# Assignment 1 - CS 726

Utkarsh Kumar, 130050022

15.10.2016

## Summary

As an extension to the well known MNIST problem, this assignment involved recognizing handwritten Devnagri characters. Sticking to the basics, basic one-layer neural network with *ReLU* activation was the baseline to start at. The network worked reasonably well when run on two labels (reaching an accuracy of 90%), but performance fell unreasonably when run on the entire data with 104 labels (10%). However, a bit of pre-processing significantly boosted this performance up to 30%.

Switching to CNNs gave a substantial boost to the accuracy, with a network of four *convulation* and *max-pooling* layers with *ReLU* activations followed by three fully connected layers with *ReLU* taking the accuracy to 83%. *L2 regularization* with coefficient 0.01 was employed to avoid overfitting. For finding the optimal network parameters, *Adam Optimizer* with $10^{-4}$ was used instead of the regular gradient-descent for many reasons including faster convergence. The model reached 80% accuracy within 6000 iterations.

The entire framework, therefore, looks like

- CNN with *ReLU* with patch size $5 \times 5$ and stride of 1 in each direction followed by a max pooling of size 2. This layer takes in $64 \times 64 \times 1$ image and generates a $32 \times 32 \times$ output.
- CNN with *ReLU* patch size $5 \times 5$ and stride of 1 in each direction followed by a max pooling of size 2. This layer takes in $32 \times 32 \times 32$ sized input and generates a $16 \times 16 \times 64$ output.
- CNN with *ReLU* patch size $5 \times 5$ and stride of 1 in each direction followed by a max pooling of size 2. This layer takes in $16 \times 16 \times 64$ sized input and generates a $8 \times 8 \times 128$ output.
- CNN with *ReLU* patch size $5 \times 5$ and stride of 1 in each direction followed by a max pooling of size 2. This layer takes in $8 \times 8 \times 128$ sized input and generates a $4 \times 4 \times 256$ output.
- Fully connected layer with *ReLU* that takes in a flattened 4096 size vector output from the second layer as input and gives out a 256 vector output.
- Fully connected layer that takes in a 256 size vector output as input and gives out a 512 vector output.
- Fully connected layer that takes in a 512 size vector output as input and gives out a 104 sized vector corresponding to the number of labels.

*One iteration refers to training the network on one batch of 100 images

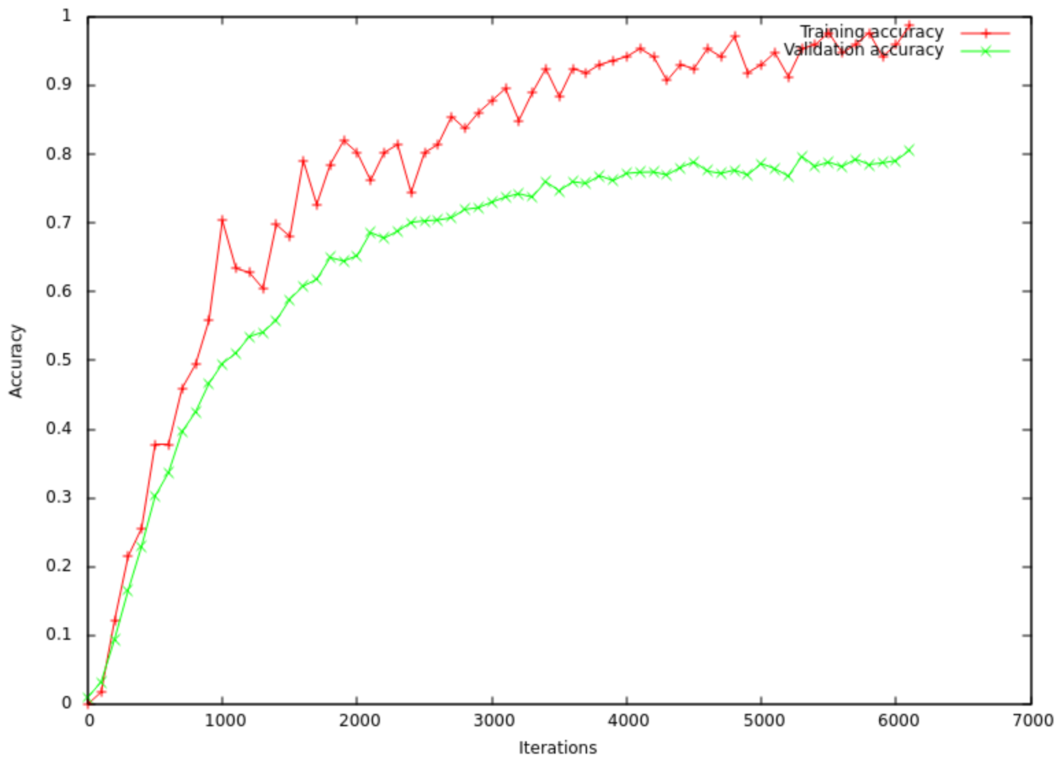

Figura 1: Training and validation accuracy vs iterations

## Analysis

We achieved the highest accuracy using the CNN described in the last section. It is this network that we would be playing around with to see the effects of varying different parameters keeping the rest constant.

### 1. Number of levels

Using one layer each of convolution leads to 51% accuracy on the validation data, significantly poorer than that using two layers.
Using two layers of each of convolution leads to 60% accuracy on the validation data, Using three layers of convolution leads to the best performance of 80%.
Increasing this to higher levels should lead to higher accuracy initially but will hit saturation later. This saturation point can be worked out with time and effort.

### 2. Width of levels

Halving the width of each layer leads to the network getting stuck at 1% accuracy.
Doubling the width of each layer had a similar effect as halving it. The network again stuck at 1% accuracy. The chosen set of parameters thus seem to be the optimal set.

## 3. Hidden unit type

Switching the hidden unit type to *sigmoid* and *tanh* led to poor performances with 1% accuracy in both cases.
Using *ReLU6* as the hidden unit type led to at par performance with *ReLU*.

## 4. Learning rate

Learning rate was 0.0001 in all the above models. Increasing it to 0.001 led to the network oscillating around the 1% accuracy mark with no signs of convergence even after 5000 iterations. Decreasing it to 0.00001 also made the network oscillate about the 1% accuracy mark even after 5000 iterations.

## 5. Regularizer

With *dropout regularization*, the validation accuracy takes over 50000 iterations to reach 80% accuracy, making it slow to converge as compared to the regular *L2 regularization*.