Network Architecture

- Encoder: Three linear layers with ReLU activation functions. Last layer outputs 16-dimensional vector.
- Decoder: Three linear layers with ReLU activation functions. Last activation function is Tanh.

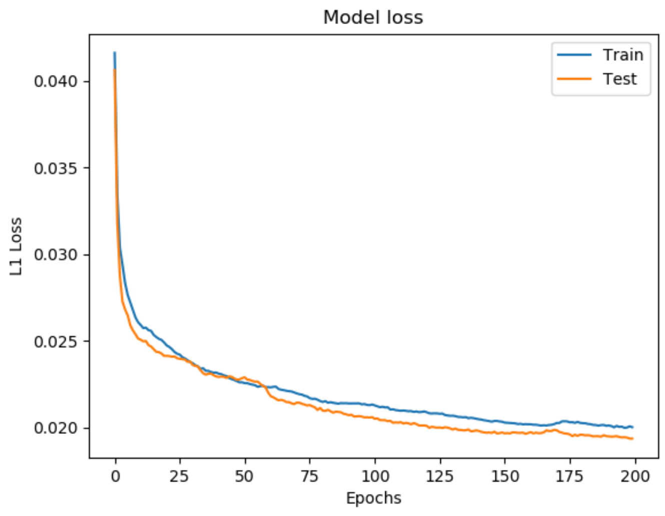

Figure 1: L1 Loss over number of epochs.

	Features							
Original	0.3784	0.4201	0.8061	0.6186	0.5947	0.5829	0.3933	0.3555
Recovered	0.1137	0.3739	0.7458	0.6246	0.5693	0.5709	0.3942	0.4198

Table 1: First 8 features of example with large loss of 1.14.

Performance The model is trained for 200 epochs. The L1 loss reached 0.020 on the training set and 0.019 on the test set (Figure 1). The mean for both bias vectors and weight matrices are around 0, but the values themselves are usually between -2 and 2 (Figure 2). The model trained on the GPU for about 5 minutes.

Table 1 shows an example that has a large loss. Each datapoint in the dataset has 33 dimensions, so only 10 features are shown. The recovered features are not exactly the same, but are pretty close. The only exception is the first feature, which differs by a lot for some reason.

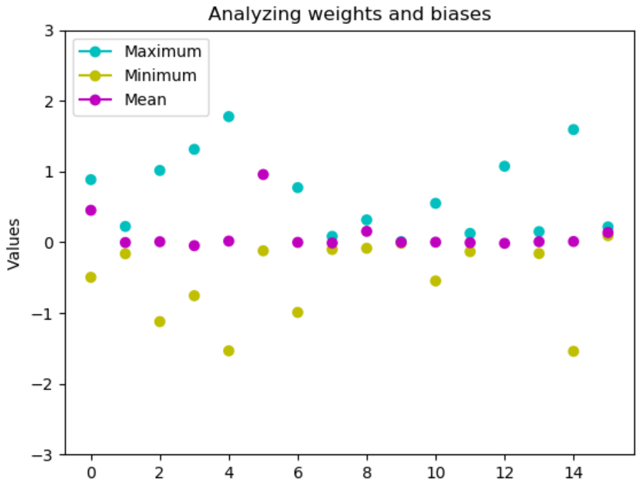

Figure 2: Final weights and biases of trained model. There are three colored points along each x-coordinate because these are the values associated with a parameter: maximum, minimum, and mean. These values indicate the largest, smallest, and average of values in the weight matrix/bias vector.

Explanation The autoencoder has two parts, an encoder and decoder. The encoder takes in a 33-dimensional vector and outputs a 16-dimensional vector. This vector is the autoencoding of the original input. Then, the decoder passes the 16-dimensional vector and outputs a 33-dimensional vector. The output should ideally be the same as the original input of the autoencoder. So, the autoencoder is essentially performing dimensionality reduction.

The autoencoder utilizes batch normalization to improve performance and stability. ReLU activation functions are used to introduce nonlinearity. For some reason, including a Tanh at the end of the decoder reduces loss. To mi-minize L1 loss, a grid search is used to determine the optimal hyperparameters. I tried different batch sizes (ranging from 100 to 1000) and learning rates (ranging from 0.1 to 0.0001). It has been found that a batch size of 500 and a learning rate of .001 can minize loss for this network. After 200 epochs, the loss does not reduce significantly, so the model stops training.