DeepLearning Lab Report - Exercise 2

Sara Al-Rawi

13. November 2018

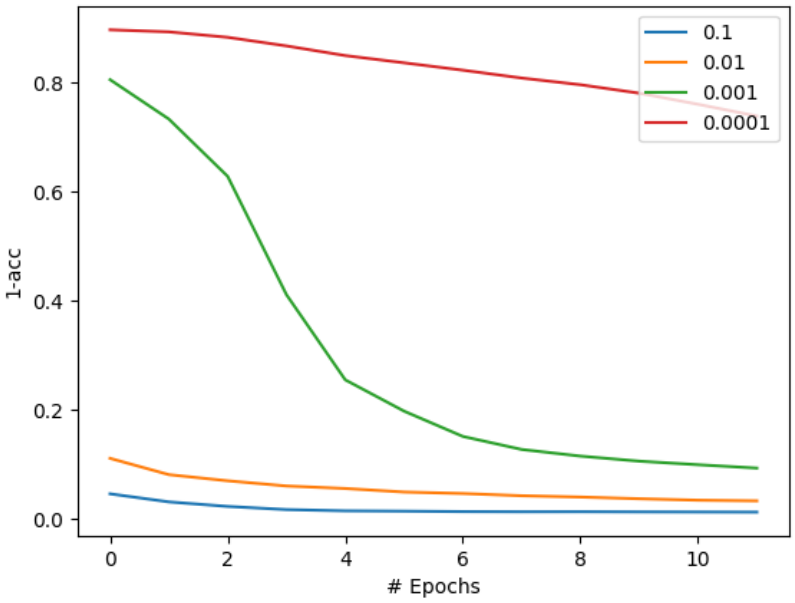
1 Introduction

In the first part of this lab work the convolutional neural has been implemented using the layers API from TensorFlow. The network consists of the following two convolutional layers, two pooling layers and fully connected lyer. For that purpose the function cnn_model_fn has been implemeted.

As a first step the network has been trained using the default learning_rate which was: 0.001. After that the model has been saved as well as the error rate which is (1-accuracy). Since the task is multiclassification problem the cross entropy has been considered as loss function. After successfully training the model, a test function has been invoked. The test function retrieves the graph through the session object.

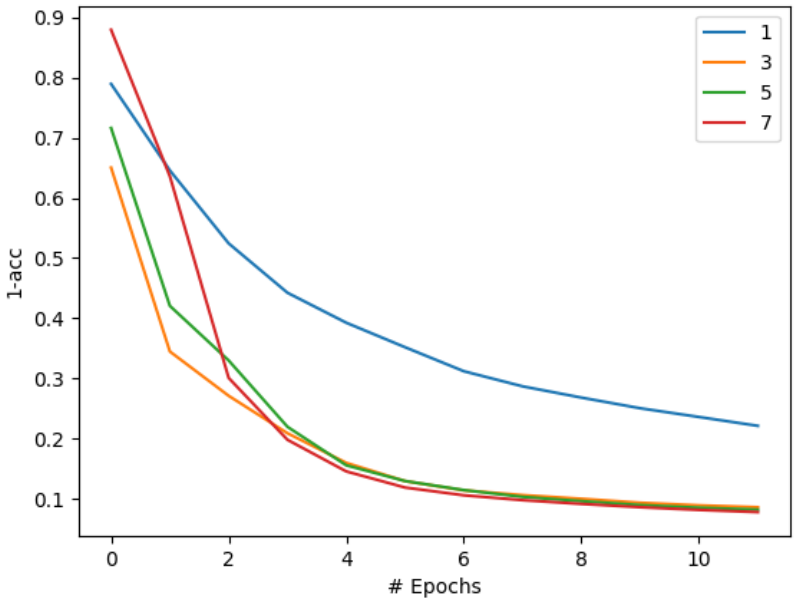
2 The Implementation and Observations

After implementing and training the network successfully and as a part from task two. Four different learning rates have been considered and the results have been collected as json files. The graph below shows the change in the error rate over 12 epochs considering different learning rates. From The graph it can be observed different trends. For the first learning rate which is the smallest one 0.0001, the lowest accuracy achieved, since the learning rate is small, the network takes longer training time in order to converge to the local minima, while when the learning rate (green trend) with the value 0.001 is considered. It can be observed that the trend is converging faster. The bigger learning rates can achieve low error rate at the beginning but with being close to the local minima the improvement in error rate will remain stable. In addition, considering big learning rate might over shoot the mini causing a high error.



As a part of observing the effect of considering different hyperparameter settings, different filter sizes have been considered during training. The graph below shows the change in the error rate over the number of epochs for different filters.

It can be observed that by decreasing the filter size the trend takes longer to converge observe the blue trend. The estreme case is filter size 1 (blue trend) converges slower than the filter size seven(red trend). It can also be observed that filter size seven (red trend) and filter size five (green trend) they start sharing almost the same error rate almost at the epoch 3.
The reason behind theses results, is by considering small filter size like one as extreme case means that the network must care very much about each pixel in the image considering low level features, therefore, the network needs longer training time in order to converge, on the other hand choosing large filter size means not considering the low level features. By choosing a filter size equal to the image size, in this case the network will behave as a fully connected network and it will ignore the low level features.



3 Hyper Parameter Optimization

In this task the hpbandster has been used. getConfig function has been implemented which returns a config_space object with integer, float and categorical hyperparameter. Unfortunately, running the random_search.py was not possible not the computer pool 54 due to ResourceExhaustedError. However, the code has been submitted and it is also available under the user alrawis on the pool computer 54 with all the files.