11/21/2016

Vaibhav Sharma

DEPAUL UNIVERSITY

Convolutional Neural Networks

**INTRODUCTION**

Convolutional Neural Networks or CNN is a type of feed-forward artificial neural network, which are made up of neurons that have learnable weights and biases. This type of network is particularly designed to classify images but its usage is not limited to just that. This type of network can be used for sentiment analysis, generate images and can be used with other advanced neural network approaches to perform more advanced analysis.

The aim of the project is to develop Convolutional Neural Network without the use of Neural Network libraries such Theano, Keras or TensorFlow. I chose this approach to understand the minute details of the working CNN. This project is not meant for industry level usage but could be a useful resource for learning purposes as it exposes features that are hidden when we use libraries. The application is currently in work-in-progress state. I will be discussing some of the improvements/enhancements in the further sections.

**LEARNING SOURCES**

The primary resource for this project was chapter 6 from the class text book – Neural Networks and Deep Learning by Michael Nielson. I have also used various online resource to concretize my understanding of the topic and backpropagation with respect to convolutional layer. Below are some of the resources I have accessed.

1. <http://neuralnetworksanddeeplearning.com/chap6.html>
2. <http://cs231n.github.io/convolutional-networks/>
3. <https://www.youtube.com/watch?v=GYGYnspV230&index=7&list=PL16j5WbGpaM0_Tj8CRmurZ8Kk1gEBc7fg>
4. <https://grzegorzgwardys.wordpress.com/2016/04/22/8/>
5. <http://jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/>
6. <http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/>

**DATASETS**

I have used 4 data sets which are variations of MNIST dataset.

1. MNIST Basic

<https://github.com/mnielsen/neural-networks-and-deep-learning/tree/master/data>



Figure 1 MNIST Basic Dataset Image Source: https://www.tensorflow.org/versions/r0.8/tutorials/mnist/download/index.html

1. Modified MNIST dataset:

<http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations>

1. MNIST Rotated



Figure 2 MNIST Rotated Dataset Image Source: http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations

1. MNIST with Background Images



Figure 3 MNIST with Background Image Source: http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations

1. MNIST with Random Background



Figure 4 MNIST with Random Background Image Source: http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations

**HOW TO INSTALL AND RUN**

1. Copy all the codes in the folder say CNN
2. Create a folder ‘data’ in CNN and download the datasets in that folder
   1. For MNIST Basic do not unzip the .gz file
   2. For all other MNIST datasets unzip the downloaded file
3. Execution
   1. Main.py – Load file in python 2.7 shell and call the main () function in the shell and enter the inputs as asked

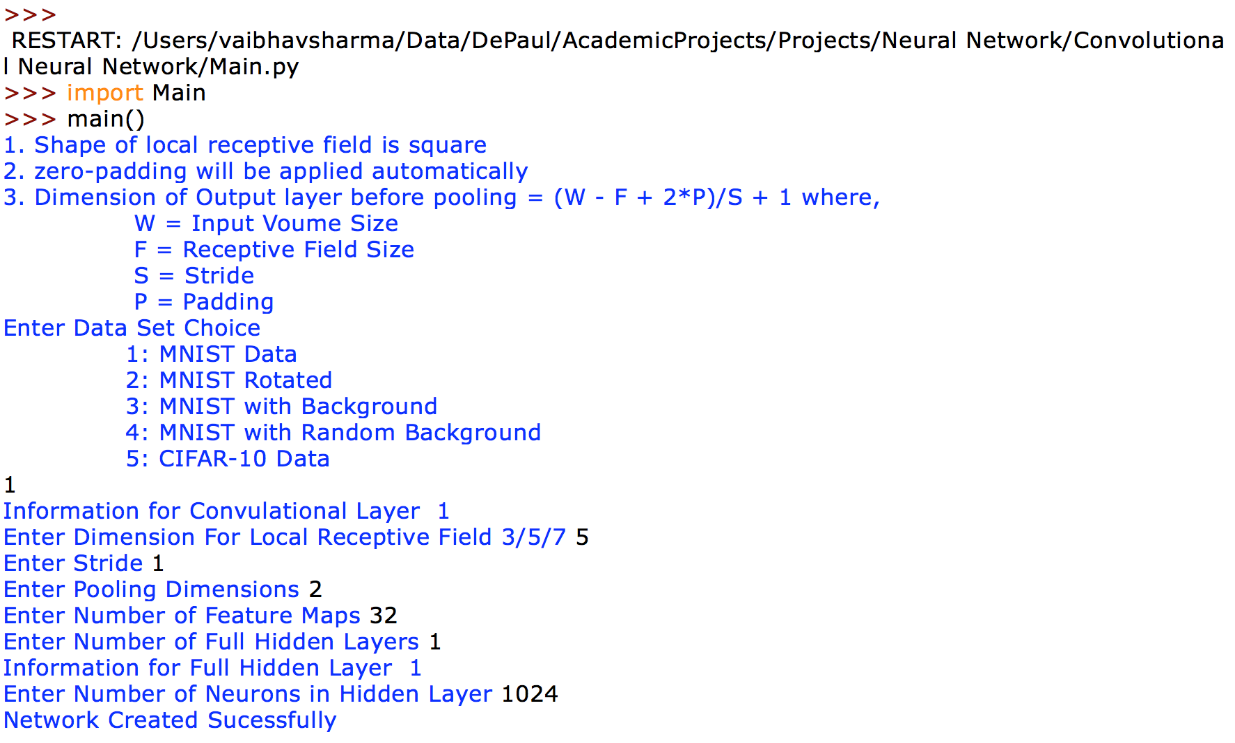


Figure 6 Sample run using Main.py

* 1. testfile.py – Load the file in python 2.7 shell. Execution will start immediately. If you want to change the parameters, you can edit the file itself.

**APPLICATION ARCHITECTURE**

When I started working on this project, my aim was to make the source code that is flexible, easy to understand and modify. As I went ahead in the development, my focus changed to get the things working for MNIST dataset. Application code in its current form is still flexible and modular but it is slightly coupled to MNIST dataset.

1. **Application Details**
2. Programming Language: Python 2.7
3. Libraries Used:
   1. NumPy – For storing and manipulating data
   2. MatPlotLib – For plotting output results
4. **Code files**
5. ***Main.py*** – This is a very basic text based UI for entering input. It asks a few set of questions to build network and start training
6. ***testfile.py*** – The purpose this file is like *Main.py* but it does not provide UI. It assumes that you know how run the application without the use of UI.
7. ***FullyConnected.py*** – Class file for fully connected layer. This is similar what is there in *Network2.py* in the Neural Network and Deep Learning text book.
8. ***Covolutional.py*** – Class file for convolutional layer. It contains all the functionalities related to convolutional layer.
9. ***Network.py*** – Class file for the complete network. In this class, we build the network’s layer object and perform training
10. ***Activations.py*** – Class file for activation functions such as Sigmoid, Relu, Softmax, Tanh
11. ***CostFunction.py*** – Class file for the all the cost functions such as Quadratic, Cross Entropy and Log
12. ***DataLoader.py*** – Utility class file for loading all the MNIST variations and CIFAR10 into NumPy arrays.
13. ***Validations.py*** – Utility class file for performing input data validation. Used by Main.py
14. ***ApplicationExceptions.py*** – Utility class that lists all the application related exceptions (in development)

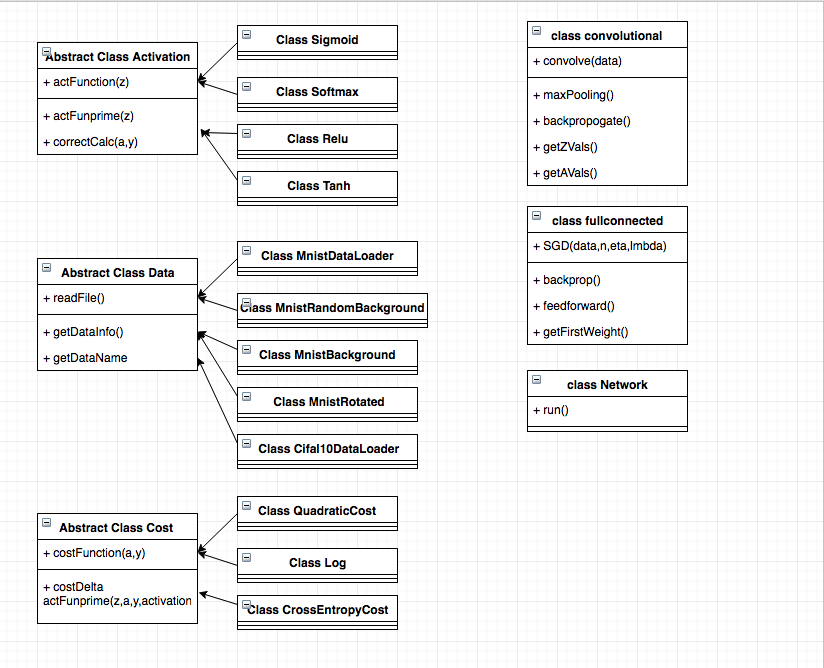
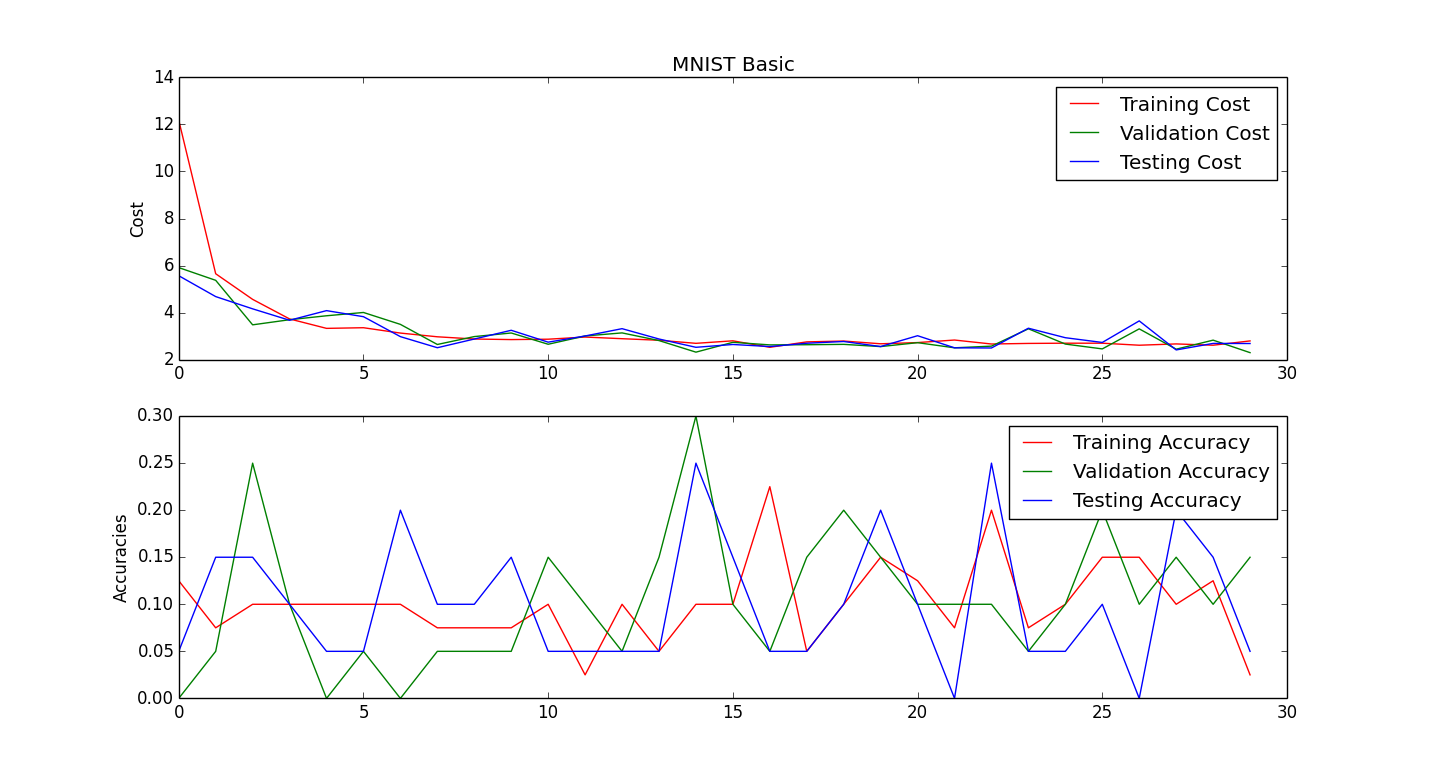


Figure 5 Class Diagram for Important Classes in the Application

**OUTPUT**

I ran the code for following network for basic MNIST dataset:

1. Convolutional Layer – 1
   1. Local Receptive Field: 5 X 5
   2. Padding: 2 layers of zero-padding over the edges
   3. Stride: 1
   4. Pooling: 2 X 2
   5. Number of Feature maps: 32
2. Fully Connected Layer – 1
   1. Hidden Layer 1: 1024
   2. Activation Function: Softmax
   3. Loss Function: Log
3. Eta – 0.3
4. L2-Regularization Lambda – 5



As it was taking a lot of time, I reduced the size of data to 20 and ran it for 30 epochs. We can clearly see the initial drop of costs because of implementation of log-likelihood. But as the input data set is small we do not see improvement in the accuracies. It looks the network has stopped learning.

I ran same network on different data sets and got the similar output.

**ENHANCEMENTS**

Currently the network is very slow and It takes hours to execute single epoch. This was expected because while developing application my focus was for it to be as close to the text as possible and not on the running efficiency. Following are some of the enhancements that I would like to put in the application:

1. Implement fast matrix multiplication and fully connected layer to convoluted layer conversion as discussed in the following page: <http://cs231n.github.io/convolutional-networks/>
2. Delete used data structures thus cleaning up the memory
3. Implement parallel processing where ever possible
4. Implement early stopping criteria
5. Implement Neural Network features such as Dropout, etc.
6. Explore ways to make application more flexible and with less dependencies

**CONCLUSION:**

I am excited with the initial success I have achieved in this project. I will continue to work on this project and post updates on GitHub.

**Link to GitHub:**

<https://github.com/vaiarrm/AcademicProjects/tree/master/Neural%20Network/Convolutional%20Neural%20Network>