Baseline Architecture Decision

Our baseline architecture consists of a single convolutional layer followed by a max pool and two fully connected linear layers, each with relu activation functions. Our architectural design goals revolved around producing a lightweight, but powerful model, which is why we limited the total network depth to 4 (7 including all activation and softmax functions). Our network is contingent upon the assumption that handwritten digits are separated only by several higher order features, which can then be connected to analyze the digit itself.

Relus were chosen as non-linear activation functions to add complexity to our network, while also preserving local linear conditions for better gradient descent convergence. Our first layer consists of a 3x3 convolution with 35 feature maps. The smaller filter size coupled with larger feature map length should extract multiple granular features for each image. After passing this through the relu network, we are left with indicators of the relative strength of each of these 35 learned features. To simplify the resulting images, we pool across 2x2 squares in the activated image to learn only the most prevalent aspects of each of the features in the image. As opposed to including additional convolutional layers, we chose the pool layer because it would inform us about the prevalence of higher order features the most quickly. Downstream filters would then be able to use that information more efficiently.

We feed the resulting output into two fully connected layers separated by a non-linear relu function. The first layer takes the 5915 outputs of the pool layer and uses each of these to transform the data into 100 outputs. Our rationale behind this was to use all the information about the prevalence of higher-order feature to produce a more granular description of the image we are analyzing. he subsequent layer then uses these 100 descriptors in a fully connected layer to produce the unnormalized log probabilities which are then passed into the softmax function to produce final probabilities.

Hyperparameter Choices

Our base we used a base learning rate of 0.01, having the rate each time training loss plateaued. Since we implemented a momentum approach to gradient descent, we found a value of 0.99 for rho worked best. Finally, we used a weight decay rate of 0.0005 and a batch size of 128. We half the learning rate each time the training loss has not increased for 30 iterations (which we also consider as a hyperparameter).

Final Test Accuracy

Our final test accuracy is: 96.95%.

Filter Depth 15:

Time to completion: 70 minutes, 200 iterations.

Accuracy: 93.55%

Convergence: graphs provided

20 minutes 50 iterations 🡪 80 minutes 200 iterations with changing filter depth to 60 for first conv (and changing linear input to 10,140).

Accuracy:

Convergence: graphs provided

Training time for very complicated model (training accuracy of 94.57%)

More depth, larger network.

Over/Underfitting