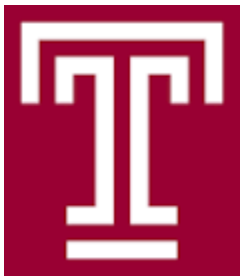
Machine Learning

Course Code – 5526

Fall 2016, Section: 01

[Final Project Report]



Submitted To:

Dr. Richard Souvenir
Associate Professor

Submitted By:

Ashis Kumar Chanda

(TU ID: 915364305)

Problem Statement:

In this project, a real-world image classification problem is given to solve by machine learning algorithms. The goal is to classify images from hotels into one of 8 classes: 'Bathroom', 'Guestroom', 'Pool', 'Gym', 'Restaurant', 'Lobby', 'Aerial View', 'Business Center'.

Project Plan:

There are many machine learning algorithms to apply for classification problems, such as, K – nearest neighbor, SVM, Neural network. However, each step has some drawbacks to classify image objects. As it is a multi-class image classification problem, I have planned to solve the problem using **Convolutional Neural Network** (**CNN**) [1]. The authors of Krizhevsky et al (2012) [2], have described the process of classify images using deep neural network. They also explained different strategy likes, uses of Rectified Linear Units (**ReLu**), data augmentation, dropout; to improve the performance of program.

Implementation Approach:

The major are summarized below:

- To implement the CNN architecture in python, I selected some popular machine learning libraries. Like: **Theano**, **Keras** [3, 4].
- Keras is a NN library that supports to create convolutional network easily. There are many API to apply or check different learning parameters.
- To design a CNN architecture, at first we need to create a model (core data structure of Keras) using Keras library.
- After declaring a sequential model, we need to define different layers. The information of each layer is given below:

Table: A summarized view of different layers

Order	Layer name	Parameters	Activation function
1	Convolutional input layer	32 feature maps with a size of 3×3	ReLu
2	Convolutional layer	32 feature maps with a size of 3×3	ReLu
3	Max Pool layer	Size 2×2.	
4	Convolutional layer	64 feature maps with a size of 3×3	ReLu

5	Convolutional layer	64 feature maps with a size of 3×3	ReLu
6	Max Pool layer	Size 2×2.	
7	Dropout set to 25%		
8	Flatten layer	(It changes our data into 1D format)	
9	Fully connected layer	512 units	ReLu
10	Dropout set to 50%		
11	Fully connected output layer	8 units	softmax

CNN model:

- In first layer, the data inputs are used in convolutional layer. The layer considers 32 feature maps with size of 3×3. Then, another convolutional layer is added with the same parameter.
- After these two layers, a max pool layer is used with size of 2×2. It will take the most responsive node of the feature map.
- Similarly, another set of convolutional layer is used with 64 feature maps with size of 3×3
- To overcome over fitting, **dropout layer** [8] is used after dense layer. It will randomly drop some data. We can set the percentage of data that would be dropped by this layer. It is a very popular way to reduce error.
- Finally, fully connected layer is used for 8 units where softmax activation function is applied to get result.

Loading Data:

- Our training data is in image format. At first we read the image files and resize each image in 100 by 100 pixels. I plotted the images to find how the images are changed. Some samples are shown below:

Figure: Testing data images after resizing in (100 by 100) shape

- The training.csv file is also loaded in a numpy array to train our data

Normalization:

- To normalize our dataset, each image value is divided by 255.
- **loss function:** As it is a classification problem, where we need to find classes of each image, categorical cross entropy is good for our model.
- **optimizer:** stochastic gradient descent (SGD) is used as an optimizer of my proposed model. I also tested with Adam **[10]**, but I found SGD is good.

Experiments:

- To find best performance, I tested the accuracy of program by taking some sample data from training set and performing validation test.
- After adding a new layer, the result is tested and compared with previous result. Suppose, whenever I got good results by adding two set of convolutional layer, then I added set of convolutional layer with 128 feature maps. But, as the result was not good, I returned to the previous layer design.
- I thought a feature map with large size like 5×5, will be better to extract feature efficiently. However, I could not find better performance using feature maps of 5×5 size.
- I changed learning rate from 0.001 to 0.01, but there were no major difference in accuracy. However, when I take small learning rate, the program takes more time/iteration to generate output. Thus, I chose 0.01 as final learning rate.
- I also checked different batch size and I found 32 are good to fit.

- The program has tested many times with several epoch values. When epoch value increased, the performance became better. But, the rate of change was very between 12 epochs to 20 epochs. Thus, the selected epoch number is 20.
- As there is a limitation of memory in my machine, I could not test the program in large valued and deeper layers.

Results:

- If we run the code, the final output will be saved in **output.csv** file.
- Keras generates an auto result summary for each epoch. It will show loss value and accuracy for validation data based on the designed model.
- There are many corrupted image file in training and testing data. I used try/catch block to avoid such corrupted/damaged files.
- As a result, the output files contained almost 19634 data, not 19648. Then, I checked the missing image files and set some random values for the images.

  (The corrupted image file numbers are: 53675, 55478, 59009, 60151, 60968, 64800, 68787, 72363, 72751, 73726, 75380, 76032, 76787, 79616)
- Again, it took long time to generate output file for huge testing data (almost 19000). In this reason, I divided the whole testing data into several parts to get result quickly. Finally, I merged all result in a single file to upload in Kaggle.

Problems:

- At first, I generate the output (testing result) file in 0 or 1 format. If one class has maximum value, then the program considered 1 for this class and other classes got 0. That time, the result was poor in Kaggle score board.
  Later, I used the prediction values in floating number and then, the result became good.

Used Library List:

- Theano
- Keras
- OS
- CSV

- PIL
- NUMPY

References:

1) http://cs231n.github.io/convolutional-networks/
2) A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
3) Theano tutorial: deeplearning.net/software/theano/tutorial/f
4) Keras tutorial:https://keras.io/
5) https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721#.d3od2grvt
6) Deep learning tutorial: http://deeplearning.net/tutorial/
7) Examples: https://github.com/fchollet/keras/tree/master/examples
8) machinelearningmastery.com/tutorial-first-neural-network-python-keras
9) Srivastava, et al. 2014, paper Dropout: A Simple Way to Prevent Neural Networks from Overfitting .
10) Adam: A Method for Stochastic Optimization, Diederik Kingma, Jimmy Ba https://arxiv.org/abs/1412.6980