READ ME

Contributors:

Thomas Arrizza Sinan Karabocuoglu Misael Ortiz Gunwoo Yong

tarrizza@umich.edu skara@umich.edu misaelo@umich.edu gwyong@umich.edu

Format as follows:

\*Code Section\* - \*Relevant section in final paper\*:

* Novel code (what part we made or adapted as opposed to what was imported from a github or library)

############################################################################

## Dataset

Generate Dataset - Section 3.1:

* We made all the code except for the following
  + download\_images\_from\_geoJson mapillary http request functions based on: https://github.com/mapillary/mapillary-python-sdk
  + Alpharemover function and with\_ztransform\_preprocess function implementation from python library: <https://github.com/JohannesBuchner/imagehash>
* The results of running the code are three datasets that have been compressed into “Street View Dataset”, “Single City Dataset”, and “Street View Dataset - 85k”.

## GAN

Build Dataloaders for GAN - N/A

* Dataloader Coords2Images loosely inspired by PS5. Specific implementation of init and getitem are new

Build GAN Architectures - Section 3.2

* The generator and discriminator classes are adapted to fit our geocoordinate implementation and image size from https://github.com/aelnouby/Text-to-Image-Synthesis/blob/master/models/gan\_cls.py

Train GAN Function - Section 4.1

* Helper function based off problem set 6, modified for two column graph outputs
* Train function inspired by problem set 6 and train codeloss functions from Text-to-Image-Synthesis github. Specific training loop, loss functions, hyper parameters, and tracked loss metrics are modified to work with the GAN model.

https://github.com/aelnouby/Text-to-Image-Synthesis/blob/master/trainer.py

## 

## CNN Model

CNN Dataloader - N/A

* GeoTrainDataset is a simple dataloader where we wrote the init and getitem functions for pairing the street view images from our dataset to their respective coordinate pairs.

CNN Classifier - Section 3.3

* GeoClassifier module is written by us, however the architecture is created by merging AlexNet (except the DropOut layer and one of the Conv layers that has 384 channels) and the same convolution layers of the last 9 convolution layers of VGG-16 models. However, for the VGG-16 models we eliminated some of the duplicated convolutions, and ended up using only 4 convolution layers. Also, BatchNorm is added by us after getting inspired by the PS5.

Train CNN Function - Section 4.2

* The train\_model function is inherited from PS5 with minor adjustments to hyper parameters, variable names, and loss tracking to fit with new model
* All helper functions are new.
* The saved model states can be found compressed as “CNN\_Benchmark.pth” and “CNN\_Advanced\_from\_Scratch.pth”

Experimental CNN Model - Section 4.3

* The function “spatial\_pyramid\_pooling” is from the paper called *“A spatial layout and scale invariant feature representation for indoor scene classification“* that can be found in our references section in our final report.
  + However, the below lines are added by us in order to introduce randomness to the pseudo code proposed in the research paper.
  + block\_sizes = [0, 4, 8, 12, 16]
  + index = np.random.randint(0, 5)
  + number\_of\_blocks = 2\*\*block\_sizes[index]

* The function “spatial\_pyramid\_pooling” is written by us, however the logic is inspired by the paper called “*Spatial pyramid pooling in deep convolutional networks for visual recognition”* that can be found in our references section in our final report.
* The full implementation of the CNN with these functions added can be found compressed in “Experimental CNN Full Implementation”