Assignment report : Image classification using Convolutional Neural Network

Akanksha Bhardwaj (RIT2015035)

Dataset

The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples.
The images are of size 28 x 28.

Architecture

The framework that has been used by me for implementing the model is Tensorflow and the training was done using Google colabotory platform.

The convolutional neural network architecture used consists of the following layers:

- ➔ The first layer is a convolution layer with 32 filters, each of 3 x 3 filter size
- ➔ The second convolutional layer has 64 filters again of 3 x 3 filter size
- ➔ First dropout layer with keep probability of 0.50
- ➔ The third convolutional layer will has 128-3 x 3 filters.
- ➔ Each conv layer was followed by a max-pooling layers each of size 2 x 2.
- ➔ After three conv and max-pooling layer a second dropout layer with keep probability of 0.50 was kept
- ➔ A fully connected layer with 128 units
- ➔ Finally an output layer of 10 units (o show percentage of classification for each digit)

Activation function:
Relu or rectified linear unit rectifier (activation function max(0,x)) was selected. The rectifier activation function is used to add non linearity to the network.

epochs = 20
learning_rate = 0.001

Experiments

The following experiments were conducted in order to choose the layers and parameters for the network:

1. Adding two Drop connect layers (after second conv layer and after third conv layer)

Since there was no out of the box drop connect layer in Tensorflow the dropout layer was used to implement the same as guided in this article

def dropconnect(W, p):
*return tf.nn.dropout(W, keep_prob=p) * p*

Loss= 0.109041, Training Accuracy= **0.96875**
Testing Accuracy at convergence (epoc = 30) : **0.99020**

2. Adding two dropout layers

First dropout layer after 2nd conv layer and second after 3rd conv layer: improved testing accuracy.

1) Training accuracy seemed to drop with dropout layers as compared to dropconnect layers.
2) With dropout layers the model converged to higher testing accuracy of **0.99240** earlier as compared to drop connect at keep_probability = 0.25
Loss= 0.068146, Training Accuracy= 0.97656

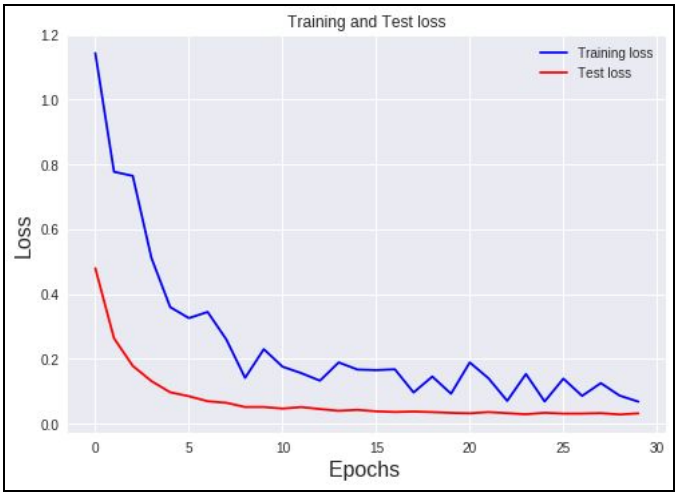


Fig1.Loss vs epochs:dropout, keep_prob = 0.25

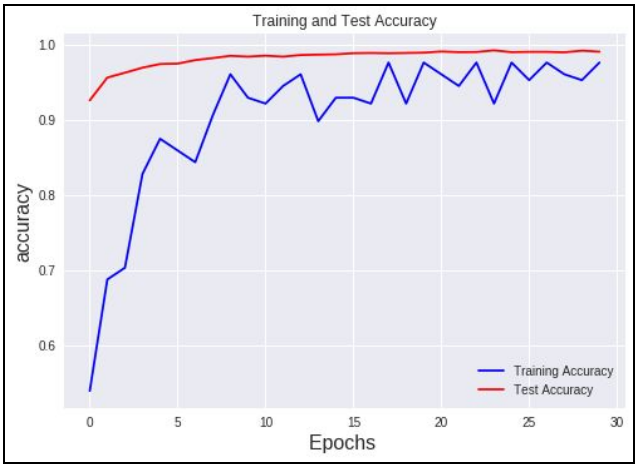


Fig2.Accuracy vs epochs:dropout, keep_prob = 0.25

3) Changing keep probability parameter for dropout layer. Keep probability implies to the ratio of nodes to be kept in the dropout layer.
    a) At greater keep_prob = 0.50 better accuracy was achieved as compared with keep_prob = 0.25

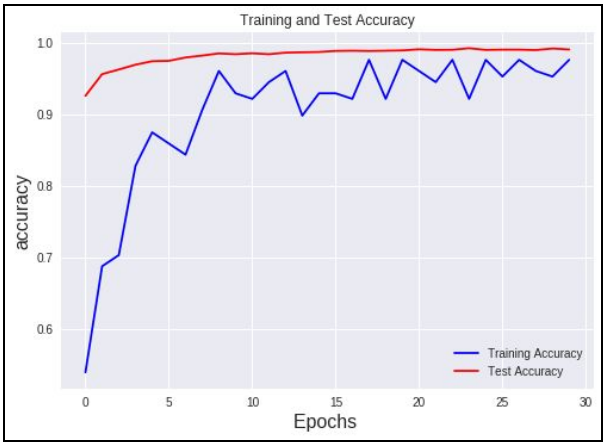


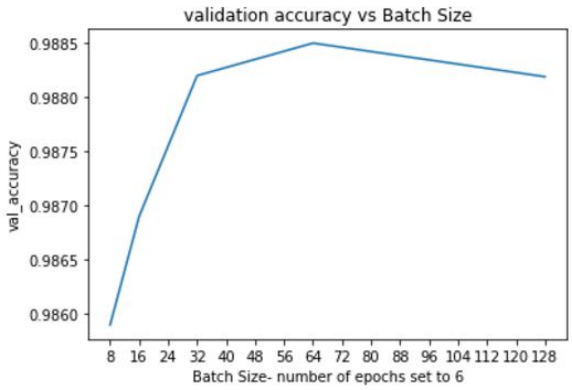
Fig3.Accuracy vs epochs:dropout, keep_prob = 0.50

Loss= 0.003373, Training Accuracy= 1.00000
Testing Accuracy: 0.99390

    b) However further increasing keep_probablity to 0.75 yielded comparatively worse testing accuracy.

Loss= 0.002206, Training Accuracy= 1.00000
Testing Accuracy: 0.**99380**

3. Changing batch size
    a. Upon decreasing batch size to 64 testing accuracy increased, although training decreased at conversion as compared to batch size of 100 and 128



    b. Training time was optimised upon decreasing the batch size between 128 to 64.

4. Changing stride :
   Keeping stride in convolution layer as 2 improved computational time and cost but keeping stride as 1 bumped up the test accuracy slightly results.

   At convergence, keeping 2 dropout layers (keep'-prob = 0.50), stride = 1:
   Loss= 0.002749, Training Accuracy= 1.00000
   Testing Accuracy: 0.99250

   At convergence, keeping 2 dropout layers (keep'-prob = 0.50), stride = 2:
   Loss= 0.025862, Training Accuracy= 0.99219
   Testing Accuracy: 0.98970

Analysis

Based on the above experiments with the CNN architecture the following things were analysed:

- DropOut and DropConnect layers help in tackling with overfitting issue and we might fail to generalize and produce not good enough results for test data. Both methods helps prevent "co-adaptation" of units in a neural network, which means

each neuron should be capable of independently extract the feature its intended to by not relying on other neurons. With dropout layer some of the nodes are dropped out by selecting a subset of the units randomly a keep probability or a drop rate while with drop connect layer some of the connections are only dropped out and not the full node.

- With experiments it was analysed that the effect becomes more apparent when there are huge no. of connections i.e. complex network. With out simple network the effects were not that prominent however the highest test accuracy achieved was with help of addition of two drop out layers.
- Batch size of small magnitude help to extract the fine details but also increase the computation time, hence batch size of 128 was selected.
- As in the case of batch size, increasing stride will consequently result in loss of details captured and hence poor results are observed.

Results/Conclusion

The layers and parameters for the network were chosen by experimenting with various combinations of layers like dropout layers, dropconnect layers and parameters like batch size, keep_probability for dropouts, stride and padding. Based on the testing best test accuracy achieved was **0.99390**
This kind of analysis helps to understand better what kind of network settings works best depending on the nature of problem. Starting from data preprocessing to training and testing the model, each phase requires fine tuning of intricate parameter settings.

References

1. http://yann.lecun.com/exdb/mnist/
2. https://www.datacamp.com/community/tutorials/cnn-tensorflow-python
3. https://nickcdryan.com/2017/06/13/dropconnect-implementation-in-python-and-tensorflow/