Deep Learning in Data Science (DD2424)
Report to Assignment 3

Cong Gao

April 25, 2020

1 Optimize the performance

I tried the four following improving methods:
1. Do a more thorough search to find a good network architecture.
I trained and tested a five-layer network whose number of nodes at the hidden layers are [200, 100, 50, 30]. Other parameters: $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 0.1$, $\lambda = 5e-3$, $n\_s = 5 * 45000/n\_batch$, $n\_cycles = 2$. I used Xavier initialization and also randomly shuffled the order of the training data after each epoch.

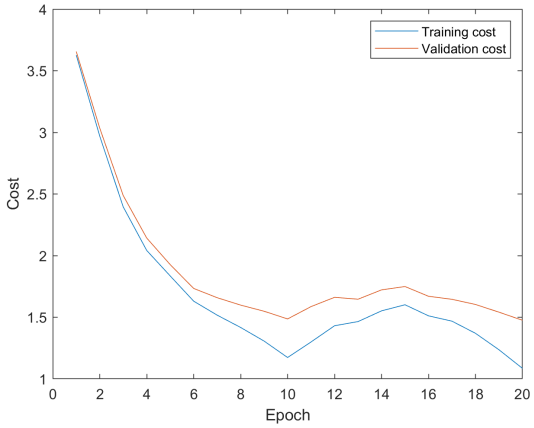

Figure 1: Cost plot

	Training data	Validation data	Test data
Accuracy	73.57%	58.50%	57.09%

Table 1: Classification accuracy with better network architecture

The classification accuracy on test set are improved by about 3.5% with a better network architecture. But making the network deeper doesn't necessarily

improve performance. For example, in the basic part of assignment 3, the classification accuracy on test set is just 50.76% with 9-layer network architecture.
2. On the basis of method 1, I did a more exhaustive random search to find good values for the amount of regularization. The range of the values I searched for λ is $[3e-3, 7e-3]$. And the best found λ is $5.2e-3$. Other parameters are the same as before.

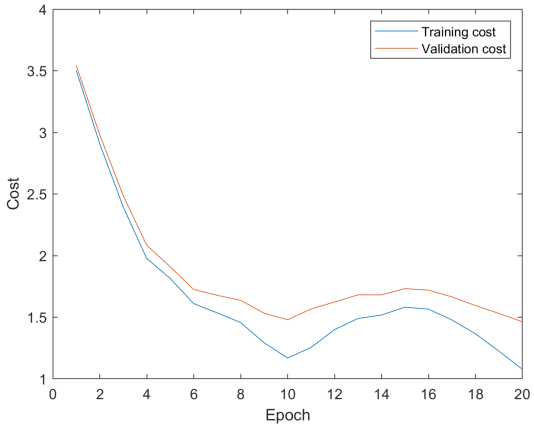

Figure 2: Cost plot

	Training data	Validation data	Test data
Accuracy	73.52%	58.48%	57.59%

Table 2: Classification accuracy with best found lambda

The performance is improved a litte (about 0.5%) but not too much.
3. Then I applied a random jitter to each image in the mini-batch before doing the forward and backward pass on the basis of method 1 and method 2.

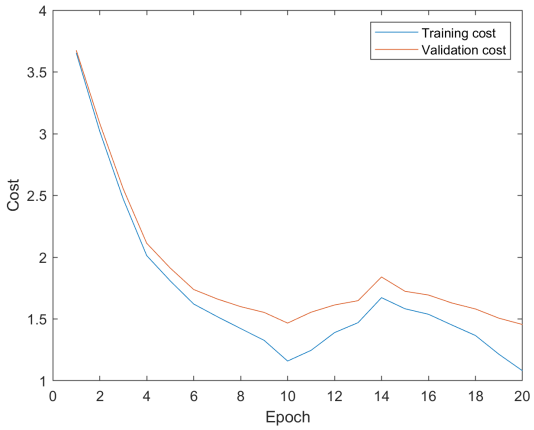

Figure 3: Cost plot

	Training data	Validation data	Test data
Accuracy	72.79%	58.18%	57.96%

Table 3: Classification accuracy with random jitter

The performance is improved a litte (about 0.5%) but not too much. And the amount of the random jitter should be chosen wisely, otherwise it would ruin the network.

4. According to the results above, we can see that the training accuracy is 72.79% and test accuracy is 57.96%, so the network needs more regularization. Then I applied dropout to the training process.

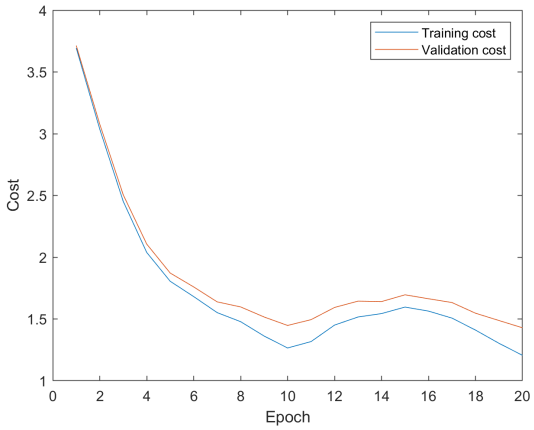

Figure 4: Cost plot

	Training data	Validation data	Test data
Accuracy	65.71%	56.90%	56.83%

Table 4: Classification accuracy with dropout

After applying dropout, test accuracy is now 56.83% which is lower than previous result. Therefore in this case, dropout doesn't help improve test accuracy, but helps alleviate overfitting.
5. According to the previous results, the best network: $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 0.1$, $\lambda = 5.2e-3$, $n\_s = 5 * 45000/n\_batch$, $n\_cycles = 2$, number of nodes of hidden layers: [200, 100, 50, 30], Xavier initialization, applying random jitter to each image in the mini-batch before doing the forward and backward pass.

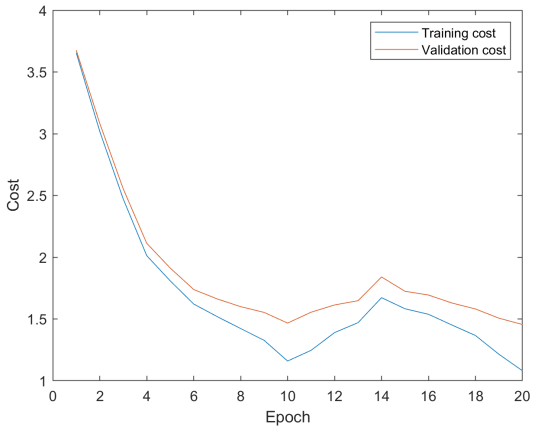

Figure 5: Cost plot

	Training data	Validation data	Test data
Accuracy	72.79%	58.18%	57.96%

Table 5: Classification accuracy of the best network

Therefore, the best test accuracy I got was 57.96%.