~~~~~~~~~~~~~~Double-view CNN walkthrough~~~~~~~~~~~~~~~

The project is our research code and contains programs that are not fully cleaned up or documented.
Please quickly read through the code and fix the specified paths (e.g. finding variables like output_dir and change them with valid ones) for data input/output on your own. Consider placing all the generated files in a folder, like ./data/

Tutorials and useful references:
The following 3 tutorials build up the fundamentals very well towards a working CNN in TensorFlow. Please read the introductions and try to make the experiment code work.
Softmax classifier:
https://www.tensorflow.org/get_started/mnist/beginners
2-level ConvNet:
https://www.tensorflow.org/get_started/mnist/pros
CIFAR classification with ConvNet:
https://www.tensorflow.org/tutorials/deep_cnn

~~~~~~
TensorFlow API references:
https://www.tensorflow.org/api_docs/python/

module load anaconda3
1. Generate synthetic dataset:
cd ./SimulationCode/generators
python simulation.py

2. Generate Fourier-Bessel basis:
cd ./xray_learning
python run_fbb.py --action 0 >run_fbb_output

3. Compute Fourier-Bessel transform (FBT): (this is slow and it could take a day ~ a few days)
cd ./xray_learning
python fbb_enet.py >fbb_enet_output
========
Ln 16-23:
[1]output_dir: FBT output path
[2]SDH('...'): synthetic dataset path

4. Generate binary batches for training/testing:
(binary batches: we condense all the generated data in batches of raw data records for fast I/O in TensorFlow. Batch files are divided into fixed length byte intervals that save a (tags, image pixels, FBT) tuple each)
cd ./xray_learning
python convert_fbb.py >convert_fbb_output
========
output_dir: binary output path
[1]input_dir: path of FBT computed in step 3
[2]SDH('...'): synthetic dataset path
~~~~~~

5. Train and test the CNN

(hint: some settings are written in ./xray_learning/run_config.yml)
Monitor the training (learning) progress by the means of observing learning error (console output) and tensorboard. Training error will stop descending and stay (roughly) constant at one point. Force quit the learning (Ctrl-C / end the job) then. No need to wait for the preset training steps to finish.

Notes:

```
cd ./xray_learning
##Use batches 0-8 for training && use batch 9 for testing
##This is changed between training and testing by editing the
'run_config.yml' as displayed below
##Best to edit run_config.yml in terminal using commands below:
      vi run_config.yml
      x => to delete selected character
      i => to insert text
      [esc] :wq => exit editor, save edits, quit view
      [esc] :q => exit editor, quit view without saving
      [esc] :q! => exit editor, quit view without saving
      cat filename => view file content

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For training change run_config.yml:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
## real train
#num_examples: 2000
#record_paths: [../xray_data/real_binary/batch-0.bin]

## real test
#num_examples: 429
#record_paths: [../xray_data/real_binary/batch-1.bin]

#synthetic test using batches 0-8, saving batch-9 for testing
num_examples: 45000
record_paths: [../xray_data/fbb_output/batch-0.bin,
    ../xray_data/fbb_output/batch-1.bin,
    ../xray_data/fbb_output/batch-2.bin,
    ../xray_data/fbb_output/batch-3.bin,
    ../xray_data/fbb_output/batch-4.bin,
    ../xray_data/fbb_output/batch-5.bin,
    ../xray_data/fbb_output/batch-6.bin,
    ../xray_data/fbb_output/batch-7.bin,
    ../xray_data/fbb_output/batch-8.bin]

## synthetic test
#num_examples: 5000
#record_paths: [../xray_data/fbb_output/batch-9.bin]
train_dir: ../xray_data/fbb_output/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

~~~~~~

For testing change run_config.yml:

~~~~~~

```
## real train
#num_examples: 2000
#record_paths: [../xray_data/real_binary/batch-0.bin]

## real test
#num_examples: 429
#record_paths: [../xray_data/real_binary/batch-1.bin]

#synthetic test using batches 0-8, saving batch-9 for testing
#num_examples: 45000
#record_paths: [../xray_data/fbb_output/batch-0.bin,
#    ../xray_data/fbb_output/batch-1.bin,
#    ../xray_data/fbb_output/batch-2.bin,
#    ../xray_data/fbb_output/batch-3.bin,
#    ../xray_data/fbb_output/batch-4.bin,
#    ../xray_data/fbb_output/batch-5.bin,
#    ../xray_data/fbb_output/batch-6.bin,
#    ../xray_data/fbb_output/batch-7.bin,
#    ../xray_data/fbb_output/batch-8.bin]

## synthetic test
num_examples: 5000
record_paths: [../xray_data/fbb_output/batch-9.bin]
train_dir: ../xray_data/fbb_output/
```

~~~~~~

##The modules anaconda3 and cuda/8.0 is needed to be loaded before training and testing can be started. This is done by typing 'module load anaconda3 cuda/8.0' into the command prompt.

```
module load anaconda3 cuda/8.0
cd ./xray_learning
(Image CNN)
(training) python -m nn_fbbenet.train -a cnn
                    CHANGE RUN_CONFIG.YML For Testing
(testing) python -m nn_fbbenet.nn_eval -a cnn >test_cnn_output
                   CHANGE RUN_CONFIG.YML For Training
(Coef CNN)
(training) python -m nn_fbbenet.train -a fbb
                    CHANGE RUN_CONFIG.YML For Testing
(testing) python -m nn_fbbenet.nn_eval -a fbb >test_fbb_output
                   CHANGE RUN_CONFIG.YML For Training

(Double-view CNN) (must have learned image CNN and coef CNN checkpoints)
(training) python -m nn_fbbenet.train -a joint
                    CHANGE RUN_CONFIG.YML For Testing
(testing) python -m nn_fbbenet.nn_eval -a joint >test_joint_output
```

~~~~~~

Expected outcome:
Image CNN can achieve mean average precision of ~0.5-0.6 with training.
Coef CNN can outperform image CNN by ~0.1.
Joint double-view CNN should be slightly better than both.

Paths and files:
./SimulationCode/generators/

- Simulation.py
- Synthetic_data/ *generated from simulation.py
  - *_varied_sm/ ^
    - *.mat ^
    - analysis/ ^
      - tagimgs/ ^
      - thumbnails/ ^
        - *.jpg ^
      - results/ ^
        - *.xml ^
      - oned/ ^
        - *.npy ^

./xray_learning/

- run_fbb.py
- nn_fbbenet/
  - model.py
  - train.py
  - nn_eval.py
- fbb_enet.py
- convert_fbb.py

/xray_data/

- fbb_output/
  - fbb.npy *generated from run_fbb.py
  - nodefect_50k/
  - sed/
  - batch-*.bin *generated from convert_fbb.py
  - imagelist *generated from convert_fbb.py

Batch Script: run by typing into command prompt:
sbatch *filename*.sh

Note:

squeue => to check status of jobs
squeue -u *yourusername* => check status user jobs
squeue -j *jobnumber* => check status of a job
scancel -j *jobnumber* => cancel job
vi *filename* => view and edit script in terminal

```
#!/bin/bash
#SBATCH -N 1 ## number of nodes
#SBATCH -n 1 ## number processes
#SBATCH --gres=gpu:1 ## request nodes and 4 gpus per node
#SBATCH -A PQ0005 ## your account
#SBATCH --mail-type=BEGIN, END, FAIL ##notifies when job begin,end, fails
#SBATCH --mail-user=rlashley@bnl.gov #email for notification
#SBATCH -J train_cnn ## job name
#SBATCH -o train_cnn_output ## file to which stdout will write to
#SBATCH -e train_cnn_err ## file to which stderr will write to
#SBATCH -p long ## partition (queue) to use
#SBATCH -t 24:00:00 ## time requesting

# if your submitting shell did not do this you
# need do this in batch script
module load anaconda3 cuda/8.0

cd ./hpcgpfs01/scratch/rlashley/dtyu-sidl-5fafe18a286a/dtyu-sidl-
5fafe18a286a/xray_learning
#run your code

python -m nn_fbbenet.train -a cnn
```