# **Description of the Structure**

1. **Model Structure and brief explanation on the structure of the model-**

We have used a feedforward autoencoder structure as shown below

50

100

100

200

200

Output (batch\_size,331)

Input (batch\_size,331)

MSE with Adam

We have used a bottle neck with 50 neurons and the input layer has 200 neurons. Our dataset has around ~10k data points and 331 features. As the amount of data available is relatively small the input layer neurons were kept at 200. It also suits our encoder design as the data dimension gets reduced from 331 to 200 in the first layer and data gets compressed.

The 50 neuron bottleneck is the compression point. Here latent features of our noisy data are generated in 50 dimensions and the green decoder tries to learn back the original data without the noise reconstructed from these latent features.

Dropouts are added between the layers and l2 regularization is used to help with overfitting.

1. **Performance Evaluation-**
   1. **Accuracy Evaluation-**

We report the train and test Mean Absolute Error(MAE), Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE)

**Accuracy evaluated on training+ validation data**

|  |  |
| --- | --- |
| Train Metric | Value |
| MAE | 0.11 |
| RMSE | 0.09 |
| MAPE | 0.16 |

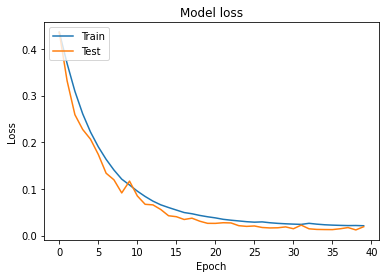
**Accuracy evaluated of test dataset assigned**

|  |  |
| --- | --- |
| Test Metric | Value |
| MAE | 0.09 |
| RMSE | 0.09 |
| MAPE | 0.10 |

We see that the MAPE on test dataset is lower than that of the training dataset. While this can point to overfitting but in this case it is due to the very low number of test data samples (20) compared to the training samples (10k).

* 1. **Training times, epochs, curves-**

We train the network with a high batch size of 2048 and let it run for 40 epochs. Early stopping on test loss is used to prevent overfitting. The Train and Test losses which are mean squared error are plotted over a function of epochs below-



As we can observe, a liberal patience parameter did not stop the training at 10 epochs as test loss decreased after increasing for a couple of epochs. The curves are smooth due to the high batch size being used.

The whole training took approximately 10 seconds

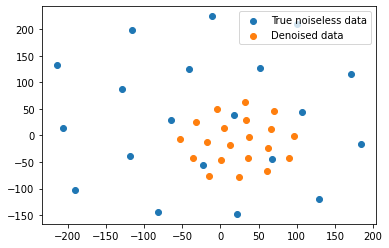
* 1. **Hyper parameters-**

The following hyperparameters are finally being used after some brief experimentation-

|  |  |
| --- | --- |
| Hyperparameter | Value |
| Layer Regularization | L2 with value=10-3 |
| Dropout value | 0.5 for 200 neurons, 0.4 for 100 neuron layer |
| Optimizer | Adam |
| Learning Rate | 0.003 |
| Loss Function | Mean Squared Error |
| Epochs | 40 |
| Batch size | 2048 |
| Early Stopping Criteria | Validation loss increasing over 7 epochs |

* 1. **Visualization of data-**

Each datapoint has 331 features. As 331 is a prime number we cannot visualize it as an image with pixel values. We try to visualize the 20 datapoint test noise less data and data denoised by the autoencoder in a 2d tsne plot below



We observe that the data points of the denoised data have a similar pattern to that of the true noiseless data, only that the distance of the datapoints from the centre is less indicating a slight underprediction of the values.

1. **Discussion and Conclusions-**

Autoencoders present an efficient way to denoise data. We were able to obtain reasonable MAPE values on the test dataset with a trained autoencoder. The advantages of autoencoders are as follows-

* + 1. Autoencoders are great at learning compressed representation of the input data when a lot of data is available and hence can be used for feature extraction.
    2. Autoencoders are unsupervised machine learning models and do not require any labels for feature generation.
    3. When used on a supervised task like image denoising with it is very effective and data efficient

The disadvantages are follows-

1. Autoencoders are lossy compressors. The decoder cannot reconstruct the original data due to reconstruction loss.
2. Autoencoders learns from as much data as possible. If there is little relevant data, it may not learn the relevant part and learn irrelevant information.
3. Reconstructing data from the lower dimensional features loses model interpretabilty