THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Machine Learning II
Exam #2
Due: *November-26-2018*

Q1:Training Convolution networks with Caffe on MNIST dataset

The objective of this miniproject is to become familiar with Caffe for training deep convolution networks on large datasets. You will be provided with a sample program that loads in the MNIST data set and sets up an example convolution network to be trained on it. (See `http://caffe.berkeleyvision.org/gathered/examples/mnist.html` for a discussion of this example.) The instructions below provide a rough framework of what you should do for the miniproject. You should experiment with using Caffe on this large dataset. The project is open-ended. Learn as much as you can about using deep convolution networks, and relate what you have learned in your project report. For all parts below, and any other experiments you run, include the results into one PDF file, and upload it to the BlackBoard. Include all program listings, plots, command line printouts, discussion, etc.

Very Important Note: Your workflow for the exam should be as follows:

- **First:** Create a folder and title it as your name followed by an underscore and exam2. For example, I would create a folder and save it as `amirjafari_exam2`. **All** of your work for the exam should be saved in this folder.
- **Second:** Since part of the goal is to communicate what you have learned about multilayer networks by experimenting in Caffe, you will write a report to document your learning process. Type your answers for each question into a word document. Be sure to include program listings, plots, command line output, etc. if necessary. Save your word document as a single PDF file and name it `exam1_report`.
- **Third:** Create a new folder, name it `report`, and move the PDF file of your report into it.
- **Fourth:** Create another new folder and name it `code`. All the code you write for the exam should be placed in this folder. You should have different scripts for each question and you should give them informative names such as `Q1_sol.py`, `Q2_sol.py`, `Q4_3layer_sol.py`, etc. Name the modified prototxt files for each questions that we can run it independently.
- **Fifth:** Zip the main folder and upload it to Blackboard. It should be titled similarly to the main folder (i.e., `amirjafari_exam2.zip`.
- Failing to do these steps result in a reduced grade.

1. Download `train_mnist.py`, `get_mnist.sh` and `create_mnist.sh` from `https://github.com/amir-jafari/Deep-Learning/tree/master/Caffe_/Mini_Project` and put them in the directory where you want to run your programs.
2. Run the following commands on terminal in the your working path directory:

```
chmod 777 create_mnist.sh
chmod 777 get_mnist.sh
./get_mnist.sh
```

3. Open up the `create_mnist.sh` and change the path to of EXAMPLE and DATA to the directory that you are working with. Then run the following commands.

```
./create_mnist.sh
```

4. By this time you should have the train and test lmdb files of the mnist data set.

5. Download `lenet_solver.prototxt`, `lenet_train_test.prototxt` from GitHub and put them in the directory. Change the path of net in `lenet_solver.prototxt` to your working path directory.

6. Run the program `train_mnist.py` in PyCharm and investigate and verify its performance. You may need to change the line "`my_root =`" to the appropriate path.

7. Investigate the kernels in the two convolution layers. Can you identify kernels that would be useful for particular numerals?

8. How does the performance of the convolution network compare with the multilayer networks that you used in Exam#1?

9. Change the size of the minibatches (`batch_size` parameter). If you make the batch size very large, does it affect the computation time significantly? Describe the advantages and disadvantages of increasing the batch size. Find a good choice.

10. Use a dropout layer at layer `fc1`. Make `fc1` the top and the bottom for the dropout layer. (See https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf for a description of dropout.) Does dropout improve the testing error?

11. Experiment with different numbers of layers and different numbers of kernels. Maintain the total number of weights and biases in the network, while increasing the number of layers in the network. Describe how the performance changes as the number of layers increases – both in terms of training time and performance.

12. Try one other training function from the list on this page: http://caffe.berkeleyvision.org/tutorial/solver.html. Compare the performance with gradient descent.

Bonus-Q2:Training Convolution networks with Caffe on CIFAR10 dataset

Answer all the above questions with CIFAR10 data sets and train it with Caffe. You are allowed to use any source to solve this question.