TRIBHUWAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS, PULCHOWK

A SYSTEM OF VEHICULAR MOTION SENSING OVER BRIDGE AND IMPACT ANALYST

By:

Amit Paudyal (073BEX404)

Nabin Rai (073BEX418)

Nirdesh Bhattarai (073BEX420)

Shiva Bhandari (073BEX442)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

APRIL, 2021

LETTER OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "**A System of Vehicular Motion Sensing Over Bridge and Impact Analyst**" submitted by **Amit Paudyal, Nabin Rai, Nirdesh Bhattarai and Shiva Bhandari** in partial fulfillment of the requirements for the Bachelor's degree in Electronics and Communication Engineering.

Supervisor: **Dr. Nanda Bikram Adhikari**, Associate Professor
Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus

Internal Examiner: **Prof. Dr. Ram Krishna Maharjan**, Head of Department
Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus

External Examiner: **Mr. Manoj Ghimire**, CEO
Raralabs

Head of Department: **Prof. Dr. Ram Krishna Maharjan**, Professor
Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus

DATE OF APPROVAL:

COPYRIGHT

ABSTRACT

The analysis of traffic load and its impact on road bridges has become an important method to ensure the longevity of the bridges. Heavy and rapid moving vehicles cause huge strain and deflection of bridges. The study of vehicular vibration response of bridges and its trend over a period allows the continuous assessment of bridge's deteriorating conditions. Visual inspections and surveys only give an outlook on the condition of the exterior of the bridge. A network of sensors measuring dynamic characteristics such as strain, deflection, and vibration can assist in understanding the overall bridge structure conditions. During the analysis, an array of high-precision inertial sensors were deployed on both the old and the new sections of the Thapathali-Kupondole bridge situated over the Bagmati river. Data is extracted from the Inertial Measurement Unit (IMU) sensors. Visualization of obtained data gives the vibrations in different sections of the bridge. Similarly, statistical analysis shows the nature of obtained data and machine learning algorithms have been applied to detect anomalies and predict future trends. Isolation forest and LSTM autoencoders detect the occurrence of high vibration events, due to heavy vehicles, and flag them as anomalies. Furthermore, ARIMA models allow for efficient modeling of the trends of vibrations. Frequency domain analysis of vibration signals has also proved fruitful in extracting dominant frequency components. A simple prototype of the data acquisition system has also been successfully tested to record, store and retrieve the sensor data which indicates that an array of IMU sensors can be used for impact analysis as well as structural health monitoring.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to the Department of Electronics and Computer Engineering for providing us with this opportunity to carry out the project.

We would also like to extend our gratitude to Dr. Nanda Bikram Adhikari sir for providing us with required platform and motivation. It would not have been impossible to complete the project without his scholarly advice and expert guidance.

We would also like to thank Technology Sales Pvt. Ltd. for supplying us with all the necessary equipment as well as to all our seniors and friends who have guided us through this project.

We would like to acknowledge all the authors of the research papers and other articles that helped us understand the required concepts and algorithms better.

Amit Paudyal
073bex404.amit@pcampus.edu.np
Nabin Rai
073bex418.nabin@pcampus.edu.np
Nirdesh Bhattarai
073bex420.nirdesh@pcampus.edu.np
Shiva Bandhari
073bex442.shiva@pcampus.edu.np

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AIC	Akaike Information Criterion
ARIMA	Auto Regressive Integrated Moving Average
CPU	Central Processing Unit
CSV	Comma Separated Values
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
I/O	Input Output
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
LSTM	Long Short Term Memory
MQTT	Message Queuing Telemetry Transport
NN	Neural Network
PSD	Power Spectral Density
RNN	Recurrent Neural Network
SARIMA	Seasonal Auto Regressive Integrated Moving Average
SHM	Structural Health Monitoring
TSV	Tab Separated Values

1. INTRODUCTION

1.1. Background

One of the key infrastructures of transportation is bridges in the context of Nepal. It has extreme significance in public transportation. The safety and serviceability condition of such infrastructure is now a critical issue. There have been many accidents about bridges collapsing due to aging and being used past their life expectancy which has cost lives and properties. Also due to the high demand for transport capacity, heavy traffic loads have been detected in the bridges which degrade their structures more rapidly. In Nepal, mostly in Kathmandu, the bridges suffer from the heavy traffic of vehicles. This results in the unwanted vibration of the bridges which leads to degradation of the structures. Moreover, these bridges are used past their life expectancy without proper monitoring and examining parameters that continuously make bridges structurally deficient. According to the reports many bridges have been in operation past their ages without monitoring those entities that deteriorate the quality of the bridges. Thus, it is of utmost importance to safeguard and properly monitor such critical infrastructure to prevent any life-threatening situation from surprising us. Thus, the project features one of the highly busy and heavily loaded bridges of Kathmandu valley, the Thapathali bridge, which links two districts Kathmandu and Lalitpur.

There are two bridges at Thapathali, an old one and a new one. This project focuses on real-time sensory data collected from these bridges, their storage, and sensory data analysis and visualization to classify anomaly parameters. Similarly, a data acquisition system was developed using Raspberry Pi. The data acquisition system receives stores and displays data from wireless sensor nodes in near real-time. This project also focuses on frequency domain analysis of collected vibration data.

1.2. Problem Statement

According to Nepal Bridge Standards-2067, all permanent bridges shall be designed with a minimum operational life of 50 years, but there have been several cases of bridges collapsing prematurely resulting in loss of properties and even lives. The operational life of such structures indicated at design time does not match the actual operational life in most cases. This could easily be the result of improper construction work but could also be the result of ignoring the dynamic environmental factors and mismanaged load and traffic conditions. These factors significantly deviate or diminish the life span of structures like the bridge. All of the above-mentioned points further supplement the need for a periodic monitoring and inspection system.

One of the places with the strongest need for such a system is the Kathmandu valley. The densely populated capital is the most active city in the whole country. And, with the increase in population, increases the demand and utilization of infrastructures like transportation. According to the Traffic Division, roughly 3.23 million vehicles have been registered across the country in the fiscal year 2074/75, and among them, more than 1.17 million vehicles were registered in the Bagmati zone alone. This means that roads in Kathmandu look like parking spaces due to heavy traffic conditions. One of the heavily jammed places is the Kupondole-Thapathali bridge. It is clear that the actual age of the bridge is severely affected and is quite less than the initially predicted value due to heavy vehicular loads or traffic congestion. To add to the condition there are dynamic load effects, meaning that the bridges suffer from varying loads of vehicles falling under different categories according to their weight. So, the collapse of many bridges is simply a result of ignorance and a serious lack of information about the geometric and dynamic characteristics of the bridge.

Thus, a system that can monitor the operating conditions in real-time and detect anomalies along with the associated risk is required. The system will also provide ideas on how to manage traffic congestion to keep the load on the bridge within a limit. The amount of research in this particular field has not been as sound as it needs to be. The major constraints being the complexity of the environment under study and its components.

1.3. Objective

The main objective of this project is to design a data acquisition system capable of handling large amounts of sensor data, process the data and perform analysis on the data to design a prediction model to accurately monitor the condition of bridge under different type of stress applied.

Our objectives can be broken down and listed as below:

- To design an instrumentation system capable of vehicular motion sensing over bridge.
- Pre-processing of synchronized data for cleaning up irregular and redundant data.
- Analysis and visualization of enormous volumes of data using various tools.
- To develop a system capable of monitoring the motion of vehicles and the condition of bridges and predict associated risks, in the long run, using machine learning.
- To record and separate the occurrence of high amplitude vibrations on the bridge.

1.4. Application

The completed form of our project is expected to have following applications:

- This project can be utilized in analyzing the impact of vibration on old structures and designing appropriate renovation methods.
- This project can be utilized by automobile companies in assessing the impact of different vehicles on different surfaces.
- This project may be helpful for structural engineers to analyze different properties of bridges like natural frequencies and material characteristics.
- This project can be helpful for transportation officials to assess the impact of different vehicles based on their size and weight.

1.5. Feasibility Analysis

To analyze the viability of the project idea, a feasibility assessment is performed. It is up to the project undertakers to determine if the project can effectively solve the problem as stated in the problem statement. This is done through a feasibility assessment. And based on analysis, the decision is taken whether to proceed, postpone or cancel the project. There are several entities or components in our project and require feasibility analysis not only to carry out the project but also to know if the project has to be redesigned or scalable or not. The following areas fall under feasibility assessment:

1.5.1. Economic Feasibility Analysis

Economic feasibility checks whether the cost required for complete system development is feasible using the available resources in hand. It is worthwhile to note that the cost of the resources and overall cost deployment of the system should be kept at a minimum while the operation and maintenance cost for the system should be within the capacity of the organization and the data visualization and analysis tools are all open sources and the sensors implemented are optimum in cost and performance, the system can be considered economically feasible for the development. However, we consistently need to perform economic feasibility through each further step ahead.

1.5.2. Technical Feasibility Analysis

Technical Feasibility Analysis examines whether the proposed system can be designed to solve the defined problems and meet requirements using available technologies in the given problem domain. The system is said to be feasible technically if it can be deployable, operable, and manageable under the current technological context of the global market. Numerous factors are associated with the assessment of technical feasibility such as the right choice of technology type, use of standard technology, and familiarity of project team members with the technology used. The sensors used for the project are not easily available in the market. Thanks to Technology Sales Pvt. Ltd. for providing the necessary sensors and supporting us technically. For the accurate and precise measurement of data from the sensor, it is calibrated extremely well and a separate and easily operable decompiling software has been used for extracting sensor data. To transform and understand data such as

trends and insights, visualization tools are implemented by the use of open-source resources easily available. Hence, the project can be considered technically feasible.

1.6. System Requirement

1.6.1. Hardware requirement

- Inertial Measurement Unit (IMU) Sensor.
- WiFi Module
- Raspberry Pi Microcontroller
- Camera

1.6.2. Software requirement

- Python.
- Machine learning library and frameworks (TensorFlow, Keras, etc.)
- Data Analysis and Visualization libraries (Pandas, matplotlib, etc.)
- Node-RED
- Arduino IDE
- InfluxDB
- Mosquitto
- Grafana

2. LITERATURE REVIEW

Time series prediction has been one of the most widely used fields in the machine learning environment. Machine learning techniques have been extensively used to analyze and manipulate different structure parameters and also predict the vibration data based on historical data. The Nationwide bridge survey uses deep learning techniques [1] and a model-free damage detection approach uses machine learning techniques [2] to evaluate the bridge serviceability in accordance with real-time sensory data or archived bridge-related data such as traffic status, weather conditions, and bridge structural configuration. Applying deep learning [3]and machine learning perspective to Structural Health Monitoring (SHM) consists of training algorithms so that it can classify the structure state of health-based only on data about a given damage-sensitive feature [4] and a Wireless Sensor Network (WSN) for SHM is designed [5]. LSTM NN can achieve the best prediction performance in terms of both accuracy and stability [6]. The algorithms for vehicle detection and axle estimation based on the findings are derived and their performance is evaluated and discussed [7]. Since different machine learning models can provide good predictions and results so different approaches can result in better performances. Moreover, hybrid models of ARIMA and ANNs are often compared with mixed conclusions in terms of the superiority in forecasting performance [8, 9]. The LSTM prediction method has an excellent prediction capability which can be used to predict the future deflection values with good accuracy and mean square error [10]. Similarly, frequency domain analysis can be crucial in the case of the behavior of structures such as bridges based on the vibration data available. The analysis of the dynamic behavior of the structure can be done using spectral analysis methods [11]. Different anomaly detection techniques can provide useful information regarding structures from accelerometer vibration data results. One of the anomaly detection algorithms is isolation forest. It is used in time series data for detecting anomalies such as one mechanism to minimize the risk of fraudulent credit card transactions utilize a detection technique for ongoing transactions [12].This paper presents the performance analysis of isolation forest algorithms in fraud detection of credit card transactions. Similarly, the isolation forest algorithm is effectively used to detect anomaly instances for the streaming data [13] wherein such data any traditional anomaly detection algorithm that

attempts to store all the dataset fails to perform successfully. Likewise, the LSTM autoencoder algorithm which uses the reconstruction loss parameter for detecting anomalies is also widely used for time-series datasets. Cyber security [14] is one application of LSTM autoencoder where intrusion detection is evaluated based on determining the probability of a sequence being reconstructed by the model representing normal data. The sequences with a low probability value are classified as an anomaly. Time series anomaly detection using LSTM autoencoder network with one-class support vector machines provides better performance in detecting anomalies in sales [15].

3. THEORETICAL BACKGROUND

3.1. Time Series

Time series data refers to a sequence of observations recorded at regular intervals. The regular intervals can be a year, month, week, day, hour, minute, seconds, or even milliseconds. This frequency of observation defines the type of time series. Sensor data for anomaly detection and forecasting, server performance like CPU usage, I/O load, memory usage, network bandwidth consumption, stock prices data to detect trends, weather data for forecasting are some examples of time series data. Time series data represents how a process changes over time. Time series analysis deals with the understanding of various aspects of the inherent nature of the series so that we are better informed to create meaningful and accurate forecasts. This also involves creating assumptions as well as interpreting the given series data. From the time series analysis, we can get a better picture of the components of the time-series data. A brief description of the time series components are as follows:

Trend: The trend is observed when we see an increasing or decreasing slope observed in the time series. This describes the increasing or decreasing behavior of the time series data. This is usually present for a longer period.

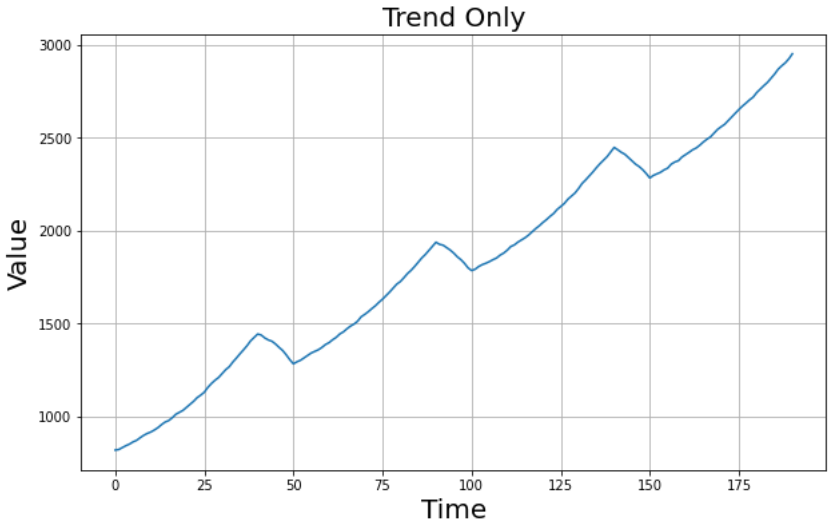

Figure 3.1: Trend

Seasonality: Seasonality is observed when there is a distinct repeated pattern observed

between regular intervals due to seasonal factors. This could be because of the month of the year, the day of the month, weekdays, or even time of the day.

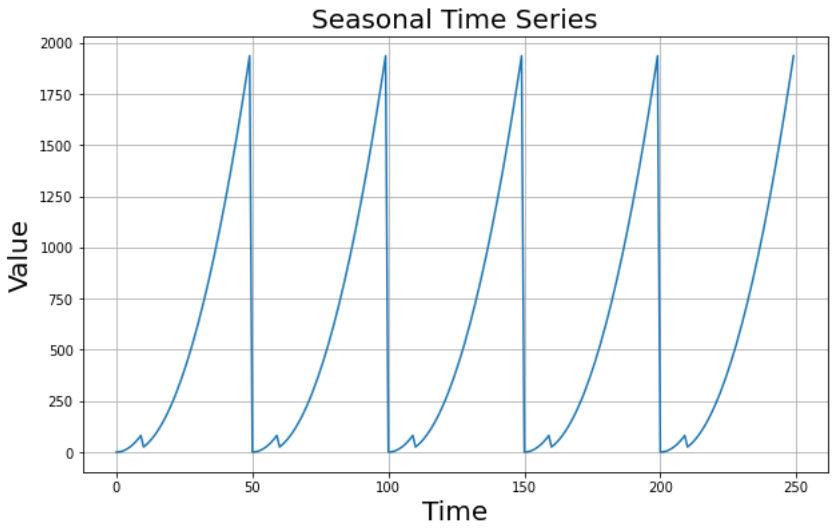

Figure 3.2: Seasonality

Cyclicity: It happens when the rise and fall pattern in the series does not happen in fixed calendar-based intervals. That means if the pattern is not fixed calendar-based frequencies then it is cyclic.

3.1.1. Stationary and Non-stationary Time Series

Stationarity is the property of time series data. Stationary series can be defined as the time series in which statistical parameters like mean, variance, and autocorrelations are constant over time as well as the seasonal components are not present. And the time series which does not have statistical parameters constant over time and seasonality components are present is a non-stationary time series. Since most of the statistical forecasting methods are designed based on stationary so non-stationary time series need to be converted to stationary time series by some transformation. Fig. 3.3 and Fig. 3.4 show the examples of stationary and non-stationary time series data.

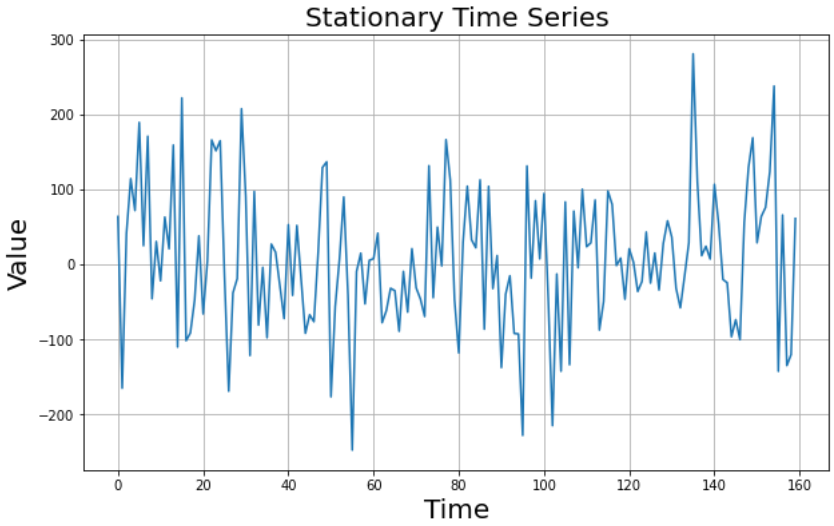

Figure 3.3: Stationary Time Series

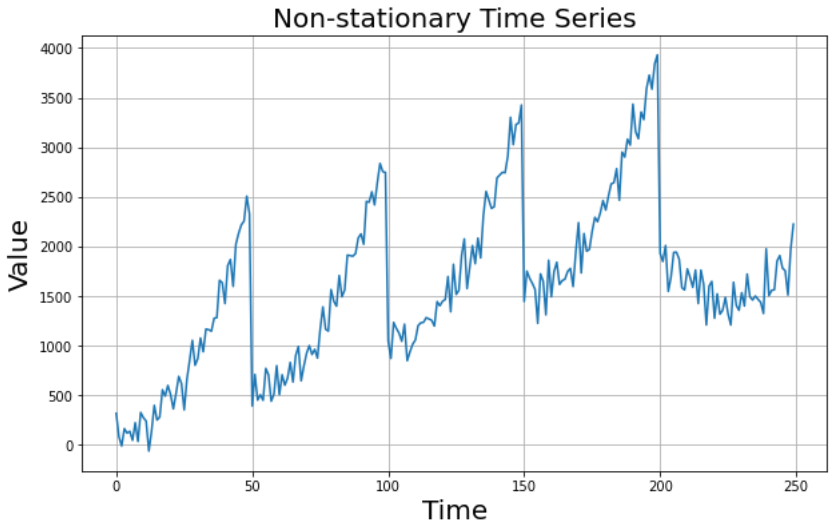

Figure 3.4: Non-stationary Time Series

Time Series Forecasting makes use of the best fitting model essential to predicting the future observation based on complex processing current and previous data. The ARIMA/SARIMA model, LSTM model, RNN model are some well-known models for time series forecasting.

3.2. LSTM

Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. The LSTM NN can overcome the issue of backpropagated error decay through memory blocks, and thus shows the superior ability for time series prediction with long temporal dependency.

Problem with RNN: Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they will have a tough time carrying information from earlier time steps to later ones. So, if you are trying to process a paragraph of text to do predictions, RNN's may leave out valuable information from the beginning. During backpropagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are those values used to update a neural network's weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it does not contribute too much learning. The gradient update rule can be illustrated from the following equation:

$$Newweight = weight - learningrate * gradient \tag{3.1}$$

So, in recurrent neural networks, if layers get a small gradient update then they stop learning. Those are usually the earlier layers. So, because these layers do not learn, RNN's can forget what is seen in longer sequences, thus having a short-term memory. The solution to short-term memory is LSTM. LSTM has gated architecture that solves the problem of learning relationships with long and short sequences in data. These gates regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions.All state-of-the-art results based on recurrent neural networks are achieved with the LSTM networks. LSTM is good at processing long sequences. The LSTM cell can be as illustrated as shown in Fig. 3.5.The operations within the LSTM cells are different from that of RNN's ones.

Concept: The main components of LSTM are cell states and gates. The cell state acts as

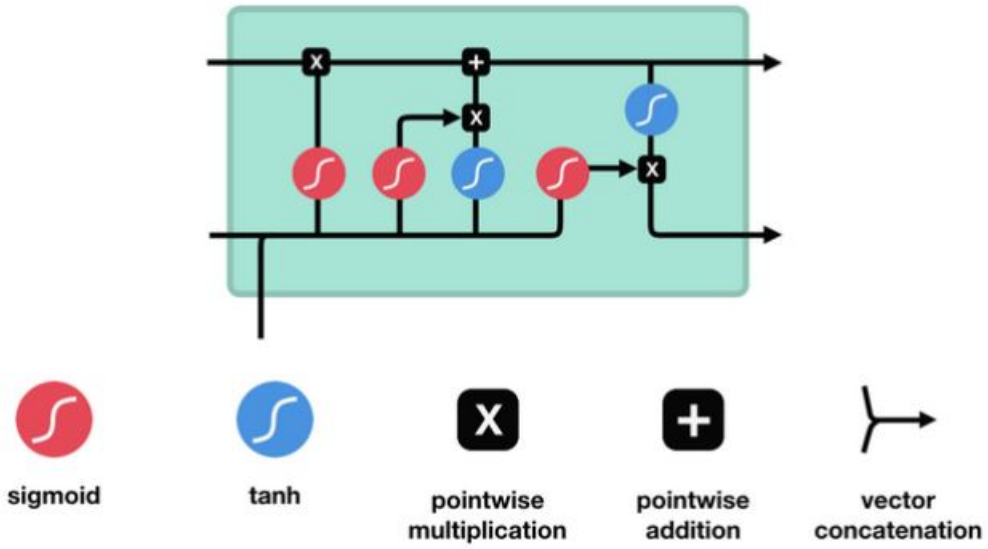

Figure 3.5: LSTM cell and its operation

a “transport highway” that transfers relative information down the sequence chain. It can be thought of as the “memory” of the network. The cell state carries relevant information throughout the processing of the sequence. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information gets added or removed to the cell state via gates. The gates are different neural networks that decide which information is to be allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

Tanh activation: The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1. It ensures that the values stay between -1 and 1, thus regulating the output of the neural network. You can see how the same values from above remain between the boundaries allowed by the tanh function.

Sigmoid activation: A sigmoid activation is like the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappear. The network can learn which data is not important therefore can be forgotten or which data is important to keep.

3.2.1. Types of Gates in LSTM

Forget Gate: This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

Input Gate: To update the cell state, we have the input gate. First, we pass the earlier hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. The hidden state and current input are passed into the tanh function to squish values between -1 and 1 to help regulate the network. Then the tanh output is multiplied with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output. After the input gate operations, there is enough information to calculate the cell state. First, the cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us our new cell state.

Output Gate: The output gate decides what the next hidden state should be. Here, the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. The tanh output is multiplied with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

In a nutshell, what these gates do is that the forget gate decides what is relevant to keep from prior steps; the input gate decides what information is relevant to add from the current step and the output gate determines what the next hidden state should be. Following are the relevant equations for these gates:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{3.2}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{3.3}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{3.4}$$

where, i_t, f_t and o_t represent input gate, forget gate and output gate; σ represents Sigmoid activation function; w_x represents weight for respective gate neurons; h_{t-1} represents output of the previous LSTM block; x_t represents input of current time stamp and b_x represents biases for respective gates.

The equations used for the cell state and the final output are as follows:

$$\hat{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c) \tag{3.5}$$

$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t \tag{3.6}$$

$$h_t = o_t * \tanh(c_t) \tag{3.7}$$

where, $\hat{c}_t$ represents candidate for cell state or memory at time stamp t; c_t represents cell state or memory at time stamp t and h_t represents output of the current lstm block; w_c represents weight for cell state or memory and bc represents bias for cell state.

3.3. Auto Regressive Integrated Moving Average (ARIMA)

An ARIMA model is a class of statistical models for analyzing and forecasting time series data. ARIMA is an acronym that stands for Autoregressive Integrated Moving Average. It is a generalization of the simpler Autoregressive Moving Average and adds the notion of integration. ARIMA models can capture some complex relationships as it takes account lagged or past values and error terms. This acronym is descriptive, capturing the key aspects of the model itself. The major components of the ARIMA model are composed of three elements each of which can be represented as a separate model as follows:

- **Autoregressive (AR):** Since the past values of time series data have a significant impact on the current and future values or time points so ARIMA model accounts for this concept through the AR component. This model uses the dependent relationship between an observation and some lagged observations. In other words, this model has a changing variable which depends on its lags or earlier values. A weight is assigned to each of the earlier terms by non-linear optimization methods. This model relies on auto-regression where autoregression can be thought of as the regression of a variable on past values of itself.

- **Integrated(I):** This part represents the use of differencing of raw observations to make the time series stationary. It also means that a time series that needs to be differenced to make it stationary is an integrated version of the stationary time series. If a time series is stationary, we do not need to differentiate the data. Here, the current data is replaced by the difference between the current data and closest previous data, i.e., current data(x_t) = current data(x_t) - previous data(x_{t-1})

- **Moving Average (MA):** This model uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

The non-seasonal ARIMA model can be viewed as ARIMA(p,d,q) model where;
p: Number of lagged or past terms in auto regressive part
d: Number of nonseasonal differencing required for making the time series stationary.
q: Number of lagged forecast errors or moving average term.

The general forecasting equation of non-seasonal ARIMA model can be shown as follows:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \varepsilon_t + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \ldots + \phi_q \varepsilon_{t-q} \quad (3.8)$$

where, Y_t represents the predicted value of y; p is the order of AR term and $\beta_1, \beta_2, ..., \beta_p$ represent the auto regressive parameters; q is the order of MA term and $\phi_1, \phi_2, ..., \phi_q$ represent the moving average parameters; ε_t represents error term at time stamp t and α represents a constant.

Identification of order of differencing in ARIMA model: To fit an ARIMA model the first step is to decide the order of differencing required for making the time series stationary. Usually, the correct amount of differencing is given by the lowest amount of differencing which yields a time series that fluctuates around a well-defined mean value and whose autocorrelation function (ACF) plot decays rapidly to zero. We need to perform extra differencing if the series still shows a long-term trend, or otherwise lacks a tendency to return to its mean value, or if its autocorrelations are positive out to a high number of lags. The stationarity of the time series can be checked with the help of Augmented Dickey Fuller's (ADF) test. It results in a test statistic and a p-value. This test has the following assumptions:

Null Hypothesis: p-value > 0.05; time series is stationary

Alternate Hypothesis: p-value < 0.05; time series is non-stationary.

Identification of AR and MA terms in ARIMA model: Once the time series is made stationary, the AR and MA terms are identified from the ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots. The ACF plot is the plot of the coefficients of correlation between a time series and lags of itself. The PACF plot is a plot of the partial correlation coefficients between the series and lags of itself. Partial autocorrelation is the correlation between the time series and its lag after excluding the effect of the intermediate lags. The AR term can be determined by observing and inspecting the PACF plot. Similarly, the MA term can be determined by inspecting the ACF plot.

3.4. Seasonal Auto-Regressive Integrated Moving Average (SARIMA)

The SARIMA model is similar to the nonseasonal ARIMA model in terms of structure but it employs seasonal AR and MA terms. Therefore, a seasonal ARIMA model can be

characterized as ARIMA (p,d,q) x (P, D, Q) where P, D, and Q represent the additional seasonal AR, seasonal differencing, and seasonal MA terms respectively.

3.5. Autocorrelation Function Plot (ACF)

Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis. Here we can infer that the Autocorrelation function plot will let you know how the given time series is correlated with itself.
Generally in an ARIMA model, we make use of either the AR term or the MA term. We use both of these terms on very rare occasions. We use the ACF plot to decide which one of these terms we would use for our time series:

- If there is a **Positive autocorrelation at lag 1** then we use the AR model
- If there is a **Negative autocorrelation at lag 1** then we use the MA model

After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed. The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags. At last, finding the correct model is an iterative process.

3.6. LSTM Autoencoder

LSTM autoencoder is an implementation of autoencoder for sequence data that uses an encoder-decoder architecture. Autoencoder is a neural network model that tries to learn a compressed representation of the input. It takes input data that is bigger and encodes it into smaller vectors. We can use LSTM layers to put them in the autoencoder type of way.
LSTM autoencoders can learn complex representations of sequence data and are used in image, video, text, audio, and time-series data. The LSTM network can be organized into an architecture called the Encoder-Decoder LSTM that allows the model to be used to both

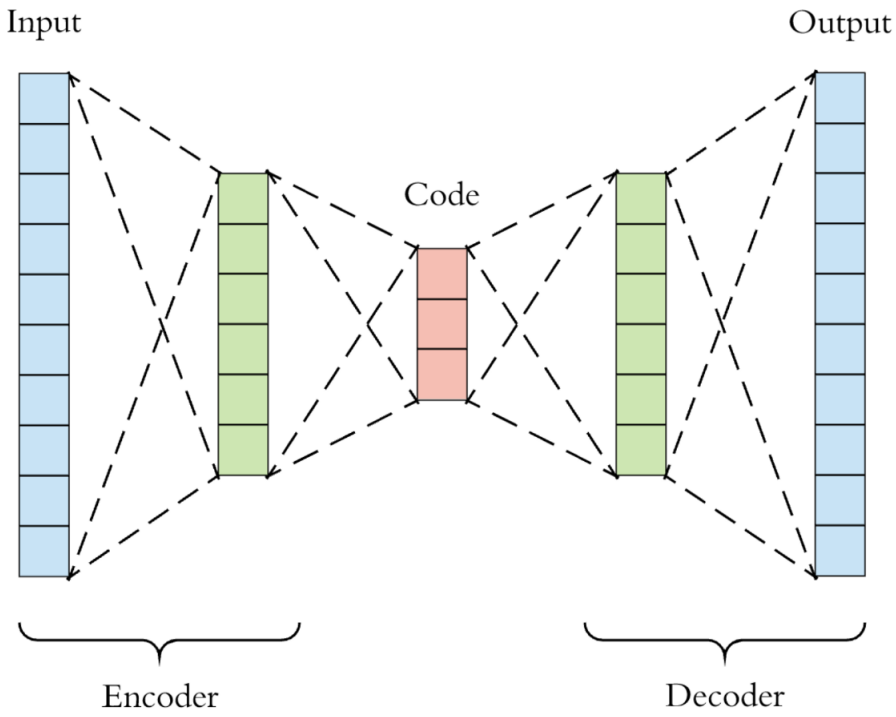

Figure 3.6: LSTM autoencoder architecture

support variable-length input sequences and to predict or output variable-length output sequences. Fig. 3.6 depicts the general architecture of LSTM autoencoders. In other words, the LSTM autoencoder architecture enables capturing temporal dependencies of the data. The encoder LSTM model reads the input sequence step by step. After reading in the entire input sequence, the hidden state or output of this model represents an internal learned representation of the entire input sequence as a fixed-length vector. This vector is then provided as an input to the decoder model that interprets it as each step in the output sequence is generated. This method can be used for anomaly detection. When we train the LSTM autoencoder the objective is to reconstruct the input sequence as best as possible. This is accomplished by minimizing a loss function known as reconstruction loss. Some common examples of reconstruction loss are cross-entropy loss and mean square error. Once the training data is completed then we look at the new data that is coming and then reconstruction loss is looked at and if the reconstruction loss is within acceptable limits then we say that there is no anomaly but if it is not at an acceptable limit then it is an anomaly. In LSTM autoencoders the layers are that of LSTM.

3.7. Isolation Forest

Isolation forest is an unsupervised learning algorithm mainly for anomaly detection that works on the principle of isolating anomalies. An anomaly is an observation or an event that deviates a lot from the other events present in the dataset. In very big datasets, all the anomalies present may not be detected by human eyes as they may follow very complicated patterns. Many other techniques used for anomaly detection are based on building a profile of so-called what is "normal" and anomalies are identified as those events or observations in the dataset that do not conform to the normal profile.

In the case of Isolation Forest, it is not the same. It is based on the decision tree algorithm. Instead of building a normal profile or a model of normal events, it explicitly isolates anomalous points in the dataset. In comparison to other profile-based anomaly detection techniques, this technique creates a fast algorithm (low time complexity) with a low memory demand. It is also known as **iForest**.

The core algorithm was initially proposed by Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou [16] in 2008 based on the assumption that the anomalous data are few and different from other normal data points. It follows that since the anomalies are few and different they are relatively easier to isolate than normal data. It builds an ensemble of **isolation trees(iTrees)** for the dataset. The anomalies are the points that have shorter path lengths on the iTrees.

To isolate a data point, the algorithm recursively generates partitions as shown in Fig. 3.7 on the sample by randomly selecting an attribute and then randomly selecting a split value for that attribute between the minimum and maximum values allotted for that attribute. However, from the analysis, it is clear that the anomaly point requires lesser partitions to isolate than the normal points.

The recursive partitioning is represented using a tree structure called isolation tree and the number of partitions required to isolate a point from other points is represented as the length of the path within the tree, in reaching the terminating node starting from the root.

Let X = $\{x_1,....,x_n\}$ be a set of d-dimensional points where $X^1 \subset X$. An isolation tree is defined as a data structure that has following properties:

- For each node T in the tree, T is either an external node with no child or an internal

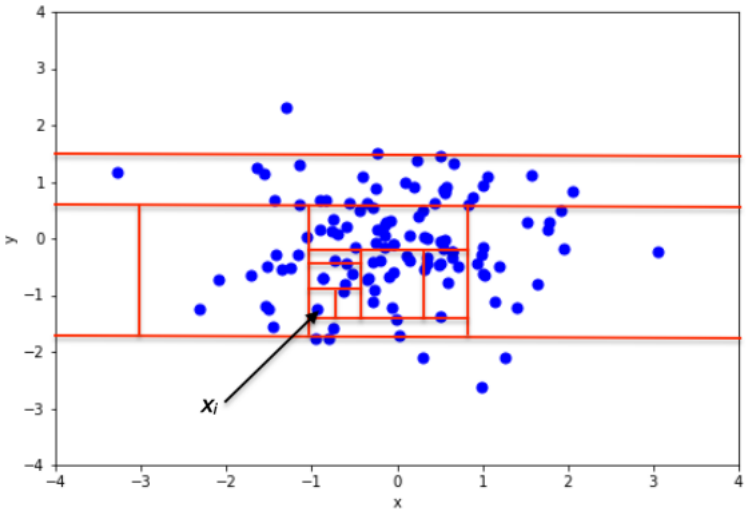

Figure 3.7: Representation of Isolation Forest Partitioning of a Point

node with one test and two daughter nodes T_l and T_r.

- A test at node T consists of an attribute q and a split value p such that test p<q determines the traversal of a data point to either T_l or T_r.

Now, to build an iTree, the algorithm recursively divides the X^1 by randomly selecting an attribute q and a split value p until the node has only one instance or until all the data at the node have the same values.

When the iTree is fully grown each point in X is isolated at one of the external nodes. The anomalous points are those that have shorter path lengths. Since the isolation forest does not need to isolate all the normal data points it can frequently ignore the large majority of the training sample. Thus it works well even when the sample size is small.

Some properties of isolation forest are as follows:

Swamping: When normal instances are too close to anomalies, the number of partitions required to separate anomalies increases, a phenomenon known as swamping, which makes it more difficult for iForest to discriminate between anomalies and normal points. One of the main reasons for swamping is the presence of too much data for anomaly detection, which implies one possible solution to the problem is sub-sampling. Since iForest responds very well to sub-sampling in terms of performance, the reduction of the number of points in the sample is also a good way to reduce the effect of swamping.

Masking: When the number of anomalies is high, it is possible that some of those aggregates

in a dense and large cluster, making it more difficult to separate the single anomalies and, in turn, to detect such points as anomalous. Similar to swamping, this phenomenon is also more likely when the number of points in the sample is big and can be alleviated through sub-sampling.

3.8. Frequency Domain Analysis

The time-domain analysis of the vibration data limits us with few parameters to quantify the strength of vibration profile such as amplitude, peak to peak, and RMS value. However, in vibration analysis, understanding the frequency profile is crucial. Therefore, the sensor data in the time domain is converted to the frequency domain using appropriate algorithms. This allows us to visualize the effects of noise filtering and other windowing and filtering techniques. By frequency domain analysis we can observe the dominant frequencies involved during the passing of light, medium, and large weight vehicles.

3.8.1. Fast Fourier Transform (FFT)

It is one of the most widely used algorithms to calculate the Discrete Fourier Transform (DFT) of a signal. It does not represent a transform different from the DFT but it is a special algorithm for faster implementation of DFT. FFT requires comparatively fewer arithmetic operations (multiplications and additions) than DFT which reduces the computational time than that of DFT. The fundamental principle of all FFT algorithms is based on that of decomposing the computation of DFT of a sequence of length N into successively smaller DFTs. The complexity of the DFT can be related to $O(N^2)$. This complexity is reduced to $O(N \log_2 N)$ in FFT which is a lot in terms of computational time. The standard N-point DFT of a length-N sequence is given by:

$$X_k = \sum_{n=0}^{N-1} x[n] exp(-i2\pi kn/N) \qquad 0 \leq k \leq N-1 \tag{3.9}$$

There are different variations or versions for implementing the FFT of signals such as decimation-in-time FFT, decimation-in-frequency FFT but the basic idea is the same as to reduce the computational complexity of DFT.

3.8.2. Power Spectral Density (PSD)

In the real-world vibration data, there can be random vibrations which means they can have multiple frequencies at the same time. When there are a finite number of dominant frequencies, the performance of FFT is good but when analyzing the random vibration signals, the power spectral density is used. The power spectral density comes into action when dealing with random signals or random processes. The PSD characterizes the random vibration signals. It determines how the power of a given signal is distributed over frequency. Since it deals with stochastic processes, we can only estimate the power spectral density. One of the forms of the estimation is squaring the FFT with its conjugate form, i.e., taking the square of the absolute value of FFT. The equation used for estimating PSD using the FFT is as follows:

$$P(k) = \frac{1}{N}|X(k)|^2 \tag{3.10}$$

4. METHODOLOGY

4.1. System Block Diagram

The overall system block diagram of our project is depicted in Fig. 4.1. The description of these blocks are described in subsequent subsections.

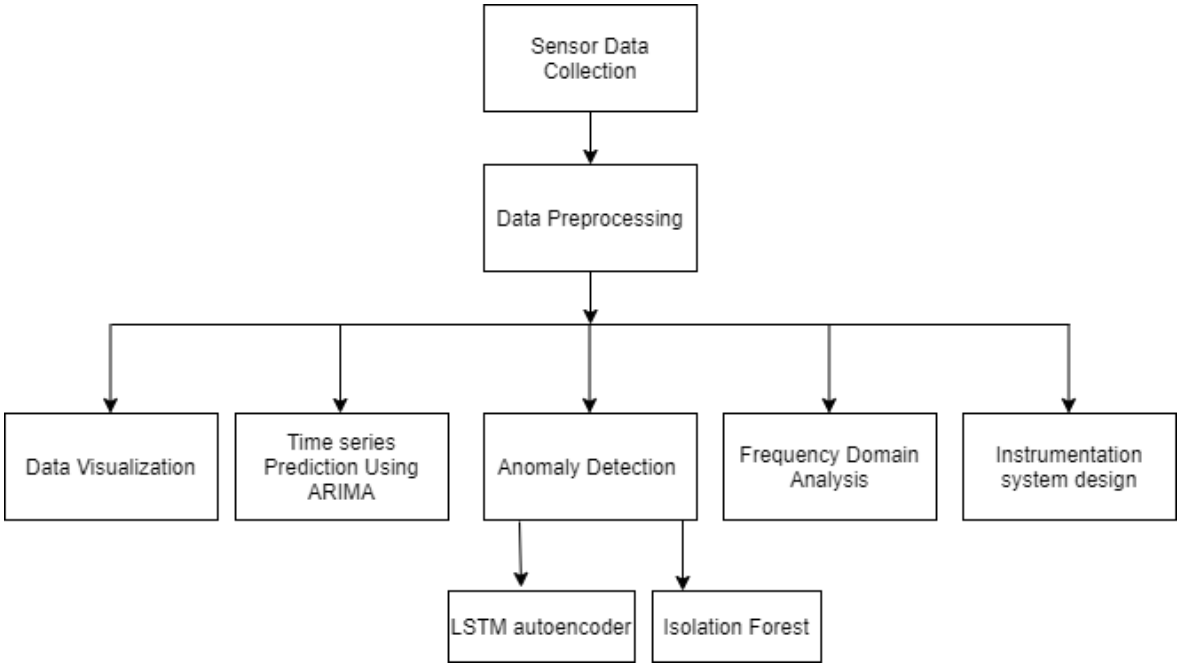

Figure 4.1: System Block Diagram

4.2. Sensor Data Collection

There are two beam-type bridges each having a length of 184m and a width of 6m. The old bridge was constructed in 1967 AD whereas the new one was built in 1995AD. Before sensor installation, it was essential to identify and locate the sections for optimum vibration recording. So, testing of possible sites for sensors was done at five different locations on the bridge road surface. Fig. 4.2 depicts the test points for observation before selecting the sensor deployment location.

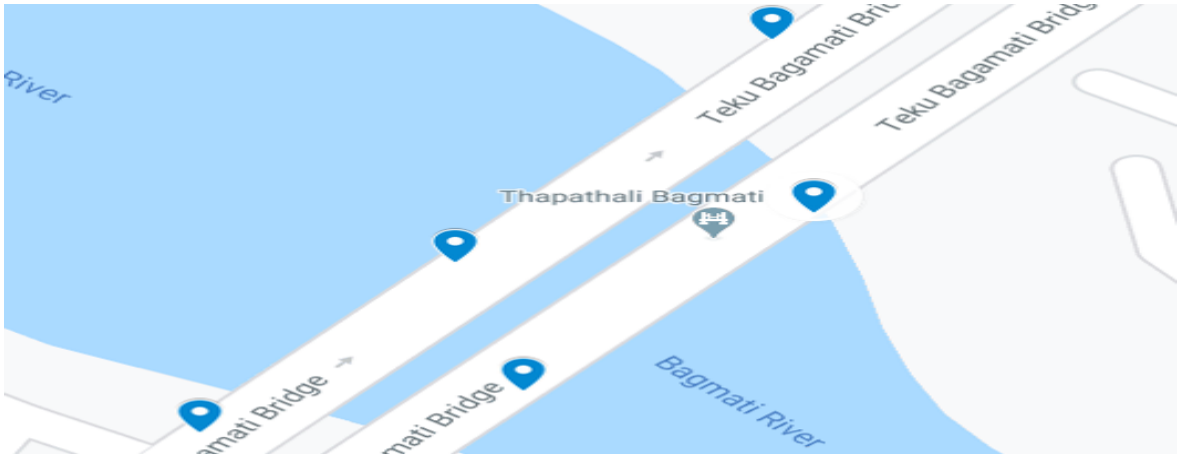

Figure 4.2: Test points before sensor installation

Case 1: When the Sensors are placed under the Bridge

Six sections along with the sensor's orientation arrangement were selected for sensor deployment under the Thapathali-Kupondole bridge as depicted in Fig. 4.3. The sensors were fixed on the structure under the bridge. The data were extracted manually from the SD Card on the sensor at a fixed time interval. The sensor installation line was selected under the bridge as shown in Fig. 4.4.

Figure 4.3: Illustration of Sensors installation in the Thapathali-Kupondole bridge.

Inertial parameters and temperature values were collected over 10 days from 2020-02-18 to 2020-02-28 with a sampling rate of 200 ms. The dataset consists of 13 fields covering acceleration, angular velocity, angular displacement, and magnetic field in three directions and a temperature parameter. However, we discuss and analyze the vibration(acceleration)

Figure 4.4: Sensor Installation Site

of the bridge in the vertical direction (az) in this section. Due to interruptions and failure in the sensors, a significant amount of data has been extracted only through sensor 1 and sensor 4. As we can observe that the data collection period for this case is longer, i.e., in days.

Case 2: When the sensors are placed on the surface of the bridge

This time around the sensors was placed at the surface of the bridge. Midpoints of the spans of the bridge along the footpath were used as data collection points as shown in Fig. 4.5. It was observed that the amplitude of vibration was more prominent when the sensors were placed at the surface of the bridge.

Figure 4.5: Sensor Installation Surface

It is worthwhile to note that during this case of sensor data collection the sensor data collected are of shorter duration as it was not possible to record on the surface for a longer duration and this data was mostly fruitful for anomaly detection and frequency domain analysis purposes.

4.3. Data Pre-processing

The text file obtained from the sensor was converted to a .csv file format and read as a pandas data frame. The data frame sample before the preprocessing step is shown in Fig. 4.6.

	address	Time(s)	ChipTime	ax(g)	ay(g)	az(g)	wx(deg/s)	wy(deg/s)	wz(deg/s)	AngleX(deg)	...	GPSV(km/h)	q0	q1	q2	q3	SV
0	NaN	11:25:33.867	NaN	0.4375	-0.2700	0.7588	-132.0801	77.2095	46.3867	-14.2822	...	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	11:25:33.976	NaN	0.3345	-0.2969	0.7412	-73.9136	-0.4272	17.9443	-29.9762	...	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	11:25:33.976	NaN	0.2441	-0.5122	0.5308	-21.3013	2.1973	42.3584	-44.2584	...	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	11:25:33.976	NaN	0.1650	-0.8037	0.5757	-13.3667	-1.0986	27.7100	-53.4375	...	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	11:25:33.976	NaN	0.1196	-0.7998	0.5718	0.0610	-7.9956	31.1279	-55.1129	...	NaN	NaN	NaN	NaN	NaN	NaN

Figure 4.6: Data Sample Before Preprocessing

During the preprocessing step, the unnecessary data columns and NaN values were

removed. The missing values were interpolated. Then for smoothing data noise filtering was performed. The preprocessed data frame sample after the preprocessing step is shown in Fig. 4.7.

rtime	ax(g)	ay(g)	az(g)	wx(deg/s)	wy(deg/s)	wz(deg/s)	AngleX(deg)	AngleY(deg)	AngleZ(deg)	T(°)	hx	hy	hz
2020-02-18 01:30:00.000	0.4375	-0.2700	0.7588	-132.0801	77.2095	46.3867	-14.2822	-31.5033	7.9871	20.03	-260	-4	-57
2020-02-18 01:30:00.100	0.3345	-0.2969	0.7412	-73.9136	-0.4272	17.9443	-29.9762	-26.0815	9.4427	20.07	-285	108	-116
2020-02-18 01:30:00.200	0.2441	-0.5122	0.5308	-21.3013	2.1973	42.3584	-44.2584	-20.5115	18.0231	20.02	-326	182	-155
2020-02-18 01:30:00.300	0.1650	-0.8037	0.5757	-13.3667	-1.0986	27.7100	-53.4375	-12.7332	22.9120	20.00	-390	239	-180
2020-02-18 01:30:00.400	0.1196	-0.7998	0.5718	0.0610	-7.9956	31.1279	-55.1129	-8.9429	25.4498	19.99	-432	257	-198

Figure 4.7: Data Sample After Preprocessing

4.4. Time Series Prediction Using ARIMA

The methodology of ARIMA modeling can be illustrated in Fig. 4.8.

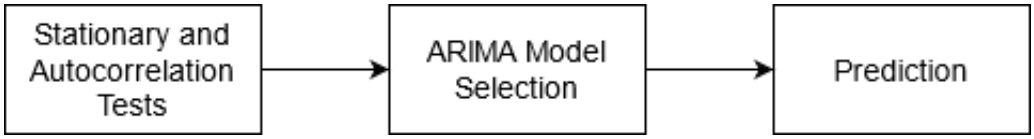

Figure 4.8: ARIMA Block Diagram

First, the original time series data and one-time differenced data were inspected for stationarity using the Augmented Dickey-Fuller test. The p-value is less than 0.05 for both cases. However, the one-time differenced series shows stronger stationarity. The summary of the dickey fuller test is as shown in Fig. 4.9 and Fig. 4.10.

```
Dickey Fuller Test:
Test Statistic                -1.641222e+01
p-value                        2.577130e-29
#Lags Used                     4.200000e+01
Number of Observations Used    1.410900e+04
Critical Value (1%)           -3.430814e+00
Critical Value (5%)           -2.861745e+00
Critical Value (10%)          -2.566879e+00
dtype: float64
```

Figure 4.9: Augmented Dickey Fuller test for one time differenced series

```
Dickey Fuller Test:
Test Statistic                  -3.060276
p-value                          0.029636
#Lags Used                      42.000000
Number of Observations Used    14110.000000
Critical Value (1%)             -3.430814
Critical Value (5%)             -2.861745
Critical Value (10%)            -2.566879
dtype: float64
```

Figure 4.10: Augmented Dickey Fuller test for non differenced series

Then for selecting the best fit of the ARIMA model the auto Arima function was used. This function fits the best ARIMA model to a univariate time series data according to a provided information criteria; AIC in our case. Auto Arima function iterates with our time series data and then returns models with corresponding AIC values as shown in Fig. 4.11.

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=759.824, Time=13.20 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=7998.326, Time=2.77 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=4282.238, Time=2.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=760.033, Time=7.92 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=7996.329, Time=0.25 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=760.619, Time=7.36 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=759.002, Time=22.13 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=761.862, Time=9.83 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=2628.822, Time=3.75 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=759.580, Time=13.57 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=1898.870, Time=3.46 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=761.666, Time=17.60 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=757.113, Time=1.98 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=759.977, Time=1.44 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=2626.834, Time=0.68 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=757.693, Time=3.23 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=757.454, Time=6.37 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=4280.246, Time=0.32 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=758.735, Time=1.87 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=1896.887, Time=1.30 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=759.778, Time=2.63 sec

Best model:  ARIMA(2,1,1)(0,0,0)[0]
Total fit time: 123.731 seconds
```

Figure 4.11: Auto ARIMA function summary

Again, setting the difference parameter to 0 in auto_ arima search, an ARIMA model(3,0,1)

was found to be the best fit as shown in Fig. 4.12.

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=760.961, Time=16.84 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=40168.474, Time=0.65 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=7636.376, Time=4.31 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=27985.508, Time=3.14 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=40166.474, Time=2.75 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=761.808, Time=38.29 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=761.814, Time=25.19 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=759.922, Time=27.61 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=758.742, Time=23.93 sec
 ARIMA(3,0,0)(0,0,0)[0] intercept   : AIC=inf, Time=5.36 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=759.432, Time=34.16 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=4158.129, Time=3.59 sec
 ARIMA(4,0,0)(0,0,0)[0] intercept   : AIC=inf, Time=6.38 sec
 ARIMA(4,0,2)(0,0,0)[0] intercept   : AIC=761.752, Time=46.77 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=757.112, Time=6.39 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=760.194, Time=5.64 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=inf, Time=1.02 sec
 ARIMA(4,0,1)(0,0,0)[0]             : AIC=757.801, Time=7.57 sec
 ARIMA(3,0,2)(0,0,0)[0]             : AIC=758.304, Time=7.87 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=4156.152, Time=0.77 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=758.868, Time=4.93 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=inf, Time=1.82 sec
 ARIMA(4,0,2)(0,0,0)[0]             : AIC=760.203, Time=5.53 sec
```

Figure 4.12: Auto ARIMA function summary (d=0)

However, when used for prediction in a new test set, both of these models fail. This is due to the presence of seasonality. Therefore, a Seasonal ARIMA(SARIMA) model was fitted and the best-fit parameters were obtained using the same *auto_arima*() function.

For seasonal modeling, the entire time series was first resampled to a rate of 2 hours to minimize the seasonal factor. From the first Dickey-Fuller test as shown in Fig. 4.13, the p-value is obtained as 0.11 which means the series is not stationary. Therefore, the time series was differenced once and the new p-value was obtained as 0.009 as shown in Fig. 4.14. From the auto_arima search as in Fig. 4.15, a SARIMA(0,1,1) (1,1,0,12) was found to be the best fit.

```
Dickey Fuller Test:
Test Statistic                  -2.501971
p-value                          0.114996
#Lags Used                      12.000000
Number of Observations Used    105.000000
Critical Value (1%)             -3.494220
Critical Value (5%)             -2.889485
Critical Value (10%)            -2.581676
dtype: float64
```

Figure 4.13: Augmented Dickey-Fuller test summary for SARIMA

```
Dickey Fuller Test:
Test Statistic                  -3.430914
p-value                          0.009949
#Lags Used                      12.000000
Number of Observations Used    104.000000
Critical Value (1%)             -3.494850
Critical Value (5%)             -2.889758
Critical Value (10%)            -2.581822
dtype: float64
```

Figure 4.14: Augmented Dickey-Fuller test summary for SARIMA after differencing once

```
ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=1.07 sec
ARIMA(0,1,0)(0,1,0)[12]             : AIC=139.061, Time=0.02 sec
ARIMA(1,1,0)(1,1,0)[12]             : AIC=129.693, Time=0.15 sec
ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.57 sec
ARIMA(1,1,0)(0,1,0)[12]             : AIC=139.102, Time=0.04 sec
ARIMA(1,1,0)(2,1,0)[12]             : AIC=130.193, Time=0.41 sec
ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=0.86 sec
ARIMA(1,1,0)(0,1,1)[12]             : AIC=inf, Time=0.59 sec
ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=2.21 sec
ARIMA(0,1,0)(1,1,0)[12]             : AIC=130.401, Time=0.08 sec
ARIMA(2,1,0)(1,1,0)[12]             : AIC=130.364, Time=0.20 sec
ARIMA(1,1,1)(1,1,0)[12]             : AIC=131.381, Time=0.30 sec
ARIMA(0,1,1)(1,1,0)[12]             : AIC=129.383, Time=0.13 sec
ARIMA(0,1,1)(0,1,0)[12]             : AIC=138.972, Time=0.04 sec
ARIMA(0,1,1)(2,1,0)[12]             : AIC=129.948, Time=0.32 sec
ARIMA(0,1,1)(1,1,1)[12]             : AIC=inf, Time=0.91 sec
ARIMA(0,1,1)(2,1,1)[12]             : AIC=inf, Time=2.20 sec
ARIMA(0,1,2)(1,1,0)[12]             : AIC=131.373, Time=0.19 sec
ARIMA(1,1,2)(1,1,0)[12]             : AIC=inf, Time=1.41 sec
ARIMA(0,1,1)(1,1,0)[12] intercept   : AIC=131.373, Time=0.33 sec

Best model:  ARIMA(0,1,1)(1,1,0)[12]
Total fit time: 12.077 seconds
```

Figure 4.15: Auto ARIMA function summary for SARIMA model

4.5. Anomaly Detection

4.5.1. LSTM Autoencoder

The methodology for LSTM autoencoder includes the following steps:

Sensor data collection: For the LSTM autoencoder model the sensor data collected are time series data taken at a frequency of 50Hz at different locations on the bridge for a relatively short interval of time of 10-15 minutes. The detailed process of sensor data collection is mentioned under the topic "Sensor Data Collection" above.

Data preprocessing step: For the data preprocessing step the data collected from sensors are transformed into such a state that is suitable for machine learning purposes. At first, the missing data are looked for checking missing values and secondly interpolating them as well as for removing NaNs. Then, the data preprocessing step also includes proper scaling of our data using either the *MinMaxScaler()* function or *StandardScaler()* function. The main feature of the dataset for anomaly detection is selected as vertical accelerometer vibration data, i.e., az(g) column.

LSTM Autoencoder architecture: An LSTM autoencoder structure is created. A sequential model in Keras is created and then LSTM layers are added with units specified. Here parameter 'return sequences' has to be true if we have to stack layers. The autoencoder architecture is provided with two components; encoder (for compressing the input) and decoder (for trying to reconstruct the input). Two LSTM layers are used in the encoder section which is used for compressing the time-series input data. Since LSTM autoencoders are provided with encoder and decoder parts so there is a repeat vector that is a bridge between encoder and decoder parts. The compressed representation is then decoded using a decoder which essentially has two LSTM layers. Finally, we have a time-distributed layer that outputs the same dimensions as input data. The optimizer used is adam optimizer and the loss function used is mean absolute error. A standard scaler is used which standardizes the features by removing the mean and scaling to unit variance. Similarly, a 'look back' parameter is also used which simply acknowledges the number of previous samples to be considered in our LSTM autoencoder model. Another important thing is to add dropout layers for coping with overfitting. Different models were tried and the model that gives the best fit for our dataset by tuning different hyperparameters was selected as the best among them. It was observed that LSTMs are a bit slow. The model

summary is as shown in Fig. 4.16.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128)               66560
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 40, 128)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 40, 128)           131584
_________________________________________________________________
dropout_3 (Dropout)          (None, 40, 128)           0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 40, 1)             129
=================================================================
Total params: 198,273
Trainable params: 198,273
Non-trainable params: 0
```

Figure 4.16: Model summary of LSTM autoencoder

The model is best fitted with given epochs=50, batch size=32, and validation split=0.2, learning rate = 0.005, which are the hyperparameters of the model. We observe the train and test set loss as shown in Fig. 4.17.

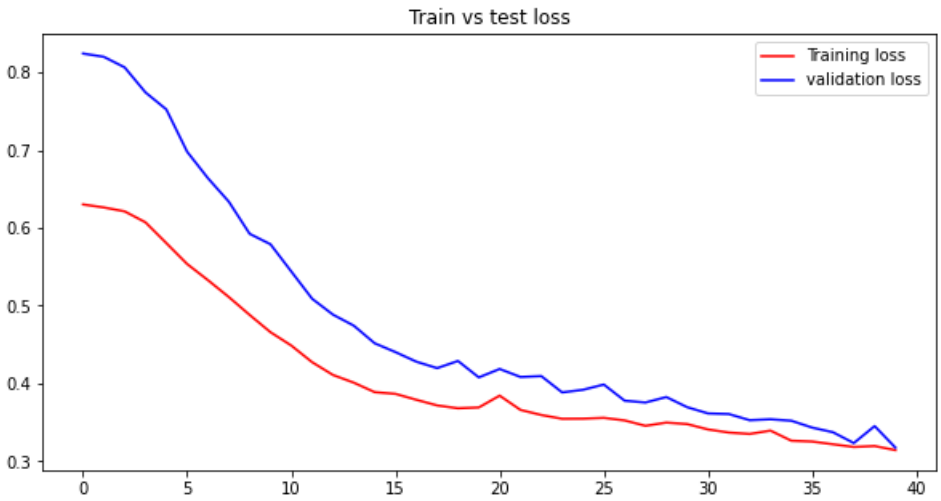

Figure 4.17: Train and test loss for epoch =50, batch size=32 and learning rate=0.005

The model is predicted in the train set and then its mean absolute error is calculated. For determining the threshold for anomaly detection the histogram plot is observed and the threshold is set at 90% distribution. The histogram plot is as shown in Fig. 4.18.

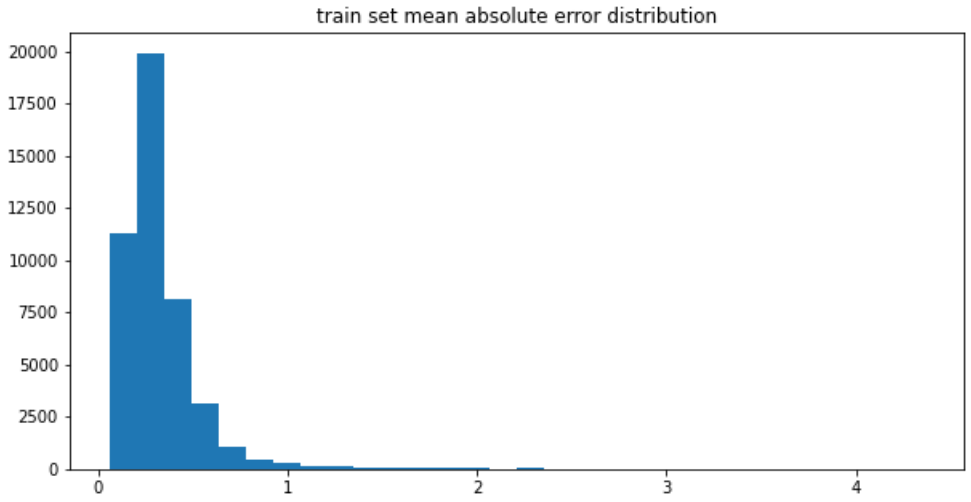

Figure 4.18: Histogram for train set mean absolute error distribution

The same procedure is done for the test set and its histogram for mean absolute error was obtained as shown in Fig. 4.19.

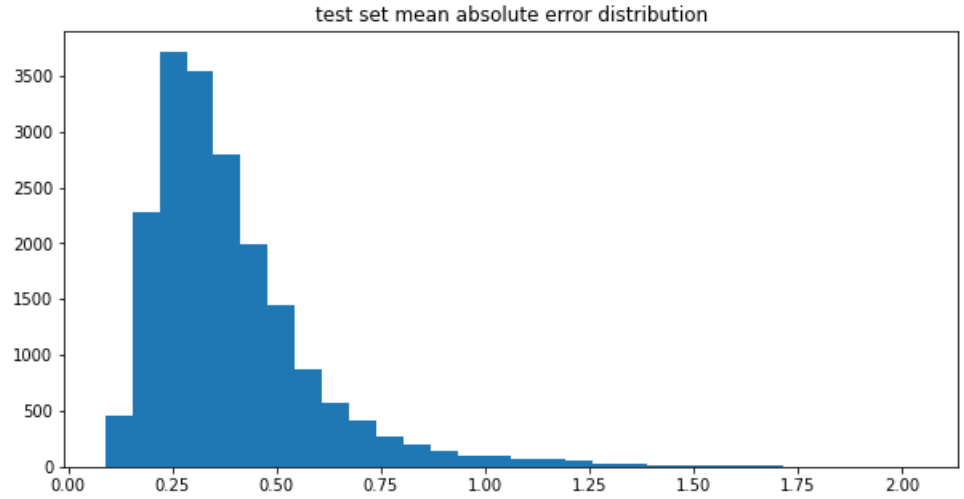

Figure 4.19: Histogram for test set mean absolute error distribution

As the threshold is determined in the above steps, the main task is to classify the dataset as a binary classification task. If the reconstruction loss (mean absolute error in our case) is higher than the threshold then the point is anomalous. Alternatively, if the loss is lower than the threshold for a point then the point is normal.

4.5.2. Isolation Forest

It consists of mainly two steps:

- In the first step, a training dataset is used to build the iTrees.

- In the second step, each instance in the test set is passed through the iTrees built in the first step and a proper anomaly score is assigned to that instance based on the algorithm.

Once all the instances in the test set have been assigned an anomaly score, it is possible to mark as "anomaly" any point whose score is greater than a predefined threshold, which depends on the domain the analysis is being applied to.
At first, the dataset is passed through sklearn.StandardScaler() for standardization purposes before passing through the algorithm. One of the hyperparameters of isolation forest is the contamination parameter that has to be tuned to set the percentage or proportion of data points assumed to be anomalous. The other hyperparameters to be tuned are the number of estimates and the maximum sample. Since initially the level of contamination in our dataset is not known so the model is run without any hyperparameters. In an Isolation forest, no distance metrics are used so it saves computation times. Usually, isolation forest assumes high values as contamination but setting value high will also make some inliers as anomalies. For isolation forest, parameters to be tested are the number of trees or number of estimators and sampling size. For the isolation forest algorithm we have the following major steps:

- **Training the forest:** Here a function called iForest (X, no Of Trees, sample size) is defined that returns a set of iTrees, where each tree is made by a certain subsampling size of random samples from the input data. The subsampling size, as well as the number of trees, can be varied for good results. Similarly, the method iTree(i/p data, current tree height, height limit) is created which returns a traversing tree. It selects an attribute at random, as well as a separation value at random, and then constructs a binary tree recursively until the value of the current tree height is smaller than the height limit. The height limit is typically chosen as the average tree height to make the algorithm more efficient as it does not have to traverse through the whole tree.
  Alternately training can be done by creating a 'model' variable and instantiating the Isolation Forest class. For proper tuning of the model, four parameters (or hyperparameters) are passed to the Isolation forest method which are:
  Number of Estimators: This parameter refers to the number of base estimators or

trees in the ensemble, i.e., the number of trees that will be built in the forest. This is an integer parameter whose default value is 100.

Max samples: This parameter refers to the number of samples to be taken to train each base estimator (or tree). If this max samples parameter is set more than the number of samples provided then all the samples will be used to build all the trees in the forest. The default value of this parameter is 'auto' where the value will be equal to min(256, number_ of_ samples).

Contamination: Our algorithm is quite sensitive to this parameter. It refers to the expected proportion of the outliers in the dataset.

Max features: All the trees are not trained on all the features available in the dataset. This parameter refers to the number of features to draw from the total features to train each tree whose default value is one.

- **Evaluating anomaly score:** The anomaly scores for each instance are found in this section. To do it, a function Pathlength(instance, iTree, current path length) is used which returns the length of the path, for each instance in the dataset. The value of the current path length is updated recursively, and the final value is then assigned as the length of the path for that particular instance. In this way, anomaly scores are calculated.

  Alternatively, once the model is defined and fitted anomaly scores and outlier columns are found. Here at first the values of the anomaly scores column are calculated by calling the $decision_function()$ of the trained model and passing 'az(g)' as a parameter that represents the vertical vibration data. Then after the values of outlier columns are calculated by observing the trained data anomaly score distribution using a histogram plot. The anomaly score is then plotted in a histogram plot as shown in Fig. 4.20.

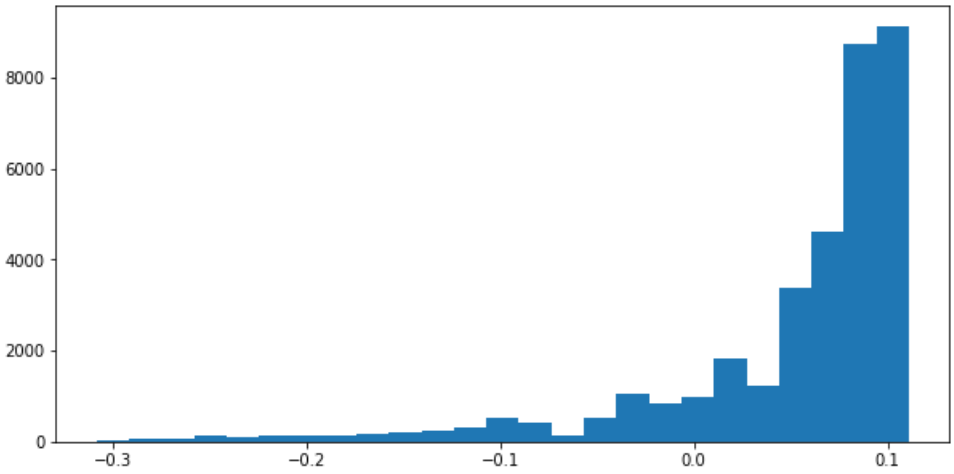

Figure 4.20: Histogram plot of Anomaly Scores of Dataset

From the Histogram plot of Anomaly Scores of Dataset, a good threshold can be set for isolating the anomaly points from normal points instead of previously assuming contamination level. As seen from Fig. 4.20 those points that have anomaly scores less than -0.2 are separated as anomalous points and based on that threshold a separate outlier column is identified where each data point is marked as anomalous or not (1 for outlier and 0 for normal point). However when the contamination level is set to a fixed value then the value of the outlier column is calculated by calling $predict()$ function of the trained model and passing 'az(g)' as the parameter.

4.6. Frequency Domain Analysis

The Fast Fourier Transform of signals was calculated for three cases: a no-load condition, the passing of a medium-sized vehicle like sedans and SUVs, and the passing of a large-sized vehicle like buses and RCC trucks. For each of the cases, the time duration was set as five seconds. At first, a rolling mean was applied to remove high-frequency noise from the data. Then, the accelerometer drift was removed using a Butterworth high pass filter with a cut-off frequency of 1 Hz. The FFT of the filtered signal was computed using the $fft()$ function inside the NumPy module. Similarly, a spectrogram was also plotted using the $signal.spectrogram()$ function of the scipy module.

4.7. Data Acquisition System

A data acquisition system was designed using widely available sensors and microcontrollers. Fig. 4.21 depicts the block diagram of the data acquisition system and it consists of the following sections:

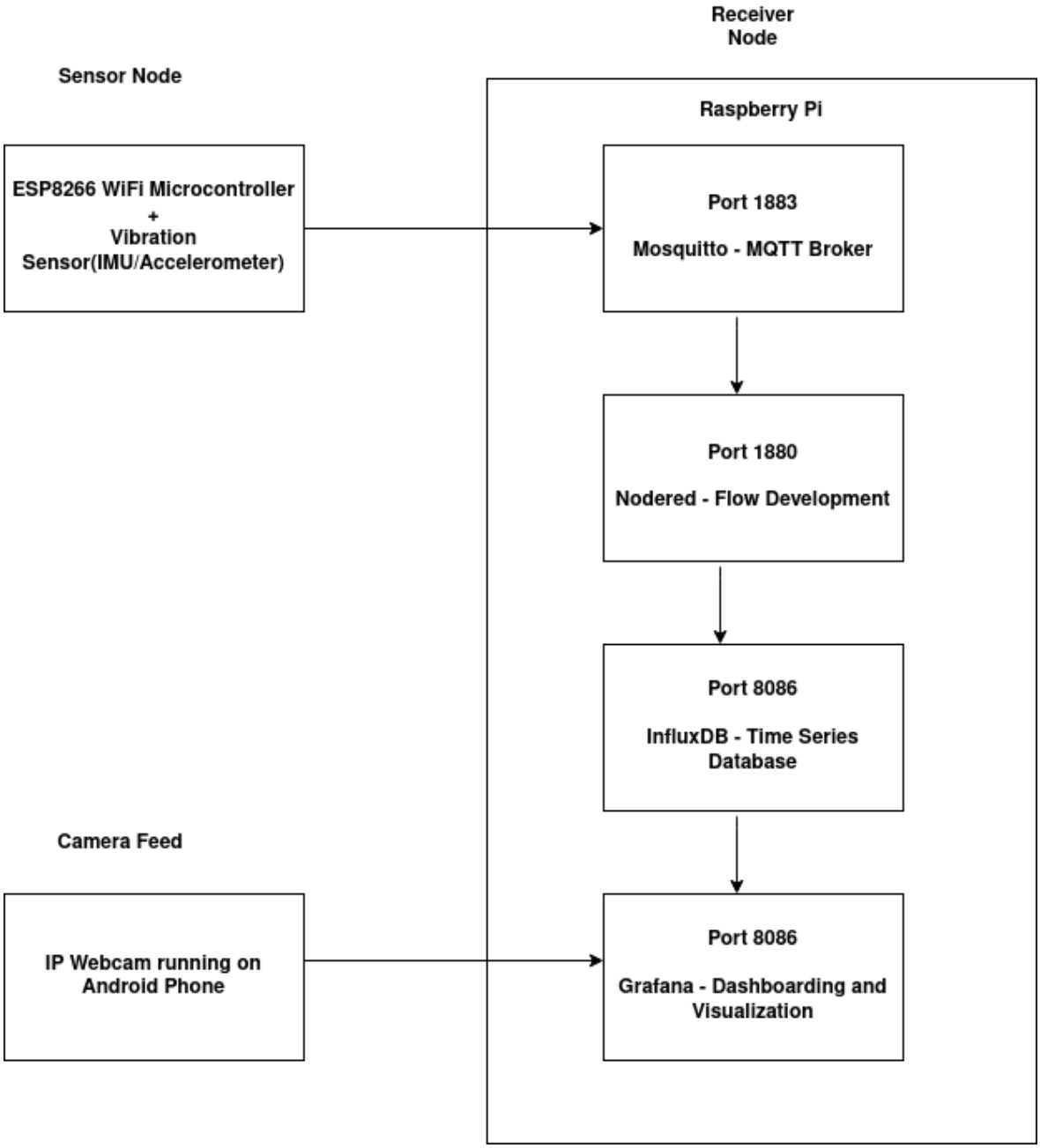

Figure 4.21: Block diagram of data acquisition system

4.7.1. Sensor Node

The sensor node consists of an MPU6050 inertial measurement unit coupled with an ESP8266 WiFi microcontroller. The MPU6050 can output accelerometer, gyroscope, and magnetometer data along with the temperature. The microcontroller reads the values from the inertial sensor and transmits them to the Raspberry Pi which works as a local server. The WiFi microcontroller acts as an MQTT client which publishes messages when it connects to the Mosquitto MQTT broker running inside the Raspberry Pi. Fig. 4.22 depicts the connection of the microcontroller (NodeMCU) and MPU6050.

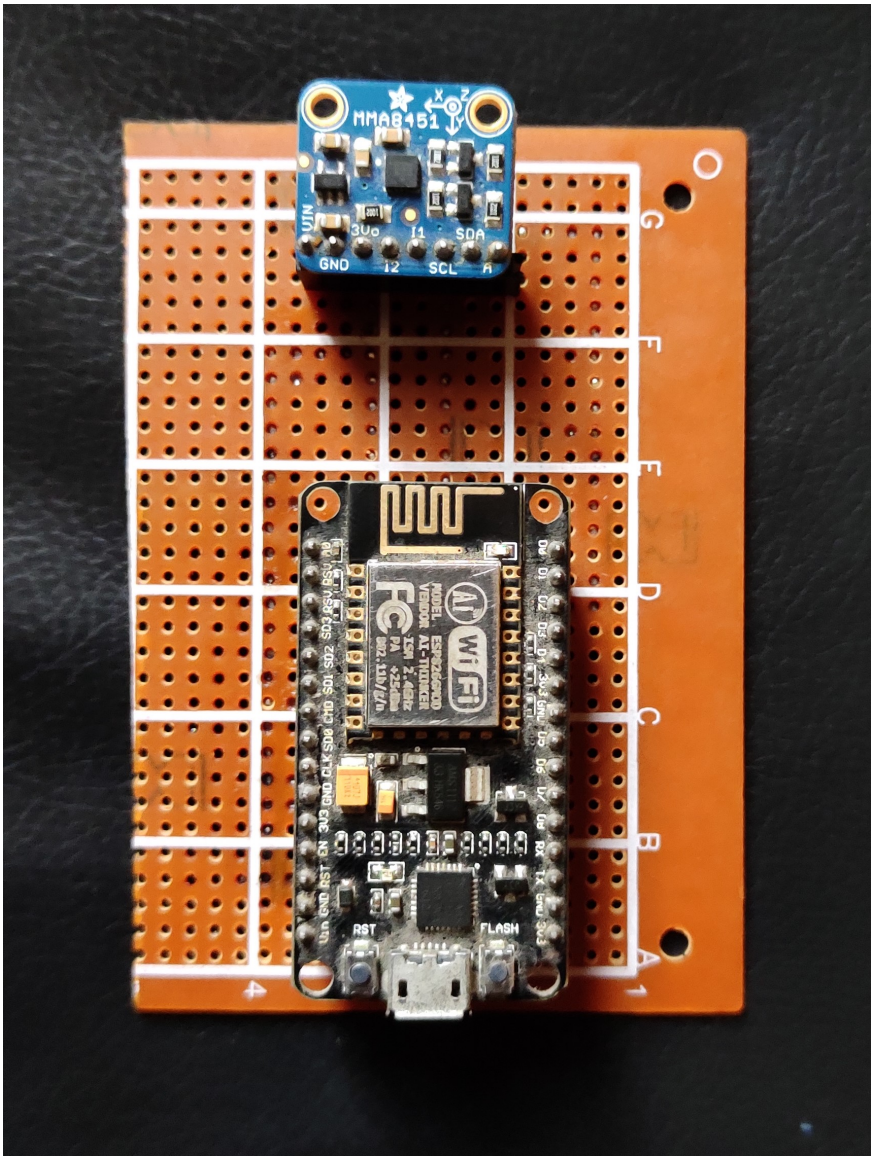

Figure 4.22: NodeMCU and MPU6050 connected using matrix board

4.7.2. Receiver Node

The receiver node consists of a Raspberry Pi as shown in Fig. 4.23 single-board computer running multiple applications that listens, stores, and displays messages transmitted by the sensor node.

Figure 4.23: Raspberry Pi

There is a stack of four applications running on the Raspberry Pi that support the design of an Internet of Things (IoT) based data acquisition system. They are described below.

- **Mosquitto:** It is an MQTT broker that receives the messages published by the sensor node. Then, it filters out messages based on topic and it is also responsible for transmitting those messages to devices subscribing to that topic. Mosquitto runs as a Docker container in the Raspberry Pi on TCP port 1883.

- **Nodered:** It is a flow-based development tool that can be used to connect hardware devices, APIs, and different online services. It provides a browser-based graphical

editor which can be used to connect multiple services as flows. The flow shown in the figure below defines the entire process running on the raspberry pi. The 'MQTT in' node receives the message from the broker as a string of values separated by a comma. The parse node then separates those values and creates a JSON object which is then fed to the database. Node-RED runs as a Docker container on the TCP port 1880 of the Raspberry Pi.

- **InfluxDB:** It is an open-source time-series database that is optimized for time series data. Therefore, it is very suitable for near real-time monitoring of sensors and devices. The JSON object outputted by the parser is fed to the acceleration measurement in the sensor database. InfluxDB runs as a docker container on port 8086 of the Raspberry Pi.

- **Grafana:** It is an open-source dashboarding and visualization tool that can display real-time analytics from multiple sources of data. It supports multiple types of databases and can display multiple types of data in various forms such as graphs, bar charts, gauge indicators, heatmaps, etc. Likewise, it supports various refresh rates and can display values in near real-time. Grafana runs as a Docker container on port 3000 of the Raspberry Pi.

4.7.3. IP Webcam

An Android phone camera app was used as a wireless camera that transmitted live streams to the raspberry pi which was transferred to the Grafana dashboard.

5. RESULT AND ANALYSIS

5.1. Statistical Analysis

The data collected for this project as in case 1 is the measurement of inertial parameters and temperature collected over for 10 days from 2020-02-18 to 2020-02-28 with a sampling rate of 200ms. The data set consists of 13 fields covering acceleration, angular velocity, angular displacement, and magnetic field in three directions and a temperature parameter. However, we discuss and analyze mainly the vibration (acceleration) of the bridge in the vertical direction(az) in this section. Due to interruptions and failure in the sensors, a significant amount of data has been extracted only through sensors 1 and 4.

There are several statistical tests known as normality tests that can be used for inferring whether the data appears as drawn from a Gaussian distribution. For this purpose, the D'Agnostino's K2 test was used. In this test, the p-value can be interpreted as follows:

If p <= alpha; then reject the null hypothesis H0; not normally distributed
If p > alpha; then accept the null hypothesis H0; normally distributed

The value of alpha was fixed to 0.05 and then this test was applied to the data sets of sensor 1 and sensor 4. The result of the statistical test for sensor 1 data was found to be as follows: Statistics=14917213.591, p=0.000.

From the results, we can conclude that the data for sensor 1 data was not normal and thus reject the null hypothesis **H0**. Similarly, the results of the statistical test for sensor 4 data were found to be as follows: Statistics=559346.244, p=0.000. From the results, we can conclude that the data for sensor 4 data was not normal and thus reject the null hypothesis **H0**.

Moreover, the kurtosis and skewness were calculated. The skewness defines the symmetricity of the distribution. If the distribution is symmetric it looks the same to the left and the right from the center position. Similarly, kurtosis defines how much of the distribution is in the tail, i.e., whether the data is heavy-tailed or light-tailed. If the kurtosis

is greater than 3 then the distribution has longer and fatter tails whereas if the kurtosis is less than 3 then distribution has shorter and thinner tails. Following results were obtained for sensor 1 data: Kurtosis = 6488.68643 and skewness = -68.81102 and following results were obtained for sensor 4 data: Kurtosis = 1088.71485 and skewness = -32.88305.

Since the kurtosis of our sensor 1 and sensor 4 data were greater than three which implies that the data is heavily tailed. Similarly, the skewness of our sensor 1 and sensor 4 data were less than zero which implies that the data is highly skewed on the left side.

The summary of data obtained from sensor 1 and sensor 4 is shown in Table. 5.1 and Table. 5.2 respectively.

	Count	**Mean**	**SD**	**min**	**25%**	**50%**	**75%**	**max**
az(g)	4248290	1.01574	0.00863	-0.8276	1.0151	1.0156	1.0166	1.6987

Table 5.1: Vibration data distribution for sensor 1

	Count	**Mean**	**SD**	**min**	**25%**	**50%**	**75%**	**max**
az(g)	209290	1.00044	0.0579	-1.3691	1.002	1.0024	1.0029	1.1816

Table 5.2: Vibration data distribution for sensor 4

5.2. Analysis of Vehicular Motion on Bridges (Case 1)

Visual analysis of the vertical acceleration plot of sensor 1 as shown in Fig. 5.1 shows a seasonal trend occurring over a period of a day. This indicates that there is not much difference in traffic density on normal weekdays. The overall acceleration amplitude is lower on holidays due to less movement of traffic. The time series has been re-sampled at a rate of five minutes for better visual clarity.

The trend of vertical acceleration in sensor 5 is the same as that of sensor 1. Due to sensor failure, the data has been recorded for only three days. This indicates that the traffic density follows the same trend in both the old bridge and the new bridge.

From the vertical acceleration plot from Fig. 5.2, we can see that maximum impact on the bridge occurs during midday and minimum impact occurs during early hours of the day. This indicates that there is peak traffic density at midday and least during midnight. Similarly the

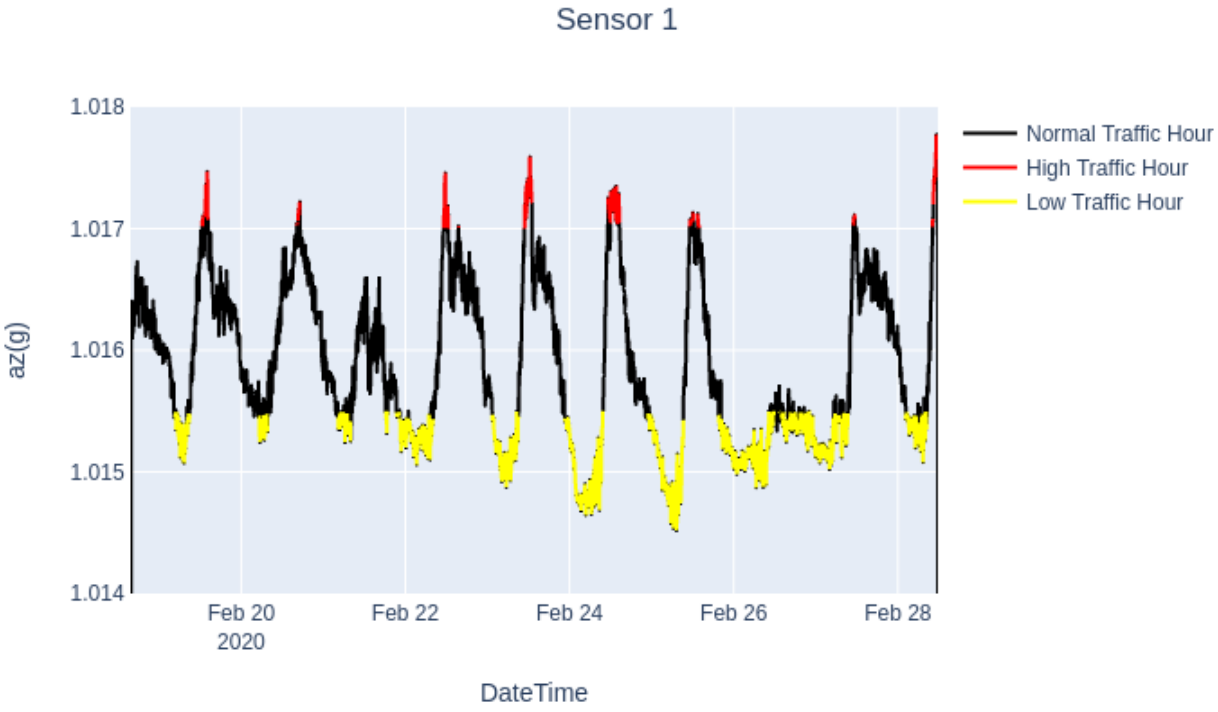

Figure 5.1: Plot of az for 10 days from sensor 1

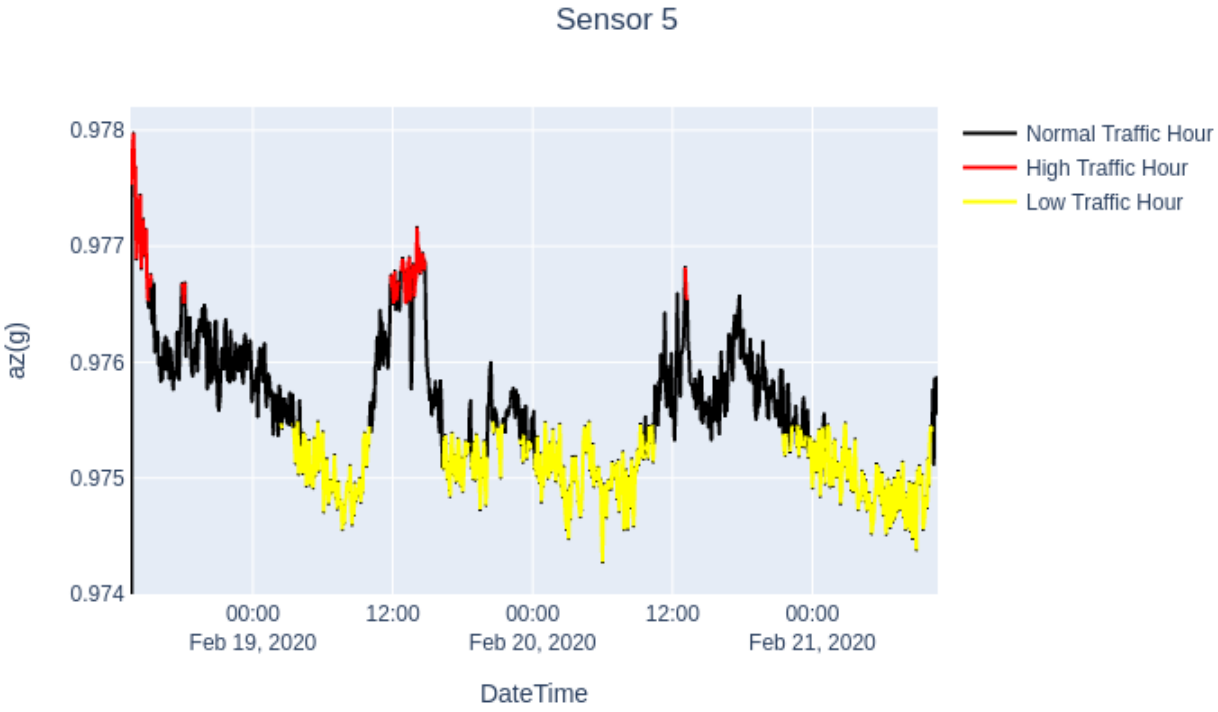

Figure 5.2: Plot of az for 3 days obtained from sensor 5

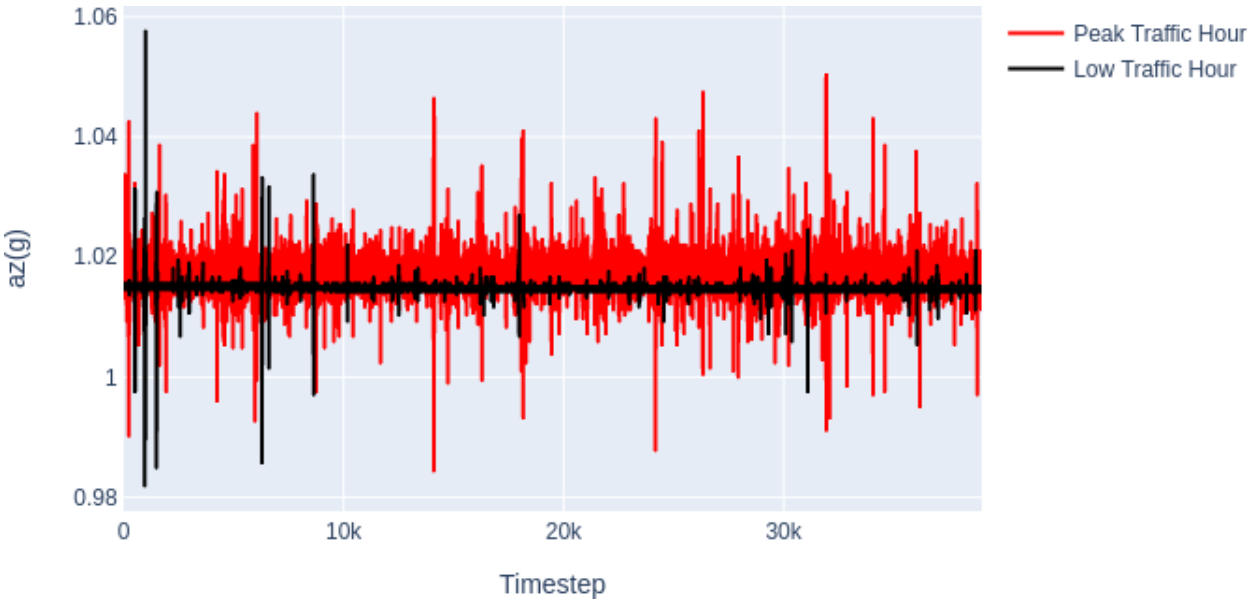

Figure 5.3: Vertical vibrations in peak traffic hour vs low traffic hour as seen in the data obtained from sensor 1

vertical vibration on the bridge at peak traffic hour and low traffic hour has been illustrated in Fig. 5.3.

The overall vibration is higher during normal days and low on holidays as seen in Fig. 5.4. From the comparative plot as shown in Fig. 5.5, it is observed that the z-axis acceleration provides vertical vibration of the bridge as the value varies about 1g. The y axis acceleration is found to be the lateral vibration of the bridge as it has significant amplitude but not maximum. The x axis acceleration is the longitudinal vibration direction of the bridge as it has lowest amplitude. Since the most impact is along the z-axis so the plots of z-axis vibration data are mainly considered.

5.3. Analysis of Vehicular Motion on Bridges (Case 2)

The data collected from sensors at various locations on the surface of the bridge can be depicted from Table. 5.3.

Kupondole side (new bridge)	Point 1	Point 2	Point 3	Point 4	Thapathali Side
Old bridge	Point 1	Point 2	Point 3	Point 4	

Table 5.3: Observation points taken along the bridge

The observations of vibration data as obtained from 2021-02-07 10:17:45 to 2021-02-07

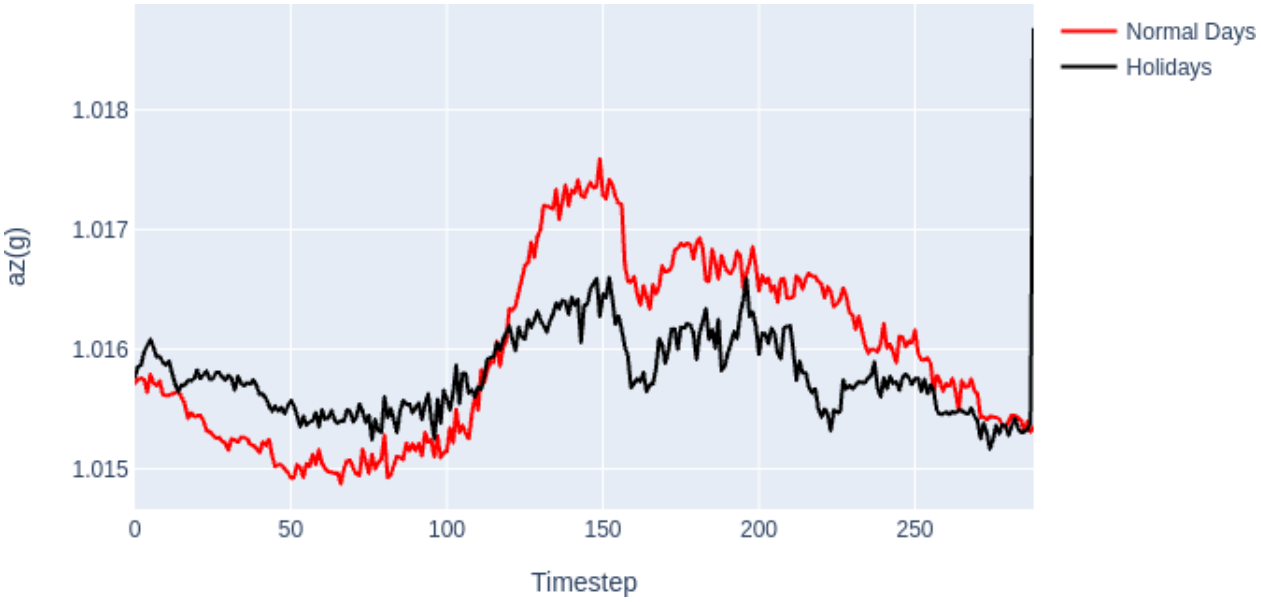

Figure 5.4: Vertical vibrations during normal days vs holidays as seen in the data obtained from sensor 1

Figure 5.5: Acceleration in 3 different axes as seen from the data collected from sensor 1

10:32:53 at point3 on new bridge are as follows:

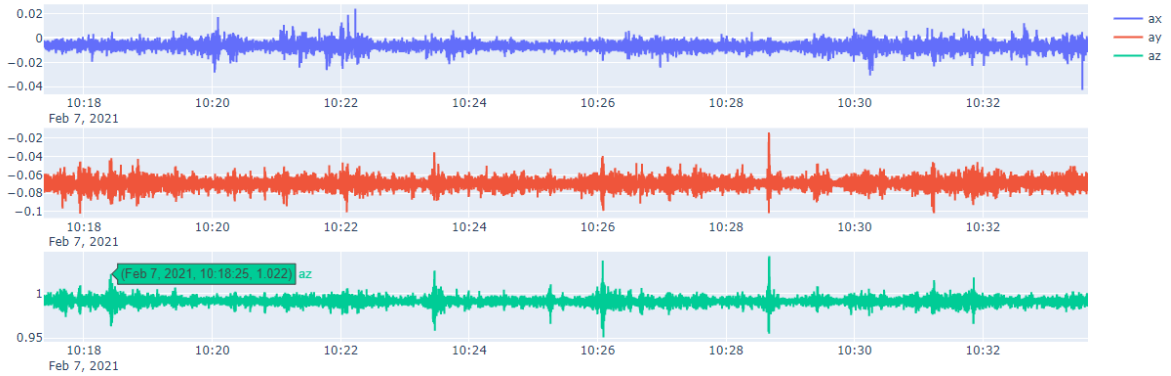

Figure 5.6: Acceleration in 3 different axes observed at point 3 on new bridge (Feb 7)

The observations of vibration data as obtained from 2021-02-09 16:14:50 to 2021-02-09 16:35:10 at point3 on new bridge) are as follows:

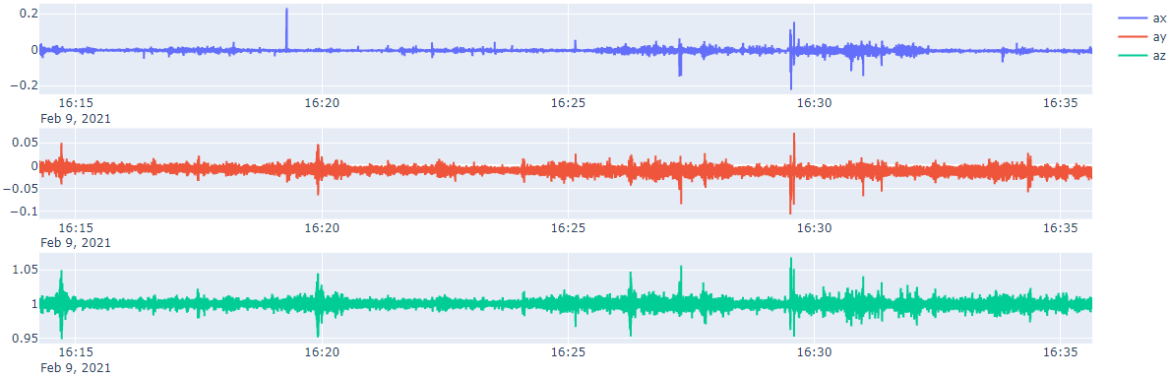

Figure 5.7: Acceleration in 3 different axes observed at point 3 on new bridge (Feb 9)

The observations of vibration data as obtained from 2021-02-13 14:04:51 to 2021-02-13 14:11:45 at point2 on old bridge are as follows:

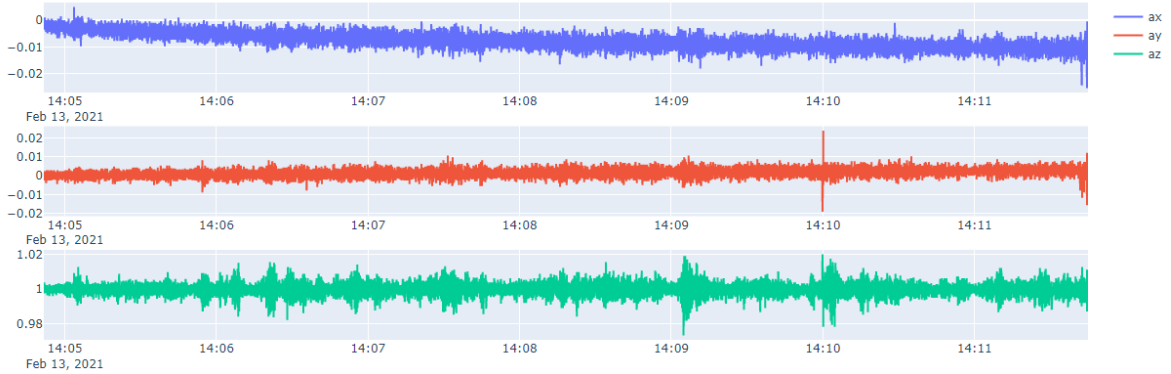

Figure 5.8: Acceleration in 3 different axes observed at point 2 on old bridge

The observations of vibration data as obtained from 2021-02-13 13:00:51 to 2021-02-13 13:13:27 at point2 on new bridge are as follows:

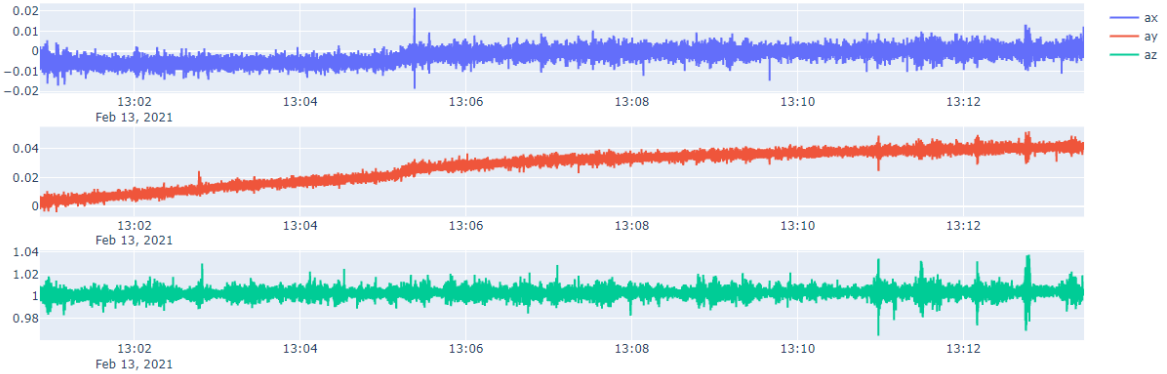

Figure 5.9: Acceleration in 3 different axes observed at point 2 on new bridge

The observations of vibration data as obtained from 2021-02-13 13:18:09 to 2021-02-13 13:29:52 at point3 on new bridge are as follows:

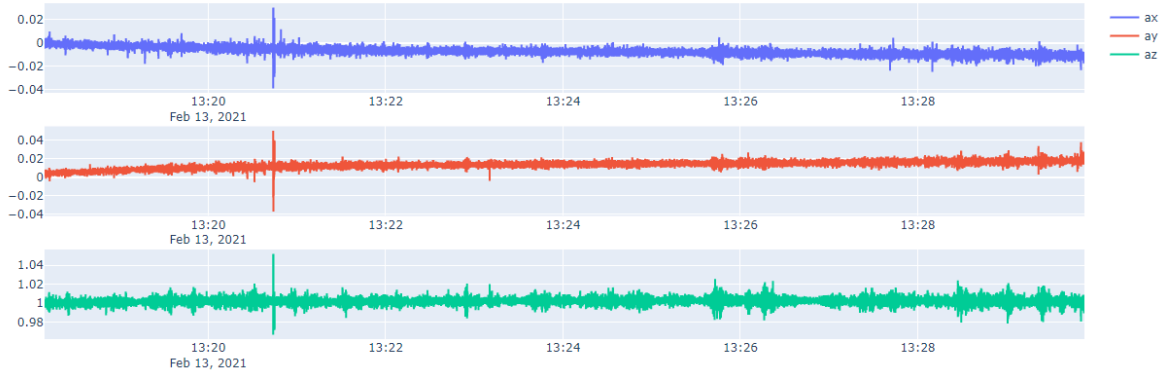

Figure 5.10: Acceleration in 3 different axes observed at point 3 on new bridge (Feb 13)

The observations of vibration data as obtained from 2021-02-13 13:37:38 to 2021-02-13 13:47:22 at point4 on new bridge are as follows:

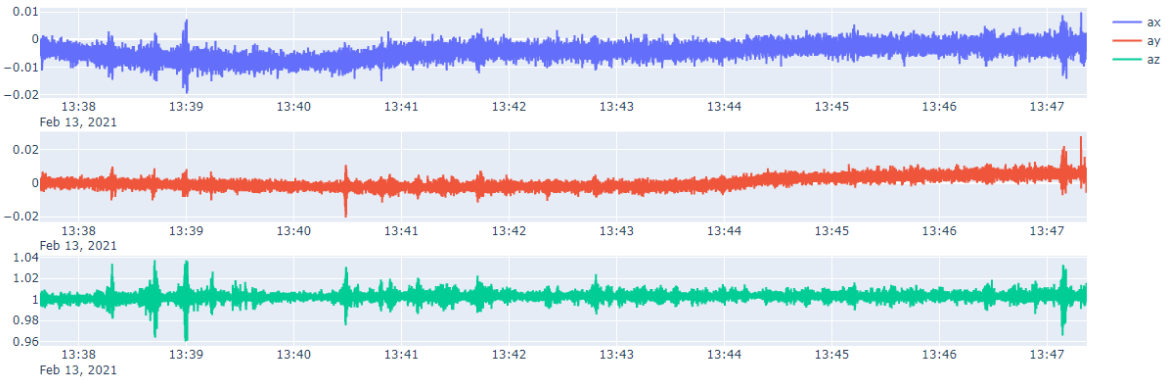

Figure 5.11: Acceleration in 3 different axes observed at point 4 on new bridge

It is clear from three axis acceleration plot the major variation is along the vertical direction so main feature for machine learning algorithms is vertical acceleration data, i.e., az(g) column of dataset. Most distinct observations of vehicular movement are captured through vertical acceleration data. Further analysis is performed on the z-axis component as it carries most of the variation and energy.

From the plots of vibration data it was found that heavy vehicles such as sajha bus, rcc truck, tipper have high amplitude of vibration and also their vibration persists a little longer than vibration from other medium size vehicles. Fig. 5.12 shows vertical vibration of bridge from time 2021-02-13 13:38:10 to 2021-02-13 13:39:43. Distinct vibrations were observed at points shown in Table. 5.4.

Vehicle Passed	**Time Stamp (Around)**
Water Tanker	13 : 38 : 18
Tipper	13 : 38 : 41
Tipper	13 : 38 : 59
Nepal Yatayat	13 : 39 : 15

Table 5.4: Time stamp of passing heavy vehicles

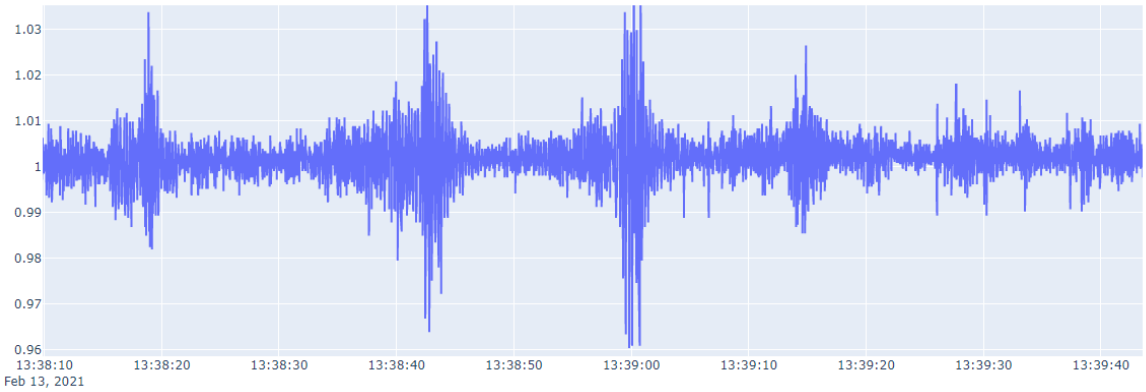

Figure 5.12: Time stamp of passing heavy vehicles

During sensor data collection it was found that the amplitude of vibration also depends on the speed of the vehicle, even though the primary factor is the weight of the vehicle. For instance, it was observed that if a heavy vehicle like the "Sajha bus" was moving at a very low speed along the bridge then its vibration amplitude was relatively lower than when the same vehicle with a similar loading condition was observed at high-speed condition. Similarly, it was observed that the level of vibration was different at various points along the bridge. Also observing vibration data at two bridges have different levels of vibration for vehicle detection.

5.4. Modeling Using ARIMA

The results of forecasting with ARIMA(2,1,1) and ARIMA(3,0,1) are shown in the Fig. 5.13 and Fig. 5.14.

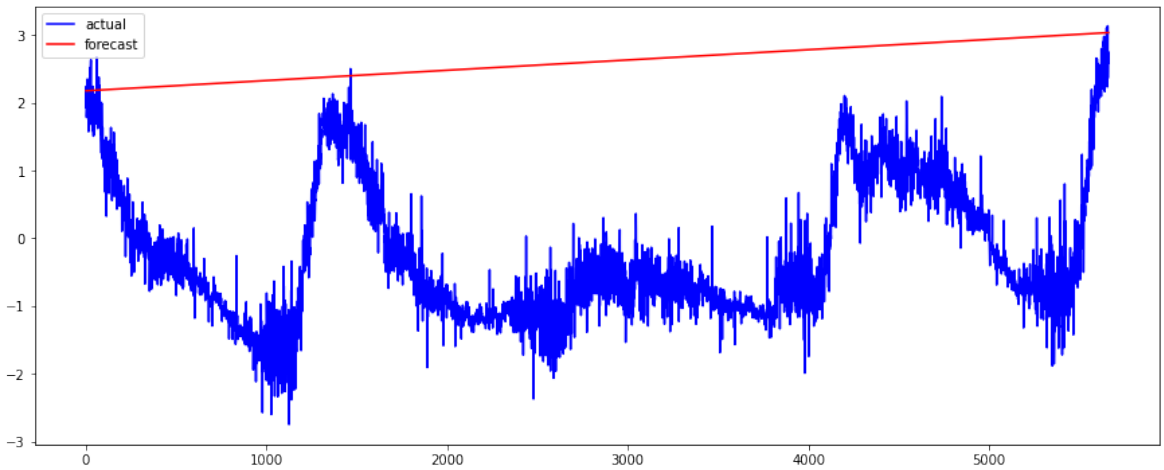

Figure 5.13: Actual plot vs plot predicted by ARIMA(2,1,1)

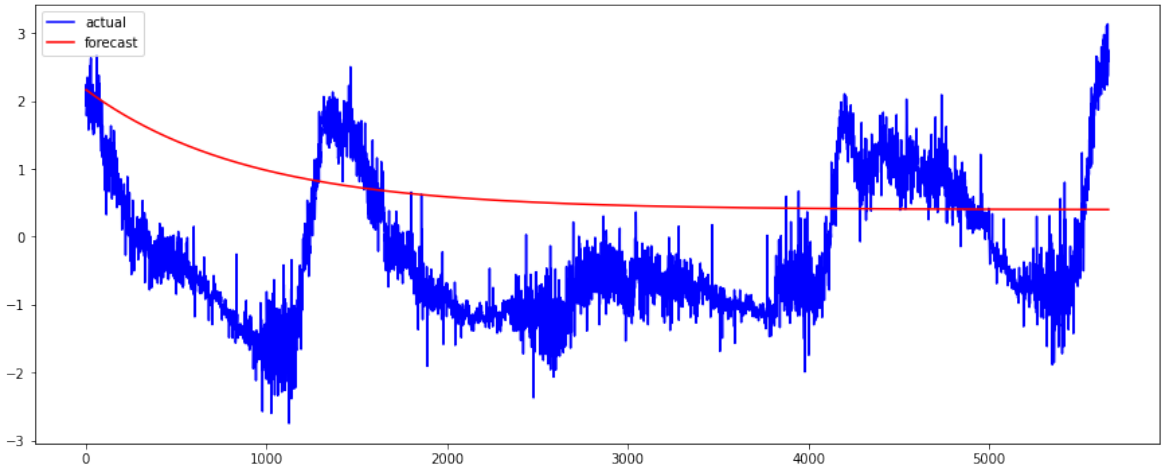

Figure 5.14: Actual plot vs plot predicted by ARIMA(3,0,1)

However, the seasonal ARIMA seems to model the test series with greater accuracy which is shown in Fig. 5.15. The MSE for SARIMA(0,1,1) (1,1,0,12) forecasting was found to be 1.8. Similarly, the results of out-of-sample forecasting using the same SARIMA model can be shown in Fig. 5.16.

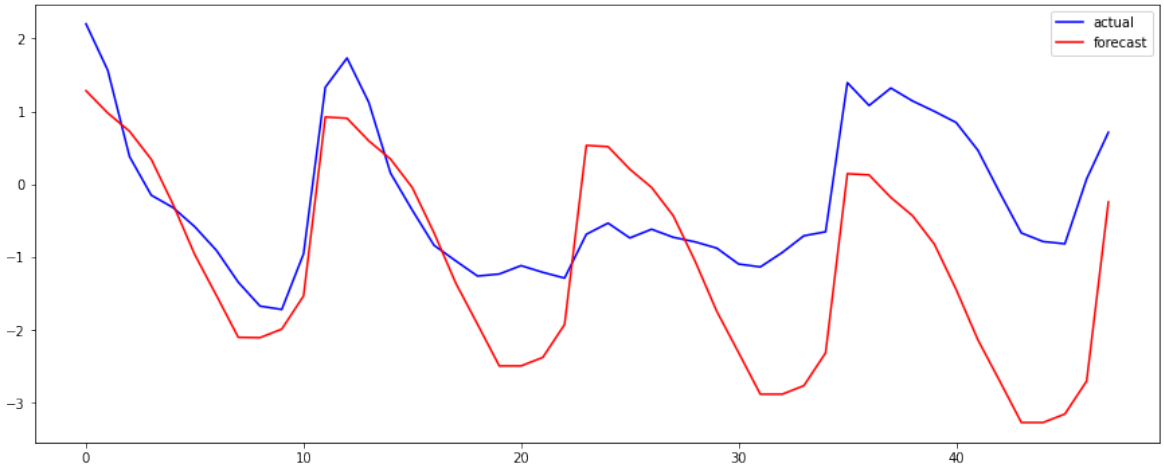

Figure 5.15: SARIMA forecast vs actual test set

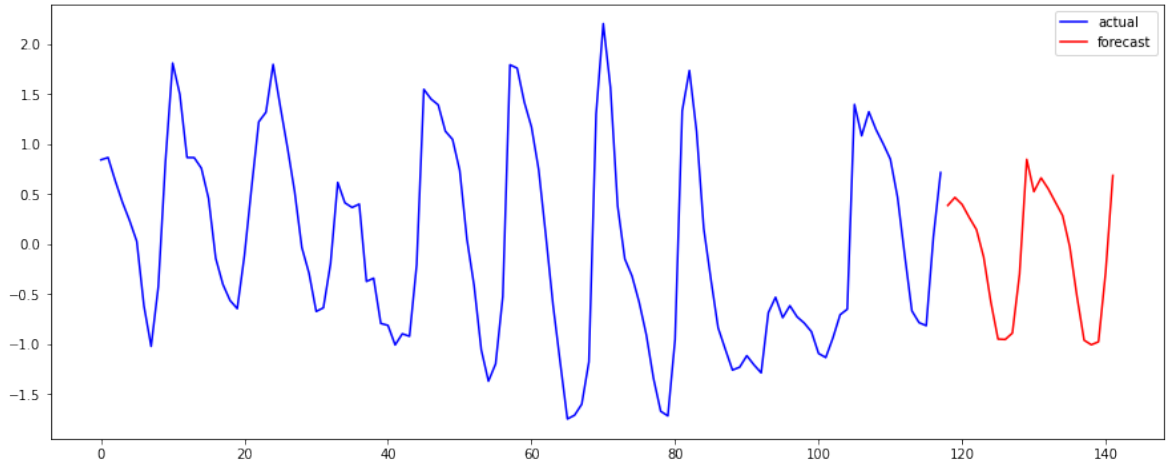

Figure 5.16: Out of sample forecasting with SARIMA(0,1,1) (1,1,0,12)

5.5. Anomaly Detection

5.5.1. LSTM Autoencoder

The LSTM autoencoder is fitted and anomalies are detected based on the threshold level set from observations in the distribution of train set mean absolute error. The plot of anomalous data and normal data can be observed as in Fig. 5.17.

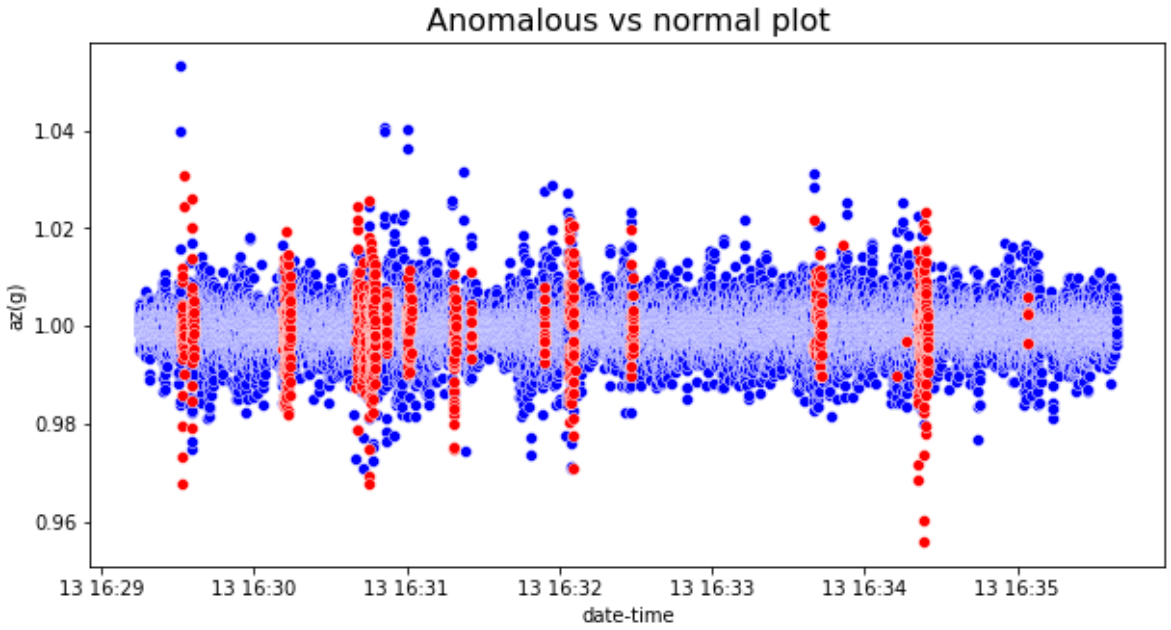

Figure 5.17: Anomalous vs normal data plot

From the observations of anomalous and normal data points, we see that the isolation of anomaly points is not as expected. This is because the LSTM autoencoders isolate anomalies

based on the threshold set on the reconstruction loss. During the training phase, it is required to pass only normal data points as input to the LSTM autoencoder architecture. This is because the reconstruction loss for anomalous data i.e. peak amplitudes becomes much greater than the threshold and are thus flagged as anomalies. But, the data obtained from the sensor was random, and hence segregation of low amplitudes and high amplitudes was difficult. This was the reason for the unexpected isolation of anomaly points using the LSTM autoencoder.

5.5.2. Isolation Forest

The Fig. 5.18 depicts the plot after isolating the points as anomalous and normal by the Isolation Forest algorithm when the contamination level is at 1%.

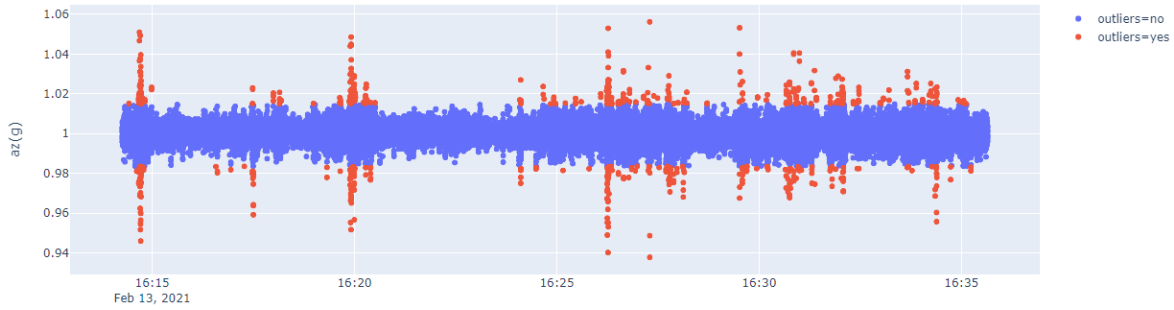

Figure 5.18: Isolation of anomalous points setting contamination level at 1%

But better anomaly detection is possible when we set the threshold by observing the anomaly score based on the path length. We get the dataframe as shown in Fig. 5.19 where anomalous points are isolated if anomaly scores less than the threshold value. The outlier column for the anomaly point has the value '1'.

Time(s)	ax(g)	ay(g)	az(g)	AngleX(deg)	AngleY(deg)	AngleZ(deg)	T(°)	scores	outliers
2021-02-13 16:14:17.940	0.034200	0.004900	0.986800	-0.049400	0.148300	-0.307600	31.050000	-0.213815	1
2021-02-13 16:14:19.720	0.007800	-0.008800	0.985800	-0.060400	0.137300	-0.324100	31.050000	-0.233769	1
2021-02-13 16:14:25.660	-0.013282	0.000843	1.015187	-0.060105	0.092549	-0.395502	30.909132	-0.226664	1
2021-02-13 16:14:34.800	0.002000	-0.016600	0.984400	-0.054900	0.115400	-0.488900	31.370000	-0.246286	1
2021-02-13 16:14:34.820	0.001922	-0.017447	0.983635	-0.057331	0.115614	-0.488391	31.409957	-0.253494	1
...	...	...	...	...	...	...	...	...	...
2021-02-13 16:35:14.180	-0.009066	-0.012384	0.981273	-0.561639	0.367629	1.851200	33.385758	-0.266866	1
2021-02-13 16:35:14.200	-0.008903	-0.014830	0.983117	-0.563050	0.366025	1.851200	33.349165	-0.254004	1
2021-02-13 16:35:23.320	-0.011845	-0.021471	0.986609	-0.565840	0.362626	1.758039	33.488493	-0.215749	1
2021-02-13 16:35:27.960	0.001075	-0.007893	0.986844	-0.555161	0.362965	1.626172	33.549313	-0.213232	1
2021-02-13 16:35:27.980	0.003681	-0.007880	0.986924	-0.555285	0.363970	1.626743	33.564425	-0.211786	1

Figure 5.19: Dataframe consisting of detected anomalous points

The plot after implementing the isolation forest algorithm without previously setting the contamination level is observed as in Fig. 5.20.

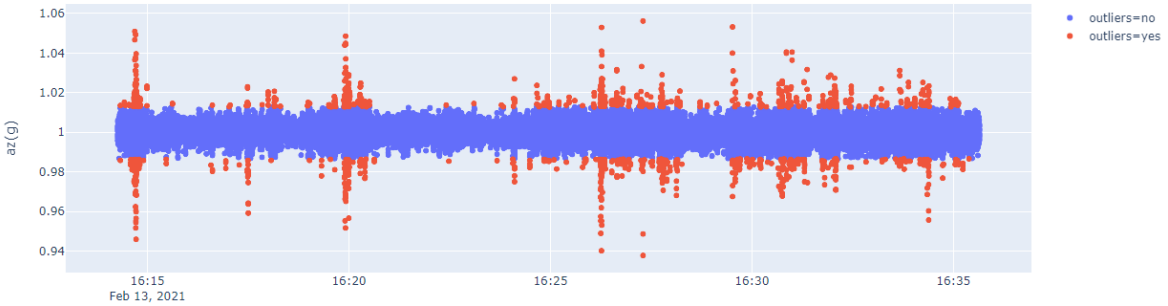

Figure 5.20: Isolation of anomalous points using threshold from anomaly scores

The evaluation of the algorithm can be done if we have the ground truth data that has both normal and anomaly points classified based on a certain threshold. But, in our case, no such ground truth data was available. For the evaluation of our model, we set a threshold for vertical acceleration data as az(g)>1.013g or az(g)<0.987g. The data points that satisfy this threshold inequality are identified as real anomaly points. This threshold was determined during the sensor data collection phase based on both the speed and the weight of the passing vehicle. When the threshold is crossed, it is classified as an anomalous event. This helped us in finding the evaluation metrics such as the ROC-AUC score, F1 score, and confusion matrix. It has to be considered that the ground truth data is required only for evaluating our model but not in the algorithm modeling or training. The ROC-AUC score and F1 score were found to be 0.98215 and 0.96337 respectively, and the confusion matrix is obtained as

shown in Fig. 5.21.

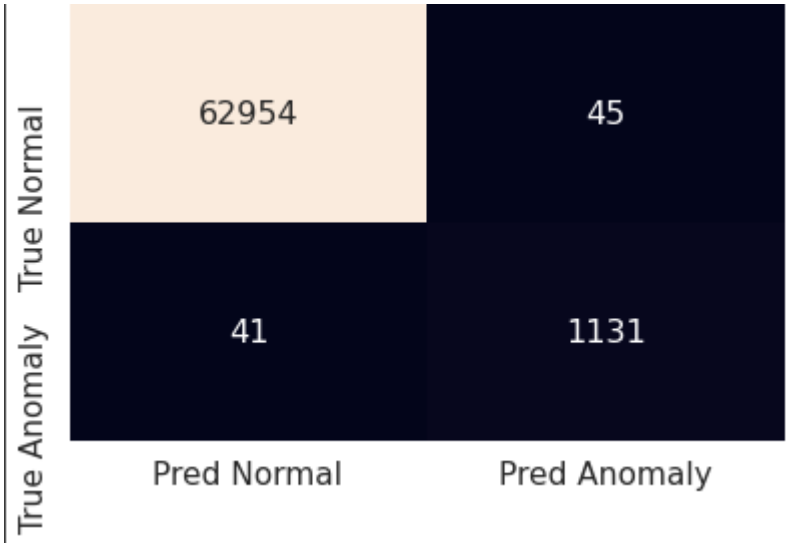

Figure 5.21: Confusion matrix for isolation forest model

5.6. Frequency Domain Analysis

From the PSD vs frequency curve, we observe the following results:

- During the no vehicle movement condition, we see significant peaks at about 1.4 Hz and 9.03 Hz as shown in Fig. 5.22, where the highest value of amplitude reaches up to 22 nano (g^2/Hz).

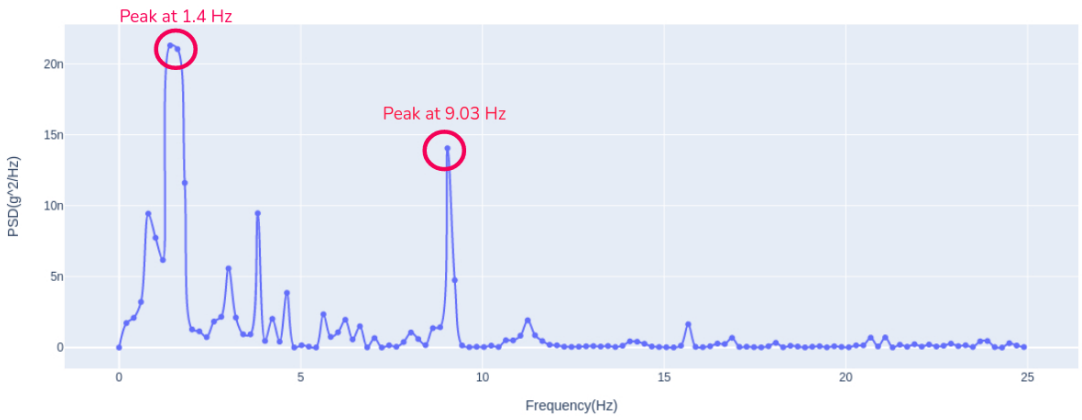

Figure 5.22: PSD when no vehicle passes through the bridge

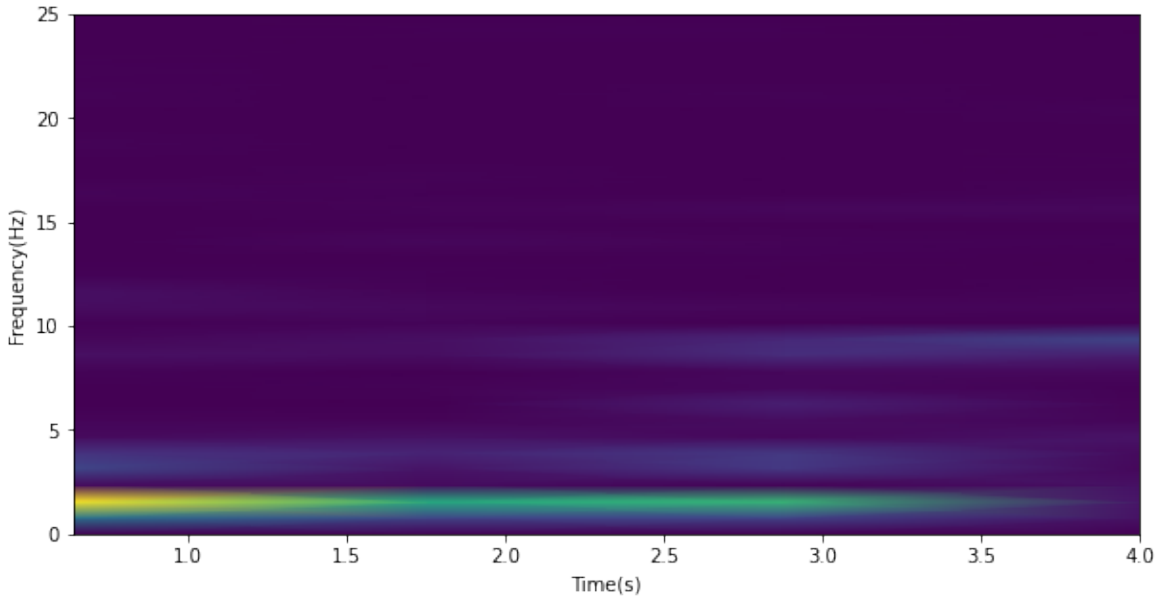

Figure 5.23: Spectrogram of signal when no vehicle passes through the bridge

- When a medium-sized vehicle like a sedan or an SUV passes through the bridge road at sufficient speed, we see significant peaks at about 1 Hz, 3.8 Hz, 9 Hz, and 15.67 Hz as shown in Fig. 5.24. The highest amplitude is about 7 times more than when no vehicle is passing through the bridge.

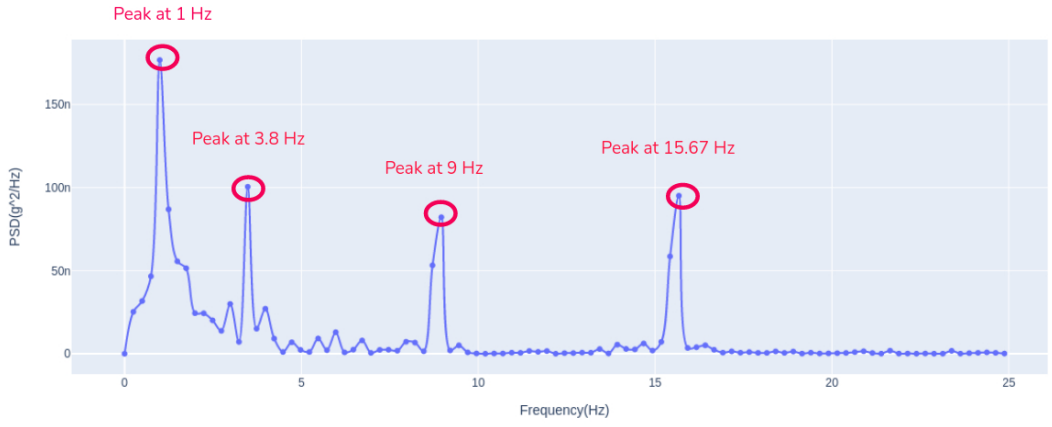

Figure 5.24: PSD when medium sized vehicle passes through the bridge

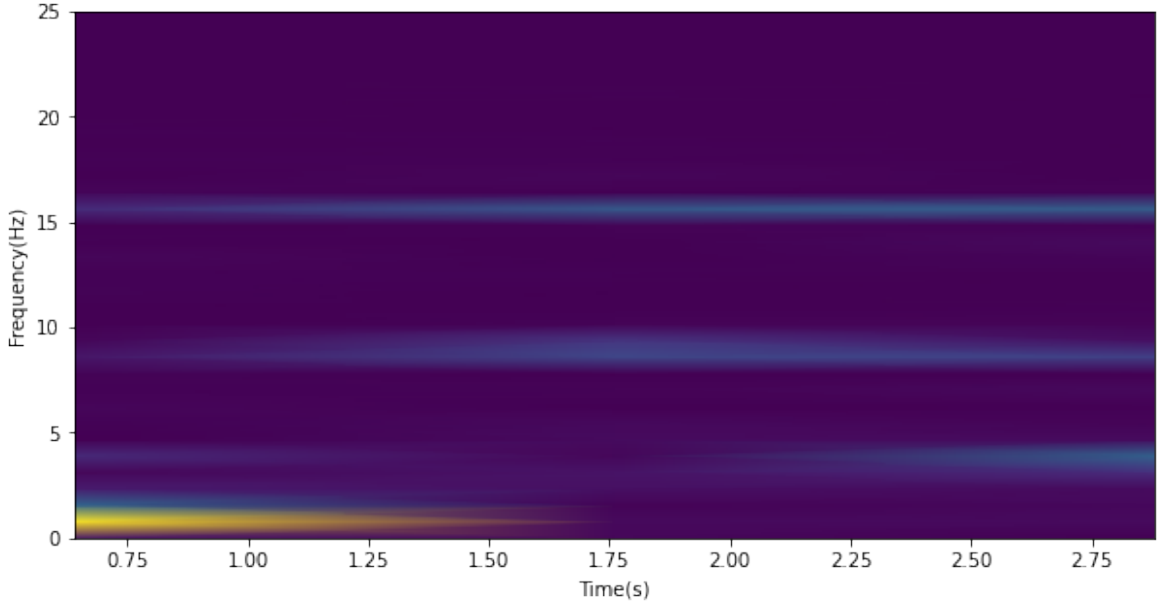

Figure 5.25: Spectrogram of signal when medium sized vehicle passes through the bridge

- When large vehicles such as buses and trucks pass through the bridge at sufficient speeds, we see significant peaks at 3.75 Hz and 8.69 Hz as shown in Fig. 5.26. The amplitude of vibration corresponding to these frequencies is found to be 10 times the amplitude obtained during the passing of a medium-sized vehicle.

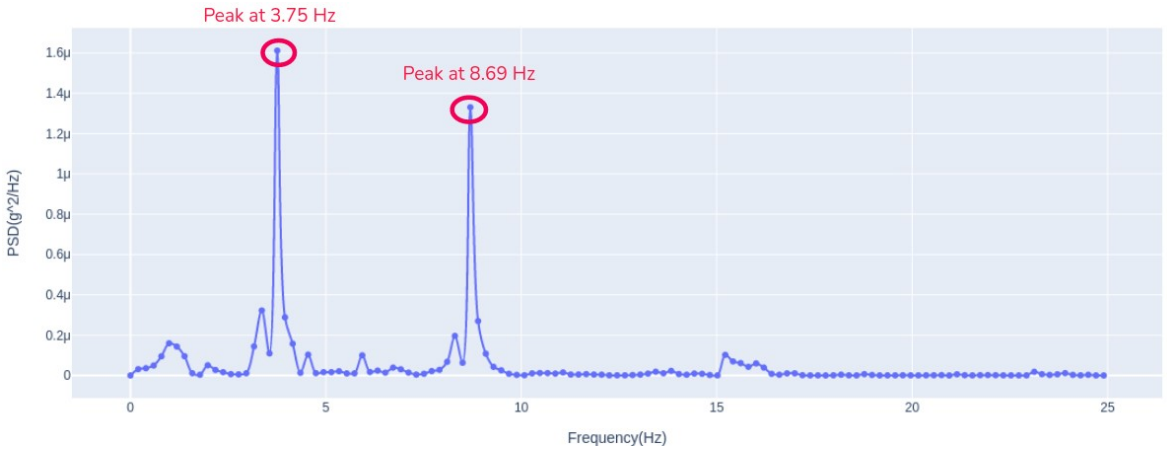

Figure 5.26: PSD when heavy vehicle passes through the bridge

Furthermore, from the spectrogram as shown in Fig. 5.23, Fig. 5.25, and Fig. 5.27, we see the evolution of different dominant frequencies over time. Here, high amplitude components are observed to have higher intensities.

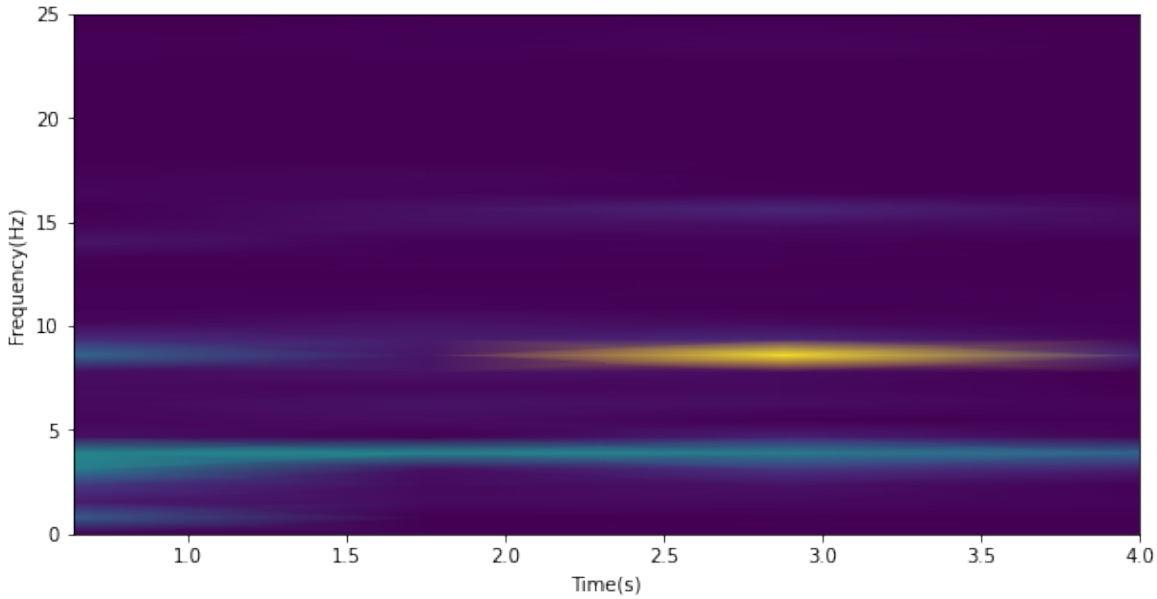

Figure 5.27: Spectrogram of signal when heavy vehicle passes through the bridge

6. CONCLUSION

The project studies the vibration characteristics of the Thapathali-Kupondole bridge using inertial measurement sensors. The vibration data obtained from those sensors is used to analyze the impact of vehicles on the bridge. Similarly, data obtained from different points on the bridge provides different types of information. For example, sensors placed on the steel girders under the bridge road provided a general trend of traffic density. Likewise, sensors placed on the surface of the bridge provided information about the impact of moving vehicles. Frequency domain analysis of obtained vibrations shows the different dominant frequencies during no vehicle movement and movement of medium-sized and large, heavy vehicles. The Isolation Forest algorithm detects the occurrence of huge vibrations caused by heavy vehicle passage. Similarly, LSTM autoencoders also detect anomalous points but better results were obtained from the Isolation Forest algorithm. ARIMA(2,1,1) model and ARIMA(3,0,1) model were used to model and predict the trends of vibrations occurring on the bridge. However, a poor performance was obtained in the test set because they could not account for the seasonality in the series. Therefore, a seasonal ARIMA of order (0,1,1) x (1,1,0,12) was found to be the best fit with an MSE of 1.8.

The algorithms used for detecting anomalies as well as prediction can be utilized in Structural Health Monitoring which is an important practice that has to be adopted for proper diagnosis of existing structures. Due to the development of highly powerful miniature sensors, non-destructive testing of structures has been easier than ever before. Data-driven structural health monitoring can provide insights into important characteristics to structural engineers and inspectors. For roads and bridges, understanding the impact of different types of vehicles can help policymakers design weight and speed limits so that structures remain healthy for a longer period of time. Likewise, understanding different parameters like vibration and displacement of different sections of structures allows timely renovation.

7. LIMITATIONS AND FUTURE ENHANCEMENT

7.1. Limitations

Some of the major limitations of this project can be summarized as follows:

- **Insufficient data**

  Structural monitoring and analysis require data collected over a large period of time, but we had access to only about 10 to 14 days of scattered data due to frequent power failures and sensor theft. Because of this, the project could not study the long-term trends and seasonal patterns. Also, as the data collected during this research was found to be insufficient, it may have resulted in the over-fitting of models.

- **Non-continuous data**

  Since the number of sensors was limited and the measurements had to be done manually, this project cannot study continuous data coming out from any single point on the bridge. Therefore, it cannot account for the changes that can take place between two measurements.

- **Inability to consider multiple vehicles simultaneously in frequency domain analysis**

  The measurement of vibrations and its corresponding frequency domain analysis was not carried out in a controlled setting with isolated medium-sized and large-sized vehicles. Since it was done during normal traffic hours, there might be contributions from other vehicles passing through the bridge. The project cannot study the impact of multiple vehicles passing through the bridge at the same time.

- **Inability to present the status of the bridge**

  Due to a lack of theoretical structural analysis, we cannot state the present condition of the bridge yet.

7.2. Future Enhancement

This project still has a lot of room for enhancements that can increase the accuracy of analysis and predictions. Some improvements that can be done are listed below:

- Design of an advanced data acquisition system allows transmission of enormous volumes of data from multiple points in the bridge to a remote server where it can be stored for a long period of time.
- Cameras can be installed in multiple strategic locations to extract information about what is exactly happening during high vibration events. This assures that the vibrations are caused by the vehicles and not due to other factors.
- The results obtained from a data driven approach can be verified with theoretical structural analysis so that necessary improvements can be made.
- The concept of vibration analysis based on a wireless sensor network system and data driven models can be extended to other areas like industrial equipment monitoring, vehicle and aircraft structure monitoring, etc.

REFERENCE

[1] Y. Liang, D. Wu, G. Liu, Y. Li, C. Gao, Z. Ma, and W. Wu, "Big data-enabled multiscale serviceability analysis for aging bridges," *Digital Communications and Networks*, vol. 2, 07 2016.

[2] A. Neves, I. Gonzalez, J. Leander, and R. Karoumi, "A new approach to damage detection in bridges using machine learning," 01 2018, pp. 73–84.

[3] M. Azimi and G. Pekcan, "Structural health monitoring using extremely-compressed data through deep learning," *Computer-Aided Civil and Infrastructure Engineering*, 11 2019.

[4] E. Khouri Chalouhi, I. Gonzalez, C. Gentile, and R. Karoumi, "Damage detection in railway bridges using machine learning: application to a historic structure," *Procedia Engineering*, vol. 199, pp. 1931–1936, 12 2017.

[5] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," 05 2007, pp. 254–263.

[6] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, 05 2015.

[7] R. Hostettler, "Traffic counting using measurements of road surface vibrations," 2009.

[8] P. Zhang, "Zhang, g.p.: Time series forecasting using a hybrid arima and neural network model. neurocomputing 50, 159-175," *Neurocomputing*, vol. 50, pp. 159–175, 01 2003.

[9] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," *Applied Soft Computing*, vol. 11, pp. 2664–2675, 03 2011.

[10] A. Guo, A. Jiang, J. Lin, and X. Li, “Data mining algorithms for bridge health monitoring: Kohonen clustering and lstm prediction approaches,” *The Journal of Supercomputing*, vol. 76, 02 2020.

[11] A. Kopacik, I. Lipták, J. Erdélyi, and P. Kyrinovic, “Structural health monitoring of bridges using accelerometers – a case study at apollo bridge in bratislava,” *Geonauka*, vol. 03, pp. 9–15, 04 2015.

[12] H. John and S. Naaz, “Credit card fraud detection using local outlier factor and isolation forest,” *International Journal of Computer Sciences and Engineering*, vol. 7, pp. 1060–1064, 04 2019.

[13] Z. Ding, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,” vol. 46, 09 2013, pp. 12–17.

[14] A. Chawla, P. Jacob, B. Lee, and S. Fallon, “Bidirectional lstm autoencoder for sequence based anomaly detection in cyber security,” 10 2019.

[15] H.-D. Nguyen, K. P. TRAN, S. Thomassey, and M. Hamad, “Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management,” *International Journal of Information Management*, 11 2020.

[16] F. T. Liu, K. Ting, and Z.-H. Zhou, “Isolation forest,” 01 2009, pp. 413 – 422.

APPENDIX

Description of the Sensor

The module integrated high precision gyro accelerometer and geomagnetic field sensor. The product can solve the current real-time motion posture of the module quickly by using the High performance microprocessor, advanced dynamic solutions and Kalman filter algorithm. The advanced digital filtering technology of this product can effectively reduce the measurement noise and improve the measurement accuracy.The attitude controller is integrated inside the module, the current attitude of the module can be accurately output in the dynamic environment, with the dynamic Kalman filtering algorithm.An internal voltage stabilizing circuit module, voltage 3.3v$\tilde{5}$v, pin compatible with the 3.3V/5V embedded system, convenient connection.High performance cortexM0 core processor operates as high as 48MHZ, take into account low power consumption and high performance. The purpose of Bluetooth BLE4.0 wireless transmission, the transmission is stable, a distance of more than 10meters. 4 ports are reserved, which can be configured as analog input, digital input and digital output respectively. The stamp hole gold plating process can be embedded in the user's PCB board. In the 4 layer PCB board process, thinner, smaller, more reliable.

Product Parameters

- **Voltage**: 3.3V to 5V
- **Current**: less than 20mA
- **Size**:19mm x 20.4mm x 2mm
- **Pad pitch**: up down 100mil(2.54mm); right left 600mil(15.24mm)
- **Measurement dimension**: Acceleration 3 axes, Angular velocity 3 axes, Magnetic field Angle 3 axes, Air pressure
- **Range**: Acceleration $\pm$ 16g; Angular velocity $\pm$ 2000 °/s; Angle X/Z $\pm$ 180°Y $\pm$ 90 °
- **Stability**:Acceleration -0.01g Angular velocity 0.05°/s

- **Attitude measurement stability**: 0.05°
- **Output content**: Acceleration, Angular Velocity, Angle Magnetic Field, Port State, Air Pressure, Height.
- **Output frequency**: 0.1Hz to 100Hz, default: 10Hz.
- **Date interface**: Serial TTL level Baud rate: 115200
- **Bluetooth transmission distance**: Greater that 10m.
- **Extended opening function**: Analog input(0 - VCC), digital input, digital output.
- **Bluetooth4.0**: Compatible Android/ IOS operating system.