CS 7180 Special Topics in AI: Deep Learning
Northeastern University, Spring 2019

Assignment 1

Name: Sabbir Ahmad, NUID: 001860911
Email: ahmad.sab@husky.neu.edu

1 Problem 1

1.1 Final Model

The Convolutional Neural Network (CNN) contains Convolutional layer, RelU as activations, MaxPooling, Flatten layer, Dense layer, Dropout layer and Softmax as final activation. The hyper-parameters and the whole network are given below. The code of the model is followed from the Udacity's Deep Learning Course, and so there are some similarities in the whole process.

1.1.1 Hyper-parameters

- Stride = 1
- Padding = 1 (so that image size remains the same)
- Filter size: 3×3
- Number of channels for first convolutional layer: 8
- Number of channels for second convolutional layer: 16
- MaxPooling size: 2×2
- Dropout probability: 0.2
- Hidden layer size for dense layer: 256
- SGD optimizer with learning rate 0.01

1.1.2 Final CNN

The CNN contains the following layers sequentially.

1. **Conv2D**: Takes $28 \times 28 \times 1$ image and generates $28 \times 28 \times 8$ tensor.
2. **ReLU**
3. **BatchNormalization**
4. **MaxPooling2D**: Takes $28 \times 28 \times 8$ tensor and generates $14 \times 14 \times 8$ tensor.
5. **Conv2D**: Takes $14 \times 14 \times 8$ image and generates $14 \times 14 \times 16$ tensor.

6. **ReLU**
7. **BatchNormalization**
8. **MaxPooling2D**: Takes $14 \times 14 \times 16$ tensor and generates $7 \times 7 \times 16$ tensor.
9. **Flatten**: Flattens the input tensor to $batch_size \times 784$ tensor for the Dense layer. Here, $16 \times 7 \times 7 = 784$.
10. **Dropout**: Dropout layer with dropout probability 0.2
11. **Dense**: Takes 784 size vector and outputs to 256 hidden layer.
12. **ReLU**
13. **Dropout**: Dropout layer with dropout probability 0.2
14. **Dense**: Takes 256 size vector and outputs to 10 hidden layer for 10 classes.
15. **Softmax**: Softmax activation used to get the final output for 10 classes.

1.2 Final Model Performance

1.2.1 Number of Parameters

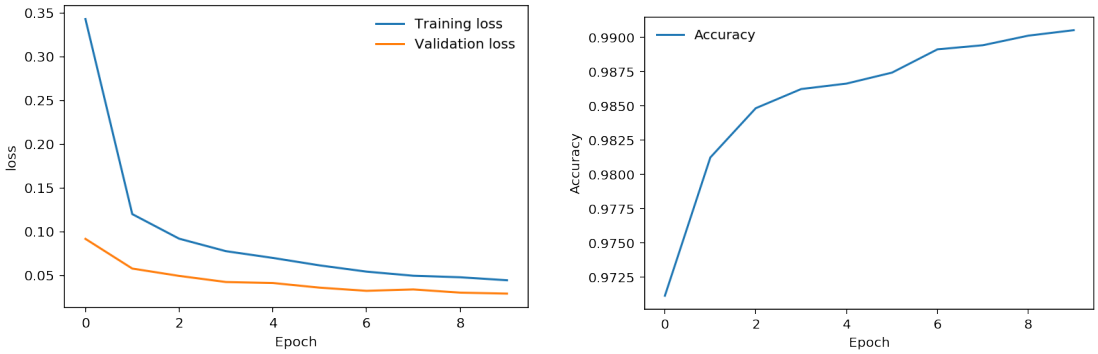
The total number of parameters in the final model is 204826.

1.2.2 Loss

The CNN model was run for 10 epochs and the training loss and validation loss was calculated for each epoch. The training loss and the validation loss for each epoch can be seen in Table 1.
As can be seen from the values, loss decreases for both training and validation with respect to epoch. This is shown illustratively in Figure 1(a).

1.2.3 Accuracy

Accuracy was also calculated for each epoch and is shown in Table 2, and Figure 1(b) shows the plot of accuracy versus epoch.



(a) Training and Validation loss versus epoch (b) Accuracy with respect to epoch

Figure 1: Model loss and accuracy versus epoch.

Epoch	Training Loss	Validation Loss
1	0.34299371	0.09168640
2	0.11998538	0.05773036
3	0.09190263	0.04934031
4	0.07759595	0.04234007
5	0.06990580	0.04121702
6	0.06135981	0.03590280
7	0.05422640	0.03227065
8	0.04953009	0.03386303
9	0.04777700	0.03020406
10	0.04437030	0.02910404

Table 1: Training and Validation loss for each epoch

Epoch	Accuracy
1	0.97114617
2	0.98123002
3	0.98482430
4	0.98622203
5	0.98662138
6	0.98742014
7	0.98911738
8	0.98941696
9	0.99011582
10	0.99051517

Table 2: Accuracy for each epoch

1.3 Different Dropout Probabilities

The final model accuracies for one epoch for different dropout probability values in range $[0, 1]$ are shown in Table 3. For one epoch, it can be seen that the accuracy is better in the range $[0.11, 0.22]$. The accuracy decreased as the dropout probability became too high.

Dropout Pr	Accuracy
0.0	0.972045
0.11111111	0.976837
0.22222222	0.974042
0.33333333	0.969848
0.44444444	0.967752
0.55555556	0.967153
0.66666667	0.960663
0.77777778	0.954972
0.88888889	0.903754
1.0	0.120607

Table 3: Caption

1.4 Discussion

- Using less Convolutional layer gave less accuracy. By using one Convolutional layer instead of two, it was seen that the highest accuracy achieved was 0.984 keeping all the other things same. Note that, with the removed Convolutional layer, the associated ReLU, BatchNormalization and MaxPooling layers were also removed.
- One interesting thing here: though generally BatchNormalization is used before Activation layer, using the opposite gave better result (accuracy more than 0.99).
- Without the dropout layers, the model also achieved required accuracy (accuracy was 0.988), but with dropout it performed better (more than 0.99). I think, as MNIST is a fairly simple dataset, the effect of dropout was not so significant.
- Without BatchNormalization, the model achieved less accuracy (0.988) than using the BatchNormalization layer (more than 0.99).

- Without both BatchNormalization and dropout layer the model achieved 0.987 accuracy. With both of them, the accuracy was more than 0.99 which demonstrates their effect. Both BatchNormalization and dropout regularizations were almost equally effective for this network and dataset.

- I think, for designing convolutional network, beginning with less number of convolutional layers and then increasing the number of layers by observing the accuracy for some number of iterations is the an effective strategy. At the same time, using BatchNormalization and Pooling layers with each convolutional layer is generally effective.

- For validating the hyper-parameters by changing only one parameters at a time has some shortcomings. For example, by using one convolutional layer, we fix a learning rate which works best for a range of learning rates. If a new convolutional layer is added to the model, the system can behave very differently. The executing might take a long time. As a result the learning rate might need to be validated again. So just changing one hyper-parameter at a time might not give the best results. Although, changing multiple hyper-parameters at the same time is harder, as the number of choices becomes exponential, and so this technique is useful.