Research To Date

**What is a Neural Network?**

A Neural Network is a data model that attempts to learn a function . It is based off the idea that a function f(x) can be approximated by splitting f(x) into piece wise components, and iteratively updating the values of these piece wise functions till we find the approximation of f(x) with the least error. In this way, Neural Networks can “predict” results given a specific input.

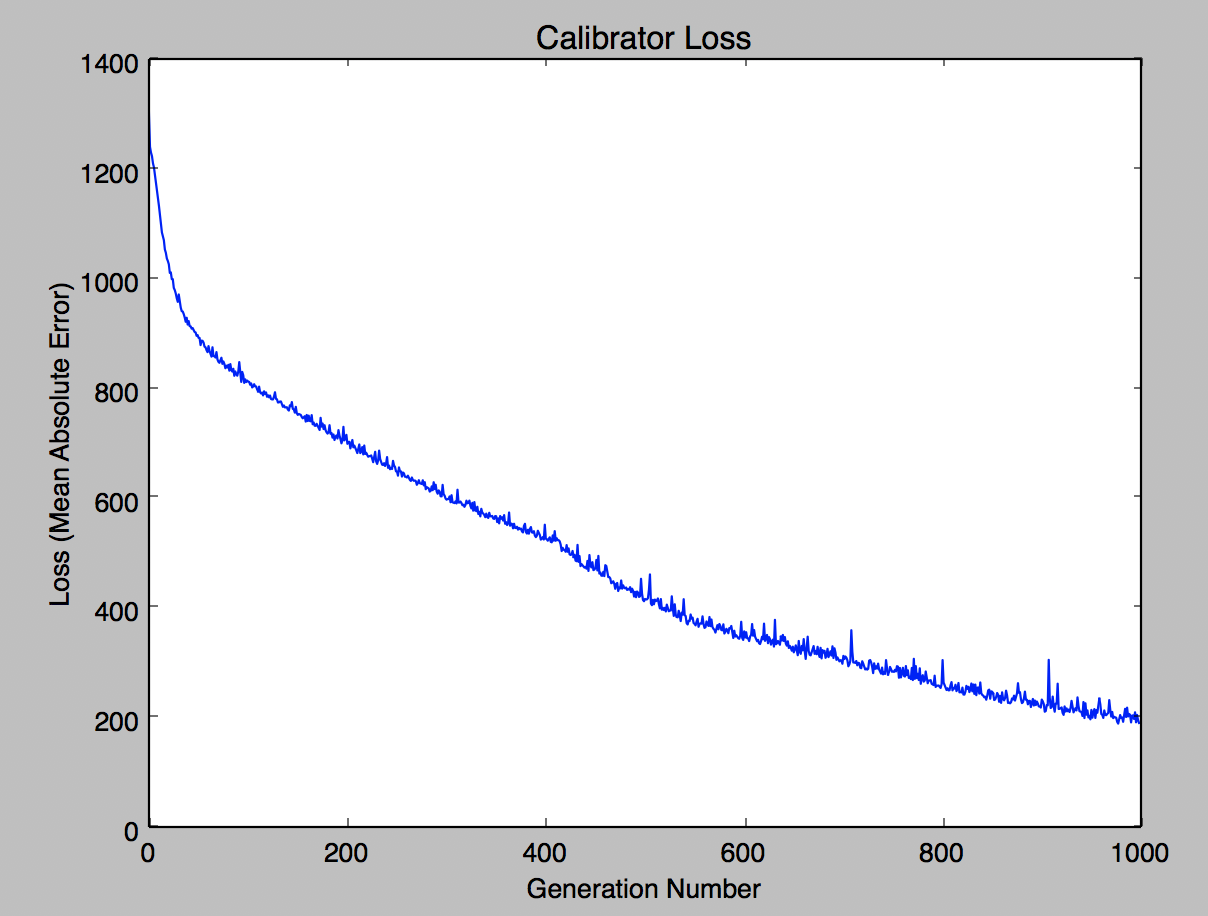
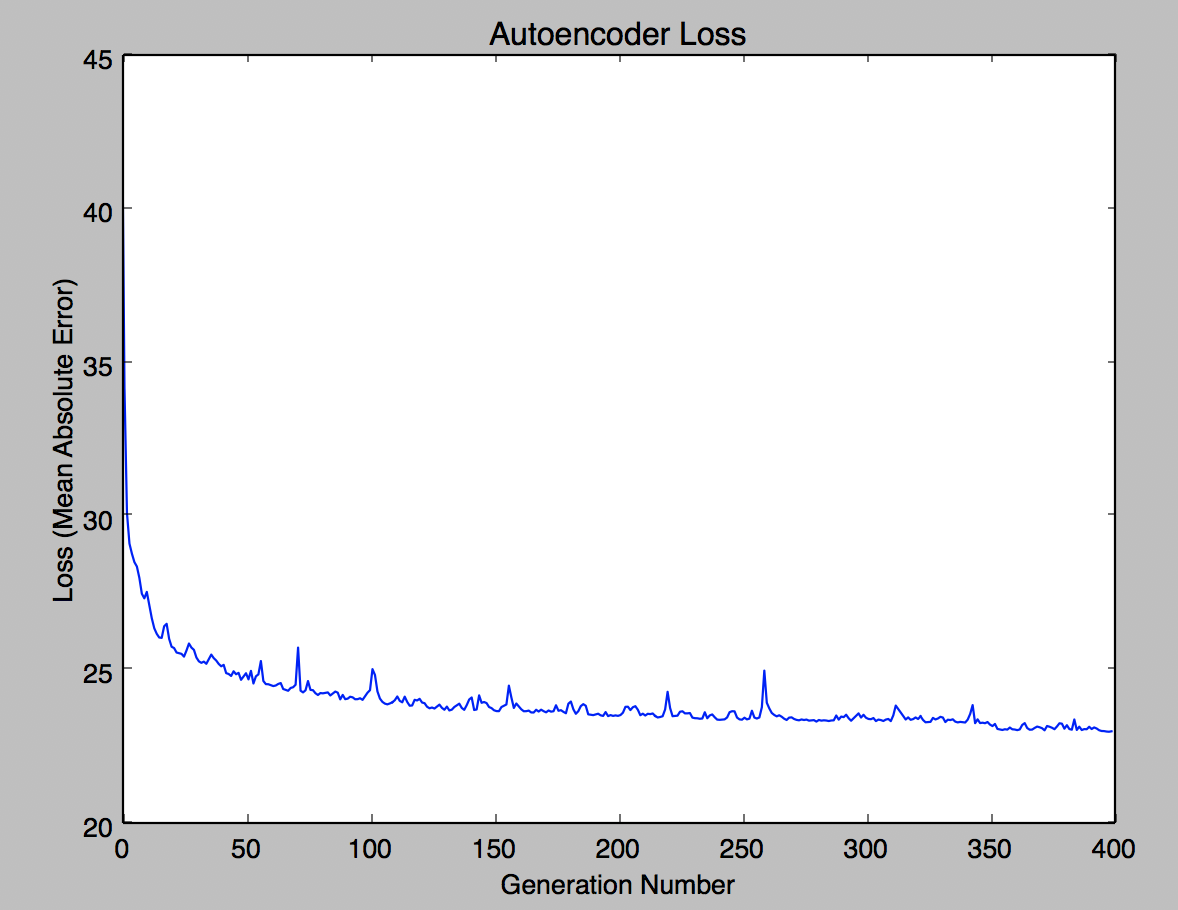
**Deep Portfolio Theory**

An autoencoder is a special type of neural network that attempts to recreate its input. In its most basic form, it consists of one input layer, one hidden layer, and one output layer. The input and output layers have the same number of neurons, while the connecting hidden layer has fewer neurons than the other 2 layers, so as to create an information bottleneck within the model to better compress data. We built an autoencoder that would attempt to recreate time series data of portfolios constructed from stocks of the same index. By training the autoencoder for multiple generations with individual portfolios, the autoencoder slowly learns a function of the market. After the network has been trained, we can pass each portfolio in through the network and use the difference of the stock’s predicted values minus the stock’s actual values as a proxy for the degree of communal information a stock/portfolio shares with the market. This provides a data-oriented approach to approximating a stock’s sensitivity to the greater market, commonly referred to as Beta.

After autoencoding the universe of stocks, we rank the stocks in order of their degree of communal information with the market. The portfolio traditionally consists of 10 stocks that share high communal information with the market and varying X stocks that share the least communal information with the market. We use the 10+X formula so that we can reduce redundancy in what the data tells us about the greater market. The stocks in between 10 and X will usually contain information that is already stored at the two tails of the distribution. In this way, we build a diverse portfolio that eliminates the idiosyncratic risks of each asset.

After we have our final portfolio of 10+X stocks, we can calibrate against a stock index to create an index tracking portfolio. Essentially, we calibrate a single time period of our portfolio against the same time period of the index we are trying to track. Once the Neural Network is fully trained, we can pass in the returns of our final portfolio to predict what the market will return.

For our research, we used data from the Fama-French Data Library. Specifically, we used monthly returns data of 100 portfolios over 60 years. Our autoencoder consisted of a single hidden layer with less than a quarter of the neurons in the input and output layers. Our Calibrator consisted of an input layer (10+X Neurons), a hidden layer ((10+x)/2 neurons), and an output layer consisting of a single neuron trained on the NASDAQ Market index.

****

**Predicting Portfolios**

Neural Networks can be used as a tool to aid investing by providing a data-justified guess as to what stock returns will be in the future. For example, let’s say that we had the returns of a single stock over 600 time periods. In order to train a neural network to predict the future based on the past, we have to properly split up our data set. We first have to choose a lookForwardPeriod (Y) and a lookBackwardPeriod (X), to establish how many time periods in the future you want to predict, and how many past time periods you want to base the prediction on, respectively. Using this value, you can create a dataset using a rolling time window, that will allow a neural network to calibrate the past X time periods to the next Y time periods, allowing neural networks to, in theory, “predict” the movements of financial returns.

We used 3 different models in order to predict future values. The first model was an Artificial Neural Network with RELU activation functions. The second model was also an Artificial Neural Network, but we used Leaky RELU activation functions, to avoid the “vanishing gradient” problem typically found in networks with RELU activation functions. The third model was a special type of Neural Network, called Long Short-Term Memory Networks (LSTM) whose neurons keep a state through training and feed their outputs back into their inputs. LSTM Networks are built specifically for sequential or time series data, because their ability to store a state within a neuron allows the network to capture features of the dataset over time.

|  |  |
| --- | --- |
| Normal ANN | 1.007756829 |
| Leaky ANN | 0.840557199 |
| LSTM | 0.427360863 |

**L2 Norm Ratios Over Test Set**