Faculty of Informatics and Computer Science

Artificial Intelligence

Time series and frequency-based model for anomaly fault detection for a desalination pump using machine and deep learning

By: Hussain Ghoneim

Supervised By

Prof. Andreas Pester

June 2022

Abstract

In Egypt, water scarcity is the major issue concerning the life of a nation that has survived for thousands of years. The Egyptian water per capita is less than 600 cubic meters per year, surpassing the water poverty limit which is 1,000 cubic meters per capita a year. The use of seawater desalination pumps employing Reverse Osmosis (RO) technology offers a solution to Egypt's water shortage challenge.

My graduation project will build up a model prototype that will detect anomalies without the need for human interventions and cut the costs of traveling and checking up on the desalination pumps. The system will be trained using time-series data that is granted to me by the faculty of Engineering at the British University in Egypt. The model will be able to classify the time series data, predict the label, and detect the anomalies from the time series using machine and deep learning techniques.

My methodology is that I have done a literature survey with a background in doing anomaly detection on real-life devices. Then starts the analysis process of the results that have come out from the literature survey the analysis process is very important to know how the writers have evaluated their models which model performs better when or where does it perform better than other models that main methods and how different they are. I have received two data sets from the faculty of engineering the first one is the normal data set which came from a desalination pump that does work in a normal condition and the second data set came from a desalination pump that does work in a cavitational condition and the goal is to detect the anomalies in cavitation data set by training on the normal data set. None of the data sets were labeled so that does make the project turns into unsupervised learning. I have done the EDA to the two data sets that I received and discovered some differences between the data sets for example the correlation of the features in the normal data set is more than the correlation of the features in the cavitation data set. Then I started to choose the main methods which were chosen according to the data set size, type, and recommendation. Then I started to choose an algorithm or a model for each method. The algorithms that are going to be used are the long short-term memory network LSTM, the Gated Recurrent Unit GRU, the Isolation Forest, and the one-class SVM. The one-class SVM approach uses the one-class classification method, the LSTM works well with the unbalanced data sets and detects anomalies by using the local and the global outlier then calculates the loss using the MAE, the Isolation Forest is the fastest between them and works by finding the anomalies by the patterns and these patterns are expected to be few and very different, and the GRU does works by discovering the correlations between among time and frequency sequence then use the Gaussian mixture to detect the anomalies. The models that were trained using the normal data set to learn it and to be able to predict the anomalies based on the threshold that the model/me chooses it. I have tested different types of methods and models and all the models were successful except the one-class SVM frequency-based model that has failed in the two

approaches the dynamic pressure and the vibration approach. This means that the one-class methods will not work as efficiently when using the frequency-based instead of the time series. So based on that discovery the anomaly detection models that I have created for this study except the one-class SVM frequency-based model could be applied to desalination pumps to have a field test. On the other side we can see that “Bi_GRU01 frequency for dynamic pressure approach” model achieved the best results and it's loss to time graph was clean without any noises. This model also had the Least MAE difference. So we need more study to know which performance measures indicated that this model will outperform the other models. Is it the MAE Cavitational loss or the MAE Difference loss. Based on these results, we could say that the dynamic pressure approach with the frequency-based data is the optimal approach for this case and that the “Bi_GRU01 frequency” model could be used as a benchmark for further studies.

Turnitin Report

ICSGraduationproject_Hussain170440

ORIGINALITY REPORT

37%	9%	14%	34%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to British University in Egypt Student Paper	33%
2	avendanoservices.com Internet Source	1%
3	A L Amutha, R Annie Uthra, J Preetha Roselyn, R Golda Brunet. "Anomaly detection in multivariate streaming PMU data using density estimation technique in wide area monitoring system", Expert Systems with Applications, 2021 Publication	1%
4	apps.dtic.mil Internet Source	<1%
5	ebin.pub Internet Source	<1%
6	www.csidesigns.com Internet Source	<1%
7	machinelearninginterview.com Internet Source	<1%

Acknowledge

Acknowledge anyone who has helped you in your work such as your supervisor, technical support staff, fellow students or external organisations. Acknowledge the source of any work that is not your own.

Supervised By: Prof.. Andreas Pester

Technical Supervised by: PhD. AmrAbdelkader

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

1.1 Overview

As the global population grows, rivers are dried due to rapid consumption and evaporation. Surface water is being evaporated at a rapid speed due to global warming. Which makes places the water scarcity issue on the top list of human major disasters in the 21st century. This places a challenge on the governments and the people of the global Earth. The fact that 71% of the total Earth's surface is covered by water gave a hint for a solution that may be the 21st century key for sustaining the human civilization to have a sight of the 22nd century. The solution is not hard as building a human colony on Mars and it's not simple as ending the second world war, but it's the only logical solution that can sustain human civilization by desalinating the seawater into drinking water. In Egypt, water scarcity is one of the major reasons for crippling all the national development plans. The growing population in Egypt with the shrinking amount of water resources caused a gradual decrease in the Egyptian water per capita, which is less than 600 cubic meters per year, surpassing the water poverty limit which is 1,000 cubic meters per capita a year. The use of seawater desalination pumps employing Reverse Osmosis (RO) technology offers a solution to Egypt's water shortage challenge.

1.2 Problem Statement

In this RO process, a High-Pressure Pump (HPP) raises the pressure of the saline feed water such that the desalination process takes place in semi-permeable membrane elements. HPP is the main element of any RO desalination plant which consumes at least 50% of the plant's electric energy. Moreover, the associated maintenance and spare parts and the employee who are being paid to monitor the pump's situation costs are very expensive. With the financial problems, there are also technical problems, there are more than six errors, and identifying the error that has caused the problem takes an amount of time and time is not that cheap to spend on diagnosing. To know which errors have caused the problem to fix it. So my model will cut off the monitoring budget and predict if this pump needs maintenance or not.

1.3 Scope and Objectives

My graduation project will build up a model prototype that will detect anomalies without the need for human interventions and cut the costs of traveling and checking up on the desalination pumps. The system will be trained using prototype time-series data that will be granted to me by the faculty of Engineering at the British University in Egypt. This prototype will classify the time series data, predict the label, and detect the anomalies from the time series using machine learning and deep learning techniques.

1.4 Report Organization (Structure)

The report is divided into seven sections. They are organized as the following, the first section is the introduction section the introduction to the project is like discussing the overview idea of the project the problem and the work methodology, and the objectives of the project. The second section is related to the study and analysis of the related work. The third section is where the discussion of the solution and the approaches that will be implemented is and how the system is designed to work. The fourth section is where the discussion of the implementation work is like discussing the dataset and how it looks like, the process of the preprocessing and visualization, the process of choosing the methods and approaches, and how these methods and approaches are implemented. The fifth section contains how the models were tested and evaluated. The sixth section is the discussion of the results. The seventh section is the conclusion of the work and the future works recommended for this project.

1.5 Work Methodology

The work methodology of this project was the following first I have done a literature survey with a background of doing anomaly detection on real-life devices, that because in real-life devices the process of detecting the anomalies differs from in the data sets like detecting fraud mails, etc. Then starts the analysis process of the results that have come out from the literature survey the analysis process is very important to know how the writers have evaluated their models which model performs better when or where does it perform better than other models that main methods and how different they are. Then I have received two data sets from the faculty of engineering the first one is the normal data set which came from a desalination pump that does works in a normal condition and the second data set came from a desalination pump that does works in a cavitational condition and the goal is to detect the anomalies in cavitation data set by training on the normal data set. None of the data sets were labeled so that does make the project turns into unsupervised learning. I have done the EDA to the two data sets that I received and discovered some differences between the data sets for example the correlation of the features in the normal data set is more than the correlation of the features in the cavitation data set. Then I started to choose the main methods which were chosen according to the data set size, type, and recommendation. Then I started to choose an algorithm or a model for each method. Then I have divided the work into four parts which are the required approaches by the faculty of engineering like the time/dynamic pressure, time/vibration, frequency/dynamic pressure, and frequency/vibration approach. Then I coded the models to this approach and then evaluated them based on the MAE loss and the number of anomalies detected and discovered that the main evaluation technique is by having a look at the distribution of the loss.

1.6 Work Plan (Gantt chart)

British University in Egypt
Faculty of Informatic and Computer Sciences
AI Specialization
Time series and frequency-based model for anomaly fault detection for a desalination pump using machine and deep learning
Supervised By PhD Andreas Pester
By Hussain Ghoneim 170440

	Start	End
Project	1/11/2021	13/6/2022

Tasks			1/11/2021	1/2/2021	20/2/2022	1/4/2022	20/4/2022	15/5/2022	1/6/2022	13/6/2022
Literature Review	1/11/2021	1/6/2021								
Data Preprocessing	1/2/2022	20/2/2022								
One class SVM	20/2/2022	1/4/2022								
Isolation Forest	1/4/2022	20/4/2022								
LSTM	20/4/2022	15/5/2022								
GRU	15/5/2022	1/6/2022								
Evaluation	1/6/2022	13/6/2022								

2 Related Work (State-of-The-Art)

2.1 Background

I have collected different papers that are interested in the same field of study as the idea of this project. The idea of this project is to use the machine and deep learning to detect anomalies in a time series real-time model. These papers address their problem and goals and then use different methods to solve different problems they face in their study. Each one of these papers contains a problem, a proposed solution, an experiment, and results. The main criteria for choosing these papers were they are a real-life anomaly detection they are all working in a multivariate time series.

2.2 Literature Survey

In [1], they have studied the Multivariate Time series Anomaly Detection and uncovered the problem of how the multivariate time series should have a different methodology in anomaly detection in comparison to the univariate time series. The occurrence of anomalies in multivariate time series data is often determined by multiple variables varying over time. Treating the multivariate time series as multiple univariate time series can't accurately locate the anomalies. So according to this finding, there is a need to propose a solution to this problem. The framework of the proposed solution consists of three core components. Which are the temporal convolution model, the graph attention model, and the threshold select model. In the first two models, we do process the data and then forecast it. Then get the Root Mean Squared Error RMSE between the forecasted value and the real value. The RMSE will be taken as the input to the threshold select model. If the input of the threshold select model exceeds the threshold value, then it will consider that an anomaly has occurred in the current moment. They have used three real-world datasets to verify the effectiveness of MTAD-TF, namely, the MSL (Mars Science Laboratory) rover, SMAP (Soil Moisture Active Passive) satellite, and SMD. To understand the noise tolerance of the model, they have carried out tests that were done with different noise levels. Five kinds of Gaussian white noise with the mean value of 0 and variance of (0.1, 0.2, 0.3, 0.4, 0.5) were added to the training set, respectively. The score of the original model without any processing had a value as high as 0.945

In [2], they have faced a problem in the process of turning their power grid system from traditional to a smart power grid that does connect. The problem was the new smart power grid contained potential risks and was more vulnerable to cyber-attacks. It was a critical problem concerning national security. So, there was a need to develop accurate and efficient anomaly detection methods to detect cyber-attacks and protect the smart grid. They have proposed an anomaly detection model for this problem, this model is based on the periodic extraction method of using a discrete Fourier transform to determine the sequence position of each element in the period by the periodic overlapping mapping. This would result in accurately describing the timing relationship between each network message. They have simulated a device failure in an industrial control environment, the time series model detects the anomaly element that does not appear in the period in which it should be in the period. The abnormality is detected and controlled this proves the validity of using the time series anomaly detection model in cyber security.

In [3], they faced the problem that while using the EEG to monitor the brain function of the patient, this monitoring produces a large amount of data. This data is coming from waveforms that in nature have high variability and were poorly defined. Which made simple linear detection models unable to detect the anomalies. Which has created a need for a model to better detect these anomalies. That was a critical clinical need. They have used Deep Belief Nets (DBNs) to build the model, a DBN is a multi-layer generative neural network that is known for its ability to model multivariate data and visualize learned features. Then they did compare the results of their DBN model to the SVM model. The result of this comparison was that the performance of both detectors is similar. The DBN outperforms the SVM by a small margin. But the DBN's threshold-based parameter search is much faster than the SVM's search parameter, which makes it easier for the DBN to find the optimal value in a limited time. This discovery suggests that the DBNs may be better suited for real-time anomaly detection applications such as multichannel EEG detection tasks.

In [4], they had faced the problem is that they have a need in Prognostic and Health Management PHM for an anomaly detection algorithm that can detect the outliers on the deep features of the data that its fluctuation amplitude changes with time. So, they have proposed a solution using the LSTM-GAN algorithm. This algorithm benefits from the LSTM capabilities in the processing of the time series data. It also benefits from the GAN capabilities in extracting data features and building the data model. They have carried out an experimental simulation on the following two sets of data the ECG and the NYC taxi. They concluded that the framework of the LSTM-GAN algorithm is suitable for time series data processing. But the error integration factor and the abnormal threshold of the LSTM-GAN algorithm need to be tuned according to specific data sets.

2.3 Analysis of the Related Work

MTAD-TF: Multivariate Time Series Anomaly Detection Using the Combination of Temporal Pattern and Feature Pattern

In [1], they have studied the Multivariate Time series Anomaly Detection and uncovered the problem of how the multivariate time series should have a different methodology in anomaly detection in comparison to the univariate time series. Because of the existing difference in dimensionality between the univariate time series and the multivariate time series, which is the univariate time series anomaly detection is an anomaly detection carried out on time series with only one variable varying over time, which means one dimension of data. But with the multivariate time series anomaly detection, is a multi-dimensional data set. The occurrence of anomalies in multivariate time series data is often determined by multiple variables varying over time. Treating the multivariate time series as multiple univariate time series can't accurately locate the anomalies. So according to this finding, there is a need to propose a solution to this problem.

The framework of the proposed solution by [1], consists of three core components. Which are the temporal convolution model, the graph attention model, and the threshold select model. After the data is processed and the multivariate time series is normalized, the next stage will be forecasting the model. Which contains two parts the temporal convolution model and the graph attention network. In the temporal convolution model, the model captures temporal patterns by a multiscale one-dimensional convolution. Which can find the temporal pattern with such other multiple periods. What has been captured in the temporal convolution model will be the input of the graph attention network. The graph attention network is going to assign a different weight based on the characteristics of its neighbor nodes for each node. The graph attention network is a network that is used to inter the relations between variables to forecast the data in the data sets. In the first two models we process the data and then forecast it and get the Root Mean Squared Error "RMSE" between the forecasted value and the real value, RMSE will be the input to the threshold select model. If the input of the threshold select model exceeds the threshold value, then it will consider that an anomaly has occurred in the current moment.

According to [1] they have used three real-world datasets to verify the effectiveness of MTAD-TF, namely, MSL (Mars Science Laboratory) rover, SMAP (Soil Moisture Active Passive) satellite, and SMD.

Dataset	MSL	SMAP	SMD
No. of attributes	55	25	38
Training subset size	58317	135183	708405
Testing subset size	73729	427617	708420
Anomaly rate (%)	10.72	13.13	4.16
Variables information	Telemetry data: computational, radiation, temperature, power, activities, etc.		CPU load, network usage, memory usage, etc.

Figure 1 Datasets information

To understand the noise tolerance of the model, they have carried out tests that were done with different noise levels. Five kinds of Gaussian white noise with the mean value of 0 and variance of (0.1, 0.2, 0.3, 0.4, 0.5) were added to the training set, respectively. As the variance of Gaussian noise increases, the data shows a downward trend. However, it also indicates that the model is still not robust enough to the addition of noise.

Dataset	MSL			SMAP			SMD		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
LSTM-NDT	0.5944	0.5374	0.5640	0.8965	0.8846	0.8905	0.5684	0.6438	0.6037
LSTM-VAE	0.5257	0.9546	0.6780	0.8551	0.6366	0.7298	0.7922	0.7075	0.7842
DAGMM	0.5412	0.9934	0.7007	0.5845	0.9058	0.7105	0.5835	0.9042	0.7093
OmniAnomaly	0.8867	0.9117	0.8989	0.7416	0.9776	0.8434	0.8334	0.9449	0.8857
MTAD-TF	0.9043	0.8988	0.9015	0.9779	0.8192	0.8916	0.9045	0.9048	0.8940

Figure 2 performance of our model and baselines

The score of the original model without any processing had a value as high as 0.945. The score of the noise with different variance: the value was stable at 0.917 from the 0.010 noise to the 0.040 noise after it decreased to 0.912 and continued to decrease. The score of the model without 1D convolution in data pre-processing as well as added noises, stayed at 0.914 until it reaches the 0.030 noise after it decreased to 0.89 and continued to decrease

Anomaly detection for power grid based on time series model

In [2], they have faced a problem in the process of turning their power grid system from traditional to a smart power grid that does connect. The problem was the new smart power grid contained potential risks and was more vulnerable to cyber-attacks. It was a critical problem concerning national security. So, there was a need to develop accurate and efficient anomaly detection methods to detect cyber-attacks and protect the smart grid.

In [2], they have proposed an anomaly detection model for this problem, this model is based on the periodic extraction method of using a discrete Fourier transform to determine the sequence position of each element in the period by the periodic overlapping mapping. This would result in accurately describing the timing relationship between each network message. Then they have performed element signal sampling which will be performed for the network flow. The sampled elements will be for the element set. Then they have performed a discrete Fourier transform on each element in the element set. Then transform the time domain signal to a frequency domain signal which is the spectral map of the model. Due to network delay or other issues, the position of each period element after mapping to a single period may be biased. To unbiased the period, they have performed aggregation on the fragment elements to a single period. If some elements are in the adjacent time fragments, they will be aggregated into the most occurring time fragments. After the aggregation operation, the elements in the time fragments that count less than the threshold value will be filtered out.

They have performed a Fourier transform on the network traffic to get the candidate period of each element. The position of each element in the period is determined by periodic overlap mapping. To ensure the accuracy of the model during the implementation process, they have divided the single period into multiple fragments and used None to indicate an empty element in the fragment.

They have simulated a device failure in an industrial control environment, the time series model detects the anomaly element that does not appear in the period in which it should be in the period. The abnormality is detected and controlled this proves the validity of using the time series anomaly detection model in cyber security.

A Novel LSTM-GAN Algorithm for Time Series Anomaly Detection

Prognostic and Health Management (PHM) has been widely applied in many fields and has made great achievements in system health management and performance monitoring. According to [4], they say that they have found a problem, saying in which the pre-used algorithms were only successful in detecting the shallow level anomalies, but they weren't able to detect outliers on the deep feature of time series data.

In [4], they have proposed a solution using the LSTM-GAN algorithm. This algorithm benefits from the LSTM capabilities in the processing of the time series data. It also benefits from the GAN capabilities in extracting data features and building the data model. When the built model output of the testing data exceeds the threshold, then it will consider that an anomaly has occurred in the current moment. In the structure of the LSTM-GAN algorithm, the LeakyRelu layer is aimed at learning the inverse gradient of the neurons of the convolutional one dimension, and the LSTM layers that are used to extract the data temporal features. The batch normalization layer keeps the input of each layer of the network which means the value of each variable has the same distribution over the structure of the network. The dropout layer helps to prevent the training model from overfitting.

In [4] they have carried out an experimental simulation on the following two sets of data the ECG and the NYC taxi. In the ECG, each data has 96 variables, and the data is divided between normal heartbeat and myocardial infraction anomaly. In the ECG also the percentage of the anomalous data is 36%. In the NYC taxi data set, each data has 48 variables, and the data is divided into normal classes where the passenger flow will be normal and the anomalous classes where the passenger flow will be either higher or lower. In the NYC taxi data set, the percentage of anomalous data is 8.3%.

Algorithms	Precision	Recall	F1	Accuracy
Isolation Forest	1.0000	0.1111	0.2000	0.7200
Local Outlier Factor	0.8889	0.1127	0.2000	0.7100
One-Class SVM	0.7222	0.3250	0.4483	0.8000
Gaussian Statistical Model	0.3600	1.0000	0.5294	0.3600
LSTM-GAN	0.7429	0.3210	0.4483	0.8100

Figure 3 Anomaly evaluation indicators of algorithms on ECG dataset

The proposed LSTM-GAN algorithm, which combines the advantage of LSTM in processing time-series data and the advantage of GAN in extracting data depth features, can achieve superior performance compared to other anomaly detection methods. They concluded that the framework of the LSTM-GAN algorithm is suitable for time series data processing. But the error integration factor and the abnormal threshold of the LSTM-GAN algorithm need to be tuned according to specific data sets.

Semi-Supervised Anomaly Detection for EEG Waveforms Using Deep Belief Nets

Neurophysiological clinical monitoring uses electroencephalography (EEG) to assess brain function and state in critically ill patients. Such monitoring often produces very large amounts of data. Even in patients with severe brain pathology, a large majority of the waveforms are still considered background or normal. The variability and poorly defined nature of these signals as well as that of background EEG require an ideal detector that would be capable of both detecting the anomalies and visualizing what separates these waveforms from the background.

In [3] they have used Deep Belief Nets (DBNs), a DBN is a multi-layer generative neural network that is known for its ability to model multivariate data and visualize learned features. They have also considered a DBN composed of logistic Restricted Boltzmann Machines (RBMs) with symmetric weights between binary visible units and binary hidden units as well as biases to the hidden layer and visible layer. They have compared the DBN anomaly detector to another anomaly detector, the one-class Support Vector Machine (SVM). To assess the performance of the detectors they use the F1 measure that incorporates both the detector precision and the detector recall because they usually improve at the expense of each other.

According to [3] both the DBN and the SVM models were trained on the same set of 500,000 unlabelled samples. The RBM layers of the DBN were first individually trained using 1-step Contrastive Divergence. The RBM and the other layers in the DBN joined together. Then they were trained with backpropagation, minimizing the log loss of the reconstruction. The reason they have chosen the DBN for anomaly detection is that the DBN in the training set will see none of the anomalous values. So, in the testing set, it will be easier for the DBN to identify them as anomalous values since they were not existing in the training set.

The results according to [3], the performance of both the DBN and the SVM detectors is similar. The DBN outperforms the SVM by a small margin. The SVM had a higher recall of 0.6529 in comparison to the DBN recall score of 0.5514. But in the precision, the DBN had a higher value of 0.4175 in comparison to the SVM which had a value of 0.3307. The only conclusion that they had drawn from this experiment is that the DBN's threshold-based parameter search is much faster than the SVM's search, making it easier to find an optimal value in a limited amount of time. This observation suggests that DBNs may be better suited for real-time detection applications involving large quantities of data, such as multichannel EEG detection tasks. They concluded the DBN outperforms the SVM in anomaly detection in a semi-supervised environment by using the F1 performance metric score. Also, the DBN has other advantages in the capability of handling and processing large data sets. From these advantages the DBN's training isn't affected by the percentage of the anomalies unlike the SVM, The

DBN’s reconstruction of the RMSE measures the depth of the anomaly, which makes the DBN capable of visualizing the anomalies in large data sets without any constraints. The DBN's role is not just detecting the anomalies, but it also has a role in visualizing the anomalies that they detect.

3 Proposed solution

3.1 Solution Methodology

My proposed methodology for determining the best approach, model, and method to detect the anomalies in the cavitation centrifugal desalination pumps. Is by comparing the results of the models using the MAE loss. The models that will be chosen will be tested by using their ability in the unsupervised learning of a dynamic dataset. The chosen models can be categorized into different methods which are one class classification method, finding local and global outliers method, patterns method, and correlation sequence with Gaussian mixture technique to detect anomalies. The algorithms that are going to be used to detect the anomalies in the dynamic unsupervised data set are the long short-term memory network LSTM, the Gated Recurrent Unit GRU, the Isolation Forest, and the one-class SVM.

One class SVM

The one-class SVM is a variation of the SVM that can be used in unsupervised learning to detect anomalies. The one-class SVM uses the one-class classification method it does work by finding the hyper-plane that separates the data given from the trained data

Isolation Forest

The Isolation Forest is similar to random forests, and it's an ensemble of binary decision trees. In Isolation Forest, the input data is processed in a tree-structured based on randomly selected features. If the sample of the input went deeper in the branches it means that it is less than the threshold value in the feature. The more deep the sample goes in the branches the less anomaly it is. This step is recursively repeated until it reaches the max depth determined or when the data point is completely isolated from the other points. The isolation forest is the fastest between them and works by finding the anomalies in the patterns and these patterns are expected to be few and very different.

LSTM

GRU

GRU is one of the important variants of recurrent neural networks and it's considered an upgrade to LSTM. GRU is simpler, similar, and faster than the LSTM networks in the implementation of models. A GRU has two gates which are the update and the reset gates. The update gate works in the long-term memory and it is responsible for defining the size of the previous data that is needed to pass to the next state which makes the model avoid the issue of vanishing gradient because this allows the model to copy the data that it needs from the past. The reset gate works with the short-term memory it's duty is to determine if the previous data needs to be neglected or not and it also decides if the previous cell state is important or not.

GRU auto encoders do work by discovering the correlations between time or frequency sequence and then using the Gaussian mixture to calculate the reconstruction loss and to detect the anomalies a threshold should be determined based on the loss density diagram.

3.2 Functional/ Non-functional Requirements

Non-Functional Requirements:

- Each model should be trained in a period that doesn’t exceed twelve hours.
- Each model should achieve an accuracy of more than 90% in detecting anomalies.
- Each model could be deployed on a 32KB memory device
- Each model could be deployed and run efficiently on a 3 KB RAM

Functional Requirements:

- Each model should be approved by the technical supervisor from the engineering faculty.
- Each model should be able to work on different data sets than the given ones in training.
- Each model should have the capability to be applied on an Arduino UNO or any similar device.
- Each model should be capable of detecting anomalies.
- Each model should be able to label the data if it’s an anomaly or normal

3.3 Design / Simulation set up

This project was implemented by using a Jupyter notebook that works with python code. I have received two data sets a normal and a cavitational data set. I have done EDA on the data sets to understand how the cavitational data set differs from the normal data set. Then I preprocessed the data sets and added a missing feature the data that the engineering faculty has required. I have also discovered some differences between the data sets for example the correlation of the features in the normal data set is more than the correlation of the features in the cavitation data set. Then I started to choose the main methods which were chosen according to the data set size, type, and recommendation. Then I started to choose an algorithm or a model for each method. The algorithms that are going to be used are the long short-term memory network LSTM, the Gated Recurrent Unit GRU, the Isolation Forest, and the one-class SVM. The one-class SVM approach uses the one-class classification method, the LSTM works well with the unbalanced data sets and detects anomalies by using the local and the global outlier then calculates the loss using the MAE, the Isolation Forest is the fastest between them and works by finding the anomalies by the patterns and these patterns are expected to be few and very different, and the GRU does works by discovering the correlations between among time and frequency sequence then use the Gaussian mixture to detect the anomalies. I have tried these models on the data set on time series-based data and frequency-based data sets. This model has detected the anomalies by training using the normal data set to learn the normal data and to detect anomalies on the cavitational data set by calculating the loss difference between the normal and the cavitational.

The model was evaluated by using the MAE loss, the number of anomalies they have detected, and the loss to time graph.

4 Implementation

4.1 Dataset

I have received two different data sets from the faculty of Engineering at The British University in Egypt one of them is normal data that has come from the desalination pump in a normal condition and the other is cavitation data that has come from the desalination pump in a cavitation condition. A cavitation condition means that this pump rapidly creates and collapses air bubbles in the fluid when the air bubbles collapse it creates shock waves that occur repeatedly when the pressure is changing. This repetition of the shock waves does erode the components of the desalination pumps.

Figure 4 Pump Impeller Showing Cavitation Damage

Figure 5 Cavitation Damage on Pump Casing

The desalination pumps are designed to work with a full flowing water supply, but in some cases, the flooded inlet is not that powerful to maintain the pressure required to prevent cavitation.

As you can see below the received dataset is in figure 6 and figure 7. The dataset received contains different features for different approaches. There are two main approaches the Dynamic Pressure approach and the Vibration approach.

The dynamic pressure approach calculates the speed of the fluid flow, and it does work in detecting the anomalies by measuring the flow rate and the differences between it so it can calculate the change in the speed of the fluid flow which increases when the bubble in the cavitation desalination pump explodes because of the shock wave of the bubble explosion.

The vibration approach detects anomalies using vibration sensors that are placed in different parts of the tank this sensor detects the direction of the water. There were two sensors placed that detect the horizontal movement and the other that detects the vertical movement.

Time: time calculated in seconds

P out bar: Pressure coming out of the bar

P in bar: Pressure getting in the bar

Q L/M: Flow rate

Vibration 1 & Vibration 2: X and y-axis vibration sensor data

Dynamic Pressure: Dynamic Pressure sensor data

```
In [2]: Normal= pd.read_csv("CentrifugalPumpNormalCondition.csv")
        Normal
```

Out[2]:

	Time	P out bar	P in bar	Q L/M	Vibration 1	Vibration 2	Dynamic Pressure
0	0.000000	1.161289	-0.084085	49.443383	0.897711	0.374315	-0.002651
1	0.000098	NaN	NaN	NaN	-0.612374	1.983866	-0.004067
2	0.000195	NaN	NaN	NaN	-0.699269	2.101856	-0.003631
3	0.000293	NaN	NaN	NaN	1.121377	1.053456	-0.010969
4	0.000391	NaN	NaN	NaN	-0.672813	-2.743689	-0.002324
...	...	...	...	...	...	...	...
919995	89.843262	NaN	NaN	NaN	-1.833754	-1.940224	0.018307
919996	89.843359	NaN	NaN	NaN	1.000499	1.311315	0.026480
919997	89.843457	NaN	NaN	NaN	-0.302599	0.097853	0.018162
919998	89.843555	NaN	NaN	NaN	-1.227691	3.129577	0.021976
919999	89.843652	NaN	NaN	NaN	1.409612	2.741606	0.025899

920000 rows × 7 columns

Figure 6 Normal dataset

```
In [2]: Cavitation= pd.read_csv("CentrifugalPumpCavitationCondition.csv")
        Cavitation
```

Out[2]:

	0	0.165204	-0.616328	27.839285	0.283178	-0.124765	0.02441
0	0.000098	0.3718	-0.616328	27.839285	-0.120759	-0.894264	0.018997
1	0.000195	NaN	NaN	NaN	0.331279	0.257127	0.016745
2	0.000293	NaN	NaN	NaN	0.509145	-0.061347	0.022920
3	0.000391	NaN	NaN	NaN	0.419741	-1.516285	0.010825
4	0.000488	NaN	NaN	NaN	0.969809	-0.061529	0.020487
...	...	...	...	...	...	...	...
1048570	102.399512	NaN	NaN	NaN	-0.415425	-0.193316	0.019324
1048571	102.399609	NaN	NaN	NaN	-0.075482	0.310897	0.009372
1048572	102.399707	NaN	NaN	NaN	0.161987	0.578513	0.013077
1048573	102.399805	NaN	NaN	NaN	-0.611381	0.048815	0.018344
1048574	102.399902	NaN	NaN	NaN	-0.320950	-0.081989	0.009118

1048575 rows × 7 columns

Figure 7 Cavitation dataset

4.2 Pre-processing & Visualization

Normal Dataset

First, as you can see in figure 6 there were several null values in the following features P out bar, P in bar, and Q L/M. When I did contact the faculty of engineering for that they replied and told me that the sensor does record at a different time rate than the other sensors so you can fill up between the numbers with the same number until the number change.

So, I wanted to see if there are other null values in the other features but there were no null values in the other features.

```
In [5]:  Normal.isnull().sum()

Out[5]: Time                      0
        P out bar            919817
        P in bar             919817
        Q  L/M               919817
        Vibration 1               0
        Vibration 2               0
        Dynamic Pressure          0
        dtype: int64
```

Figure 8 null values in Normal dataset

When I did contact the faculty of engineering for that they replied and told me that the sensor does record at a different time rate than the other sensors so you can fill up between the numbers with the same number until the number change.

```
In [6]:  Normal.replace(r'^\s*$', np.nan, regex=True, inplace=True)
         Normal = Normal.ffill()
         Normal

Out[6]:
```

	Time	P out bar	P in bar	Q L/M	Vibration 1	Vibration 2	Dynamic Pressure
0	0.000000	1.161289	-0.084085	49.443383	0.897711	0.374315	-0.002651
1	0.000098	1.161289	-0.084085	49.443383	-0.612374	1.983866	-0.004067
2	0.000195	1.161289	-0.084085	49.443383	-0.699269	2.101856	-0.003631
3	0.000293	1.161289	-0.084085	49.443383	1.121377	1.053456	-0.010969
4	0.000391	1.161289	-0.084085	49.443383	-0.672813	-2.743689	-0.002324
...	...	...	...	...	...	...	...
919995	89.843262	0.593152	-0.090887	49.371344	-1.833754	-1.940224	0.018307
919996	89.843359	0.593152	-0.090887	49.371344	1.000499	1.311315	0.026480
919997	89.843457	0.593152	-0.090887	49.371344	-0.302599	0.097853	0.018162
919998	89.843555	0.593152	-0.090887	49.371344	-1.227691	3.129577	0.021976
919999	89.843652	0.593152	-0.090887	49.371344	1.409612	2.741606	0.025899

920000 rows × 7 columns

Figure 9 Normal dataset after fixing the null issue

After fixing the null issue I added a new feature which is the pressure in the head that is equal to pressure out bar – pressure in the bar

$$H(i) = Poutbar(i) - Pinbar(i)$$

Equation 1 Pressure Head equation

```
In [42]: Normal
```

Out[42]:

	Time	P out bar	P in bar	Q L/M	Vibration 1	Vibration 2	Dynamic Pressure	H
0	0.000000	1.161289	-0.084085	49.443383	0.897711	0.374315	-0.002651	1.245374
1	0.000098	1.161289	-0.084085	49.443383	-0.612374	1.983866	-0.004067	1.245374
2	0.000195	1.161289	-0.084085	49.443383	-0.699269	2.101856	-0.003631	1.245374
3	0.000293	1.161289	-0.084085	49.443383	1.121377	1.053456	-0.010969	1.245374
4	0.000391	1.161289	-0.084085	49.443383	-0.672813	-2.743689	-0.002324	1.245374
...	...	...	...	...	...	...	...	...
919995	89.843262	0.593152	-0.090887	49.371344	-1.833754	-1.940224	0.018307	0.684039
919996	89.843359	0.593152	-0.090887	49.371344	1.000499	1.311315	0.026480	0.684039
919997	89.843457	0.593152	-0.090887	49.371344	-0.302599	0.097853	0.018162	0.684039
919998	89.843555	0.593152	-0.090887	49.371344	-1.227691	3.129577	0.021976	0.684039
919999	89.843652	0.593152	-0.090887	49.371344	1.409612	2.741606	0.025899	0.684039

920000 rows × 8 columns

Figure 10 Normal data set after adding pressure head

Then I used the box plot statistical method to find out the amount of the outliers in the normal dataset. The box plot statistical method works by finding the median feature of each feature and then multiplying it by 1.5 for the upper border of the plot and -1.5 for the lower border of the box. To make it clear I used a standard scalar that removes the mean and scales each feature to unit variance.

```
In [16]: fig = plt.figure(figsize =(20, 5))
         Normalx.boxplot()
         plt.show()
```

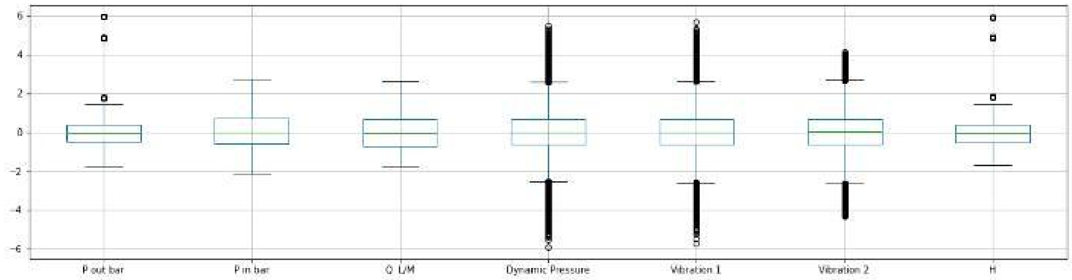

Figure 11 Normal data set box plot

Then I did get the correlation between the features in the normal dataset

```
In [17]: dataplot = sb.heatmap(Normalx.corr(), cmap="YlGnBu")
```

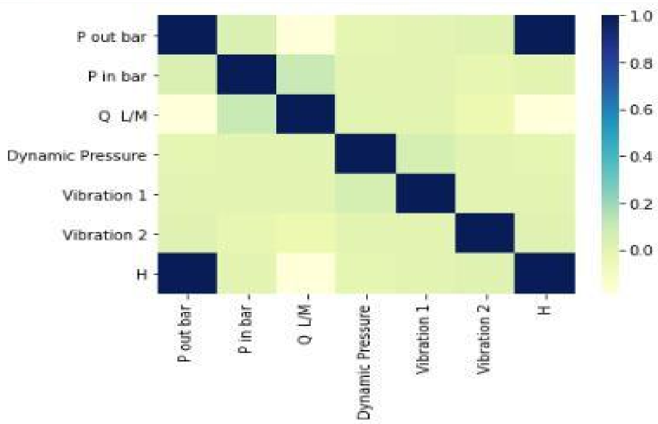

Figure 12 Normal data set correlation between features

As you can see there is a relationship between P out bar and P in bar and H and between P in bar and Q L/M which was expected.

Then I did get the density of each feature and displayed graphs to make it easier to learn how the features correlate with each other using a pair plot then I made a graph of each feature with time to see the changes that do happen and how smooth each feature.

Then I did make two data sets for each approach one for the dynamic pressure approach and the other for the vibration approach.

Cavitation dataset

First, as you can see in figure 7 there were several null values in the following features P out bar, P in bar, and Q L/M. When I did contact the faculty of engineering for that they replied and told me that the sensor does record at a different time rate than the other sensors so you can fill up between the numbers with the same number until the number change.

So, I wanted to see if there are other null values in the other features but there were no null values in the other features.

```
In [10]: Cavitation.isnull().sum()

Out[10]: Time                      0
         P out bar           1048366
         P in bar            1048366
         Q  L/M              1048366
         Vibration 1               0
         Vibration 2               0
         Dynamic Pressure          0
         dtype: int64
```

Figure 13 null values in the Cavitation dataset

When I did contact the faculty of engineering for that they replied and told me that the sensor does record at a different time rate than the other sensors so you can fill up between the numbers with the same number until the number change.

```
In [11]: Cavitation.replace(r'^\s*$', np.nan, regex=True, inplace=True)
         Cavitation = Cavitation.ffill()
         Cavitation

Out[11]:
```

	Time	P out bar	P in bar	Q L/M	Vibration 1	Vibration 2	Dynamic Pressure
0	0.000000	0.165204	-0.616328	27.839285	0.283178	-0.124765	0.024410
1	0.000098	0.371800	-0.616328	27.839285	-0.120759	-0.894264	0.018997
2	0.000195	0.371800	-0.616328	27.839285	0.331279	0.257127	0.016745
3	0.000293	0.371800	-0.616328	27.839285	0.509145	-0.061347	0.022920
4	0.000391	0.371800	-0.616328	27.839285	0.419741	-1.516285	0.010825
...	...	...	...	...	...	...	...
1048571	102.399512	0.215623	-0.624159	27.789128	-0.415425	-0.193316	0.019324
1048572	102.399609	0.215623	-0.624159	27.789128	-0.075482	0.310897	0.009372
1048573	102.399707	0.215623	-0.624159	27.789128	0.161987	0.578513	0.013077
1048574	102.399805	0.215623	-0.624159	27.789128	-0.611381	0.048815	0.018344
1048575	102.399902	0.215623	-0.624159	27.789128	-0.320950	-0.081989	0.009118

1048576 rows × 7 columns

Figure 14 Cavitation dataset after fixing the null issue

After fixing the null issue I added a new feature which is the pressure in the head that is equal to pressure out bar – pressure in the bar

```
In [14]: Cavitation
```

Out[14]:

	Time	P out bar	P in bar	Q L/M	Vibration 1	Vibration 2	Dynamic Pressure	H
0	0.000000	0.165204	-0.616328	27.839285	0.283178	-0.124765	0.024410	0.781532
1	0.000098	0.371800	-0.616328	27.839285	-0.120759	-0.894264	0.018997	0.988128
2	0.000195	0.371800	-0.616328	27.839285	0.331279	0.257127	0.016745	0.988128
3	0.000293	0.371800	-0.616328	27.839285	0.509145	-0.061347	0.022920	0.988128
4	0.000391	0.371800	-0.616328	27.839285	0.419741	-1.516285	0.010825	0.988128
...	...	...	...	...	...	...	...	...
1048571	102.399512	0.215623	-0.624159	27.789128	-0.415425	-0.193316	0.019324	0.839782
1048572	102.399609	0.215623	-0.624159	27.789128	-0.075482	0.310897	0.009372	0.839782
1048573	102.399707	0.215623	-0.624159	27.789128	0.161987	0.578513	0.013077	0.839782
1048574	102.399805	0.215623	-0.624159	27.789128	-0.611381	0.048815	0.018344	0.839782
1048575	102.399902	0.215623	-0.624159	27.789128	-0.320950	-0.081989	0.009118	0.839782

1048576 rows × 8 columns

Figure 15 Cavitation data set after adding pressure head

Then I used the box plot statistical method to find out the amount of the outliers in the normal dataset. The box plot statistical method works by finding the median feature of each feature and then multiplying it by 1.5 for the upper border of the plot and -1.5 for the lower border of the box. To make it clear I used a standard scalar that removes the mean and scales each feature to unit variance.

```
In [20]: fig = plt.figure(figsize =(20, 5))
         Cavitationx.boxplot()
         plt.show()
```

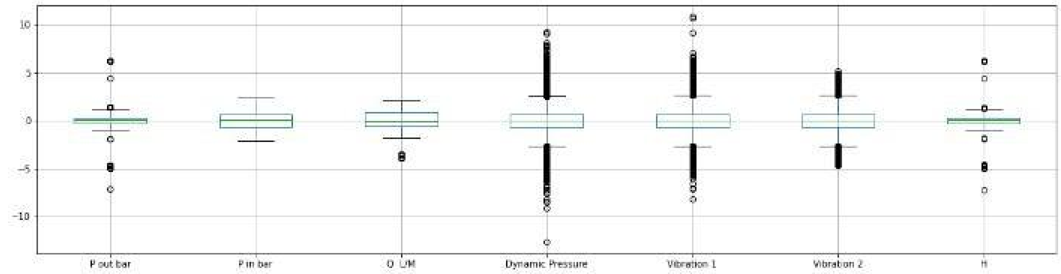

Figure 16 Cavitation data set box plot

Then I did get the correlation between the features in the normal dataset

```
In [21]: dataplot = sb.heatmap(Cavitationx.corr(), cmap="YlGnBu")
```

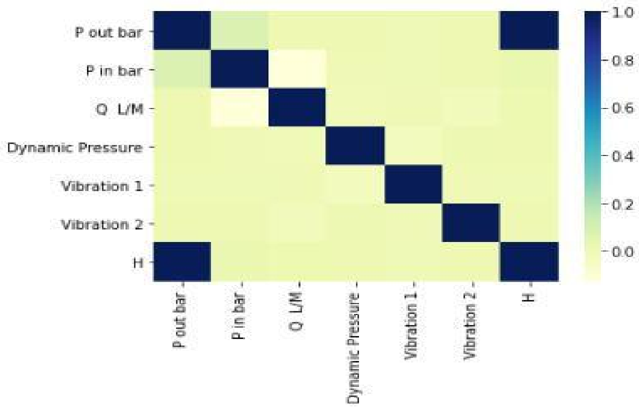

Figure 17 Cavitation data set correlation between features

As you can see there is a relationship between P out bar and P in bar and H and there are no relation between P in bar and Q L/M which was not expected and that makes difference between the normal data set and the cavitation data set.
Then I did get the density of each feature and displayed graphs to make it easier to learn how the features correlate with each other using a pair plot then I made a graph of each feature with time to see the changes that do happen and how smooth each feature.
Then I did make two datasets for each approach one for the dynamic pressure approach and the other for the vibration approach.

4.3 Methods

As mentioned before we do have four main methods that were recommended to be used to detect anomalies in unsupervised data, one-class classification, prediction error, pattern technique, and the Correlation among sequences with the Gaussian mixture technique.

One class classification

The first method is the one class classification 'OCC ' method, this method works by training the model on the normal data and then predicting whether the new data is normal or anomaly. The OCC method is used in binary imbalanced data where the model is only required to detect if the data is an anomaly or normal which is suitable in the case of this project. This method works well when the normal data has no pattern or sequence this allows the OCC to ignore the task of discriminating between the normal and the anomaly and let it focus on the task of deviation from the normal or what is expected. It is appropriate to use this method when there where no examples of anomaly cases that can be learned before training the model. The downside of this method is that if the training data contains outliers those outliers will not be detected as anomalies and will be discarded from training.

Prediction error

The second method is prediction error, this method works in predicting a sequence of the data based on what it has learned in the training since the targeted anomalies are not available in the normal dataset and only occur in the cavitation data set and we do train the model only by using the normal data set. Then it does get the loss using the MAE

$$MAE = \frac{\sum_{i=1}^{n} | y(i) - x(i) |}{n}$$

Equation 2 Mean absolute error equation

Then by using the MAE we can select a threshold that will help the model to be able to differentiate between normal and the anomaly sequences by using the loss metric.

Pattern technique

The third method is the pattern technique, this method works in finding the patterns in the data and do expect the anomaly pattern to be very different from the data it has learned and expects them to be few and not invasive so it does detect the least number of anomalies.

Correlation among sequences with Gaussian mixture technique

The fourth method is Correlation among sequences with the Gaussian mixture technique, this method mines the data dependencies in the sequence and among different sensors. This method is better at discovering the correlations among the sequence data better than any other method. Then it uses a mixture of Gaussians this mixture could learn complex information hidden and do better approximations of the original data distribution.

4.4 Time-based models

1. One class SVM algorithm

To apply the OCC method I have used the One-class SVM algorithm, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1 based on this function.

$$x(i) = \frac{(x(i) - x.\min)}{(x.\max - x(i))}$$

Equation 3 Normalization equation

Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. I have divided the normal data sets into training and testing so the model will be trained using the normal dataset so it can learn the normal patterns in the data and be able to detect the anomalies. After training the model I used the model to detect the anomalies in the cavitation data set. Then I visualized where did the anomaly occur by using the pressure head to time graph, Dynamic pressure to time graph, and the flow rate to time graph.

2. LSTM models

To apply the prediction error method I have used the LSTM, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I added time back to the data sets. Then I used the normal data in the training of the model and used the cavitation data in the testing. Then I changed the dimension of the training and testing data to meet the requirements of the data dimension of the LSTM. Then I built an auto encoder LSTM model. I have designed the model to be an auto encoder to get the reconstruction error to evaluate it using threshold criteria to predict the anomalies by looking at the trend of error in the normal and the cavitation data sets.

There are 3 different models of LSTM the LSTM01, LSTM02, and Bi_LSTM01.

LSTM01

```
In [132]: #Starting the LSTM model
          model = keras.Sequential()

          model.add(layers.LSTM(25, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.LSTM(5, activation='relu', return_sequences=False))
          model.add(layers.BatchNormalization())

          model.add(layers.RepeatVector(features))

          model.add(layers.LSTM(5, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.LSTM(25, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

          model.compile(optimizer='adam', loss='mae')
```

Figure 18 LSTM01 model for dynamic pressure approach

```
In [70]: #Starting the LSTM model
         model = keras.Sequential()

         model.add(layers.LSTM(36, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.LSTM(6, activation='relu', return_sequences=False))
         model.add(layers.BatchNormalization())

         model.add(layers.RepeatVector(features))

         model.add(layers.LSTM(6, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.LSTM(36, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

         model.compile(optimizer='adam', loss='mae')
```

Figure 19 LSTM01 model for vibration approach

As you can see in figures 20 and 21 the LSTM01 model is designed to take the data and then encode it to turn the input sequences into a single vector that do contains information about the entire sequence then the repeat vector is going to repeat this vector for x times (x is the total number of instances in the output sequence) then it does run an LSTM decoder to turn this sequence into the target sequence.

Bi_LSTM01

```
#Starting the LSTM model
model = keras.Sequential()

model.add(layers.Bidirectional(LSTM(25, activation='relu', return_sequences=True)))
model.add(layers.BatchNormalization())

model.add(layers.Bidirectional(LSTM(5, activation='relu', return_sequences=False)))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.Bidirectional(LSTM(5, activation='relu', return_sequences=True)))
model.add(layers.BatchNormalization())

model.add((layers.Bidirectional(LSTM(25, activation='relu', return_sequences=True))))
model.add(layers.BatchNormalization())

model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

model.compile(optimizer='adam', loss='mae')
```

Figure 20 Bi_LSTM01 model for dynamic pressure approach

```
#Starting the LSTM model
model = keras.Sequential()

model.add(layers.Bidirectional(LSTM(36, activation='relu', return_sequences=True)))
model.add(layers.BatchNormalization())

model.add(layers.Bidirectional(LSTM(6, activation='relu', return_sequences=False)))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.Bidirectional(LSTM(6, activation='relu', return_sequences=True)))
model.add(layers.BatchNormalization())

model.add((layers.Bidirectional(LSTM(36, activation='relu', return_sequences=True))))
model.add(layers.BatchNormalization())

model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

model.compile(optimizer='adam', loss='mae')
```

Figure 21 Bi_LSTM01 model for vibration approach

As you can see in figures 22 and 23 the Bi_LSTM01 model is the same as the LSTM01 model but it has the bidirectional feature, the bidirectional feature means that the input data will go in two directions in the stream. The bidirectional feature works by adding a reversed LSTM layer after each layer which makes it able to learn from both sides.

LSTM02

```
In [23]: #Starting the LSTM model
model = keras.Sequential()

model.add(layers.LSTM(25, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(5, activation='relu', return_sequences=False))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.LSTM(5, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(25, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(5, activation='relu', return_sequences=False))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.LSTM(5, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(25, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

model.compile(optimizer='adam', loss='mae')
```

Figure 22 LSTM02 model for dynamic pressure approach

```
In [64]: #Starting the LSTM model
model = keras.Sequential()

model.add(layers.LSTM(36, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(6, activation='relu', return_sequences=False))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.LSTM(6, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(36, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(6, activation='relu', return_sequences=False))
model.add(layers.BatchNormalization())

model.add(layers.RepeatVector(features))

model.add(layers.LSTM(6, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.LSTM(36, activation='relu', return_sequences=True))
model.add(layers.BatchNormalization())

model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

model.compile(optimizer='adam', loss='mae')
```

Figure 23 LSTM02 model for vibration approach

As you can see in Figures 24 and 25 the LSTM02 model is LSTM01 stacked over an LSTM01 this makes it more capable of learning the complex relationship between the features as you remember the correlation between the features was not so much and some features were independent of each other.

3. Isolation forest

To apply the pattern technique I have used isolation forest where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1 based on this function. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I added the time feature back to the data frame. I have divided the normal data sets into training and testing so the model will be trained using the normal dataset so it can learn the normal patterns in the data and be able to detect the anomalies. After training the model I used the model to detect the anomalies in the cavitation data set. Then I visualized where did the anomaly occur by using the pressure head to time graph, Dynamic pressure to time graph, and the flow rate to time graph.

4. GRU models

To apply the Correlation among sequences with the Gaussian mixture method I have used the GRU, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I added time back to the data sets. Then I used the normal data in the training of the model and used the cavitation data in the testing. Then I changed the dimension of the training and testing data to meet the requirements of the data dimension of the GRU. Then I built an auto encoder GRU model. I have designed the model to be an auto encoder to get the reconstruction error to evaluate it using threshold criteria to predict the anomalies by looking at the trend of error in the normal and the cavitation data sets.

There are 2 different models of GRU the GRU01 and Bi_GRU01.

GRU01

```
In [20]: #Starting the LSTM model
         model = keras.Sequential()

         model.add(layers.GRU(25, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.GRU(5, activation='relu', return_sequences=False))
         model.add(layers.BatchNormalization())

         model.add(layers.RepeatVector(features))

         model.add(layers.GRU(5, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.GRU(25, activation='relu', return_sequences=True))
         model.add(layers.BatchNormalization())

         model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

         model.compile(optimizer='adam', loss='mae')
```

Figure 24 GRU01 model for dynamic approach

```
In [104]: #Starting the LSTM model
          model = keras.Sequential()

          model.add(layers.GRU(36, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.GRU(6, activation='relu', return_sequences=False))
          model.add(layers.BatchNormalization())

          model.add(layers.RepeatVector(features))

          model.add(layers.GRU(6, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.GRU(36, activation='relu', return_sequences=True))
          model.add(layers.BatchNormalization())

          model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

          model.compile(optimizer='adam', loss='mae')
```

Figure 25 GRU01 model for vibration approach

As you can see in figures 26 and 27 the GRU01 model is designed to take the data and then encode it to turn the input sequences into a single vector that do contains information about the entire sequence then the repeat vector is going to repeat this vector for x times (x is the total number of instances in the output sequence) then it does run a GRU decoder to turn this sequence into the target sequence.

Bi_GRU01

```
In [19]: #Starting the LSTM model
         model = keras.Sequential()

         model.add(layers.Bidirectional(GRU(25, activation='relu', return_sequences=True)))
         model.add(layers.BatchNormalization())

         model.add(layers.Bidirectional(GRU(5, activation='relu', return_sequences=False)))
         model.add(layers.BatchNormalization())

         model.add(layers.RepeatVector(features))

         model.add(layers.Bidirectional(GRU(5, activation='relu', return_sequences=True)))
         model.add(layers.BatchNormalization())

         model.add((layers.Bidirectional(GRU(25, activation='relu', return_sequences=True))))
         model.add(layers.BatchNormalization())

         model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

         model.compile(optimizer='adam', loss='mae')
```

Figure 26 Bi_GRU01 model for dynamic approach

```
In [59]: #Starting the LSTM model
         model = keras.Sequential()

         model.add(layers.Bidirectional(GRU(36, activation='relu', return_sequences=True)))
         model.add(layers.BatchNormalization())

         model.add(layers.Bidirectional(GRU(6, activation='relu', return_sequences=False)))
         model.add(layers.BatchNormalization())

         model.add(layers.RepeatVector(features))

         model.add(layers.Bidirectional(GRU(6, activation='relu', return_sequences=True)))
         model.add(layers.BatchNormalization())

         model.add((layers.Bidirectional(GRU(36, activation='relu', return_sequences=True))))
         model.add(layers.BatchNormalization())

         model.add(layers.TimeDistributed(Dense(X_train.shape[2])))

         model.compile(optimizer='adam', loss='mae')
```

Figure 27 Bi_GRU01 for vibration approach

As you can see in figures 28 and 29 the Bi_GRU01 model is the same as the GRU01 model but it has the bidirectional feature, the bidirectional feature means that the input data will go in two directions in the stream. The bidirectional feature works by adding a reversed GRU layer after each layer which makes it able to learn from both sides.

4.5 Frequency-based models

1. One class SVM algorithm

To apply the OCC method I have used the One-class SVM algorithm, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I used the fast Fourier transform function to get the frequency in each feature then I visualized a graph that shows the FFT amplitude to the frequency. Then I have set the output of the fast Fourier transform function to real numbers of type float. I have divided the normal data sets into training and testing so the model will be trained using the normal dataset so it can learn the normal patterns in the data and be able to detect the anomalies. After training the model I used the model to detect the anomalies in the cavitation data set. Then I visualized where did the anomaly occur by using the pressure head to time graph, Dynamic pressure to time graph, and the flow rate to time graph.

2. LSTM models

To apply the prediction error method I have used the LSTM, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I used the fast Fourier transform function to get the frequency in each feature then I visualized a graph that shows the FFT amplitude to the frequency. Then I have set the output of the fast Fourier transform function to real numbers of type float. Then I added time back to the data sets. Then I used the normal data in the training of the model and used the cavitation data in the testing. Then I changed the dimension of the training and testing data to meet the requirements of the data dimension of the LSTM. Then I built an auto encoder LSTM model. I have designed the model to be an auto encoder to get the reconstruction error to evaluate it using threshold criteria to predict the anomalies by looking at the trend of error in the normal and the cavitation data sets.

There are 3 different models of LSTM the LSTM01, LSTM02, and Bi_LSTM01 which are the same as the time sequence models.

3. Isolation forest

To apply the pattern technique I have used isolation forest where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I used the fast Fourier transform function to get the frequency in each feature then I visualized a graph that shows the FFT amplitude to the frequency. Then I have set the output of the fast Fourier transform function to real numbers of type float. Then I added the time feature back to the data frame. I have divided the normal data sets into training and testing so the model will be trained using the normal dataset so it can learn the normal patterns in the data and be able to detect the anomalies. After training the model I used the model to detect the anomalies in the cavitation data set. Then I visualized where did the anomaly occur by using the pressure head to time graph, Dynamic pressure to time graph, and the flow rate to time graph.

4. GRU models

To apply the Correlation among sequences with the Gaussian mixture method I have used the GRU, where I normalized the data coming from the normal and cavitation data sets without time, by normalization I mean that I have scaled the data from -1 to 1. Then I used a standard scalar to scale the data which will make the algorithms more efficient in detecting the anomalies and easier to see the pattern of the data between different data sets. Then I used the fast Fourier transform function to get the frequency in each feature then I visualized a graph that shows the FFT amplitude to the frequency. Then I have set the output of the fast Fourier transform function to real numbers of type float. Then I added time back to the data sets. Then I used the normal data in the training of the model and used the cavitation data in the testing. Then I changed the dimension of the training and testing data to meet the requirements of the data dimension of the GRU. Then I built an auto encoder GRU model. I have designed the model to be an auto encoder to get the reconstruction error to evaluate it using threshold criteria to predict the anomalies by looking at the trend of error in the normal and the cavitation data sets.

There are 2 different models of GRU the GRU01 and Bi_GRU01 which are the same as the time sequence models.

5 Testing and evaluation

5.1 Testing

Each model is tested with the same dataset for two approaches the dynamic pressure approach and the vibration approach and by using both the time series and the frequency sequence.

Time series & frequency based

Note the same model can work for both data without any interruptions all the difference is in the preparation of the data as mentioned above in the 4.4 and 4.5 sections.

Note that the number of features in the dynamic pressure approach is 5 features while in the Vibration approach it is 6 features.

A. One-Class SVM

The one-class SVM model was trained with a kernel of RBF, gamma equals 0.01, and nu equals 0.01. The RBF kernel is chosen because it's a good mapping function and can be used in large data sets with high dimensions, the RBF kernel is very costly in time. The gamma and the nu were tuned to have the highest accuracy.

B. Isolation Forest

The Isolation Forest model was trained with contamination equal to 0.05 and a random state equal to 1. The contamination was tuned to have the highest accuracy.

C. LSTM01

In figures 20 and 21, you can see the LSTM01 model does have 10 layers. In the beginning, the model works in encoding the input data, the first layer is an LSTM layer the model does receive the input that the model is going to encode the number of units is equal to the number of features to the power of 2 and the return sequence is set to true so the learned weights can be transferred to the following layer. In the second, fourth, seventh, and ninth layers I do use batch normalization to normalize the output of the previous layers. The third layer is an LSTM layer the model receives the encoded input that the model is going to learn the number of units is equal to the number of features and the return sequence is set to false because setting it to true would disturb the decoding process. In the fifth layer, the repeat vectors work as a bridge between the encoding and the decoding with units equal to the number of features. The sixth layer is an LSTM layer model do receive the encoded input that the model is going to send to the next layer to decode it the number of units is equal to the number of features and the return sequence is set to true. The eighth layer is an LSTM layer the model does receive the decoded output of the previous layer that the model is going to learn the number of units is equal to the number of features to the power of 2 and the return sequence is set to true. The tenth layer is a time-distributed layer that makes sure that all dimensions of the data don't change. The model was compiled using the ADAM optimizer and the loss was calculated by using the MAE.

D. Bi_LSTM01

In figures 22 and 23, you can see that the Bi_LSTM01 model has the same structure is same as the LSTM01 structure, but every LSTM layer is bidirectional instead of unidirectional.

E. LSTM02

In figures 24 and 25, you can see that the LSTM02 model has the same structure is same as the LSTM01 structure, but it's a stacking of 2 LSTM01.

F. GRU01

In figures 26 and 27, you can see the GRU01 model does have 10 layers. In the beginning, the model works in encoding the input data, the first layer is a GRU layer the model does receive the input that the model is going to encode the number of units is equal to the number of features to the power of 2 and the return sequence is set to true so the learned weights can be transferred to the following layer. In the second, fourth, seventh, and ninth layers I do use batch normalization to normalize the output of the previous layers. The third layer is a GRU layer the model receives the encoded input that the model is going to learn the number of units is equal to the number of features and the return sequence is set to false because setting it to true would disturb the decoding process. In the fifth layer, the repeat vectors work as a bridge between the encoding and the decoding with units equal to the number of features. The sixth layer is a GRU layer model do receive the encoded input that the model is going to send to the next layer to decode it the number of units is equal to the number of features and the return sequence is set to true. The eighth layer is a GRU layer the model does receive the decoded output of the previous layer that the model is going to learn the number of units is equal to the number of features to the power of 2 and the return sequence is set to true. The tenth layer is a time-distributed layer that makes sure that all dimensions of the data don't change. The model was compiled using the ADAM optimizer and the loss was calculated by using the MAE.

G. Bi_GRU01

In figures 28 and 29, you can see that the Bi_GRU01 model has the same structure is same as the GRU01 structure, but every GRU layer is bidirectional instead of unidirectional.

5.2 Evaluation

We can evaluate the models in two ways the first way is by using the MAE loss and the anomalies detected and the other way is by visualizing the Density to loss graph. Some models can't be evaluated using any of these ways which are the One-class SVM, Isolation Forest, One-class SVM frequency, and the Isolation Forest frequency model. These models will be evaluated by the degree of the similarity between their output and the benchmark model output.

Anomalies detected by Models that can't be evaluated

Dynamic Pressure

Name of the model	Anomalies detected
One Class SVM	63000
Isolation Forest	37137
One Class SVM frequency	235267
Isolation Forest frequency	68092

Vibration

Name of the model	Anomalies detected
One Class SVM	63720
Isolation Forest	22951
One Class SVM frequency	307934
Isolation Forest frequency	60262

The number of Anomalies that were detected is quite close to what other models detected, but the "One Class SVM frequency-based" has shown a major failure in differentiating between the anomalies and the normal data points due to how the one-class classification methods work and the frequency data patterns and sequences. The isolation forest was the fastest and the most efficient and the number of anomalies detected and where they were detected was very similar to the outstanding model that has performed very well with the ‘Bi_GRU01 frequency’ in both the dynamic pressure approach and the vibration approach.

MAE Loss table

MAE Difference = | MAE Normal - MAE Cavitation |

Dynamic Pressure

Table 1 MAE loss table comparison Dynamic Pressure Approach

Name of the model	MAE Normal	MAE Cavitation	MAE Difference	Anomalies detected
LSTM01	0.0252	0.1057	0.0805	53274
Bi_LSTM01	0.0240	0.0858	0.0618	54692
LSTM02	0.0338	0.1933	0.1595	50171
GRU01	0.0298	0.1250	0.0952	41113
Bi_GRU01	0.0260	0.1370	0.111	51158
LSTM01 frequency	0.0445	0.0647	0.0202	38680
Bi_LSTM01 frequency	0.0404	0.0767	0.0363	40911
LSTM02 frequency	0.0438	0.1158	0.072	29908
GRU01 frequency	0.0443	0.0656	0.0213	41474
Bi_GRU01 frequency	0.0421	0.0515	0.0094	40267

Based on the MAE loss table evaluation for the Vibration approach, it would be easy to say that the "Bi_GRU01 frequency model" outperforms the other models. The method used in this evaluation was using the MAE difference loss with a score of 0.0094. Overall it would be pleasant to say that the frequency-based models performed better than the Time series-based models.

Vibration approach

Table 2 MAE loss table comparison Vibration Approach

Name of the model	MAE Normal	MAE Cavitation	MAE Difference	Anomalies detected
LSTM01	0.0279	0.1304	0.1025	50214
Bi_LSTM01	0.0222	0.0819	0.0597	60011
LSTM02	0.0321	1.2638	1.2317	19986
GRU01	0.0301	0.0703	0.0402	42349
Bi_GRU01	0.0257	0.0937	0.068	46563
LSTM01 frequency	0.0488	0.0755	0.0267	23616
Bi_LSTM01 frequency	0.0438	0.1181	0.0693	8573
LSTM02 frequency	0.0545	0.1865	0.1320	26983
GRU01 frequency	0.0511	0.0697	0.0186	30303
Bi_GRU01 frequency	0.0457	0.0607	0.015	30239

Based on the MAE loss table evaluation for the Vibration approach, it would be easy to say that the "Bi_GRU01 frequency model" outperforms the other models. The method used in this evaluation was using the MAE difference loss with a score of 0.015. Overall it would be pleasant to say that the frequency-based models performed better than the Time series-based models.

Density loss graph

In the loss time graph the cleaner the model the fewer noises it has the better it is and the more noises it has the worse it is because it will be more difficult to choose a threshold value.

You can view the rest of the graphs in the Appendix I section I have only presented the best two models that performed in both the vibration approach and the dynamic pressure approach.

Dynamic Pressure

Bi_GRU01 frequency

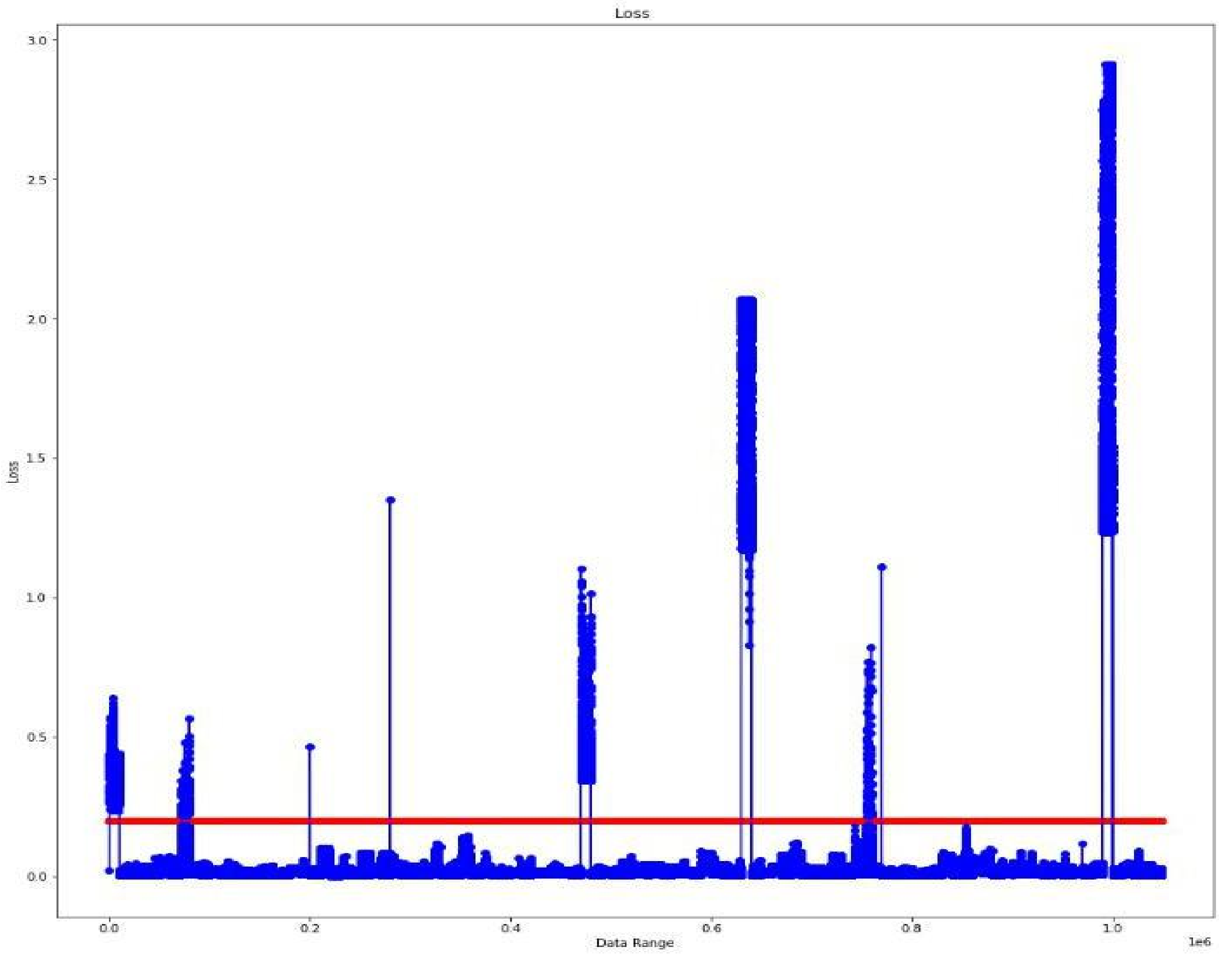

Figure 28 Figure 36 Bi_GRU01 frequency dynamic pressure approach Loss to time graph

The MAE metrics evaluate the model's ability to learn. The best model that was able to learn the normal dataset is the model that had the lowest MAE Normal loss. Which is the "Bi_LSTM01 for the vibration approach" but as you can see in figure 39 in the Appendix I section, the model was having a problem with the noises so we can't evaluate the model only by using the MAE Normal loss.

The MAE loss method had approved its efficiency when it was used to see which model reached the least MAE cavitational loss, we can see that the "Bi_GRU01 frequency for dynamic pressure approach" model achieved the best results and the loss to time graph was clean without any noises. This model also had the Least MAE difference so we need more study to know which performance that has caused this model to outperform the other models is it the MAE Cavitational loss or the MAE difference.

6 Results and Discussions

As you can see in section 5.2 which is the evaluation section. You can see that all the models expect one-class SVM and isolation forest can be evaluated using two types of metrics the MAE loss performance measure and the loss to time graph. In the MAE loss performance the measure the lower the MAE difference or the MAE cavitational the better the model is. In the loss time graph the cleaner the model the fewer noises it has the better it is like figure 28 and the more noises it has the worse it is because it will be more difficult to choose a threshold value.

The MAE metrics evaluate the model's ability to learn based on the results all the models were able to learn the Normal data set but due to their different methods of dealing with the cavitational data set they had no common ground. So to measure the performance of these models we are using the MAE cavitational least loss method in measuring the performance of these models. This method has been approved for its efficiency when it was used to see who is the model with the least MAE cavitational loss we can see that the "Bi_GRU01 frequency for dynamic pressure approach" model achieved the best results and its loss to time graph was clean without any noises. This model also had the Least MAE difference. So we need more studies to know which performance measures indicated that this model will outperform the other models.

Based on these results, we could say that the dynamic pressure approach with the frequency-based data is the optimal approach for this case and that the "Bi_GRU01 frequency" model could be used as a benchmark for further studies.

7 Conclusions and Future Work

7.1 Summary

The models that were trained using the normal data set to learn it and to be able to predict the anomalies based on the threshold that the model/me chooses it. I have tested different types of methods and models and all the models were successful except the one-class SVM frequency-based model that has failed in the two approaches the dynamic pressure and the vibration approach. This means that the one class methods will not work as efficiently when using the frequency-based instead of the time series. So based on that discovery the anomaly detection models that I have created for this study except the one-class SVM frequency-based model could be applied to desalination pumps to have a field test. Based on the results, we could say that the dynamic pressure approach with the frequency-based data is the optimal approach for this case and that the "Bi_GRU01 frequency" model could be used as a benchmark for further studies.

7.2 Issues & Future Work

The limitation in this project is all in one sector which is evaluating the model because it's hard to evaluate the models that are working in an unsupervised data set. To give a real evaluation of the model we will need to test the model on a desalination pump to see if it works or not. While some model's performance was evaluated using the MAE or by the loss to time graph, there were four models that we have failed in evaluating their process the four models are the ("One class SVM", "Isolation Forest", "One class SVM frequency-based", and "Isolation Forest frequency-based").

This project may be used in the continuation of the faculty of engineering effort to have a smart desalination pump that does detect anomalies in live monitoring.

References

[1] Q. He, Y. J. Zheng, C.L. Zhang, and H. Y. Wang, "MTAD-TF: Multivariate Time Series Anomaly Detection Using the Combination of Temporal Pattern and Feature Pattern," School of Science, Beijing University of Civil Engineering and Architecture, p. 9, 2020.

[2] Lizong Zhang, Xiang Shen, Fengming Zhang, Minghui Ren, Binbin Ge, and Bo LI, "Anomaly detection for power grid based on time series model," School of Computer Science and Engineering, Beihang University, p. 5, 2019.

[3] Drausin Wulsin, Justin Blanco, Ram Mani and Brian Litt, "Semi-Supervised Anomaly Detection for EEG Waveforms Using Deep Belief Nets," University of Pennsylvania, p. 6, 2010.

[4] Guangxuan Zhu, Hongbo Zhao, Haoqiang Liu, and Hua Sun , "A Novel LSTM-GAN Algorithm for Time Series Anomaly Detection," Hefei Innovation Research Institute, Beihang University, p. 6, 2019.

[5] A. CHOUDHURY, "Gated Recurrent Unit – What Is It And How To Learn," 16 10 2020. [Online]. Available: https://analyticsindiamag.com/gated-recurrent-unit-what-is-it-and-how-to-learn/

[6] J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras", Machine Learning Mastery, 2022. [Online]. Available: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/. [Accessed: 07- May- 2022].

[7] T. LazyProgrammer, Deep Learning: Recurrent Neural Networks LSTM, GRU, and more RNN machine learning architectures in Python and Theano, 2016.

[8] R. Yu et al., "LSTM-EFG for wind power forecasting based on sequential correlation features", Future Generation Computer Systems, vol. 93, pp. 33-42, 2019. Available: 10.1016/j.future.2018.09.054.

[9] K. Team, "Keras documentation: LSTM layer", Keras.io, 2022. [Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/. [Accessed: 05- May- 2022].

[10] Y. Tang, "Build a GRU RNN in Keras - PythonAlgos", PythonAlgos, 2022. [Online]. Available: https://pythonalgos.com/build-a-gru-rnn-in-keras/. [Accessed: 05- May- 2022].

[11] "What is Pump Cavitation and How Do I Avoid It?", Central States Industrial, 2022. [Online]. Available: https://www.csidesigns.com/blog/articles/what-is-pump-cavitation-and-how-to-prevent-it. [Accessed: 13- Jun- 2022].

[12] D. Trocel, "Vibration Analysis of Centrifugal Pumps", Power-MI, 2022. [Online]. Available: https://power-mi.com/content/vibration-analysis-centrifugal-pumps. [Accessed: 13- Jun-2022].

[13] D. Trocel, "Vibration Analysis of Centrifugal Pumps", Power-MI, 2022. [Online]. Available: https://power-mi.com/content/vibration-analysis-centrifugal-pumps. [Accessed: 13- Jun-2022].

[14] "Isolation Forest | Anomaly Detection with Isolation Forest", Analytics Vidhya, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/. [Accessed: 13- Jun- 2022].

[15] "What is One-Class SVM ? How to use it for anomaly detection?", Machine Learning Interviews, 2022. [Online]. Available: https://machinelearninginterview.com/topics/machine- learning/what-is-one-class-svm-how-to-use-it-for-anomaly-detection/. [Accessed: 13- Jun- 2022].

[16] Ergen, T. and Kozat, S., 2020. Unsupervised Anomaly Detection With LSTM Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 31(8), pp.3127-3141.

[17] Wu, W., He, L., Lin, W., Su, Y., Cui, Y., Maple, C. and Jarvis, S., 2020. Developing an Unsupervised Real-time Anomaly Detection Scheme for Time Series with Multi-seasonality. IEEE Transactions on Knowledge and Data Engineering, pp.1-1.

Appendix I

Density loss graph

Dynamic Pressure

LSTM01

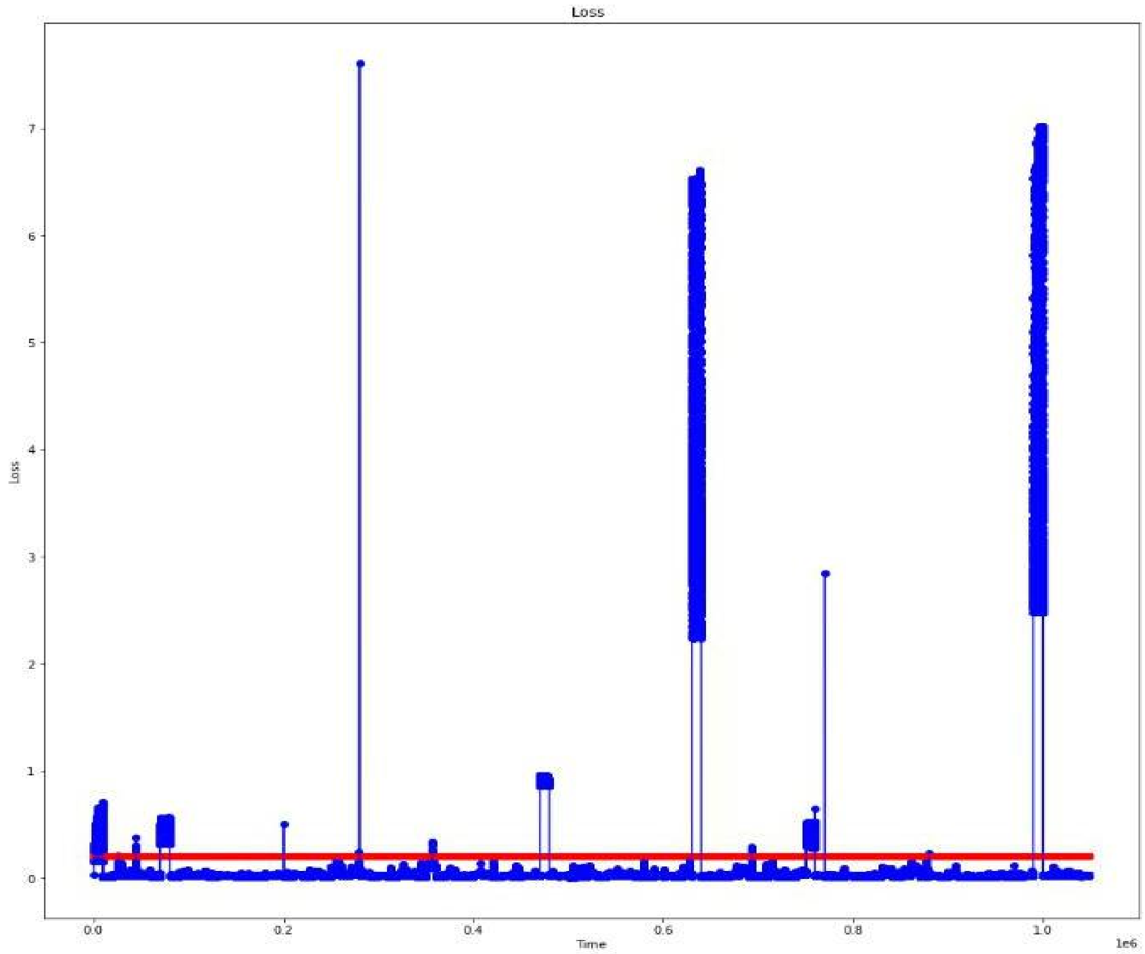

Figure 29 LSTM01 dynamic pressure approach Loss to time graph

Bi_LSTM01

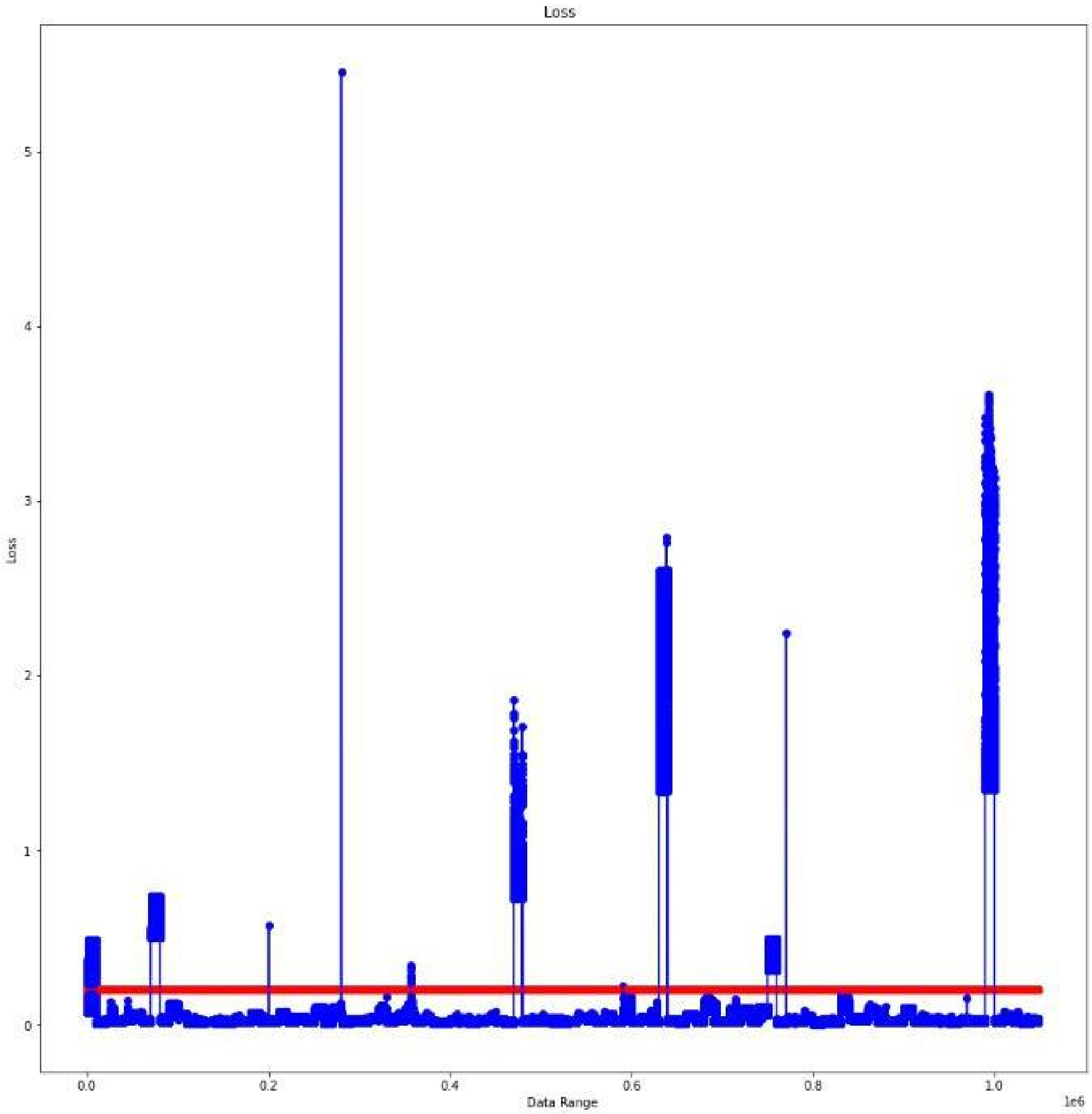

Figure 30 Bi_LSTM01 dynamic pressure approach Loss to time graph

LSTM02

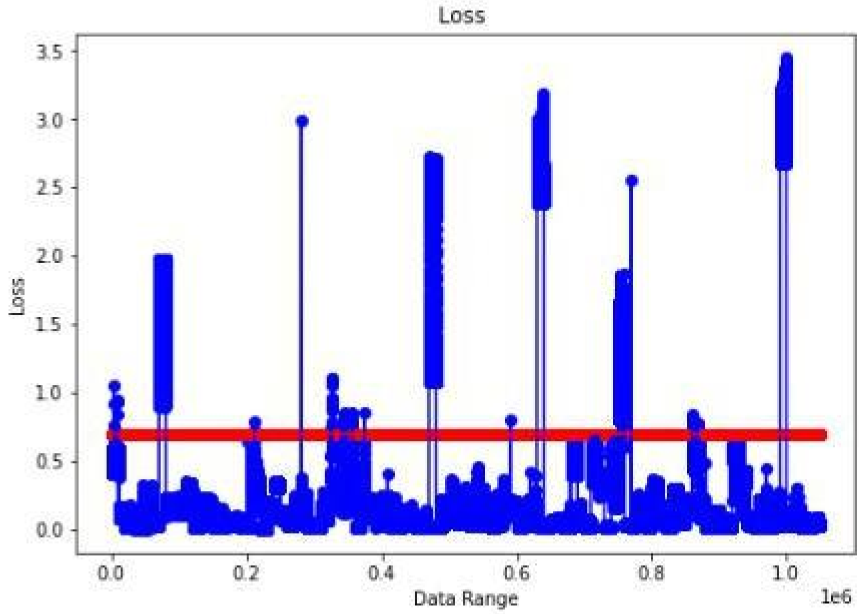

Figure 31 LSTM02 dynamic pressure approach Loss to time graph

GRU01

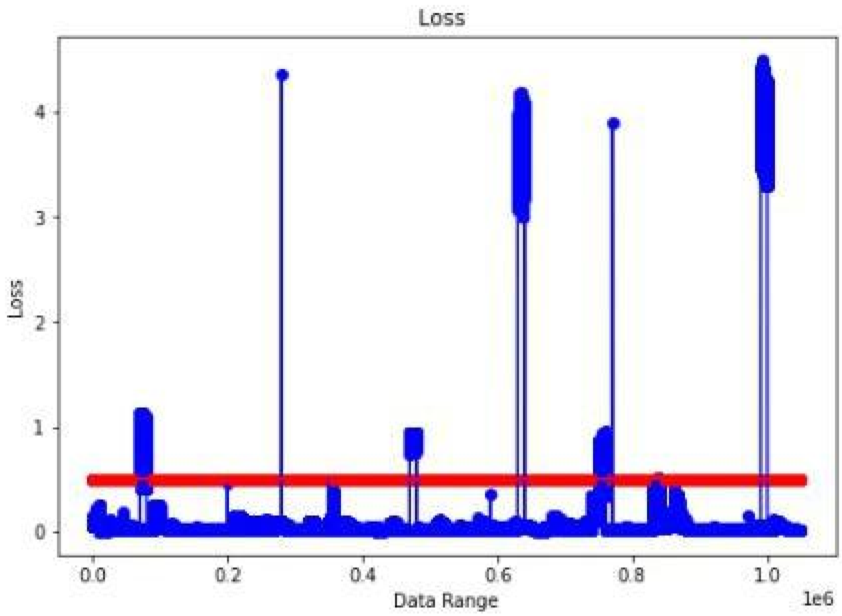

Figure 32 GRU01 dynamic pressure approach Loss to time graph

Bi_GRU01

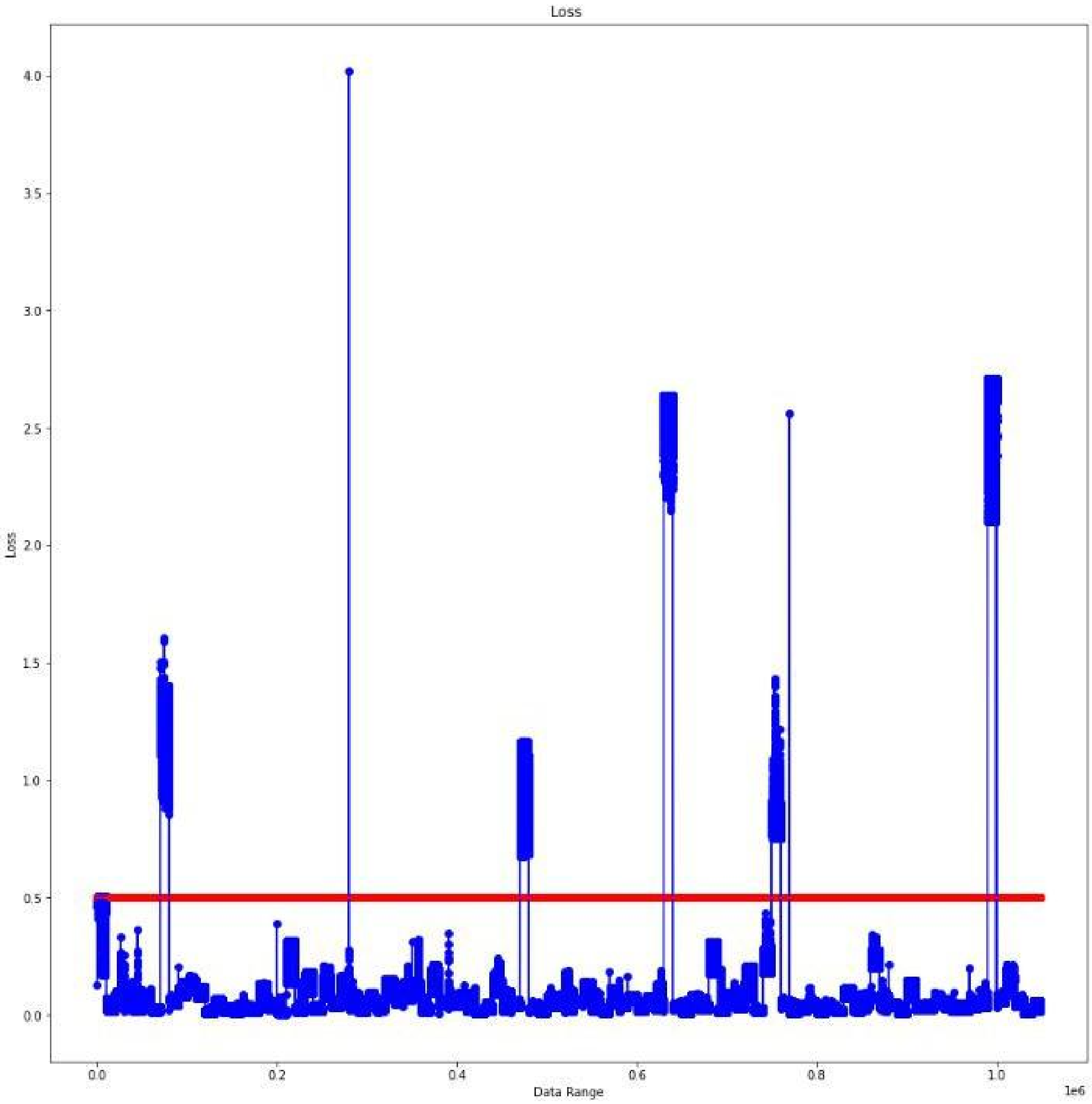

Figure 33 Bi_GRU01 dynamic pressure approach Loss to time graph

LSTM01 frequency

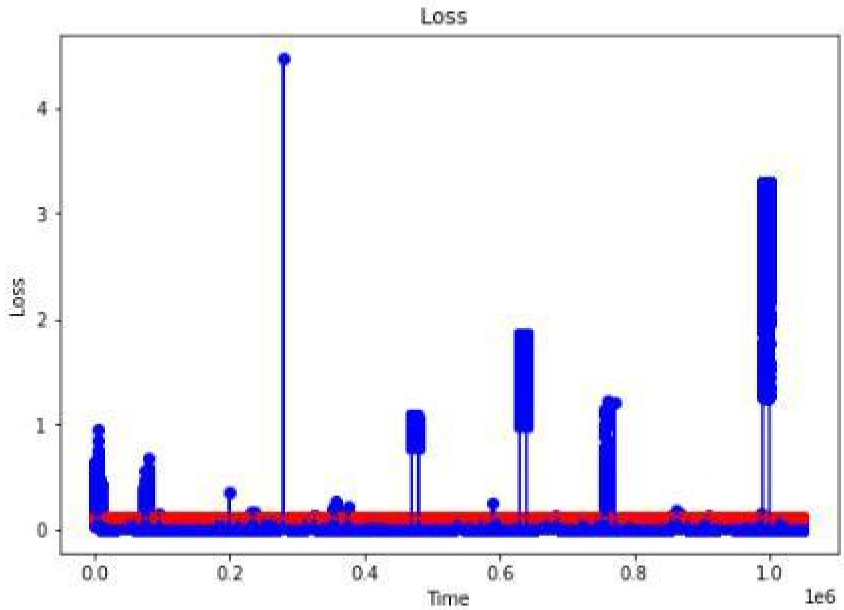

Figure 34 LSTM01 frequency dynamic pressure approach Loss to time graph

Bi_LSTM01 frequency

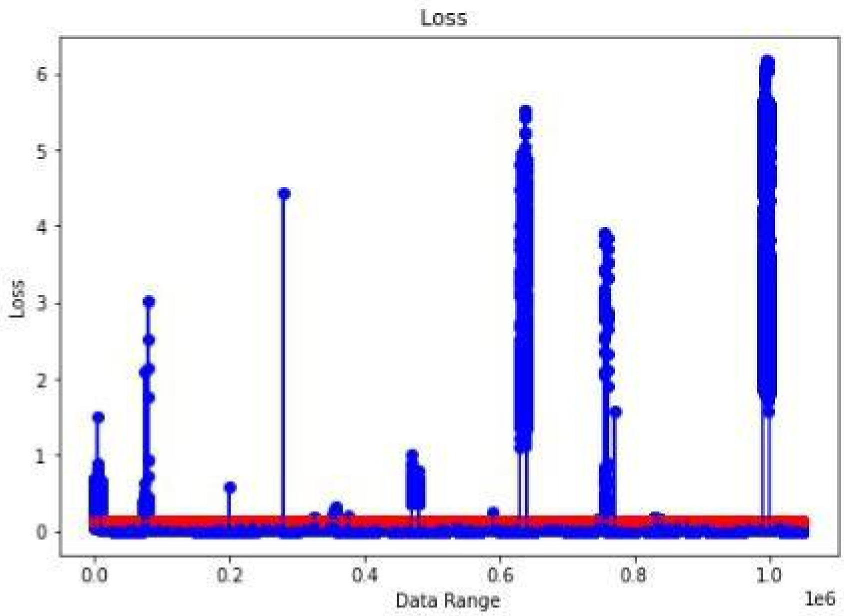

Figure 35 Bi_LSTM01 frequency dynamic pressure approach Loss to time graph

LSTM02 frequency

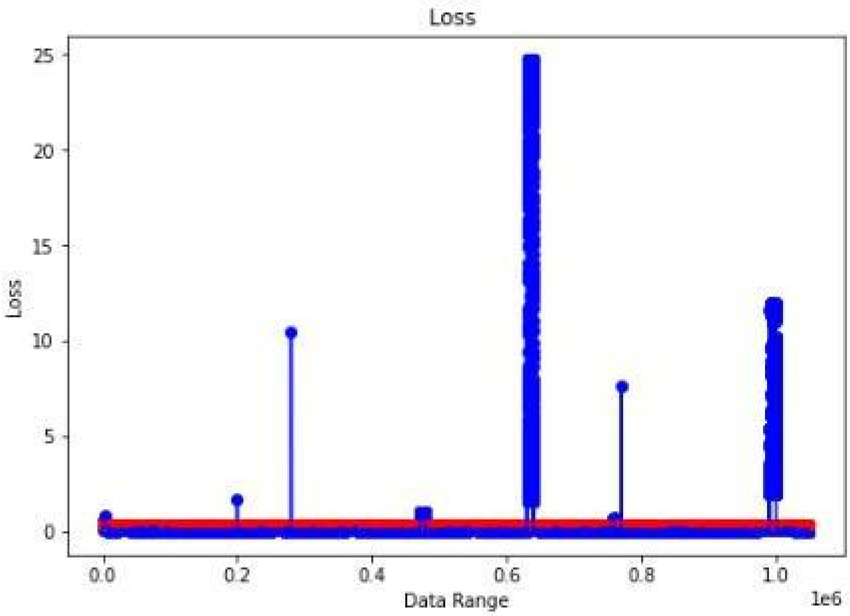

Figure 36 LSTM02 frequency dynamic pressure approach Loss to time graph

GRU01 frequency

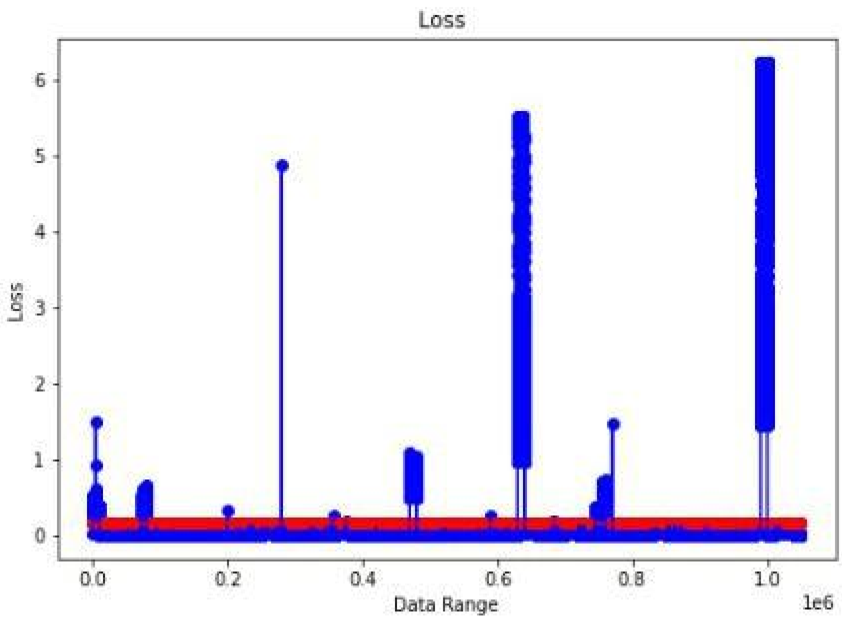

Figure 37 GRU01 frequency dynamic pressure approach Loss to time graph

Bi_GRU01 frequency

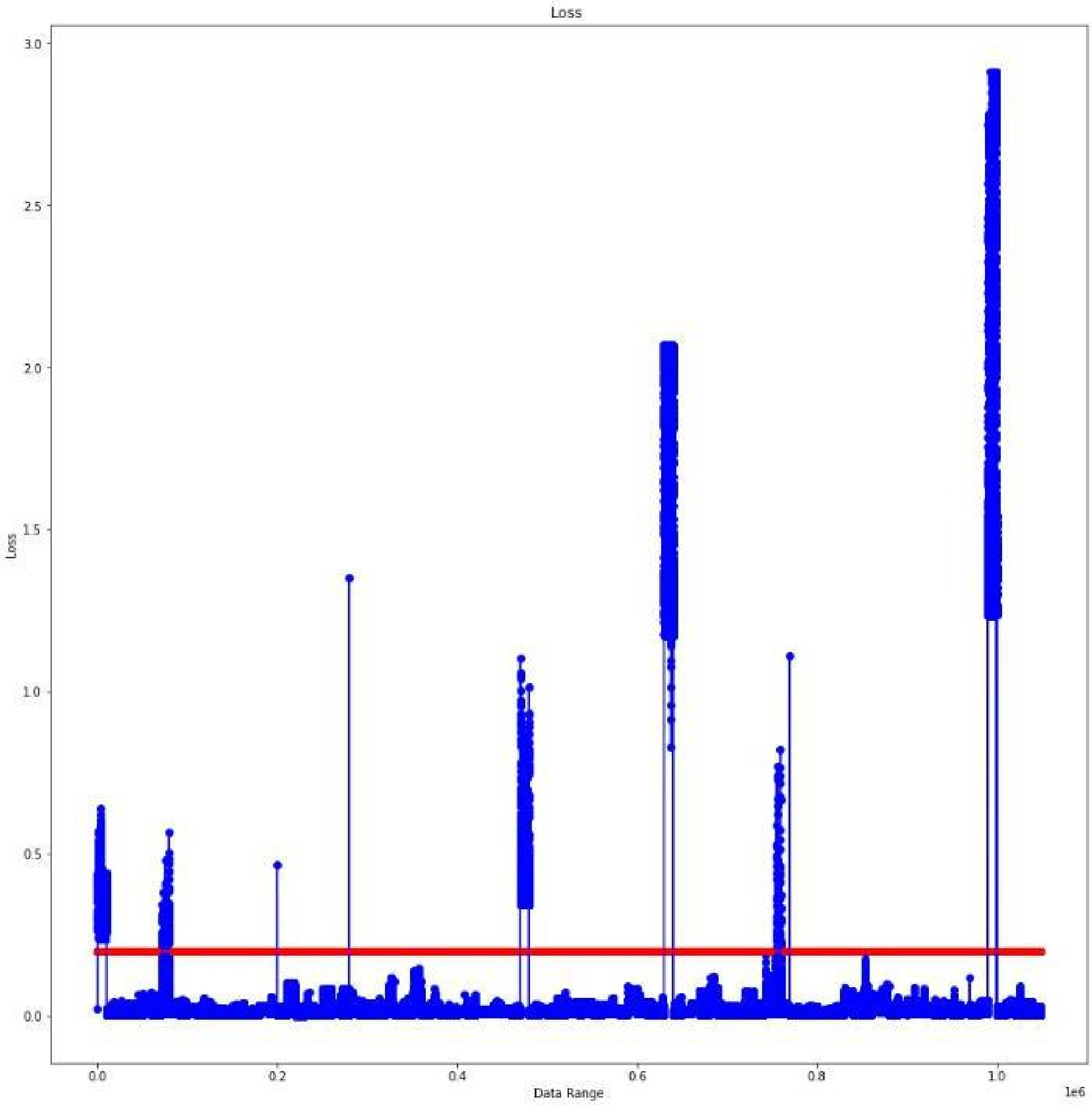

Figure 38 Figure 36 Bi_GRU01 frequency dynamic pressure approach Loss to time graph

Vibration

LSTM01

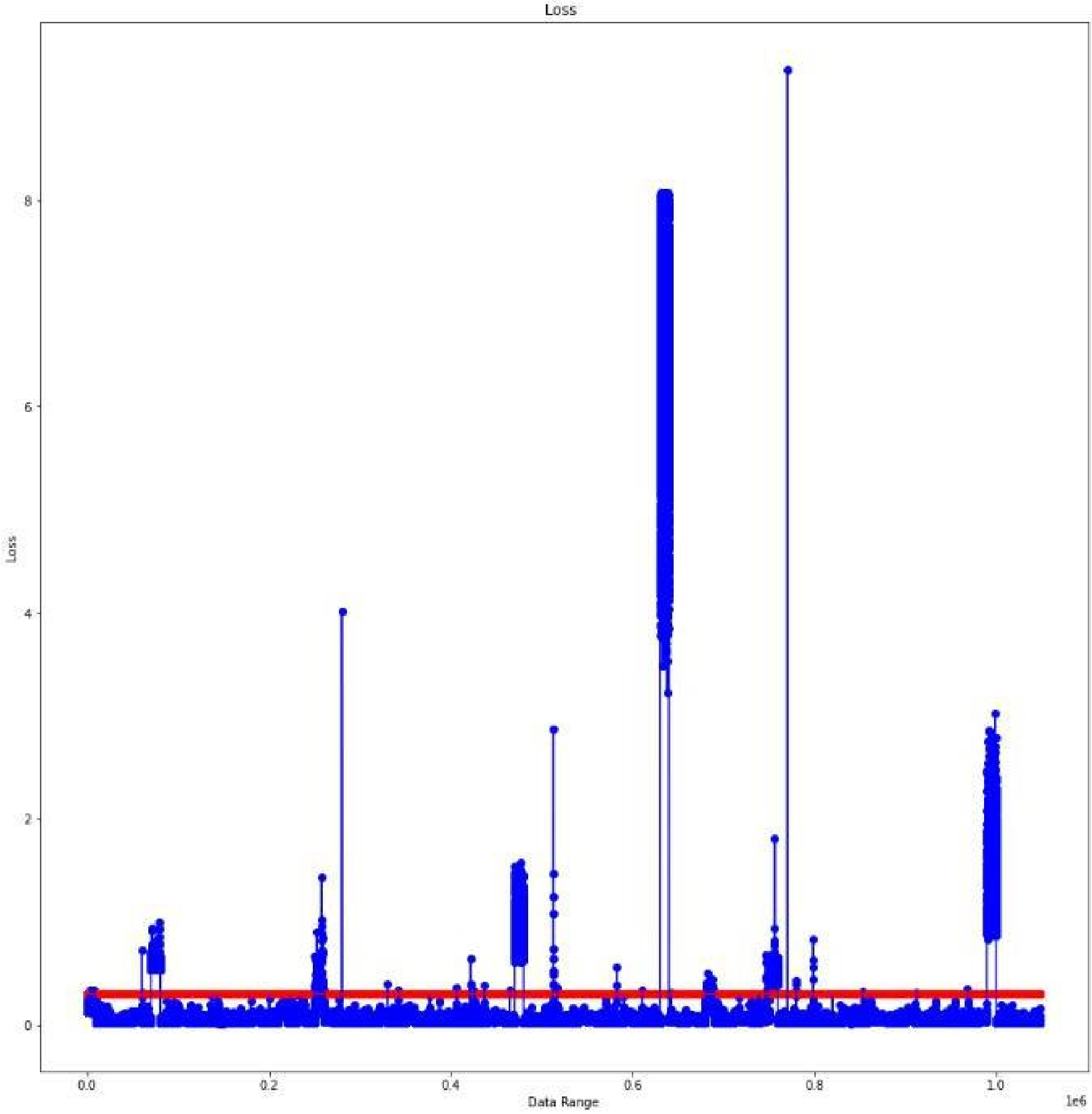

Figure 39 LSTM01 vibration approach Loss to time graph

Bi_LSTM01

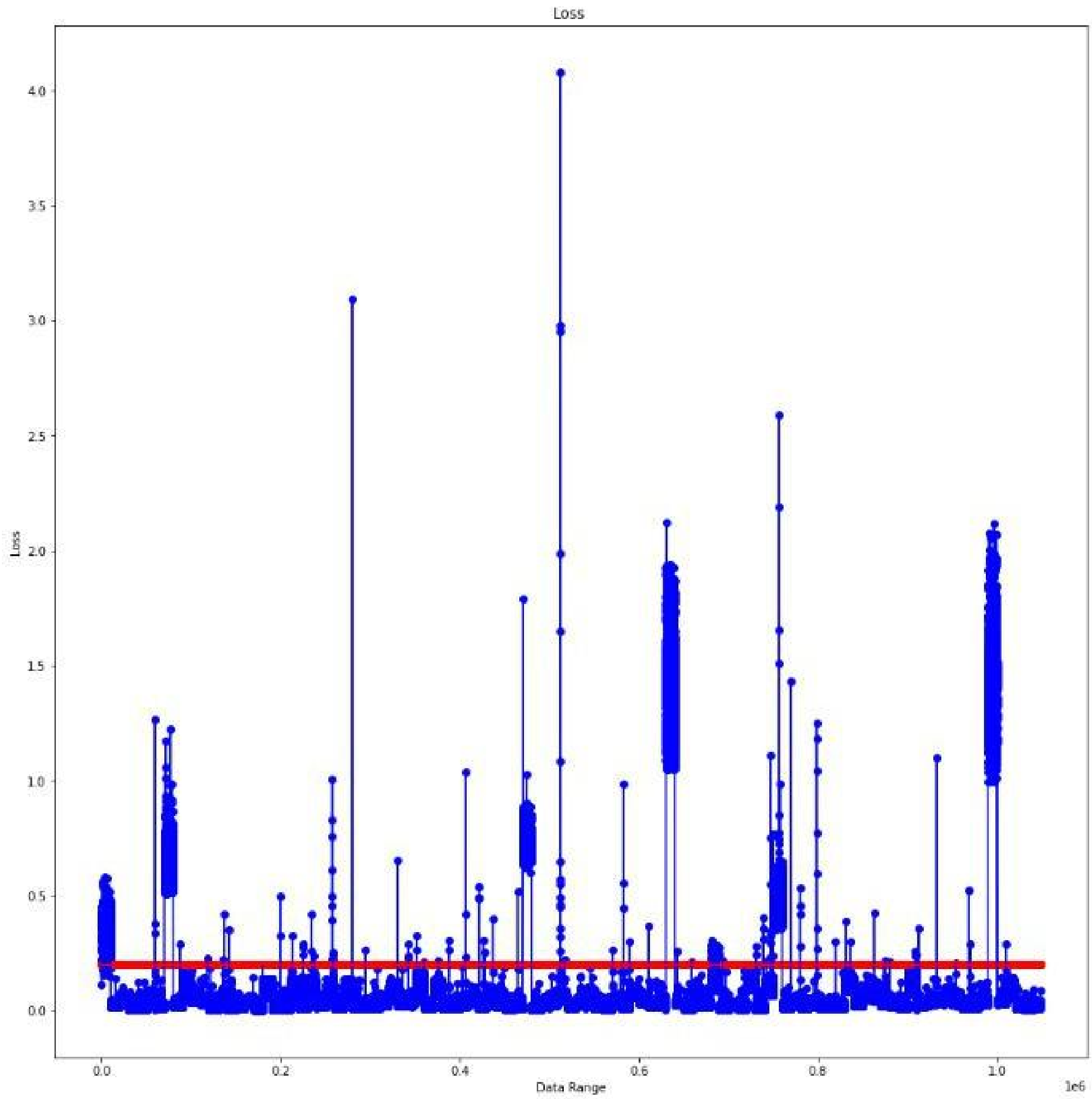

Figure 40 Bi_LSTM01 vibration approach Loss to time graph

LSTM02

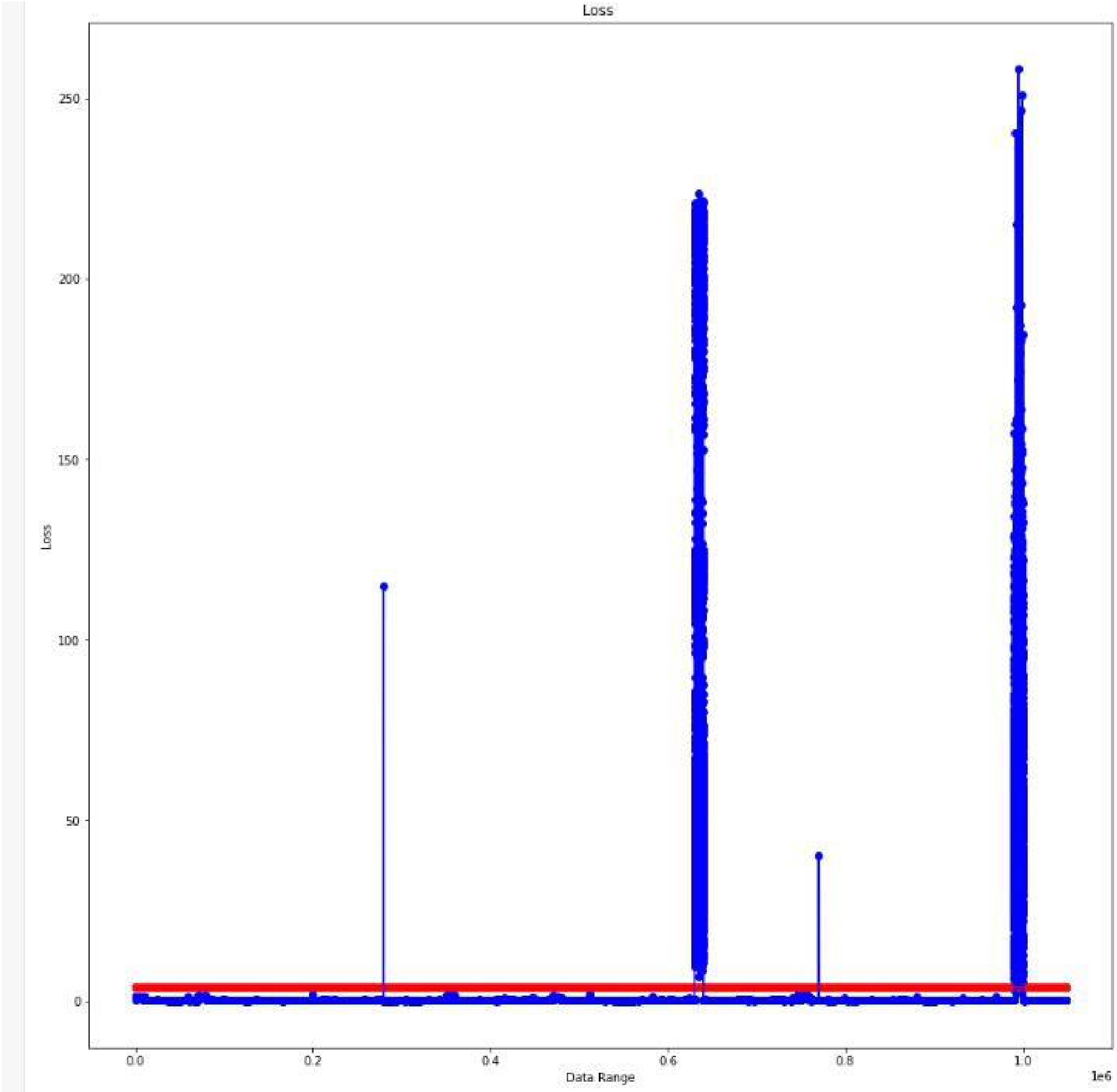

Figure 41 LSTM02 vibration approach Loss to time graph

GRU01

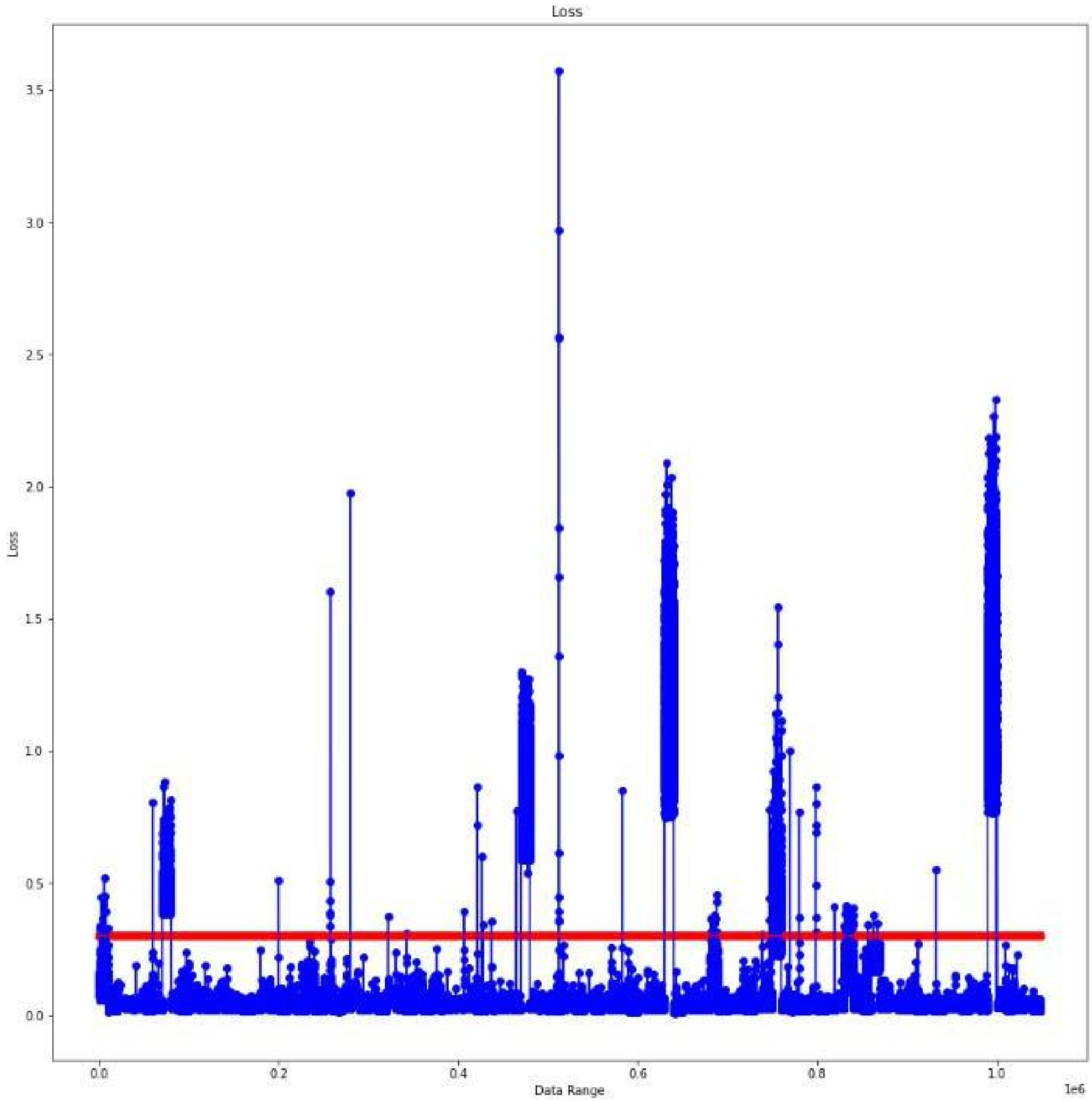

Figure 42 GRU01 vibration approach Loss to time graph

Bi_GRU01

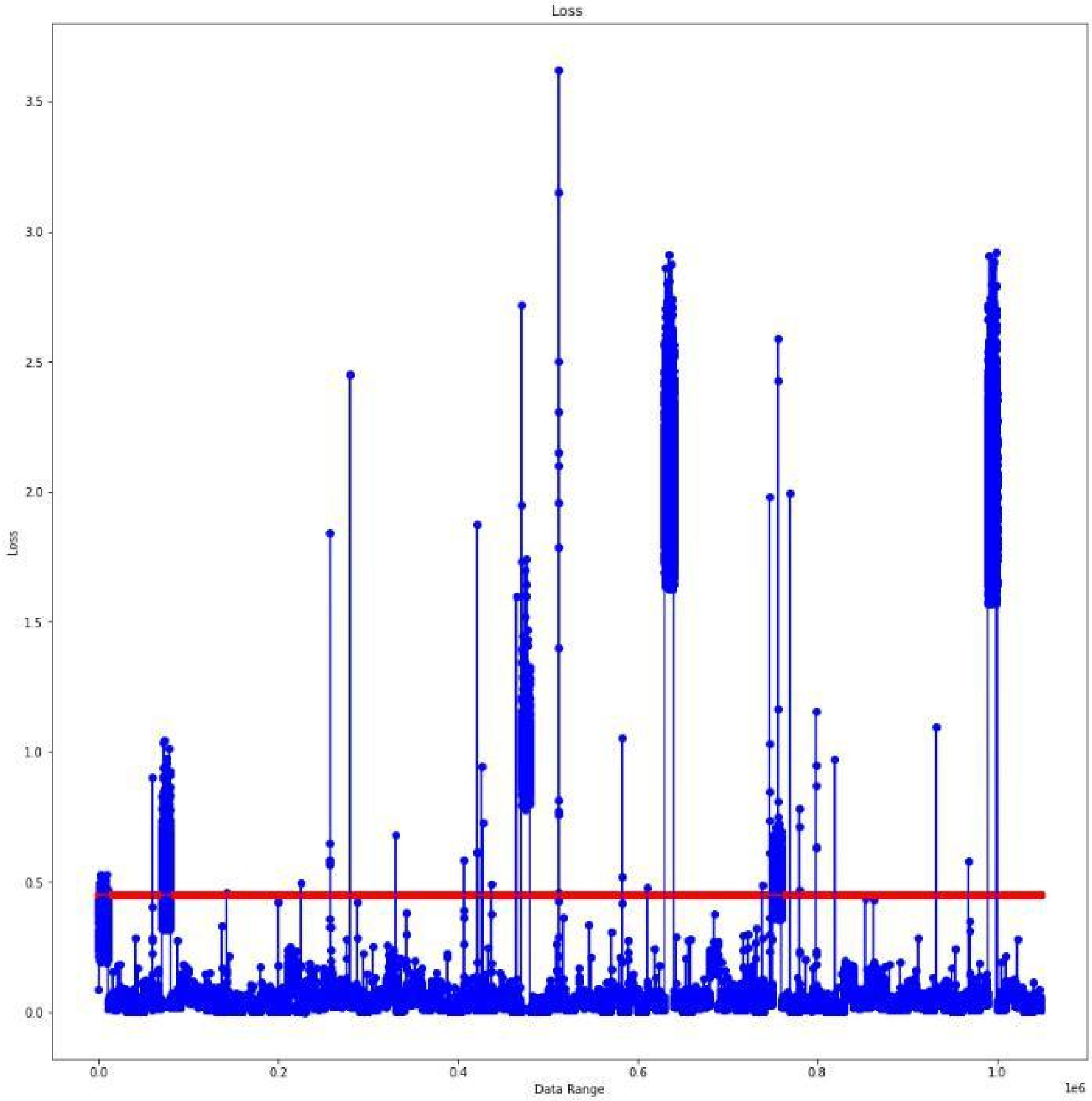

Figure 43 Bi_GRU01 vibration approach Loss to time graph

LSTM01 frequency

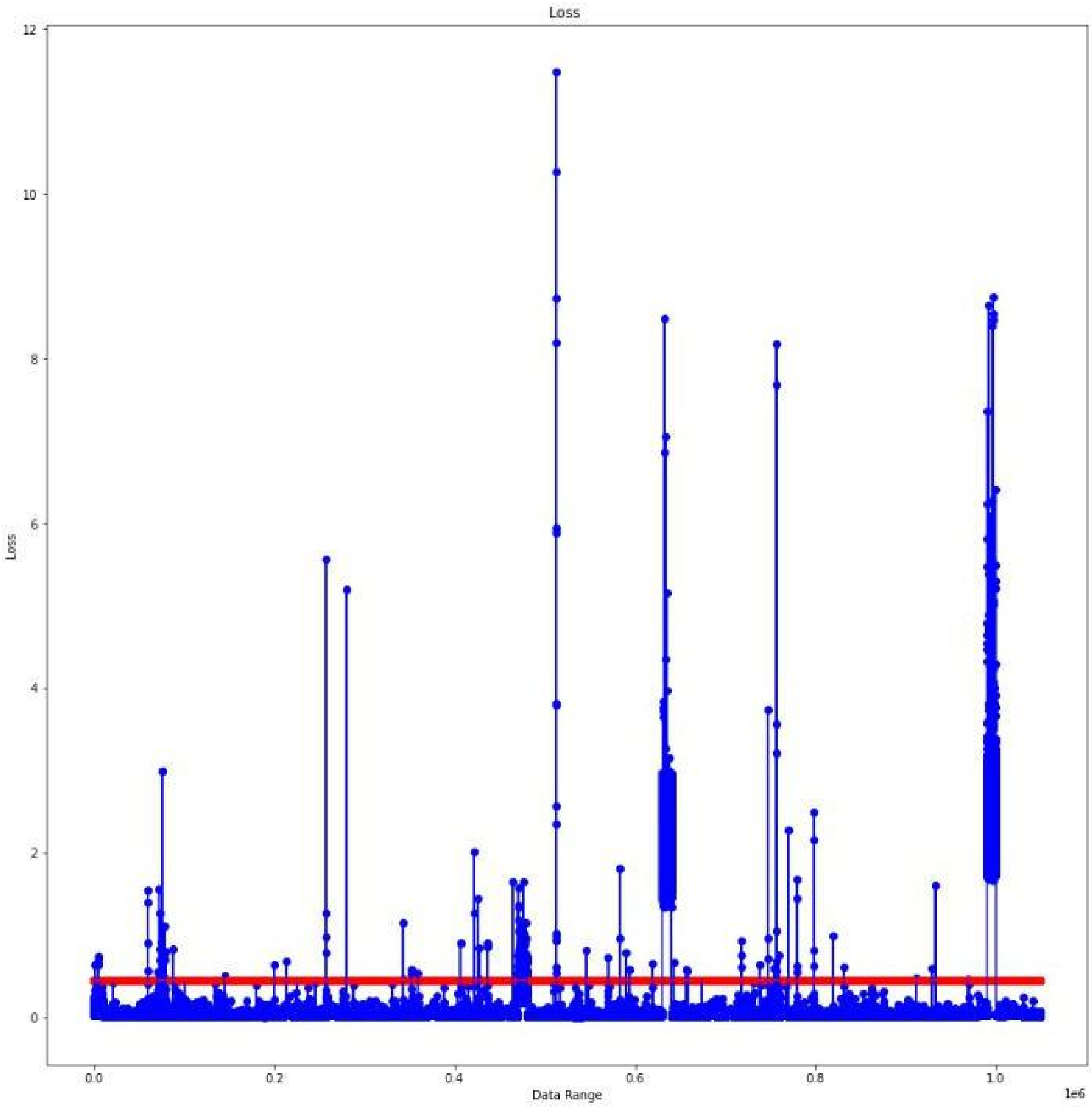

Figure 44 LSTM01 frequency vibration approach Loss to time graph

Bi_LSTM01 frequency

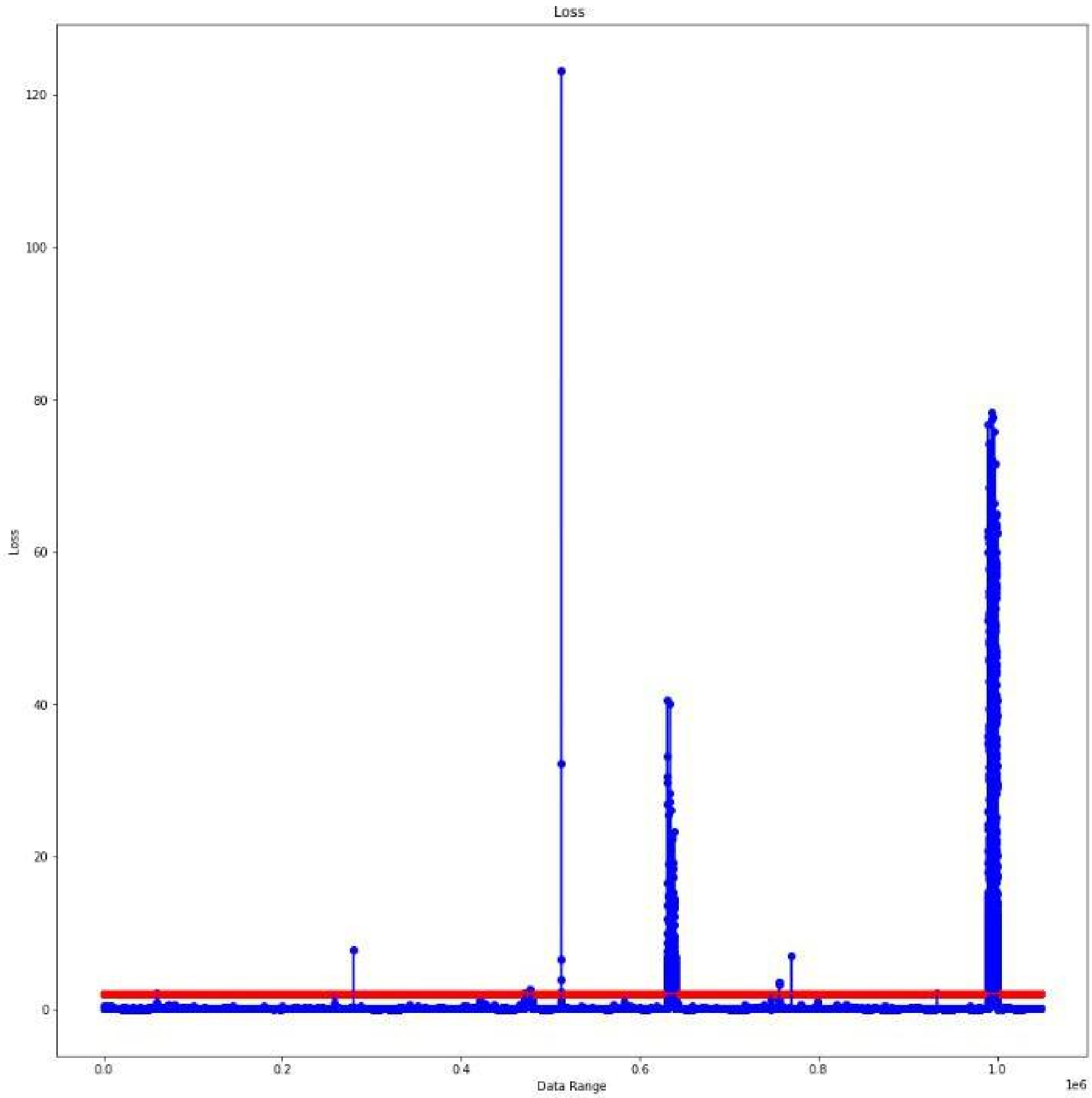

Figure 45 Bi_LSTM01 frequency vibration approach Loss to time graph

LSTM02 frequency

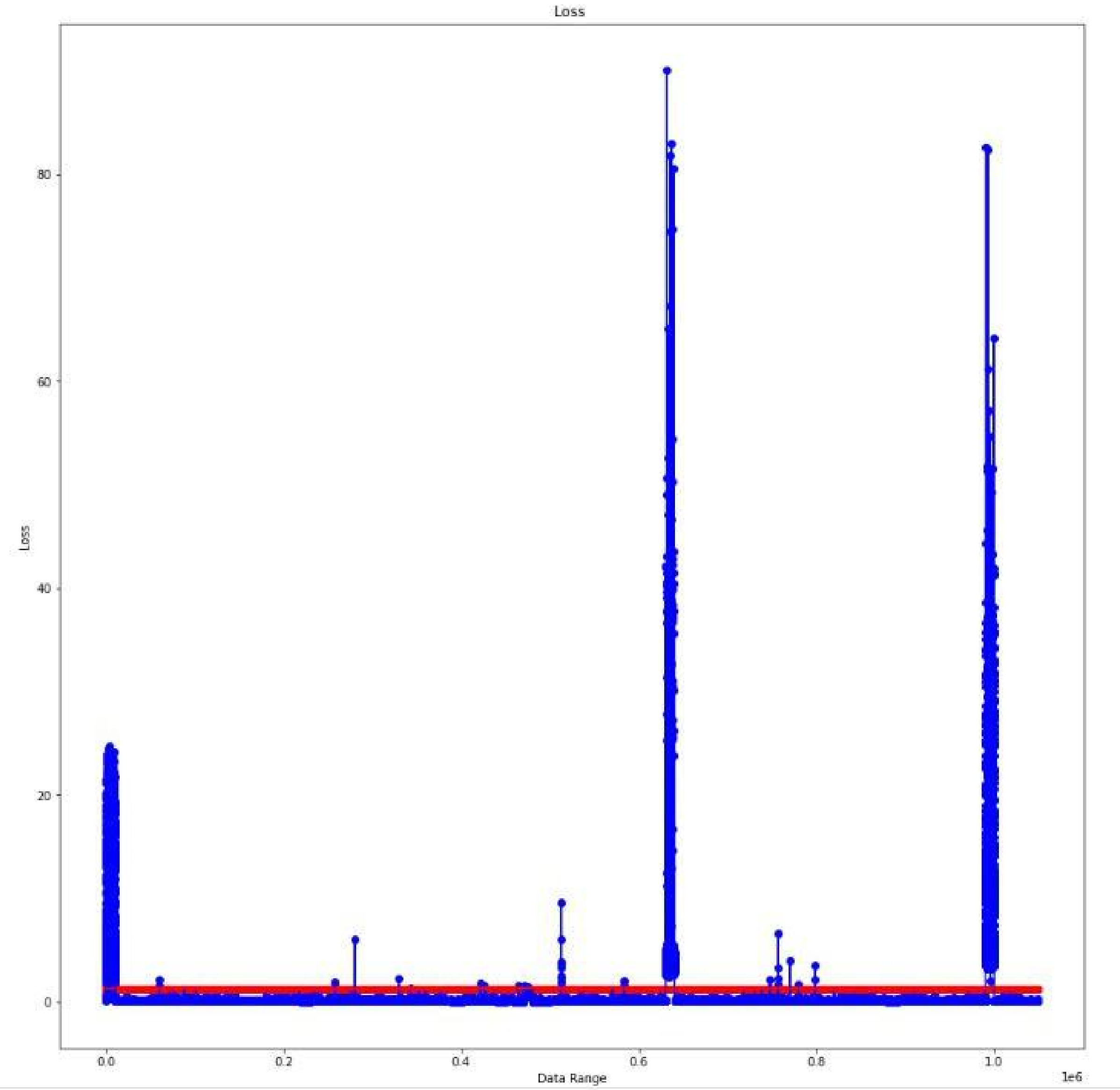

Figure 46 LSTM02 frequency vibration approach Loss to time graph

GRU01 frequency

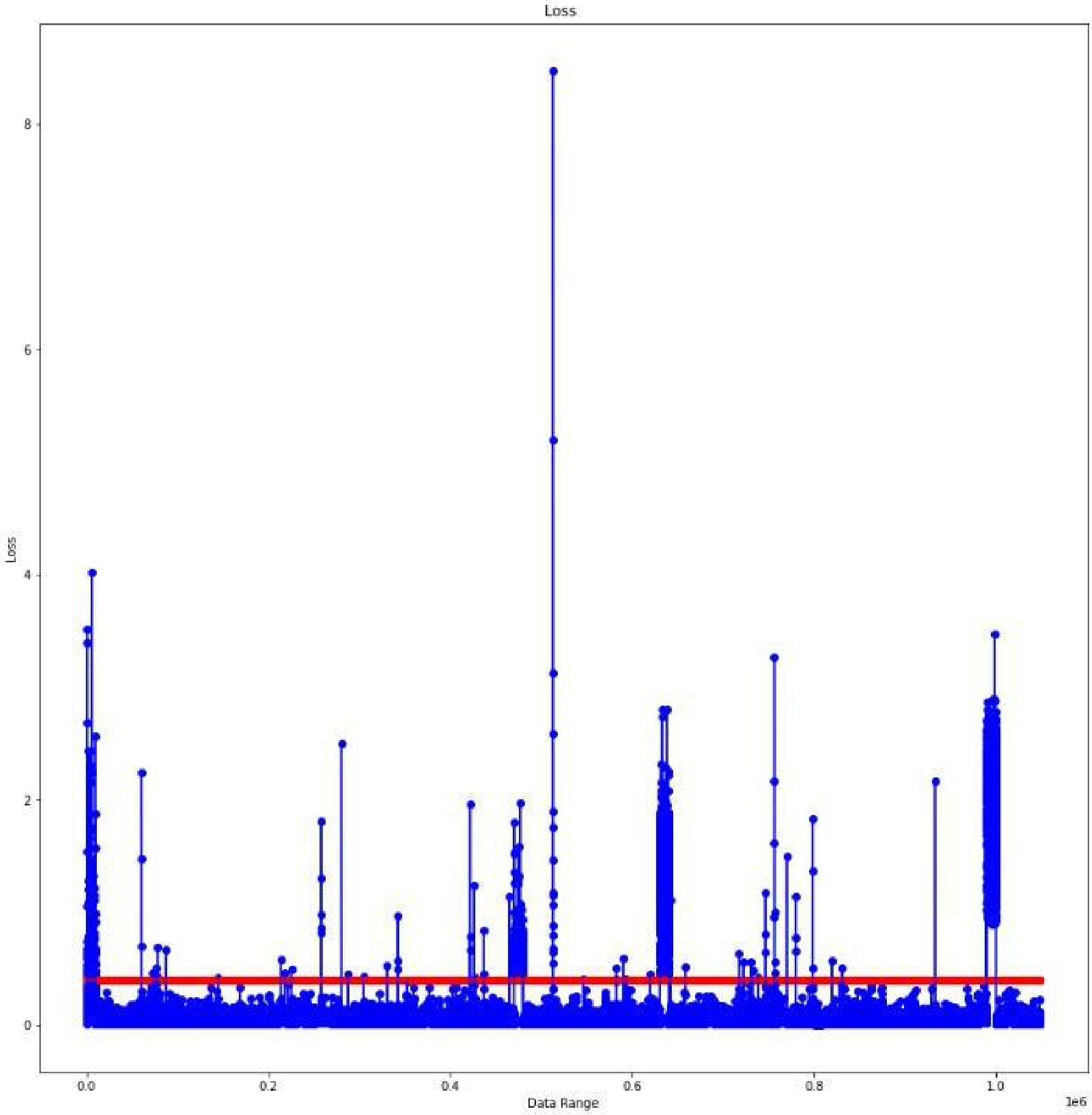

Figure 47 GRU01 frequency vibration approach Loss to time graph

Bi_GRU01 frequency

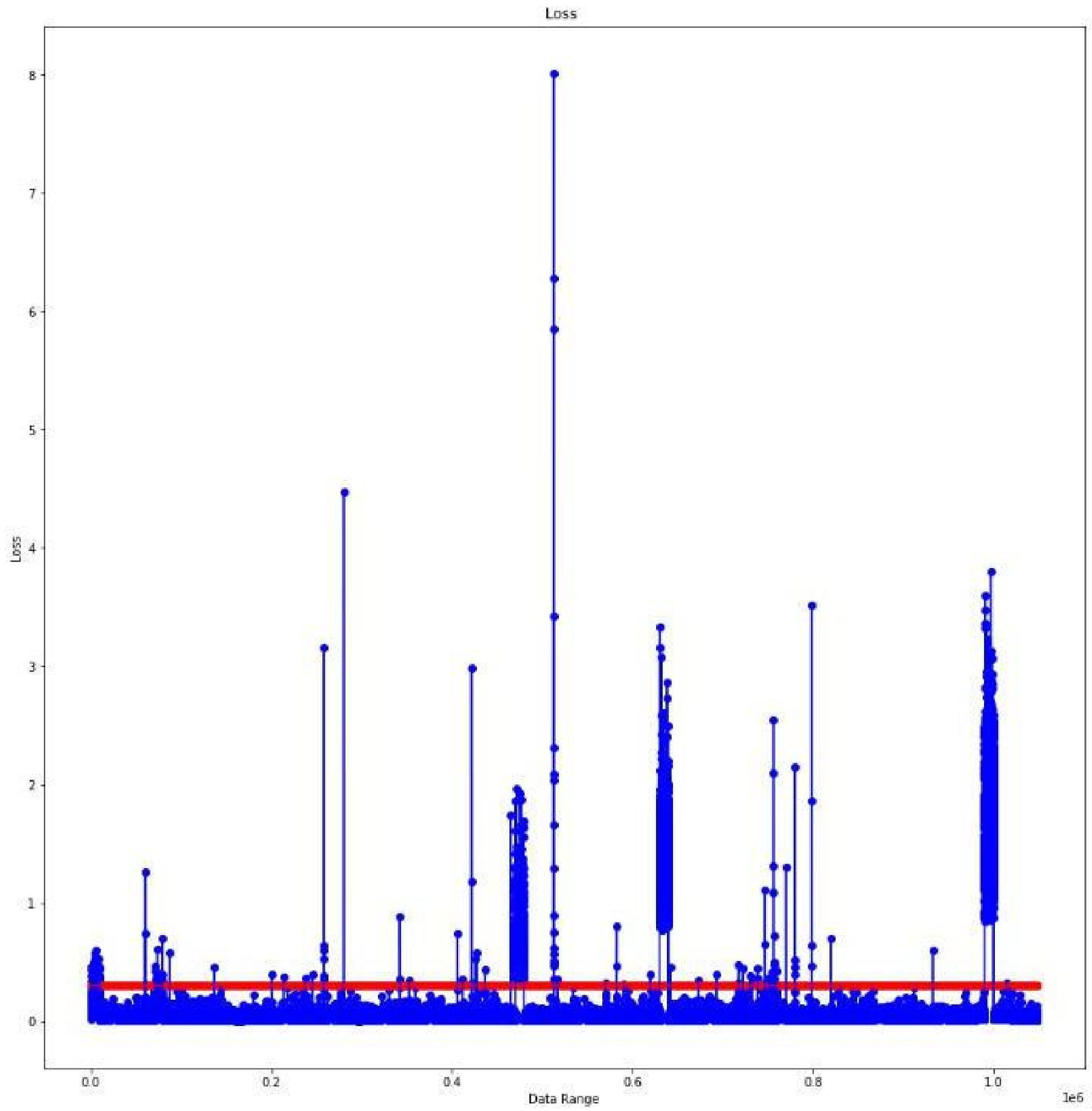

Figure 48 Bi_GRU01 frequency vibration approach Loss to time graph

Appendix II

Libraries used

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sb
import plotly.express as px
from scipy.fftpack import fft
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
import seaborn as sns

from tensorflow.keras import Sequential
from tensorflow import keras
from tensorflow.keras import layers
from keras import regularizers
from keras.layers import Input, Dropout, Dense, LSTM, TimeDistributed, RepeatVector
from keras.models import Model
```