CGAN Scripts:

Discriminator.py -> Contains the Discriminator model
model.py -> Contains the generator model
train.py -> To train the CGAN, run this script.

In this train.py script:
1. To normalize the images, can change the following parameters to get the intensity range of both input and gt to 0-1 range.
        max_im = 1
        max_gt = 1

2. Assign the hdf5 file containing the input and the gt images (All training and testing instances) -> img_dir = '*.h5'

3. Write train(80%) and test(20%) indices to a text file without overlapping. Give those two separate text files in the class HDF5Dataset:
if isTrain:
        fold_dir = "train.txt" #Text file with training indices
else:
        fold_dir = "test.txt". #Text file with testing indices

valid.py -> Save the output images of the saved best performing generator(i.e. saved in checkpoint folder) for the testing instances.
In this script img_dir = 'test.txt' contains the testing indices (same text file we gave in the train.py script).

FCN Scripts:

model.py -> Contains the model
train.py -> Training script

In this train.py script:
1. To normalize the images, can change the following parameters to get the intensity range of both input and gt to 0-1 range.
        max_im = 1
        max_gt = 1

2. Assign the hdf5 file containing the input and the gt images (All training and testing instances) -> img_dir = '*.h5'

3. Write train(80%) and test(20%) indices to a text file without overlapping. Give those two separate text files in the class HDF5Dataset:
if isTrain:
        fold_dir = "train.txt" #Text file with training indices
else:
        fold_dir = "test.txt". #Text file with testing indices

valid.py -> Save the output images of the saved best performing

generator(i.e. saved in checkpoint folder) for the testing instances.
In this script img_dir = 'test.txt' contains the testing indices (same text file we gave in the train.py script).
h5_dir = '*.h5' # contains the hdf5 file with the testing instances.
predict_path = 'Predicted_rmse/epoch_' + str(epochs) +'/' # Saves the output images in a separate location.

To train a model in the harvard cluster:

First, log in to the cluster via ssh.

Run a python script in server with GPU.

1. load Anaconda, cuda and cudnn
        module load Anaconda3/5.0.1-fasrc01
        module load cuda/9.0-fasrc02 cudnn/7.4.1.5_cuda9.0-fasrc01

2. create a new environment with the latest python3 and some dependencies needed by TensorFlow. (Required to be done only when you need a new environment)
        conda create -n tf1.12_cuda9 python=3.6 numpy six wheel

3. activate the conda environment. (That you created in step 2. When you are accessing the same environment again, you need to follow only step 1 and 3)
        source activate tf1.12_cuda9
4. use pip to install tensorflow and other required libraries such as pytorch
        pip install --upgrade tensorflow-gpu==1.12

To assign GPU:
        srun -p gpu --pty --mem 2000 -t 0-01:00 --gres=gpu:1 /bin/bash   (RAM 2000mb and time 1 hour, 1 GPU)

Finally go to the folder with the train script and run the code.
        python3 train.py