Project 7 Report : Convolutional Neural Networks

Sajal Kumar

Implementation details on parameters

My implementation allows change to the following parameters.

- `batch_size` : Size of one batch when training the CNN using mini-batches (Default : 100).
- `random_seed` : Random state/ seed for CNN, batch generation, etc (Default : 1).
- `learning_rate` : Learning rate for CNN optimizer (Default : $1e-4$).
- `epochs` : Number of iteration the CNN would go through before convergence (Default : 10).
- `c1_kernel_size` : Kernel size for 1st conv layer (Default : 3).
- `c2_kernel_size` : Kernel size for 2nd conv layer (Default : 3).
- `c1_op_channel` : Number of open channels for 1st conv layer (Default : 4).
- `c2_op_channel` : Number of open channels for 2nd conv layer (Default : 2).
- `p1_pool_size` : Pool size for 1st pool layer (Default : 2).
- `p2_pool_size` : Pool size for 2nd pool layer (Default : 4).

This along with other criteria mentioned in the project description (e.g. padding = 'valid', strides = (1, 1), etc.) represent the standard setting. All other parameters, not mentioned in the project description were kept at their default values.

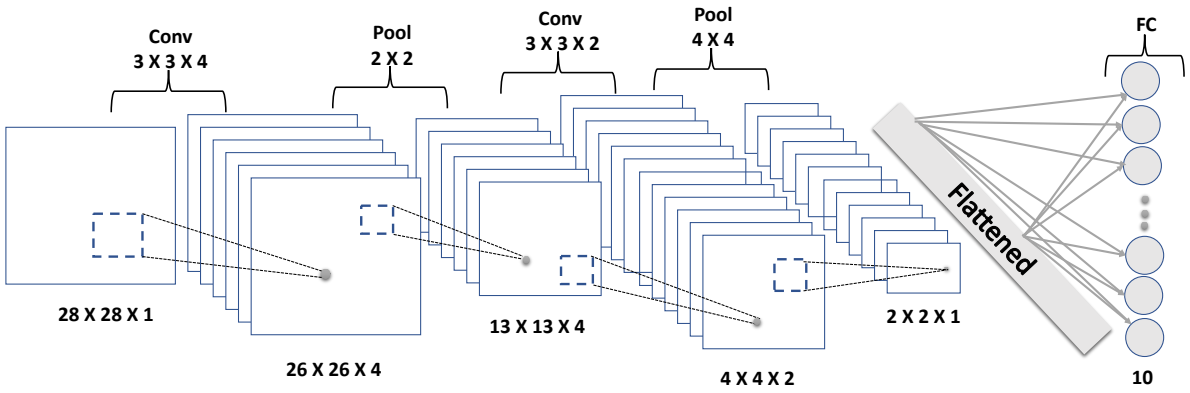

Figure 1: CNN architecture for given configuration.

CNN architecture

Figure 1 shows the architecture for the configuration provided as a part of the project assignment. The architecture has two convolution tensor layers, two pooling tensor layers and one fully connected tensor layer with 10 output units, that translate to the probability profile of the response. The following are the dimension of each tensor layer.

- Input layer tensor with dimensions = [`batch_size` 28 28 1]
- The 1st convolution layer with dimensions = [`batch_size` 26 26 4] with kernel size = [3 3] and filters (or number of output channel) = 4.
- The 1st pooling layer with dimensions = [`batch_size` 13 13 4] with pooling size = [2 2].
- The 2nd convolution layer with dimensions = [`batch_size` 4 4 2] with kernel size = [3 3] and filters (or number of output channel) = 2.
- The 2nd pooling layer with dimensions = [`batch_size` 2 2 1] with pooling size = [4 4].
- The fully connected softmax layer with dimensions = [`batch_size` 10], which is also used as the output layer.

Performance evaluation on MNIST

Here I report predictive accuracy of CNN on MNIST training and testing data. The original data was standardized using the following formula:

$$X_{standard} = \frac{X - \mu}{\sigma_X} \tag{1}$$

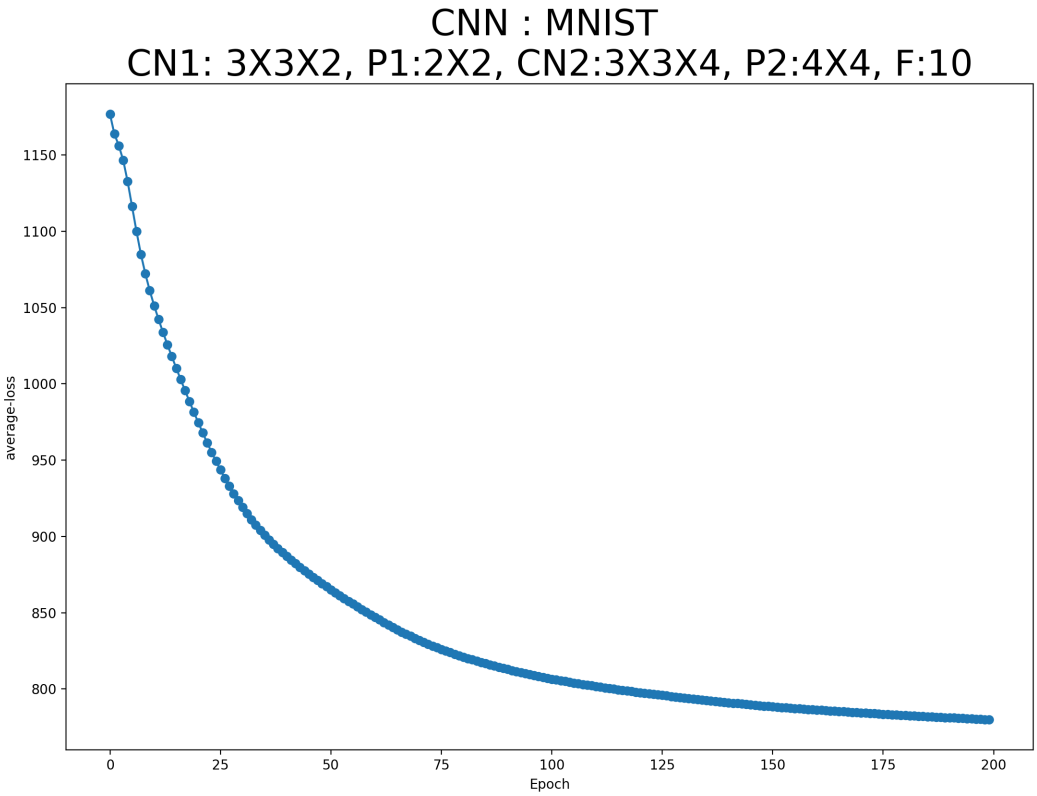

Figure 2: Decrease in average loss with epoch on standard setting

Where X is the original feature space of MNIST data set, μ is a vector of attribute wise mean and σ_X is the standard deviation of the entire X feature space.

I also split the data into 3 sets, training, testing and validation. The data was first split (stratified) into 85% training and 15% testing. The 85% training data was then further split into 85% training data and 15% validation data.The validation data was used by the CNN to improve upon its performance.

The training data was supplied to the CNN in mini-batches and the loss was computed on the validation data-set. The accuracy obtained on training and testing data was 21.6% and 21.63% respectively, which is very poor. To analyze the effect of epochs, I let the CNN run for 200 epochs and while the performance for the 200th epoch was significantly better at 44% accuracy for both datasets. Figure 2 shows that increasing the epoch number further would not have much effect on the accuracy.

A more comprehensive evaluation

I found that a CNN is quite sensitive to change in its parameters. Here I present 4 configurations touching upon 4 different parameter combinations.

config	**pred on train**	**pred on test**
1	21.60%	21.63%
2	97.43%	96.45%
3	97.45%	96.48%
4	97.83%	97.19%

Table 1: Performance change for different configuration of parameters.

Table 1 shows the affect of different parameters on the the CNN, where *config* is given as :

1. Standard setting
2. `c1_kernel_size`=5, `c1_op_channel`=32, `c2_kernel_size`=5, `c2_op_channel`=64 and `p2_pool_size`=2
3. `c1_kernel_size`=5, `c1_op_channel`=32, `c2_kernel_size`=5, `c2_op_channel`=64, `p2_pool_size`=2 and `learning_rate`=0.01
4. `batch_size` = 64, `c1_kernel_size`=5, `c1_op_channel`=32, `c2_kernel_size`=5, `c2_op_channel`=64 and `p2_pool_size`=2

As can be seen, there is no contest between config 1 and the others and the reason is that the CNN specific configuration (used in 2) for both convolution / pooling layer works really well for the MNIST dataset. This configuration closely resembles the config used in the textbook with the exception of strides, that are still set at the value that was provided in the project description.

The next interesting effect is of learning rate and batch size, while there is no major change in the final accuracy, figure 3 shows there loss decrease over 10 epochs.

Even though config 2, 3 and 4 show similar performance, it can be seen that they start at different loss values. With config-3 having the best starting point. It is however interesting to see that while config-4 starts at the worst point, it is slightly better than config-2 and config-3 on accuracy. Thus, learning rate, CNN specific parameters and batch size can all effect the performance of a CNN.

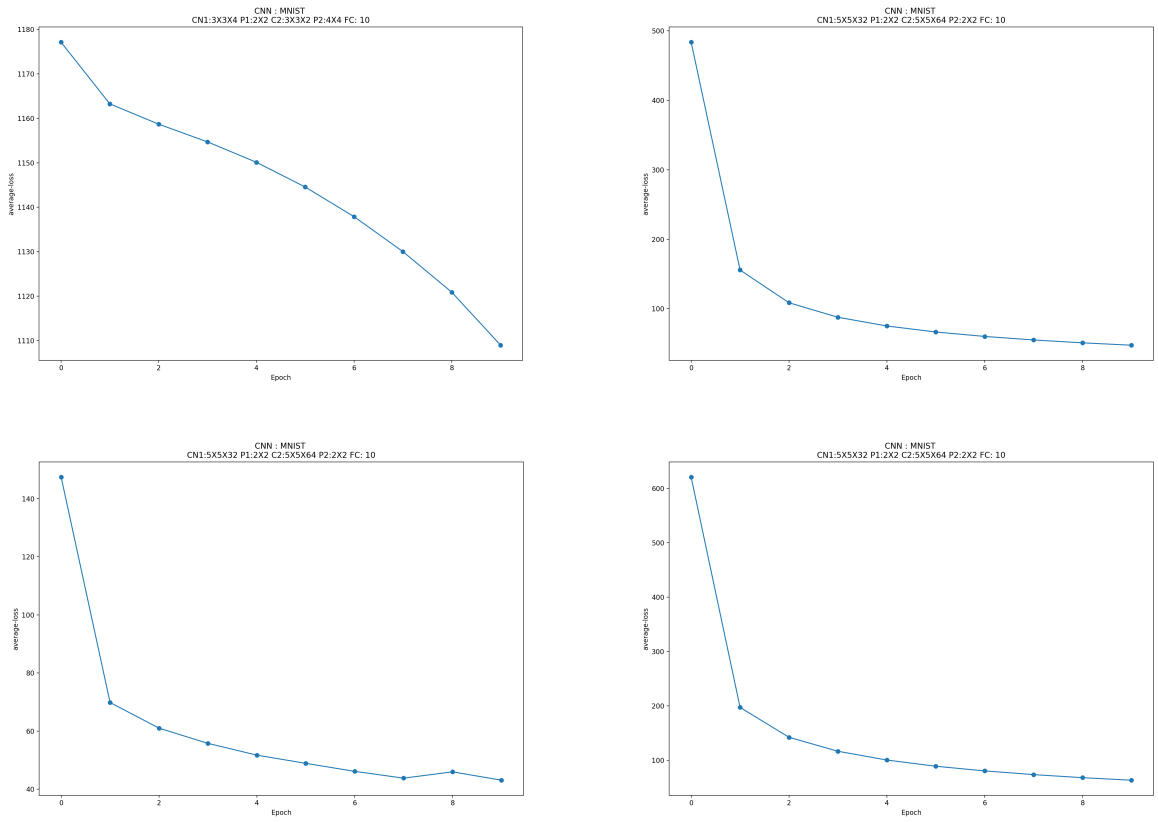

Figure 3: Decrease in average loss with epoch on 4 different parameter configuration. Top Left : Config 1, Top Right: Config 2, Bottom Left : Config 3, Bottom Right : Config 4