Deep Learning - HW1 Report

Nadav Oved - 200689768
Aviv Sugarman - 305652729
Github Link: https://github.com/nadavo/DeepLearningCourse/tree/hw1

Model Architecture

9 layer network built according to the following structure:

```
[input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> output]
(1): View(784)
(2): Linear(64)
(3): ReLU
(4): Linear(64)
(5): ReLU
(6): Linear(128)
(7): ReLU
(8): Linear(10)
(9): LogSoftMax
Number of parameters:  64010
```

We chose to build our network with 4 linear layers (layer size in bits), with 3 ReLU transfer layers in between each middle linear layer and LogSoftMax transfer layer before output.
We used the following parameters:

- Batch Size = 64
- Learning rate = 0.1
- Loss Criterion = ClassNLLCriterion (Negative Log Likelihood)
- Optimization Algorithm = SGD (Stochastic Gradient Descent)
- Total # of Epochs = 10

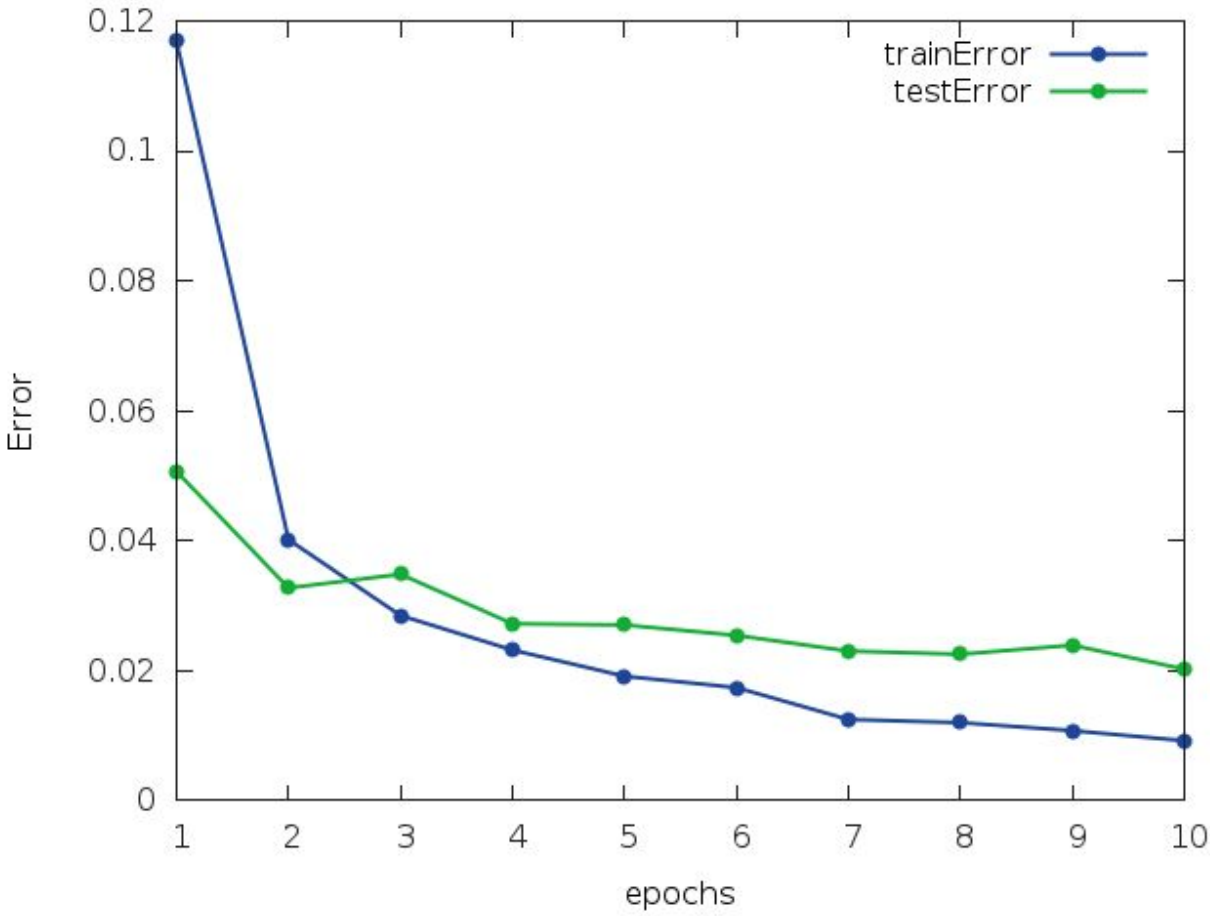
Training Procedure

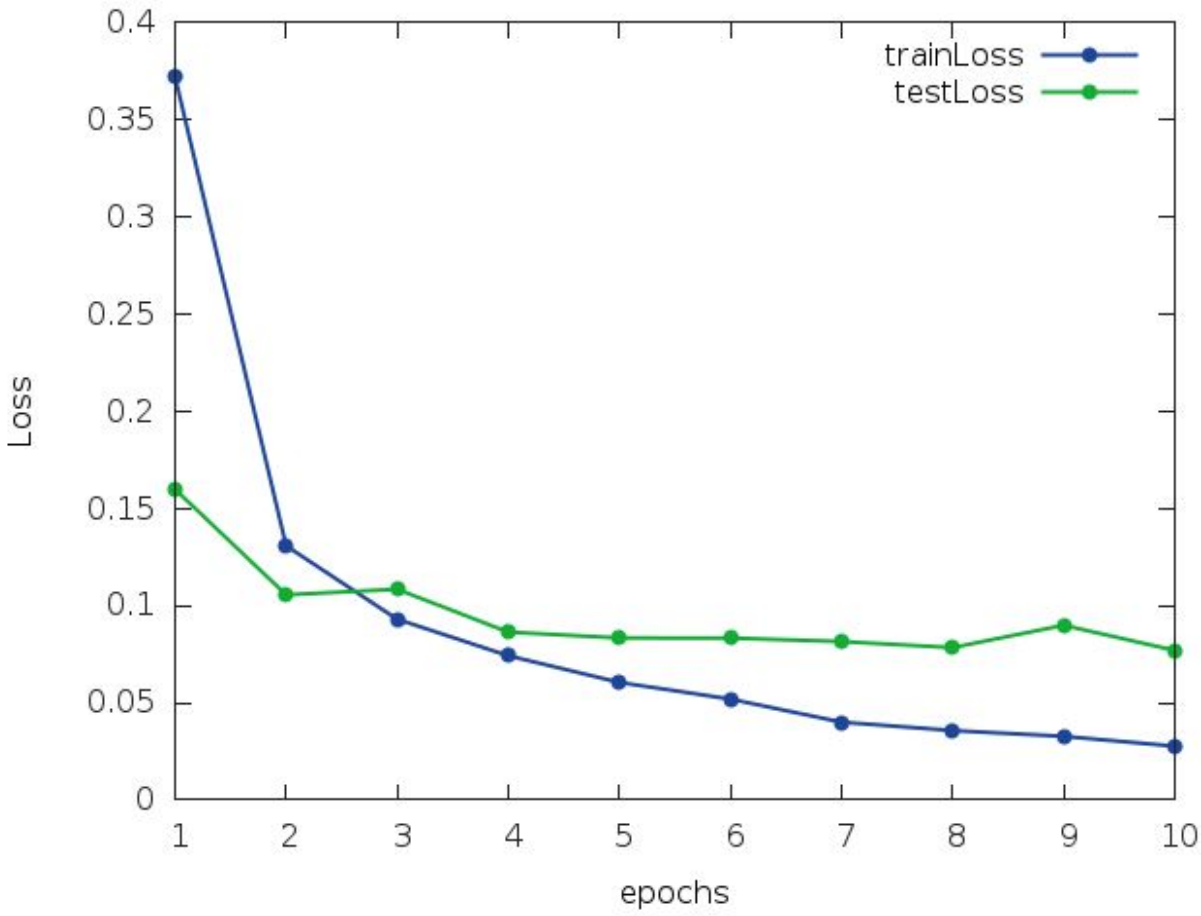
We started off by normalizing the data (training and test sets) according to the mean and standard deviation calculated on the training set.
We then constructed our fully connected neural network according to the architecture described above, and defined our loss criterion to be a Negative Log Likelihood criterion.
We defined our batch size to be 64, learning rate to be 0.1 and proceeded to run the following flow for 10 epochs:

1. Shuffle the data and labels to produce a random input sequence to the network
2. Input the entire training set in batches of 64, were for each batch we did:
    a. Forward propagation of data samples into network
    b. Loss and Error calculations according to our defined loss criterion
    c. Backward propagation to calculate gradients for network weights using Stochastic Gradient Descent as our optimization algorithm.
3. Calculate confusion matrix, average loss and average error for network predictions on training and test sets respectively.

After 10 epochs are done we finished training the network, saved it to a file (MyModel.dat) and produced the Loss and Error vs epochs graphs for test and training sets.

Error and Loss Graphs





Summary and Conclusion

Our general approach was to first define a network architecture and structure built from pairs of Linear and ReLU layers, which would comply with the # of parameter constraint (<65,000) which produces the best test error after 20 epochs with a batch size of 128, the Negative Log Likelihood loss criterion and SGD for optimization.

We started by creating the structure defined above, adding more layers, removing layers, tweaking the layer sizes, their order in the network and chose the structure which gave us the best results.

Then we decreased the batch size to 32 and epochs to 10, to test if we get better results, and checked whether increasing the number of epochs improves our results and saw it didn't. We then increased the batch size to 64 and saw a surprising improvement in results.

We then tested different combinations of another loss criterion (CrossEntropy) and optimization algorithms - SGD and ADAGRAD, ADAGRAD with ClassNLLCriterion, compared all resluts and saw that our original SGD with ClassNLLCriterion combination produced the best results under the configuration described above.

We concluded, that for this data set, we get very good results with a network which isn't very deep, fairly simple in structure, has a pretty large batch size and that training it for a large number of epochs actually yielded worse results than earlier epochs.

Best Results:

```
Epoch 10:
Training error: 0.0087713447171825      Training Loss: 0.026806424329159
Test error: 0.01933092948718      Test Loss: 0.073241243926952
ConfusionMatrix:
[[     971       1       1       0       0       1       3       0       1       1]
99.183%      [class: 0]
 [       0    1128       2       0       0       1       2       0       0       0]
99.559%      [class: 1]
 [       2       2    1014       1       1       0       2       5       3       0]
98.447%      [class: 2]
 [       0       0       4     980       0       6       0       5       7       6]
97.222%      [class: 3]
 [       2       0       2       0     962       0       5       3       1       5]
98.163%      [class: 4]
 [       2       0       0       5       1     871       5       0       4       2]
97.865%      [class: 5]
 [       3       3       1       1       4       1     943       0       0       0]
98.640%      [class: 6]
 [       1       9       7       1       1       0       1    1002       2       3]
97.566%      [class: 7]
 [       2       2       1       2       3       3       2       2     955       1]
98.150%      [class: 8]
 [       3       2       0       5      16       6       0       7       4     965]]
95.734%      [class: 9]
 + average row correct: 98.052885532379%
 + average rowUcol correct (VOC measure): 96.187941431999%
 + global correct: 98.066907051282%
```