CSCE 580: Artificial Intelligence
Graduate Coding Homework: Unsupervised Learning
Due: 4/18/2022 at 11:59pm

Your code must run in order to receive credit.

While you are free to create other functions of your own, do not change the signature (including the name) of the functions provided in the file. Your code must accept the exact arguments and return exactly what is specified in the documentation.

Installation

We will be using the same `conda` environment as in Homework 0.

The entire GitHub repository should be downloaded again as changes were made to other files. You can download it with the green "Code" button and click "Download ZIP".

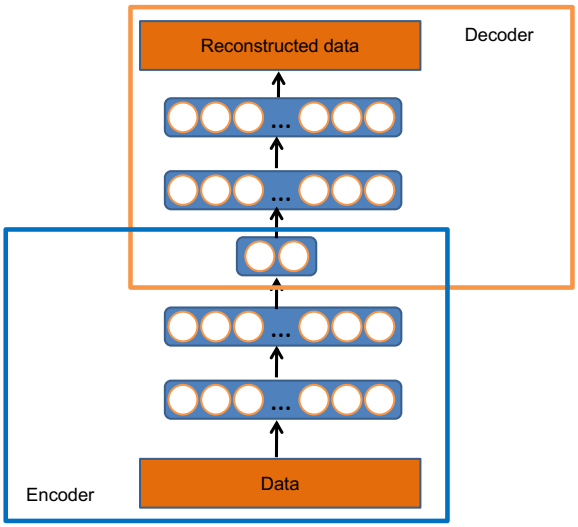
1 Unsupervised Learning with Autoencoders (100 pts)

You are an alien that has come to earth and wishes to make sense of strange symbols you have been seeing. These symbols just happen to be digits from the MNIST dataset. To try to find patters in the dataset, you want to reduce these 784 dimensional symbols to two dimensions so that you can visualize them. One way to do this using neural networks is with an *autoencoder*. An autoencoder, as shown in Figure 1a, is made up of an encoder and a decoder. The encoder maps input data to a *bottleneck layer* that has a dimensionality that is significantly smaller than that of the input data. In this case, the dimensionality of the bottleneck layer will be two. The decoder then attempts to map the bottleneck data back to the original input data. Therefore, for each example, the objective is to *reconstruct* the original input data. The bottleneck layer forces the neural network to learn the most relevant features in the data so that the reconstructed data is as close to the original as possible.

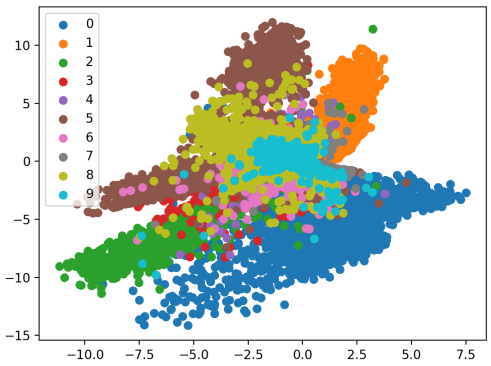
Do `python run_assignment_grad.py` to run the training of the neural network and return the encoder and decoder. After the autoencoder is trained the mean squared error will be calculated and the digits will be reduced to two dimensions using the encoder and will be plotted. An example of a plot is shown in Figure 1b. **Your neural network should obtain a mean squared error of 0.6 or less and must train in 5 minutes or less on a CPU.**

Hints: This is a regression task where we wish to minimize the mean squared error between the input and the output. See the PyTorch `MSELoss` documentation. To return the encoder and decoder separately, you should create the encoder and decoder networks separately. You can then train them together using the same optimizer by giving the optimizer the combined parameters. You can combine parameters like this: `list(encoder.parameters()) + list(decoder.parameters())`.

To read more about autoencoders, see the following paper by Hinton and Salakhutdinov. Note, this paper is using a different kind of neural network called a restricted Boltzmann machine.



(a) Autoencoder



(b) MNIST dataset reduced to two dimensions. Each point represents a digit.

What to Turn In

Turn in your implementation of `coding_hw/coding_hw_grad.py` to Blackboard.