Homework Assignment 1
(Programming Category)

Student Name:______________________

Student Session: cs6220-A

You are given three types of programming problems in the first homework assignment. Problem 1 is on data-driven deep learning system tuning and Problem 2 is on learning with hand-on experience on using deep learning models. Program 3 is an application software development problem that requires you to write code. Each problem is given two subproblems. This provides sufficient diversity for students with different backgrounds and training. You only need to choose one subproblem from one of the three Problems as your first homework. For Problem 3, feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

Post Date: on Monday of Week 2
Due Date: midnight on Friday of Week 3

Problem 1 Performance tuning for effective deep learning

This assignment provides you with experience in tuning machine learning model configurations for high performance for big data systems and applications.

(1)Software download.
You may choose one of your favorite machine learning (ML) or deep learning (DL) framework for this homework.

The popular DL frameworks are TensorFlow (google), Caffe, Torch/PyTorch, OpenCV (Intel), CNTK (Microsoft).
https://www.tensorflow.org.
https://caffe.berkeleyvision.org
http://torch.ch or https://pytorch.org
https://docs.opencv.org/master/index.html
https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine
https://keras.io
You may also use Keras, a high-level neural networks API, written in **Python**, for this homework as it is capable of running on top of TensorFlow, CNTK, or Theano.

In the rest of the problem description, I will TensorFlow (TF) to describe the homework but you can replace TF with one of your favorite ML/DL frameworks.

(2) Dataset Preparation
You are asked to use at least two types of image datasets to create your own training datasets for binary classification tasks.

(i) Color rich and object rich images like photos of people and buildings together, or color photos of cats and dogs. Such as Kaggle (https://www.kaggle.com/c/dogs-vs-cats), or CIFAR10, (https://www.cs.toronto.edu/~kriz/cifar.html), or ImageNet (https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet).

(ii) Simple grey-scale or black-white images. One example is MNIST. (http://yann.lecun.com/exdb/mnist/)

You may find such images also from other image datasets in the public domain such as Kaggle, ImageNet, or creating your own.

This programming homework provides multiple options. You are required to choose at least one option.

Problem 1.1 Comparing TensorFlow Trained CNN classifier with varying dataset complexity

Goal. The goal of this problem is to get you to learn the performance of deep neural network trained classifiers with respect to different complexity and variations of datasets.

You only need to work with an off-shell deep learning software framework such as TensorFlow as an application developer. It does not require you to write or modify any program code. Your job is simply to use TensorFlow to generate 4 CNN classifiers, one per dataset configuration, as executables for performance comparison on training and testing performance (time and accuracy).

For beginner, you are recommended to create a binary classification task. Feel free to create a classification task with k classes (k=10, 100, 1000, for example), depending on your machine capacity for running the TensorFlow training.

Requirements: You are asked to vary a set of dataset-specific parameters in your training phase setting to make observations and comparisons:

(1) the input datasets in terms of varying volumes (at least one small and one 10x larger). The number of images per class should be no smaller than 100.

(2) the input datasets in terms of varying resolutions (at least two different resolutions, e.g., 28x28, 64x64, 128x128)

(3) For each of the four datasets from the same domain (say color images from ImageNet), you are asked to use TensorFlow to train 4 different CNN classifiers and to make comparison.

(4) **Outlier Test Scenario**. When performing testing on each of the 4 CNN classifiers you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 images that do not belong to the binary classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set.

(5) Optional task 1 (only if you have more time and interest): You may choose one or two subtasks in the deliverable of Problem 2.2.

   a. Choose the one binary CNN classifier that has the highest test accuracy among the 4.
   b. For this CNN classifier and its training dataset, you are asked to use three different ratios of training size v.s. testing size and make a comparison.
   c. For this CNN classifier and its training dataset, you are asked to change the setting of #epoch to be 5x and 10x larger. Report your comparison. Hints: if your training dataset has B mini-batches, then you will need #iterations = #epochs x B.
   d. For this CNN classifier and its training dataset, you are asked to
   e. For each Use at least two different batch sizes such as 10 by 10 and 5 by 5 for an image grid size of 128 by 128.

Deliverables:

(1) Provide URL of your open source code packages

(2) Example screen shots of your execution process/environments

(3) Input Analysis: *Use a table to report your setup parameters for **each** training model built*:

   a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
   b. Choose 5 sample images and show each in their two resolution versions.
   c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
   d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, Densenet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for your training and testing experiments.

e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence. Usually it is documented for MNIST and CIFAR-10.

(4) **Output Analysis:** *Report the performance comparison and analysis of your 4 CNN classifiers*
   (a) You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (each is trained from one dataset configuration), in terms of training time, training accuracy, testing time and testing accuracy.
   (b) You are asked to record the trained model size in MB for each of the four classifiers in a table.
   (c) You are asked to provide a table of the same format but this time your test is performed on your outlier test dataset in requirement (4).
   (d) You are asked to make at least three observations from your experimental comparison of the 4 CNN classifiers.

Problem 1.2 Comparing TensorFlow Trained CNN classifier with varying training parameter configurations

Goal. The goal of this problem is to get you to learn the impact of different CNN training configurations on the performance of deep neural network trained classifiers.

You only need to work with an off-shell deep learning software framework such as TensorFlow as an application developer. It does not require you to write or modify any program code. But you will need to make changes to the TensorFlow CNN configuration default on a small selection of parameters.

Your job is to compare the TensorFlow default setting with at least 3 alternative configurations you made and generate 4 CNN classifiers as executables for performance comparison on training and testing performance (time and accuracy).

For simplicity, you are recommended to create either a binary classification task or use a classification task of 10 classes. If you have powerful GPU on your machine for running the TensorFlow training, you may choose a classification task with a larger #classes (e.g., 100, 1000, 2000, 5000).

Requirements:
(1) **Dataset:** You are asked to create a color image dataset of no smaller than 100 images from ImageNet, or CIFAR-10, CIFAR-100 or from your own collection, according to your classification task. For example, for a binary task, you may collect 50 (or 100, 500, 1000) photos of one object and 50 (or 100, 500, 1000) photos of another object.

(2) CNN training by default configuration. You are asked to use TensorFlow default configuration for CIFAR-10 (which comes with the download) to train a CNN classifier for your binary classification task. Record training time and accuracy and testing time and accuracy.

(3) You are asked to make three modifications on the TensorFlow default configuration you used to train your first binary classifier.

   a. Change the default DNN model (e.g., change LeNet to ResNet-52)
   b. Change the default CNN structure of a given DNN model, say LeNet from CNN-2 (2 hidden layers) to CNN-5.
   c. Varying the #epochs used for training CNN classifier in 4-5 values, scuh as 1 epoch (which process the dataset only one pass), 2 epochs, 5 epochs, 10 epochs. Make sure to include the default one you used to produce the CNN classifier using default configurations in (2). Record the performance in training/testing time and accuracy in a table.

(4) **Outlier Test Scenario**. When performing testing on each of the 4 CNN classifiers you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 images that do not belong to the binary classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set.

(5) Optional tasks (only if you have more time and interest, you may choose to do c) or d) instead of both):

   a) Choose the one binary CNN classifier that has the highest test accuracy among the 4 you have.
   b) For this CNN classifier and its training dataset, you are asked to use 2x, 5x increase of your datasets. Then run test performance on your trained model
   c) Retain your CNN classifier with the training and test split on the dataset of 2x bigger, 5x bigger. Compare the performance of these 3 classifers in terms of training/testing time and accuracy in a table.
   d) For this CNN classifier and its training dataset, you are asked to evaluate the impact of different resolution scales of the input image set on the CNN classifier. You are recommended to use 3 different scales of resolutions, such as 28x28, 32x32, 64x64.

Deliverables:

(1) Provide URL of your open source code packages

(2) Example screen shots of your execution process/environments for CNN classifier training and testing scenarios.

(3) **Input Analysis**: *Use a table to report your training configuration parameters*:

a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
b. choose and show 5 sample images per class.
c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.
e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence. Usually it can also be found from some documentations for MNIST and CIFAR-10.

(4) **Output Analysis:** *Report the performance comparison and analysis of your 4 CNN classifiers*

a. You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (each is trained from one dataset configuration), in terms of training time, training accuracy, testing time and testing accuracy.
b. You are asked to record the trained model size in MB for each of the four classifiers in a table.
c. You are asked to provide a table of the same format but this time your test is performed on your outlier test dataset in requirement (4).
d. You are asked to make at least three observations from your experimental comparison of the 4 CNN classifiers.

Problem 2. Hand on experience with Deep Learning Framework(s)

This assignment gets you some hand-on experience with deep learning frameworks as a user or an application developer.

(1)Software download.
You may choose one of your favorite machine learning (ML) or deep learning (DL) framework for this homework.

The popular DL frameworks are TensorFlow (google), Caffe, Torch/PyTorch, OpenCV (Intel), CNTK (Microsoft).
https://www.tensorflow.org.
https://caffe.berkeleyvision.org

http://torch.ch or https://pytorch.org
https://docs.opencv.org/master/index.html
https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine
https://keras.io
You may also use Keras, a high-level neural networks API, written in **Python**, for this homework as it is capable of running on top of TensorFlow, CNTK, or Theano.

In the rest of the problem description, I will TensorFlow (TF) to describe the homework but you can replace TF with one of your favorite ML/DL frameworks.

(2) Dataset Preparation
You are asked to use at least two different image datasets for your K-class classification task, and K >= 10.

(i) Color rich and object rich images like photos of people and buildings together, such as CIFAR-10 (https://www.cs.toronto.edu/~kriz/cifar.html). If you choose K>10, then extract K classes of images from ImageNet (https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet), or Kaggle (https://www.kaggle.com/), or your favorite photo collections or from our own photo collections.

(ii) At minimum, you should have 10 images per class. Hint: Most laptop computers can handle training of CNN classifiers on 1000 images easily.

This programming homework provides multiple subproblems. You are required to choose one of them.

If you have not had any hand-on experience with any deep learning software framework, then you are recommended to choose the first option of this problem.

Problem 2.1 Hand-on Experience with Deep Learning Framework for Building a K-class Image Classifier

Requirement.

(1) Choose your favorite deep learning framework, say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(2) Using 80% of the images as the training dataset (say 20 pictures of cat and 20 pictures of dog) and 20% (5 pictures of cat and 5 pictures of dog) as the test dataset.

(3) You are asked to show how TensorFlow was used to train a K-class classification model, from input to training in CNN layers, to the output of the trained K-class model, including the storage size of the model in MB, the training accuracy and training time.

(4) Then you use your testing images to test your K-class CNN classifier trained by TensorFlow and report the average test accuracy and test time.

(5) **Outlier Test Scenario**. When performing testing on your K-class CNN classifier you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 or more images that do not belong to the K class classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set.

(6) Choose 3 new datasets from the public domain. Using TensorFlow to produce another 3 K-class CNN classifier. Measure the training and testing accuracy and time.

Hint: Here are some example datasets.

MNIST dataset. http://yann.lecun.com/exdb/mnist/.
USPS dataset. https://www.kaggle.com/bistaumanga/usps-dataset
Traffic Sign Recognition. https://www.kaggle.com/c/traffic-sign-recognition
LISA dataset: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
p2-trafficsigns. https://github.com/ vxy10/p2-TrafficSigns
The face database.
https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
You can also find most of them at
http://yanzhaowu.me/GTDLBench/datasets/.

Deliverable:

(1) provide URL of your open source code package and dataset download.
(2) Screen shots of your execution process/environments
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
  a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
  b. choose and show 2 sample images per class for all K classes in each of the four datasets.

c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)

d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.

e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence.

(4) **Output Analysis:** *Report the performance comparison and analysis of your K-class CNN classifier:*

a. You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (each is trained from one of the 4 datasets in terms of training time, training accuracy, testing time and testing accuracy.

b. You are asked to record the trained model size in MB for the 4 classifiers in the above table.

c. You are asked to add the test accuracy and time for your outliner dataset in requirement (5) by testing using all 4 CNN classifiers.

d. You are asked to make at least three observations from your experimental comparison of the 4 CNN classifiers.

Problem 2.2. Improving CNN Classifier Performance with Some Optimization Techniques

Requirement.

(1) Choose your favorite deep learning framework, say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(2) Using 80% of the images as the training dataset (say 20 pictures of cat and 20 pictures of dog) and 20% (5 pictures of cat and 5 pictures of dog) as the test dataset.

(3) You are asked to show how TensorFlow was used to train a K-class classification model, from input to training in CNN layers, to the output of

the trained K-class model, including the storage size of the model in MB, the training accuracy and training time.

(4) Then you use your testing images to test your K-class CNN classifier trained by TensorFlow and report the average test accuracy and test time.

(5) Design your own optimization, aiming to improve the performance of your K-class CNN classifier. Hint: you may employ SVM to the fully connected layer before output of the probability confidence vector, aiming to improve the performance of your K-class CNN classifier.

(6) Choose 1 new dataset from the public domain. Using TensorFlow to produce the K-class CNN classifier for this new dataset and also to train an optimized K-class CNN classifier on this new dataset. Measure the training and testing accuracy and time for both classifiers.

Hint: Here are some example datasets.

MNIST dataset. http://yann.lecun.com/exdb/mnist/.
USPS dataset. https://www.kaggle.com/bistaumanga/usps-dataset
Traffic Sign Recognition. https://www.kaggle.com/c/traffic-sign-recognition
LISA dataset: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
p2-trafficsigns. https://github.com/ vxy10/p2-TrafficSigns
The face database.
https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
You can also find most of them at
http://yanzhaowu.me/GTDLBench/datasets/.

Deliverable:

(1) provide URL of your open source code package and the two datasets download.
(2) Screen shots of your execution process/environments
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
   a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
   b. choose and show 2 sample images per class for all K classes in each of the two datasets.
   c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
   d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for

the configuration you use to train your CNN classifiers on the two different datasets.

e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence.

(4) **Output Analysis:** *Report the performance comparison and analysis of your K-class CNN classifier:*

a. You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (two are trained on each of the two datasets, with one optimized CNN classifier for each dataset), in terms of training time, training accuracy, testing time and testing accuracy.

b. You are asked to record the trained model size in MB for both classifiers in the above table for all 4 CNN classifiers

c. You are asked to make at least three observations from your experimental comparison of the 4 CNN cla

Problem 3: Using ML or Deep Learning Software frameworks to Solve a real world big data problem.

This problem is designed for those students who are familiar with ML and AI algorithms and may have prior hand on experiences with ML or DL software frameworks, such as Spark ML library, Hadoop Mahout, Scikit-learn (http://scikit-learn.org/), Weka (http://www.cs.waikato.ac.nz/ml/weka/) or R https://bigdata-madesimple.com/10-r-packages-machine-learning/.

You are asked to work on solving a data analytic problem using your favorite ML or DL software frameworks. You may choose your own dataset and the problem on mining your dataset or use the following problems and the associated datasets:

Problem 3.1 Mining the Kaggle million songs dataset

The million songs dataset is a big data mining challenge. One of the earlier Kaggle challenges. It contains (1) the full listening history for 1M users, (2) half of the listening history for 110K users (10K validation set, 100K test set). You are asked to predict the missing half.

http://www.kaggle.com/c/msdchallenge
Statistics of the dataset:

- Large Data Set
  - 1,019,318 users
  - 384,546 MSD songs
  - 48,373,586 (user, song, count)
- Kaggle Competition: offline evaluation

- Predict songs a user will listen
- Training: 1M user listening history
- Validation: 110K users

This is a well-known big data challenge and there are software postings in the Public domain. Feel free to use any of them you found helpful.

Deliverable:

(1) URL of the dataset
(2) The subset of the data you used in your program or model training and testing.
(3) The software packages and tools or library you use in your programs
(4) The open source code you leveraged in developing your MillionSongMiner.
(5) The experiments you conducted to report your mining accuracy, ideally over three different datasets extracted from the million-songs dataset in three different sizes (e.g., 2x, 5x, 10x of your initial smaller dataset, say 1000, or 10,000 songs.)
(6) Analysis of your hand-on experiences, with three lessons learned or three observations you wish to make.

Problem 3.2 Mining the White House visitor Log dataset

The White House Visitor Log dataset available at http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Released-0827.csv . The attributes in this dataset are described at: http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Key-1209.txt . A spreadsheet of the data can be found at: http://www.whitehouse.gov/briefing-room/disclosures/visitor-records .

You are required to write an efficient program (such as using MapReduce) to find the following information:

(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST, NAMEMID) to the White House.

(ii) The 10 most frequently visited people (visitee_namelast, visitee_namefirst) in the White House.

(iii) The 10 most frequent visitor-visitee combinations.

(iv) Some other interesting statistics that you can think of.

Throughout this programming assignment, do not limit your programs to run with a single design choice (such as one reduce task only).

Deliverable:

(1) URL of the dataset
(2) The software packages and tools or library you use in your programs
(3) The open source code you leveraged in developing your visitor log miner

(4) The experiments you conducted to report your mining accuracy and time, ideally over three different datasets in three different sizes (e.g., 2x, 5x, 10x of your initial dataset.)
(5) Analysis of your hand-on experiences, with three observations you wish to make.