Deep Learning Lab Course
Second Exercise

Kinan Alzouabi

Implementing a CNN in Tensorflow:

I have Implemented LeNet with some help of 2 useful tutorials which are:
https://www.tensorflow.org/tutorials/estimators/cnn and
https://chromium.googlesource.com/external/github.com/tensorflow/tensorflow/+/r0.7/tensorflow/g3doc/tutorials/mnist/pros/index.md
They helped me to get more familiar with Tensorflow and it's concepts.

Learning Rate:

After implementing the neural network and making sure that it works correctly, we will have a look on the effect of the learning rate on the network's performance. We tried the following values for the learning rate: {0.1, 0.01, 0.001, 0.0001}. We can see the results in Table1 and Figure 2.

Table 1:

Learning rate	Minimum error	Test error
0.1	0.8936000093817711	0.886500000953674
0.01	0.011699914932250977	0.010599911212921143
0.001	0.028099894523620605	0.02959996461868286
0.0001	0.08799988031387329	0.09569990634918213

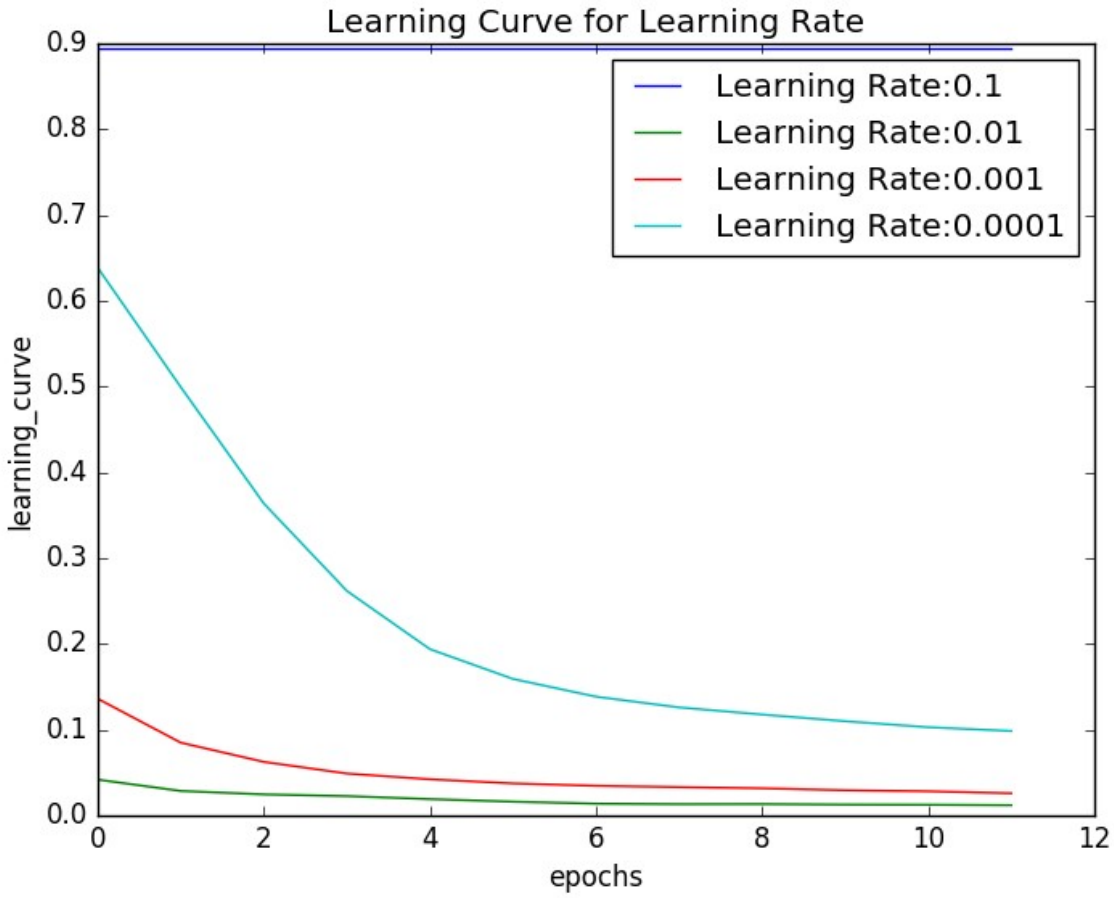

Figure 2, Learning Curves according to different learning rates

We can see that with a learning rate of 0.1, the networks doesn't learn anything because 0.1 is too big. On the other hand, 0.0001 is too small and it will take much more time to achieve an acceptable accuracy. 0.001 is good but we can see that 0.01 has the best result.

Convolution Type:

In this experiment we are trying out the following filter sizes {1, 3, 5, 7} (the same for height and width) and the results are in Table 2 and Figure 3.

Table 2:

Number of filters	Minimum error	Test error
1	0.115899920463562011	0.12019997835159302
3	0.023399829864501953	0.025299906730651855
5	0.020599842071533203	0.021399855613708496
7	0.018999814987182617	0.019199848175048828

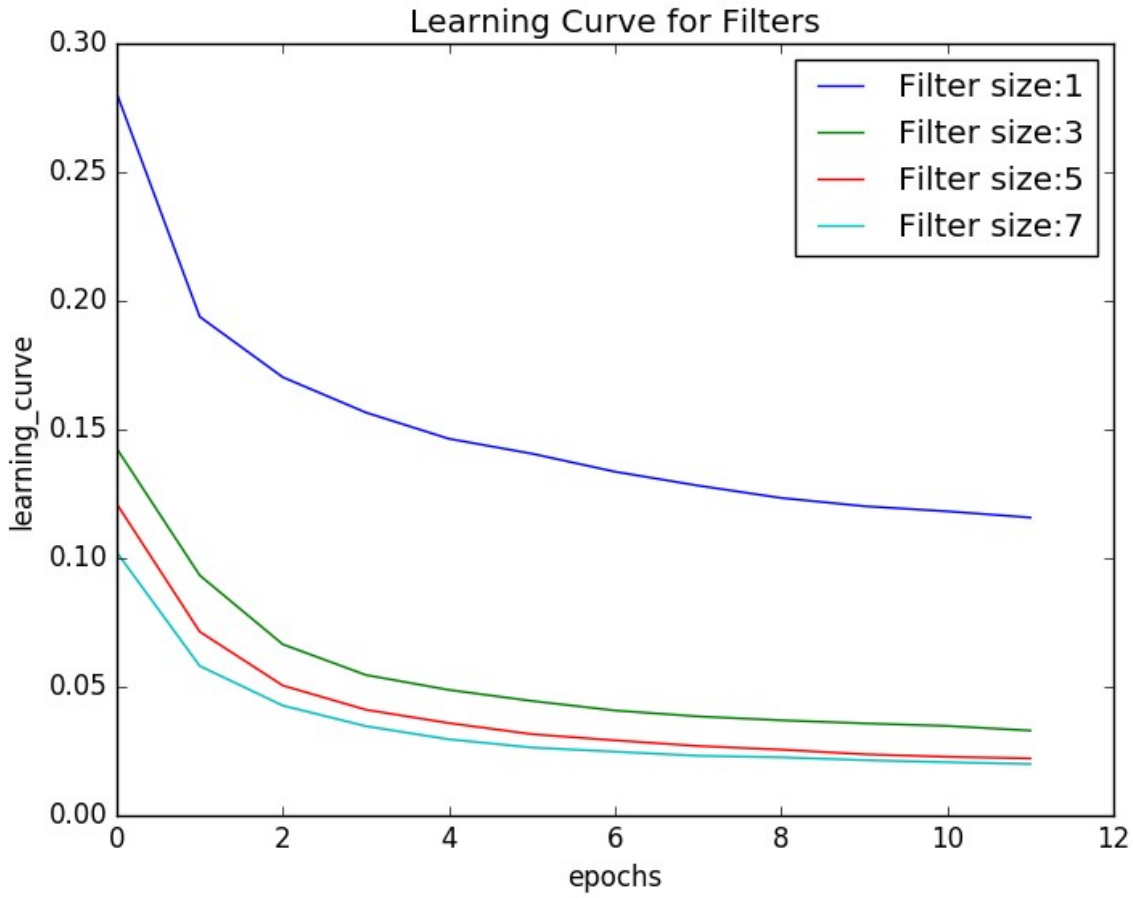

Figure 3, Learning Curves according to different filter size

We can see that the largest filter size gave the best accuracy. With a filter size of size 1, the CNN would work as a normal neural network. And with larger filter the CNN is showing it's potential. I think that if we need big amount of pixels to recognize the object then, we need larger filters. However, if the difference between objects is small or found in local features, we should use small filters.

Random Search:

In the last part of this exercise we will apply random search to automatically tune the following hyperparameters:
• learning rate ∈ [10 −4 , 10 −1]
• batch size ∈ [16, 128]
• number of filters ∈ [2 3 , 2 6]
• filter size ∈ {3, 5}
We will use the HpBandster (https://github.com/automl/HpBandSter) package for random search. In order to evaluate the progress of random search, Figure 4 shows the validation performance of the best found configuration after each iteration. Finally, the test performance of the best configuration that I found is: {'batch_size': 20, 'filter_size': '3', 'learning_rate': 0.007775791011876041, 'num_filters': 37}

Figure 4, validation performance of the best found configuration after each iteration

I have used https://colab.research.google.com to run random search code because there was a problem with pool PC. I think that google colab is a nice tool for an assignment like this one.
As a feedback I wish that we talked more about Tenserflow and the code during the lecture and less about the theory.