Assignment 2

Adnan Altukleh
Course: DV2586

1. Introduction

This study is about building an LSTM autoencoder model to detect anomalies in sensor measurements (temperature, humidity, and pressure).

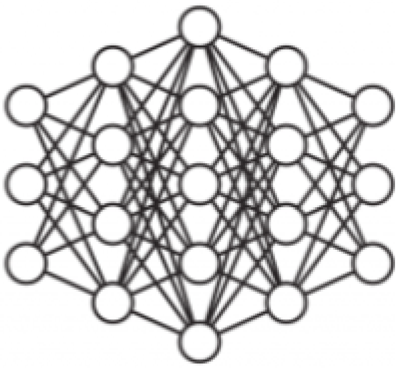
2. Data Preprocessing

At the start, checking if the data is clean is the first method to apply, which includes checking for duplicate and missing values etc. The data is clean and just needs to apply standardization which will enhance the model training stability and performance. But before standardizing the data, I had to split the data into train , test and validation, to prevent datalack in the model. Also the data class labels have been converted from categorical to numerical, finally preparing the data in sequences (number of samples, window size, number of features) to fit in the LSTM-autoencoder model.

3. Models Architecture

Each sensor has experimented with two different models; The first one is trained on just normal data where the second trained on both normal and not normal.

Deep overcomplete architecture is used in the first model.

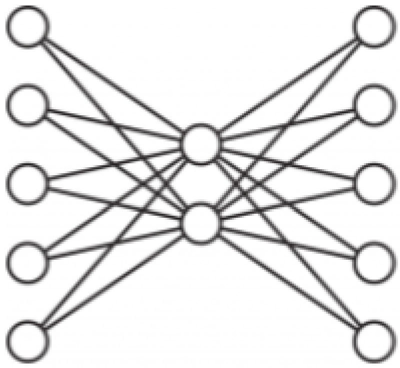


(d) Deep overcomplete

Where the first layer has 16 units and the next layer has 32 units, the next layer 64 units, the next layer has 128 and the final layer has 512.

Early stopping technique is used for preventing overfitting.

In the second model the architecture shallow undercomplete is used with 512 units.



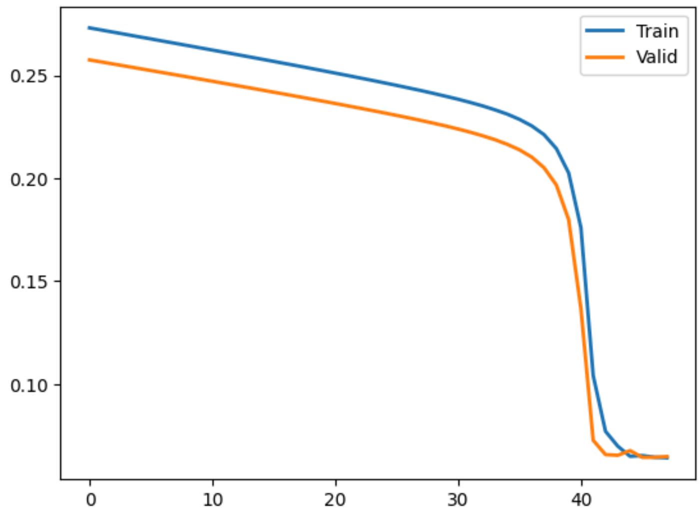
(a) Shallow undercomplete

4. Models Training

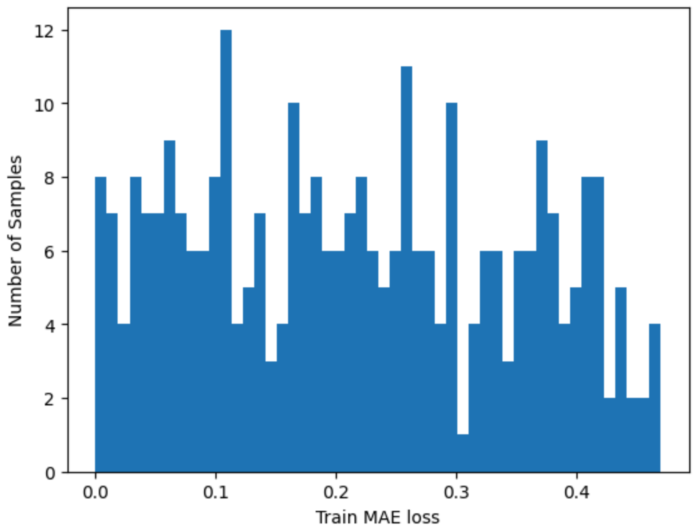
Both models were trained with a high number of epochs (400) and a small batch size (32). The Adam optimizer was used with a very small learning rate (0.0001) to enhance training results.The early stopping technique was employed to prevent overfitting.

Pressure sensor training process is shown below **were the other are in the notebook**

a) Training on just normal data for pressure sensor

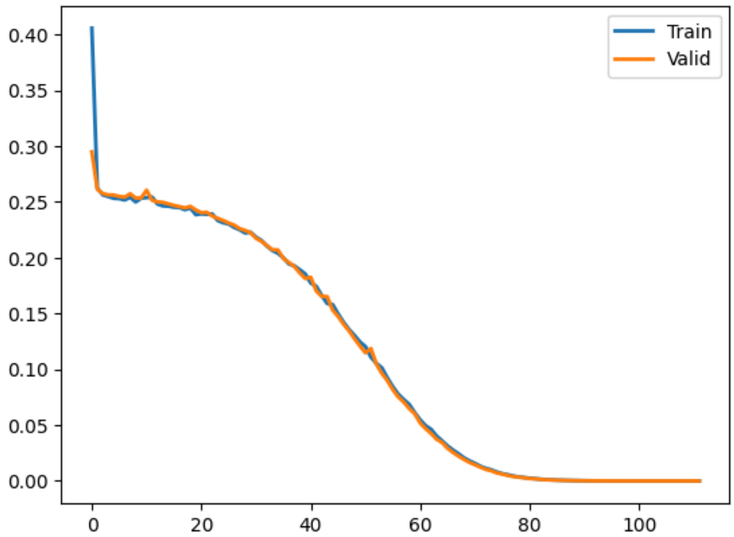


Training and validation loss in training process for pressure sensor

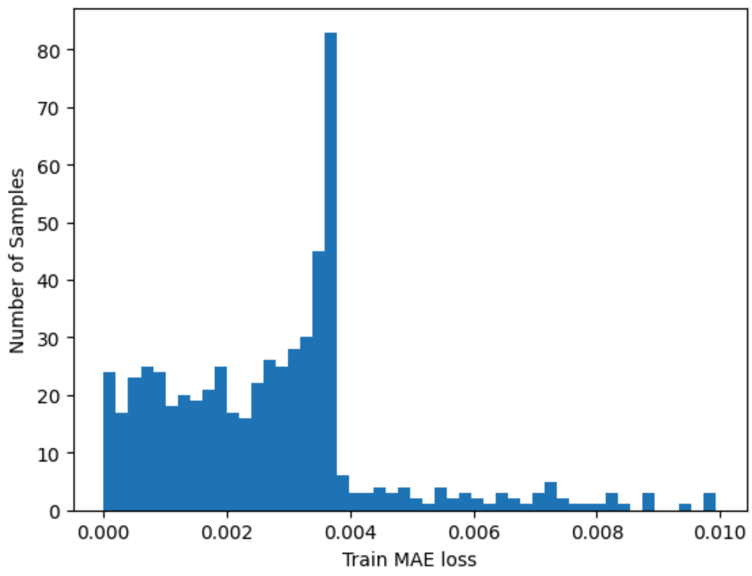


The threshold set to the max reconstruction error threshold: 0.4701170974099722

b) Training on both data normal and anomaly data for pressure sensor



Training and validation loss in training process for pressure sensor



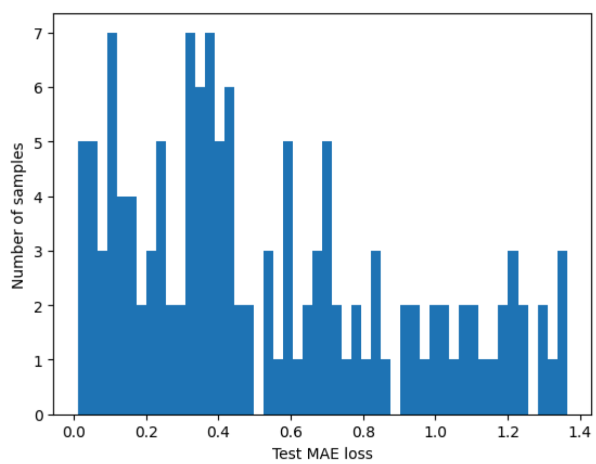
The threshold set to: 0.0037

5. Models Evaluating

To evaluate the models, a classification report is used to calculate precision, recall, and F1-score for each class, and the confusion matrix. The threshold is set based on the training mean loss, using the plot to determine the most suitable value.

6. Models Result

a) The result on test data for the first model (which is trained on just normal data) :
Mean loss plot, classifications report and confusion matrix. Pressure sensor testing process is shown below were **the others are in the notebook.**

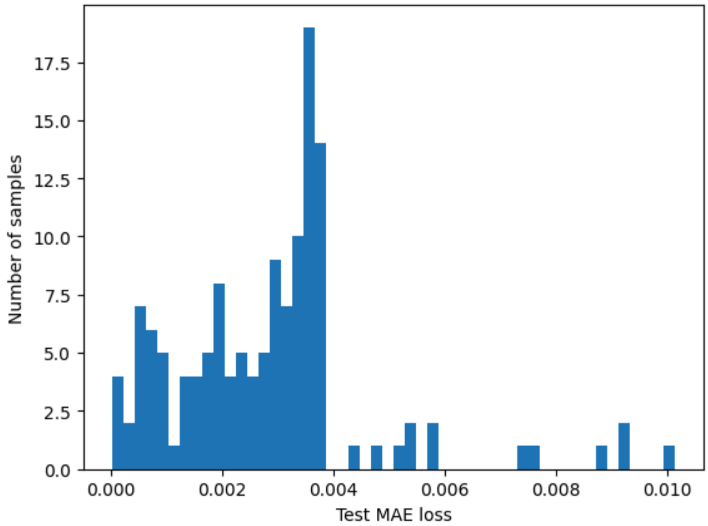


The mean loss on test data where threshold set to 0.47

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        75
           1       1.00      1.00      1.00        61

    accuracy                           1.00       136
   macro avg       1.00      1.00      1.00       136
weighted avg       1.00      1.00      1.00       136

[[75  0]
 [ 0 61]]
```

The classifications report and confusion matrix on test data where threshold set to 0.47

b) The result on test data for the second model: Mean loss plot, classifications report and confusion matrix



The mean loss on test data where threshold set to 0.004

```
              precision    recall  f1-score   support

           0       0.61      1.00      0.76        75
           1       1.00      0.21      0.35        61

    accuracy                           0.65       136
   macro avg       0.80      0.61      0.55       136
weighted avg       0.78      0.65      0.58       136

[[75  0]
 [48 13]]
```

The classifications report and confusion matrix on test data where threshold set to 0.004

7. Discussion

The performance difference between the two models and their architectures is significant. The first model, trained only on normal data with a deep architecture, achieved 100% accuracy due to the high quality and cleanliness of the data. In contrast, the second model, trained on both normal and nonnormal data, performed poorly compared to the first model. This discrepancy can be attributed to the simpler architecture of the second model and the need for hyperparameter tuning to optimize its performance.

8. Conclusion

In real-world scenarios, we do not have clearly defined classes between normal and nonnormal data as in the second model. The second model lacks the ability to identify normal data accurately due to its training process. To improve its performance, the second model requires a more complex architecture and deeper analysis. On the other hand, the first model achieved excellent results and performance due to the well-labeled classes and high data quality.