Deep learning HW1

Submitted by: Liron McLey 200307791, Matan Hamra 300280310 & Ariel Weigler 300564267

Model Architecture

Number of parameters: **64010**

We built our model such that each linear layer is some number to the power of two. We used four linear layers with transfer layers in between activating either ReLU or leaky ReLU. The criterion used for the loss function was Cross Entropy.

The model's final hyper parameters are:

1. Batch size = 50
2. Learning rate = 0.05
3. Loss criterion = CrossEntropyLoss
4. Optimization criterion = SGD with momentum
5. # of epochs = 30

Training procedure

The training and test sets was downloaded to the computer after normalization according to the mean and standard deviation of the MNIST training set. We used the architecture and parameters described above to construct our neural network. After a random shuffle of the training set, performed by the data loader, each epoch ran as follows:

1. A working batch is selected from the training set, over which the code… :
    a. Runs forward propagation
    b. Calculates the loss value and error
    c. Uses backward propagation and updates the network weights
2. The updated model is run over the test set
3. The loss and error of the test set is calculated

Summary

Since the principles of what works in neural networks are quite cryptic, as was frequently reinforced in class, our general approach was to start with a basic model and experiment with the different parameters we had control over. We started with a 3 layer model and ReLU as the activation function after each layer, keeping the number of parameters below the required limit of 65,000, a batch size of 100 and a learning rate of 0.01. At first we worked with the learning rate and epoch number to make sure our code is able to converge to at least 95%. Afterwards we played a bit with the number of layers and settled on a 4 layer model. Further experimentation led to the conclusion that for our architecture 0.05 seems to perform best for the learning rate, above ~30 epochs there seems to be no improvement in results and that a batch size of 50 works better. Next we tested some of the different optimizers presented in class (SGD w/ and w/o momentum, Adam…) and saw that SGD with momentum worked best.

Results

Our best results: **98.22%** accuracy on the test set.