Assignment 1 Report

Deep Learning – MNIST Classifier

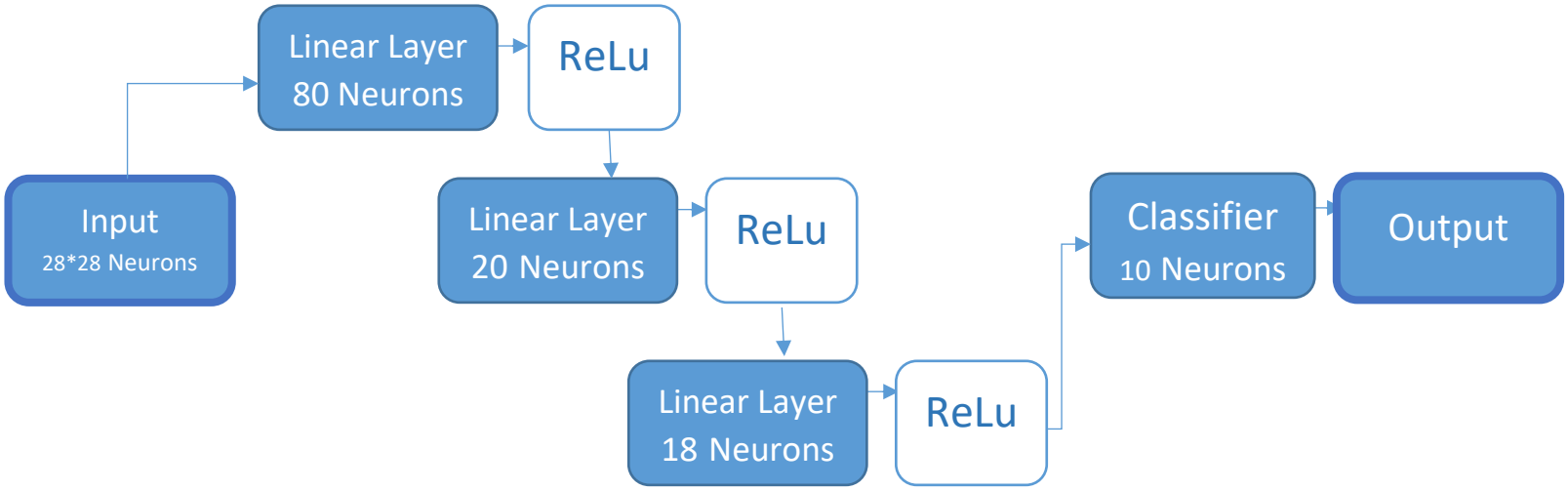
Roee Shraga & Avihay Levi

---

https://github.com/shraga89/DeepLearningCourse.git

Model Architecture:

The NN we designed is made of 3 fully connected linear layers with 1 ReLu layer in between each linear one. The final layer fully connects to a classification layer.




```
(1): nn.View(784)
(2): nn.Linear(784 -> 80)
(3): nn.ReLU
(4): nn.Linear(80 -> 20)
(5): nn.ReLU
(6): nn.Linear(20 -> 18)
(7): nn.ReLU
(8): nn.Linear(18 -> 10)
(9): nn.LogSoftMax
```

Figure 1 - Model Architecture

Training Procedure:

We started with normalizing our data with respect to the mean value and standard deviation of the training data. We shuffled the data to prevent bias and unbalanced results. Moreover, we read some papers that argue that shuffling methods helps to perform better.

The loss criterion we use is Cross Entropy, which fits better in classification tasks. We used **Adam** (Adaptive Moment estimation) as our gradient descent optimization algorithm. According to Sebastian Ruder's article that gives an overview on gradient descent algorithms, Adam tends to be the best overall pick. Nevertheless, we tried SGD, SGD with momentum, AdaDelta and RMSprop, but Adam gave us the best results.

We tuned the parameters of Adam and eventually chose: $\alpha\ (learning\ rate) = 0.003$, $\beta_1\ (first\ moment\ coefficient) = 0.9$, $\beta_2\ (seconed\ moment\ coefficient) = 0.999, \epsilon = 0.001$. We tuned the parameters based on the article that presented Adam (Kingma & Lei Ba, 2015) and based on the results we got.

We chose a batch size of 100 and ran 100 epochs.

Graphs:

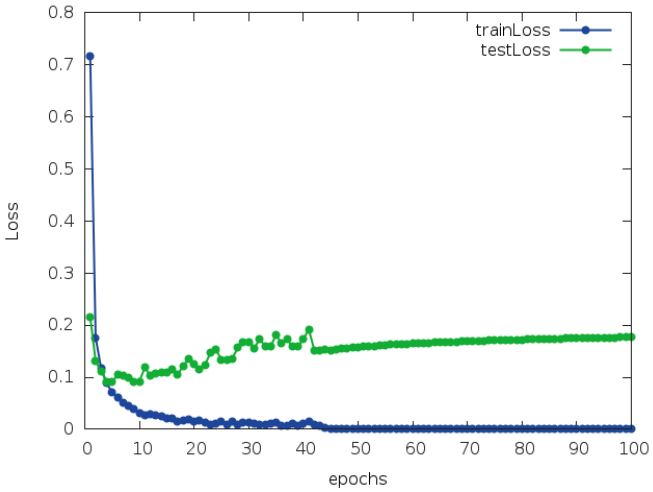


Figure 2 - Loss Convergence Graph

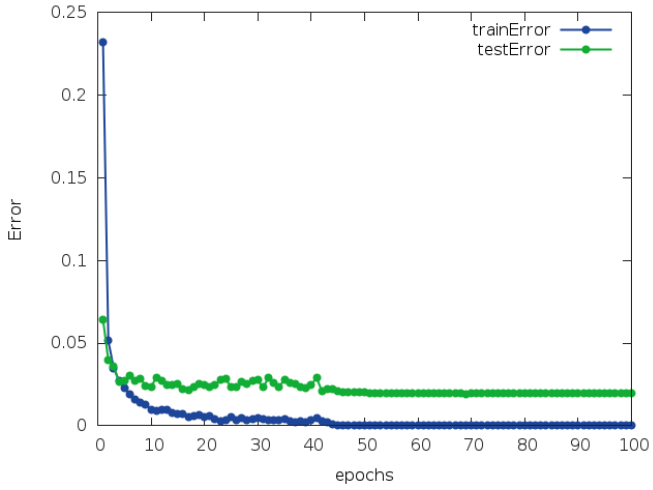


Figure 3 - Error Convergence Graph

Summary & Conclusion:

We can say that the parameters limit (of 65K parameters) together with the linearity demands (i.e. no convolution layers) made this task challenging. According to past researches, the best error rate that was achieved on this dataset is 0.21% by a CNN (DropConnect, Nov. 2016). The best results achieved by a linear NN, 0.35%, used a lot more parameters, so we had to be creative.

Our first main conclusion is that in a linear NN, the model architecture (in terms of sizes and number of layers) is crucial. The most critical part in this context is the size of the first layer – we saw that the bigger it gets, the results tend to be in a higher order of accuracy. To obtain a deep network, we ordered the layers in a decreasing order of their sizes, while maintaining the number of total parameters close to the given limit.

To further improve the performance, we chose Cross Entropy as out loss criterion (as mentioned above), Adam as our gradient descent optimization algorithm and tuned their parameters.

Given the tradeoff between batch size and number of epochs, we chose a similar number of both.