Machine Learning - Deep Learning Project 1

Anton Borries
2204600
anton.borries@hhu.de

Johannes Thiel
2164216
johannes.thiel@hhu.de

December 6, 2016

Abstract

We train a neural Network to recognize Images from the MNIST Dataset. The Accuracy only reaches 98.9% *due to Hardware constraints during training.*

I. Architecture

Our Network consists out of 4 total Layers. The first two are both Convolutional Layers and the second ones are both Fully Connected.
We used ReLu as our Activation Function for all Layers.

Table 1: *Convolutional Layers*

Layer	1	2
Input Size	28 x 28	14 x 14
Filter Size	5 x 5	5 x 5
Output Channels	32	64
Pooling	2 x 2	2 x 2
Activation Function	ReLu	ReLu

Table 2: *Fully Connected Layers*

Layer	3	4
Input Size	7 x 7 x 64	1024
Activation Function	ReLu	ReLu
Dropout	Yes	No

II. Training

We used a Batch Size of 128 to train our Network.
For the loss function we decided on using the cross entropy method.

$$C = -\frac{1}{n}\sum_{z}[y \ln a + (1-y)\ln(1-a)] \quad (1)$$

This is helping us, avoiding the learning slowdown [1].
We utilized the Adam Optimizer with a learning rate of 0.001 to train the Network.
We decided on training for 1000 Epochs because of Hardware restrains.

III. Evaluation

We reached a performance of 98.9%.
This score was calculated by validating against the MNIST Test data.
We are aware that this is a relativly low accuracy given our architecure.
If we would have trained the Network for a longer time or with more Epochs in general we are certain that we would score better.
Nevertheless this is still a higher score than the Network in Exercise 2 which scored about 96%.

IV. Randomness

The results differ from run to run although only slightly. This is due to the fact that all weights are initialized randomly (Standard deviation of 0.1).

V. Optimization iterations

All training was performed on on a Intel i5-4299U @ 1,60GHz x4 CPU.

Table 3: *Optimization Iterations*

No. of Epochs	Accuracy	Duration
100	0.9578	5 min 26 sec
300	0.9747	15 min 09 sec
1000	0.98.9	50 min 59 sec

We can see that the time it takes to tain the network is linear to the number of epochs.

VI. Change of learning rate

A higher learning rate helps us learn faster in the beginnig. In later stages though, this leads to the results changing in variance rather drastically.
Smaller learnings rates on the other hand are leading to a smaller increase in results for each epoch. However with more epochs they will get better as well. This is backed by our Network getting better the more epochs we use for training.
Also this proves that our Network would probably perform better if we trained for a longer time.

VII. Visualization

Roughly put, the Network is learning to detect Edges and their orientations.
Some Examples show the following.

- Filter 2 is detecting right facing Edges
- Filter 5 is detecting left facing Edges
- Filter 6 is detecting outer bounds
- Filter 25 is detecting bottom facing Edges

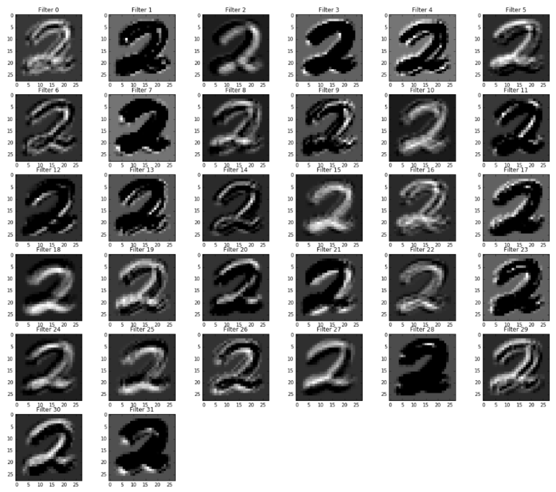

Figure 1: *Example of Activations in Layer 1*

VIII. Configuration changes

Training with 100 epochs. We tried changing the batch size.

Table 4: *Changing the Batch size*

Batch Size	Accuracy
256	0.933
512	0.944
1024	0.975
2048	0.976

As you can see. The Batch Size had no effect after reaching 1024 Changing the amount of Filters lead to no difference in the results.

IX. Minimum Configuration

Of course a definition for a minimum configuration is always dependand on how you would define "good results".
We created a Network with only two Layers. The first is a Convolutional Layer with 32 5 x 5 Filters using 4 x 4 Pooling. The second Layer is a Fully Connected Layer with 127 Neurons. Training this Network with a Batch Size of 100 lead to a accuracy of 95% which is in the same order as our complete Network.

X. No Pooling

Taking away the pooling in the Convolutional Layers led to two effects.

- Training was way slower. On our machine it was almost slower by a factor of 10
- Accuracy went down to 92%

References

[1] Cross Entropy, `http://neuralnetworksanddeeplearning.com/chap3.html`, 06 12 2016.