

Homework on convolutional networks

I decided to study the tutorial found here:

<https://www.codacy.com/app/hunkim/TensorFlow-Tutorials/file/4564407322/issues/source?bid=3237149&fileBranchId=3237149>

which implements a network on the following form

convolution → relu → max pool → dropout →
convolution → relu → max pool → dropout →
convolution → relu → max pool → dropout →
multiplication → relu → dropout → **multiplication**

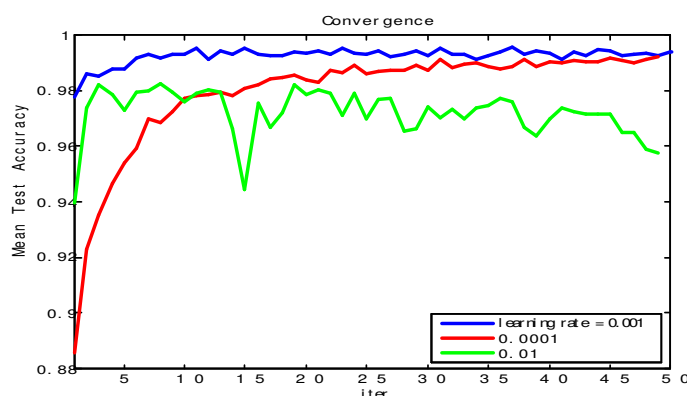
The convolution / multiplication layers consist of unknown weights w_1 w_2 w_3 w_4 and w_0 with following property

w_1 : size: 3x3 number: 32
 w_2 : size: 3x3x32 number: 64
 w_3 : size: 3x3x64 number: 128
 w_4 : size: 128x4x4 number: 625
 w_0 : size: 625 number: 10

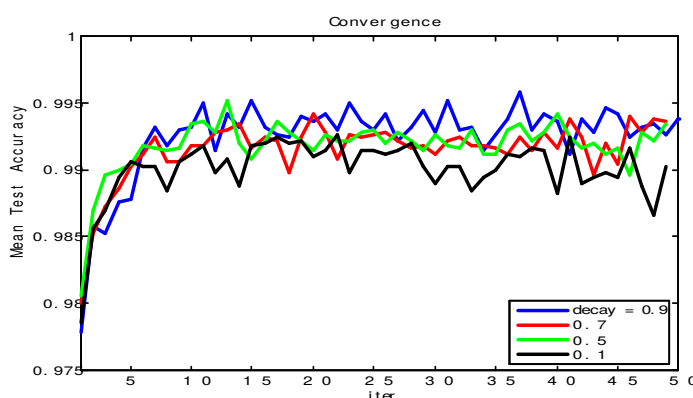
where size indicates the size of the convolution kernel and number, the number of kernels in the layer.

The network was applied to the MNIST dataset and was trained using RMSProp with decay and learning rate parameter.

Initial tests showed what happened to the learning performance when varying these parameters: Mean test accuracy was evaluated in a randomized test set of 5000 figures.

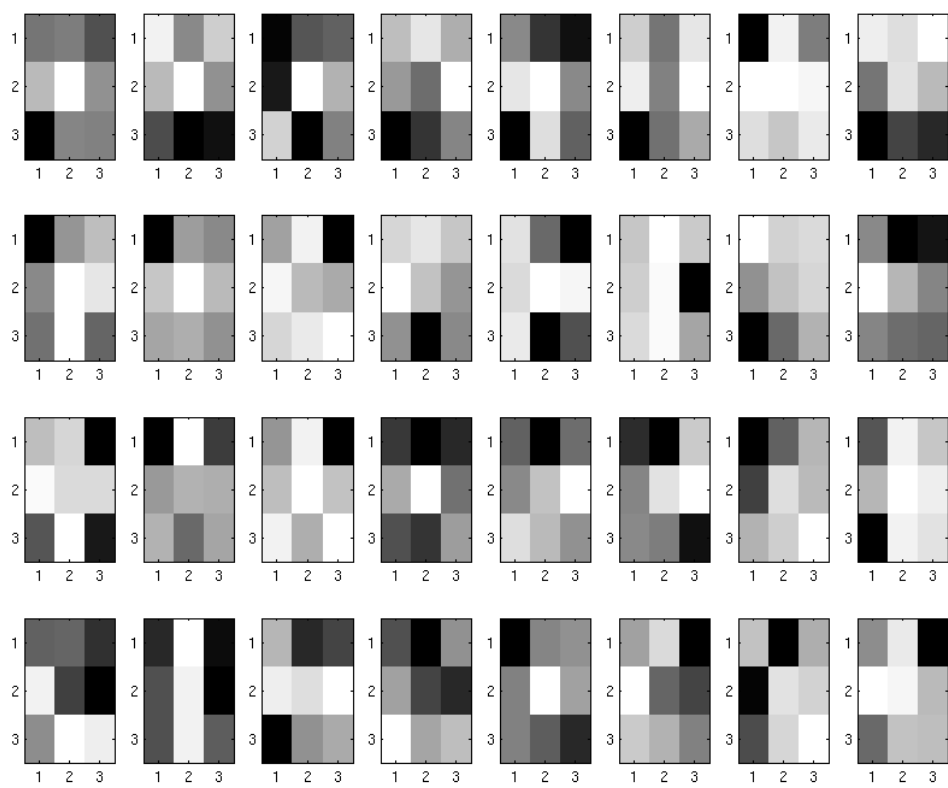


Performance w.r.t learning rate behaves as expected. It is a clear trade off between initial convergence rate and performance after convergence.



Performance trend w.r.t decay was not as strong but it seems that a smaller learning rate have similar behavior to that of a large learning rate, i.e improved initial convergence rate but decreased performance after many iterations.

The second part of the homework was to study the weights after training, the figure below shows the w1 convolution kernels. The convolution kernels deeper inside the network were to many to present here.



My hypothesis is that these kernels try to detect image gradients in various different directions.

More interesting kernels where found when increasing the size to 9x9 in the first layer, see figure below.

