Amir David Nissan Cohen 200437192
Uzi Cohen 043228311
Izik Golan 201299781

Exercise 1 Report – MNIST classification

Model

Our model is composed mainly of two parts: convolutional blocks, and fully connected layer. In order to prevent overfitting, we also incorporate dropout layers for regularization:

- Convolutional blocks are used in order to reduce the number of parameters, and capture repeating spatial pattern along the input images. The two blocks consists of:
    - Convolutional layer with 32 kernels, each of which are 5X5. We added a zero-padding of 2 to preserve dimensions, and stride 1. We used 5x5 patches in order to increase the expressiveness of the model, and capture broader patterns.
    - After the first layer we do a max pooling of 2x2, in order to reduce the number of parameters down-stream.
    - After the max pooling, we added another convolutional layer. This layer contains 64 kernels, each of which is 3X3. We again added a zero-padding of 1 to preserve dimensions, and used stride 1. We use smaller patches, since the input image is twice as small after max pooling. We double the number of filters in order to give this layer more complexity in comparison to the first convolution layer.
    - We again used 2x2 max pooling layer.
    - Each of the convolutional layers is followed by a ReLU activation function on the feature maps.
- The block of fully connected layers is getting the last convolutional layer's output as input, and outputs 10 values that correspond to each of the digits. The fully connected block is built as follows:
    - First we flatten the output of the last convolutional block, to a vector of 3136.
    - We connect that vector to a fully-connected layer with 14 nodes.
    - We connect the 14 nodes to another fully-connected layer with 10 nodes.
    - The output from these 10 nodes in the output of the network
- We incorporated dropout layers as a mean of regularization. Dropout forces the network to be more robust, and can be viewed as training an ensemble of neural networks. We drop each node with probability of 0.4. The dropout layers were placed after the last convolutional block (after flattening the feature maps), and after that first fully connected layer.
- The total number of parameters in our model is **63396**.
- Our model was able to achieve an error rate of **0.0049**, which translates to **99.51%** accuracy.
- The trained model can be found at: https://drive.google.com/open?id=0BygMyEL1SmpZM3FzbW5iMWNFemc

Optimization

- We used the CrossEntropyCriterion, which is recommended for multiclass classification problems.
- We used vanilla Stochastic Gradient Descent, with learning rate of 0.01, and trained for 600 epochs.
- We used a small learning rate since out model very quickly reached very good results and started overfitting.

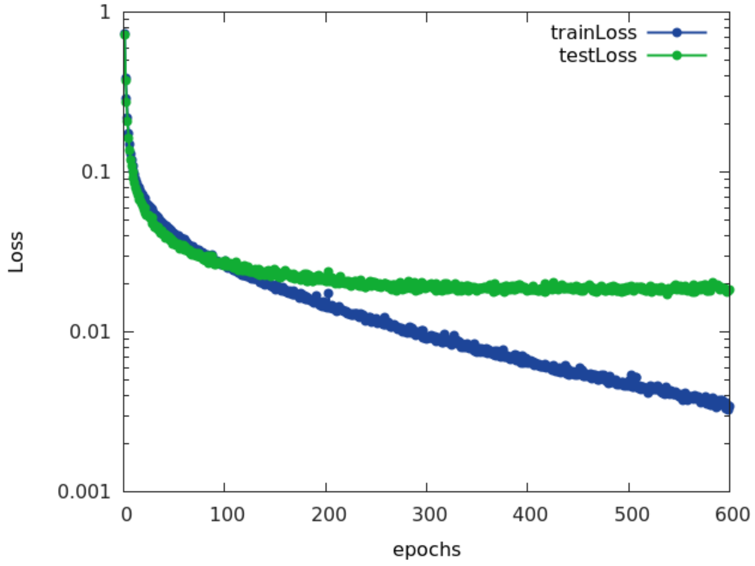
Git Repository: https://github.com/izikgo/096260-DeepLearning.git

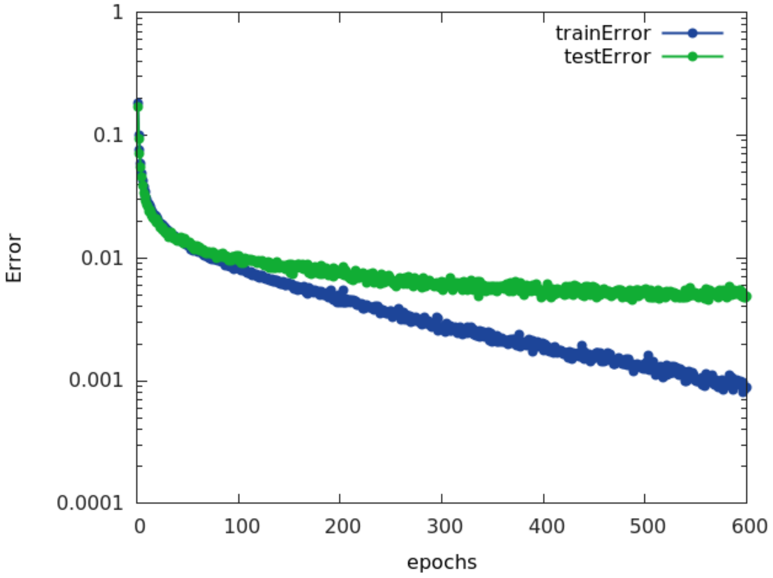
Amir David Nissan Cohen	200437192
Uzi Cohen	043228311
Izik Golan	201299781

Things we considered

1. It was clear that we will use convolution layers to capture spatial patterns and reduce the complexity of the model.
2. We normalized each image by diving it by 255 to get all pixels in the [0,1] range. Using standardization didn't help much.
3. We considered using an ensemble of networks, but the resulted networks weren't expressive enough (didn't have enough parameters) or exceeded the total number of allowed parameters.
4. We decided to add dropout as a regularization measure because the number of samples is relatively low. We did this instead of data augmentation.
5. We tried adding different activations after the first fully-connected layer, but the results with all of them were slightly worse. This is probably due to the fact that the layers themselves are small, so squashing the activation values has a negative impact which would not have happened if we were allowed to have more parameters in our network. The dropout between the fully-connected layers, prevents the two fully-connected layers from being equivalent to just 1 layer.

Plots





Git Repository: https://github.com/izikgo/096260-DeepLearning.git