IE598: Deep Learning	**Fall 2016**
Problem Set 2	
Chen Zhang, NetId: czhang49	*Handed In: October 12, 2016*

1 Target of Assignment

The target of the assignment is to apply different techniques (e.g. deep networks, batch normalization, dropout, pooling layers, convolution layers, optimization methods, data augmentation) to CIFAR10, and compare their behavior, and choose a set of configuration that works the best.

2 Base Case Description

As a base case, I used a network of 11 layers, with the first 8 being convolution layers and the last 3 being regular linear layers. The parameters used in each layer is as below:

Layer 1: (convolution) # input channel = 3, # output channel = 64, padding = 1.
Layer 2: (convolution) # input channel = 64, # output channel = 64, padding = 1.
Layer 3: (convolution) # input channel = 64, # output channel = 128, padding = 1.
Layer 4: (convolution) # input channel = 128, # output channel = 128, padding = 1.
Layer 5: (convolution) # input channel = 128, # output channel = 256, padding = 1.
Layer 6: (convolution) # input channel = 256, # output channel = 256, padding = 1.
Layer 7: (convolution) # input channel = 256, # output channel = 256, padding = 1.
Layer 8: (convolution) # input channel = 256, # output channel = 256, padding = 1.
Layer 9: input vector size = 4096, output vector size = 500.
Layer 10: input vector size = 500, output vector size = 500.
Layer 11: input vector size = 500, output vector size = 10.

The batch size is 500, number of epochs is 30 for the purpose of looking at the training effect. No batch normalization or dropout is used in the base case. The optimization method is SGD with momentum. It also uses a $(2, 2)$ pulling. Using the base case configuration, the test set accuracy is 86.16% after 30 epochs.

3 Comparison of Different Parameters

All subsequent modifications to the configuration is based on the previous base case.

3.1 Interlayer Batch Normalization

In the base case, I didn't use Batch Normalization, and achieved 86.16% accuracy in the test set. When I added interlayer Batch Normalization, the test accuracy is 85.38%, with a 0.75% drop from without Batch Normalization. This may be due to the particular network architecture and the nature of data, supposedly Batch Normalization should increase accuracy.

3.2 DropOut

Without dropout, test accuracy is 86.16% When I added dropout with $p = 0.25$, the test accuracy is 84.53%, a 1.63% drop from without dropout. Further increasing the dropout probability p result in an even lower test accuracy. This may be due to the fact that dropout is mainly for reducing overfitting. Now that the model is not being trained long enough, the effect of overfitting is not big. Supposedly if I train the model for larger number of epochs, I should have a better accuracy with dropout.

3.3 Number of Output Channel in Convolution Layer

I did three test based on the base case: i. reduce the number of output channels in all layers by $\frac{1}{2}$. ii. reduce the number of output channels in all layers by $\frac{1}{4}$. iii. reduce the number of output channels in all layers by $\frac{1}{8}$. The test accuracy is as below:

Reduce number of output channels by $\frac{1}{2}$: test accuracy 84.78%.
Reduce number of output channels by $\frac{1}{4}$: test accuracy 80.85%.
Reduce number of output channels by $\frac{1}{8}$: test accuracy 75.14%.

We can see that decreasing the number of output channel dramatically decrease the test accuracy. But also we note that decreasing the number of output channels by $\frac{1}{2}$ only decrease the test accuracy by 1.3%, but has a better training speed, so it may be worth it to decrease the number of output channels of the base case by $\frac{1}{2}$.

3.4 The Depth of Network

There are three types of convolution layer in the current architecture: 64 output channels, 128 output channels, and 256 output channels. I compared the effect of increasing each of these output channels. The result is as follows (recall base case accuracy is 86.16%):

Increase the number of 64 output channels layer by 1: 86.46%
Increase the number of 64 output channels layer by 2: 87.41%
Increase the number of 64 output channels layer by 3: 86.99%
Increase the number of 128 output channels layer by 1: 86.5%
Increase the number of 128 output channels layer by 2: 86.4%
Increase the number of 128 output channels layer by 3: 86.07%
Increase the number of 256 output channels layer by 1: 86.14%

Increase the number of 256 output channels layer by 2: 86.56%
Increase the number of 256 output channels layer by 3: 85.98%

We can see from the results that for the 64 output channels, increasing the number of layers by 2 works best. For the 128 output channels, there's no need to increase the number of layers. For the 256 output channels, increasing the number of layers by 2 works best.
We can also see that increasing the depth of the neural network does not guarantee better behavior. We still need to experiment for different cases.

3.5 Optimization Methods

The base case uses SGD with momentum as the optimization method. I tried RMS Prop method and the test accuracy is 66.22%, a 20% decrease from the base case. This shows that the SGD with momentum performs much better than RMS Prop in this problem.

3.6 Pooling Layer

I change the pulling to a $(3, 2)$ pulling, meaning there is overlap in the pulling process. The test accuracy is 86.81%, a slight increase from 86.16%.

3.7 Data Augmentation

When I flip the image and augment the data, I got a 87.45% test accuracy. This way of augmenting the data is essentially very similar to increasing the number of epochs by a factor of 2. So this behavior of increased test accuracy is expected.

4 Best Performing Configuration and Result

The batch size is 250, number of epochs is 300 for the purpose of training in reasonable amount of time. Batch normalization is used , but dropout is not used. The optimization method is SGD with momentum. A $(3, 2)$ pulling is selected. The configuration of the layers is as below (note that the number of output channels in all convolution layers are decreased by a factor of $\frac{1}{2}$ based on previous results):

Layer 1: (convolution) # input channel = 3, # output channel = 32, padding = 1.
Layer 2: (convolution) # input channel = 32, # output channel = 32, padding = 1.
Layer 3: (convolution) # input channel = 32, # output channel = 32, padding = 1.
Layer 4: (convolution) # input channel = 32, # output channel = 32, padding = 1.
Layer 5: (convolution) # input channel = 32, # output channel = 64, padding = 1.
Layer 6: (convolution) # input channel = 64, # output channel = 64, padding = 1.
Layer 7: (convolution) # input channel = 64, # output channel = 128, padding = 1.
Layer 8: (convolution) # input channel = 128, # output channel = 128, padding = 1.
Layer 9: (convolution) # input channel = 128, # output channel = 128, padding = 1.
Layer 10: (convolution) # input channel = 128, # output channel = 128, padding = 1.

Layer 11: (convolution) # input channel = 128, # output channel = 128, padding = 1.
Layer 12: (convolution) # input channel = 128, # output channel = 128, padding = 1.
Layer 13: input vector size = 2048, output vector size = 500.
Layer 14: input vector size = 500, output vector size = 500.
Layer 15: input vector size = 500, output vector size = 10.

The final test accuracy from this configuration is 87.42%. I believe with more epochs it will behave better.
The code is as attached.