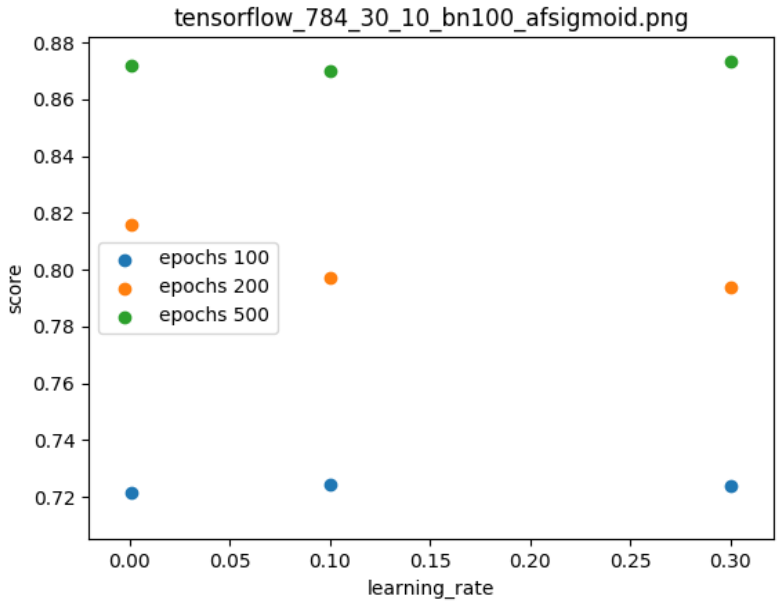

Figure 7: TF learning rate comparison

NumPy Portion

The figures placed here are the same and are in the same order as in the TF portion. The general trends noted in the TF portion will be the same as in the NumPy (NP) portion. The NP portion are the results generated using Nielson's code. Note that the NP version is not as extensive as the TF portion, as the run times were on the order of days for half of the run size.

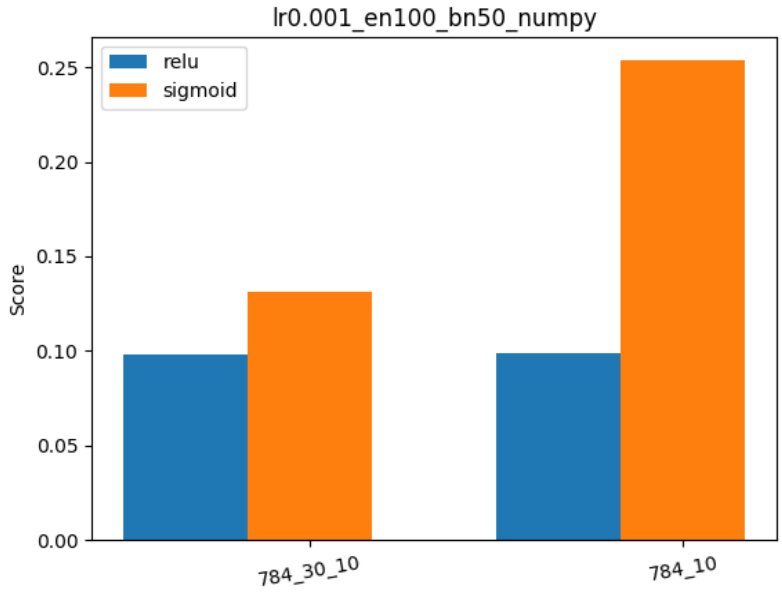

Figure 8: NP architecture accuracy comparison

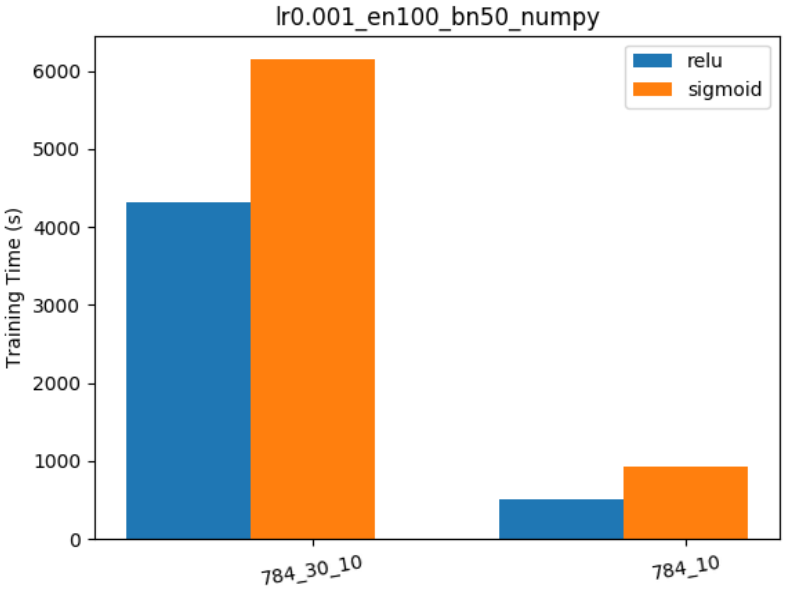

Figure 9: NP architecture training time comparison

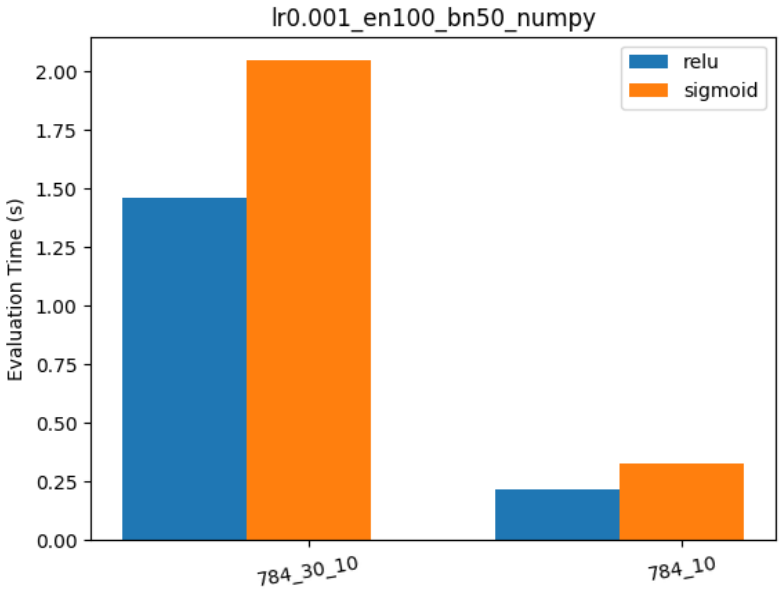

Figure 10: NP architecture evaluation time comparison

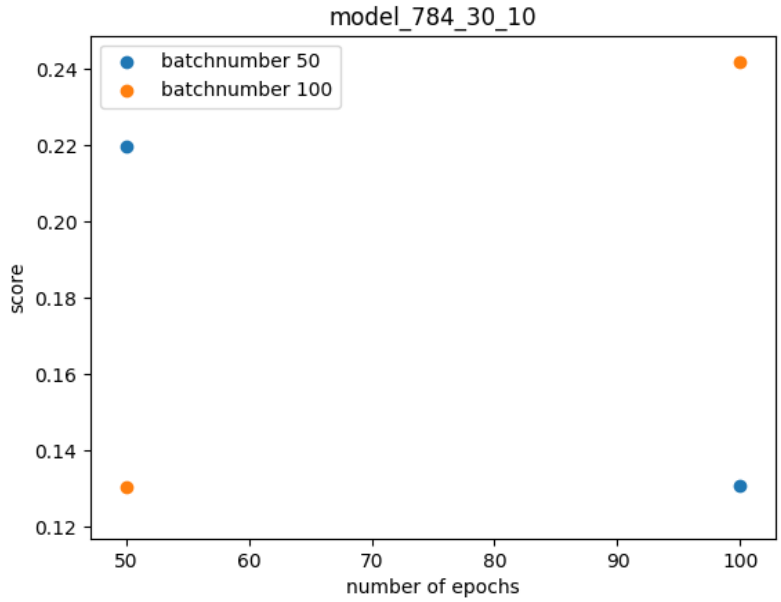

Figure 11: NP epoch accuracy comparison

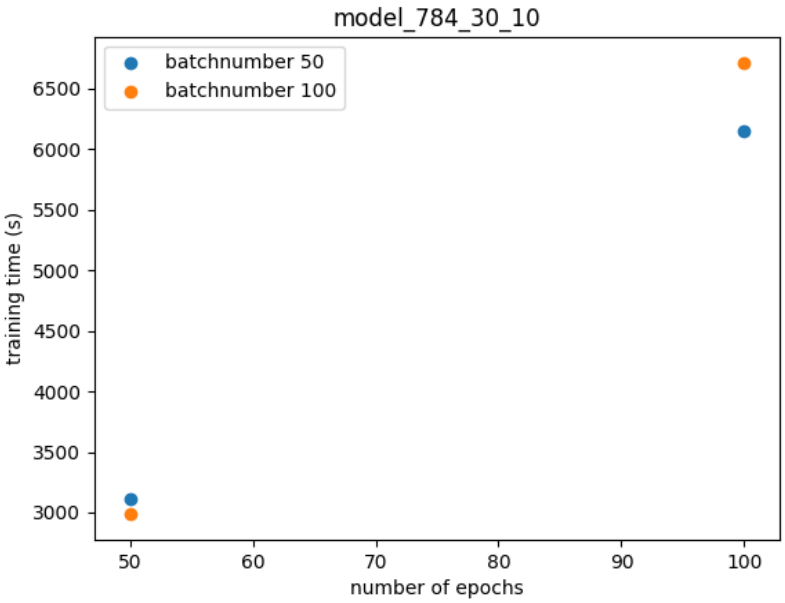

Figure 12: NP epoch training time comparison

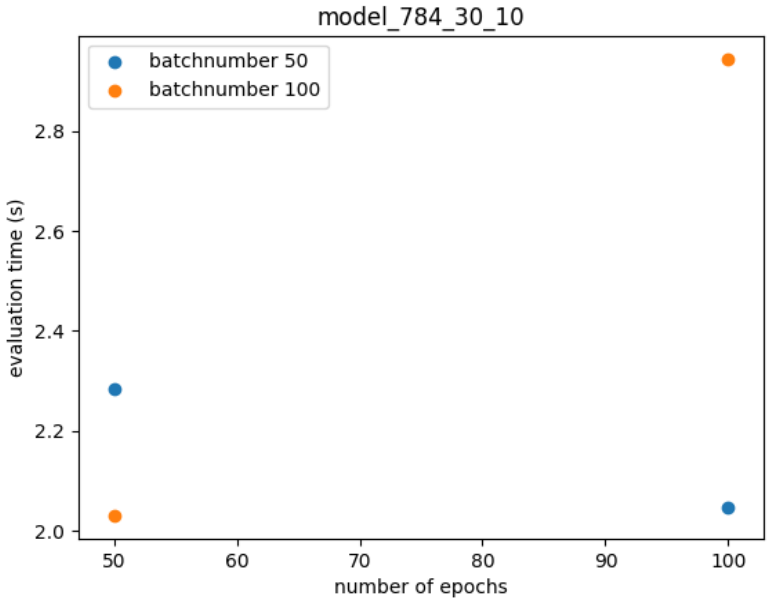

Figure 13: NP epoch evaluation time comparison

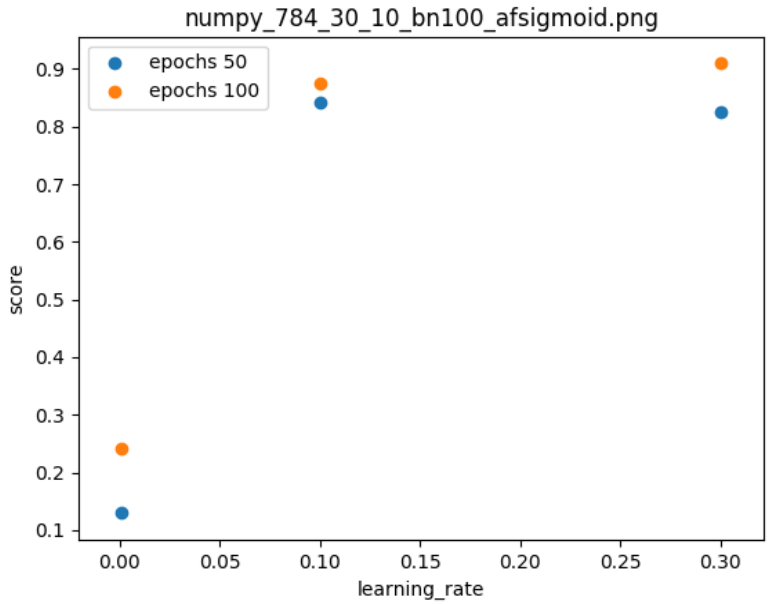

Figure 14: NP learning rate comparison

Task 3: Four Subjects

1) Batch vs. Online: This addresses when the weights of the model are updated. Batch describes that the weights of the model are updated after each epoch, while online updates the model immediately after each sample goes through the model.

2) Gradient descent vs. stochastic gradient descent: This addresses what is used to estimate the gradient used for updating the weights within the model. Gradient descent uses the entire epoch to calculate the gradient of the cost function, while stochastic gradient descent uses a randomly selected subset of the epoch (mini-batches) to approximate the gradient of the cost function.

3) Perceptron vs. Sigmoid neurons: These are activator functions which describe the behavior of a neuron firing, as well as determine how the network's weights should change. A perceptron is a step function that is continuous, but not everywhere smooth (not everywhere differentiable), while sigmoid is everywhere continuous and smooth function (everywhere differentiable) where its value can be anywhere from 0 to 1. Sigmoid is preferred to perceptron as its derivative is everywhere diffentiable making weight changes more stable and predictable during backpropagation, while the Perceptron does not have this property.

4) Feedforward vs. Backpropagation: These are terms related to the architecture of a network. Feedforward describes the process of the input signals being mapped to the output at each layer via the transfer functions (through the matrices / layers). Backpropagation is the training process which uses the errors to manipulate the weights of the network (manipulate the matrices' values) in order to get better performance during the feedforward step.