# Train.py

This file contains most hyperparameters you will be changing. Custom dataset object handles getting batches and shuffling input data. Training process is straightforward, there are two plotting functions: plot\_reconstruction and plot\_sample. Plot\_reconstruction plots the one of the images from a batch along with its reconstruction after passing through the network. Plot\_sample plots a grid of images that were created by sampling from the latent spaces of the network – this function is currently a work in progress. Currently, checkpoints are saved through training and certain values are saved for tensorboard. To Use tensorboard, run “tensorboard --logdir <path to log file>” in console. It helps to only have 1 log file in the directory otherwise it will plot all of them.

# Anvae.py

This file contains the bulk of the network code. Network is comprised of different cells, each cell has their own layers. Separating like this makes it easier to make the residual normal distributions and forming the latent spaces. Then some of these cells are grouped into encoder, decoder, discriminator, to make training specific cells possible. There are some parameters you can change in this file that determine how many latent layers there are, how many nodes of cells you have per layer, how many cells per node, etc.

The training process:

1. Pass through the encoder, saving the cells and current s for the combiner cells.
2. As you pass through the decoder, combine with previously saved combiner cells to create the latent spaces. Look at NVAE paper for a figure describing the setup of the network.
3. Once the encoder, latent spaces, and decoder have been formed, begin updating the network. The method of training is the same as described in the AAE paper, autoencoder -> discriminator -> generator.

**Autoencoder Loss**

N is the batch size, x is the input image, is the reconstruction. Simple MSE loss, applied to encoder, decoder, and some image processing cells.

**Discriminator Loss**

This is adapted from the discriminator loss equation in the AAE paper. L is the number of latent layers, N is the batch size, is the discriminator function, are from the prior and are from the posterior. This is training the discriminator to better differentiate between samples from the prior distribution, in this case I have been using a normal distribution, and samples from the latent spaces.

**Generator Loss**

Same variables as discriminator loss, however are from the posterior. This is applied to the encoder cells, this is trying to train the encoder to create a latent space distribution more similar to the prior distribution, in this case I have been using a Normal distribution.

**Spectral** **Loss**

Taken from: Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. arXiv preprint arXiv:1705.10941, 2017. This was used in the NVAE paper to help stabilize training, converted from pytorch to TensorFlow from the NVAE official implementation.

1. Gradients are calculated using gradient tape, then applied using the optimizers for autoencoder, discriminator, and generator. Paper uses Adamax optimizer, but have not experimented with others.

# Cells.py

This file contains the code for all the cells, as well as the encoder, decoder, and discriminator classes. The layers inside each cell are taken from the NVAE paper and official implementation.

# Utils.py

Helper functions

# Dataset.py

Handles loading and managing the input files. Depending on what data you are using you may need to change some parameters in this file. Right now it expects inputs to be on size 28x28 and then scales it to 32x32 which is not necessary but is a nicer dimension when upscaling and downscaling the latent spaces. This file handles shuffling the data, turning the data into batches, and handling epochs.

AAE paper: <https://arxiv.org/abs/1511.05644>

AAE code reference: <https://github.com/conan7882/adversarial-autoencoders>

NVAE paper: <https://arxiv.org/abs/2007.03898>

NVAE implementation: <https://github.com/NVlabs/NVAE>

https://github.com/tensorflow/probability/releases/tag/v0.11.0-rc1