April 24th, 2024

Spring ‘24

Final Project Report

Anomaly Detection in Spacecraft Telemetry

1. Introduction

The field of anomaly detection in spacecraft telemetry plays a crucial role in ensuring the safety and efficiency of space missions. Telemetry data, which include measurements from various spacecraft systems, provide invaluable insights into the operational status and health of spacecraft components. Detecting anomalies in this data is essential for preemptive maintenance, operational adjustments, and critical decision-making processes that can significantly impact the success and safety of space missions.

In this project, we focus on leveraging deep learning techniques, specifically convolutional autoencoders and temporal convolutional networks, to detect and analyze anomalies in time-series data from spacecraft systems. Our goal is to enhance the predictive capabilities of our models and to ensure robust detection of outliers that may indicate potential issues or system failures. By implementing a combination of data scaling and time-step analysis, we aim to improve the accuracy and reliability of our anomaly detection system, thereby contributing to safer and more efficient space operations.

This report outlines the methodologies employed in preprocessing the data, the architecture of the convolutional autoencoders and temporal convolutional networks used, and the results and implications of our findings in the context of spacecraft telemetry analysis.

1.1 What is Anomaly detection and its importance

Anomalies are unusual or unexpected data points that significantly deviate from normal operational patterns. Identifying these anomalies is crucial as they can indicate potential issues within systems, such as equipment malfunctions or failures. Anomaly detection plays a key role in early issue identification, allowing for the detection of potential problems before they escalate. This process is vital for health monitoring, ensuring the continuous operational integrity of systems. Additionally, it supports preventative maintenance, enabling timely interventions that can save costs and extend the lifespan of components. Moreover, effective anomaly detection enhances safety and efficiency, optimizing system performance and ensuring mission safety. This comprehensive approach to monitoring and maintenance is essential for maintaining robust and reliable operations across various fields.[1]

1.2 Anomaly detection in Spacecraft Telemetry

Anomalies in spacecraft telemetry represent statistical deviations that are crucial for signaling potential system issues. These anomalies appear within datasets that monitor critical parameters, such as voltage, temperature, and mechanical component speeds. Detecting these irregularities is vital for the early diagnosis of potential faults and operational problems in spacecraft systems. By identifying these deviations early, engineers can take preventative measures to address issues before they escalate, thereby ensuring the safety and functionality of the spacecraft. This process not only helps in maintaining the spacecraft's operational integrity but also optimizes its performance across various mission phases.

2. Dataset Description[2]

Our dataset encompasses several time-series data collections from various spacecraft systems, which are crucial for monitoring the ongoing performance and status of critical spacecraft components over time. Each dataset provides insight into different aspects of spacecraft operation and health. Each of these datasets plays a pivotal role in the anomaly detection framework, helping to ensure the spacecraft operates within safe and efficient parameters while alerting engineers to potential issues before they become critical.

2.1 Bus voltage (V)

This dataset measures the voltage levels within the spacecraft's electrical system, expressed in volts (V). It serves as an important indicator of the electrical system's stability. Anomalies in voltage can flag issues such as power fluctuations or failures in the electrical subsystem, which could jeopardize the entire spacecraft's functionality.

2.2 Total Spacecraft Bus Current (A)

Measured in amperes (A), this dataset captures the current flowing through the spacecraft’s electrical bus. It reflects the total electrical consumption and efficiency of the spacecraft. Variations in current can suggest problems in the electrical load and potential malfunctions in the power distribution system.

2.3 Wheel Temperature (°C)

This dataset records the temperature of the spacecraft’s reaction wheel systems in degrees Celsius (°C). The health of these components is crucial for the longevity and reliability of the spacecraft's navigational systems, as overheating can result in significant mechanical wear and potential failure.

2.4 Battery Temperature (°C)

Monitored in degrees Celsius (°C), this dataset tracks the temperature of the spacecraft’s battery units. Maintaining optimal battery temperature is vital for battery health and safety, as extreme temperatures can lead to battery failure and compromise the spacecraft's power supply.

2.5 Reaction Wheel (RPM)

Speed readings of the spacecraft's reaction wheels are critical for maintaining accurate orientation and stable navigation. This dataset provides the rotational speed of these wheels, crucial for maneuvering and stabilizing the spacecraft. Anomalies in RPM can indicate mechanical issues or misalignments that may affect the spacecraft's trajectory.

3. Preprocessing the data

3.1 Removing null values

In our spacecraft telemetry analysis, it's crucial to address null or missing values, which may result from lapses in data transmission or sensor errors. We carefully identify these gaps in our data, assessing their impact on our overall dataset. Depending on their significance, we either remove these data points or fill them in using logical imputation techniques, such as interpolation. This approach ensures our data remains robust and reliable, supporting accurate monitoring and decision-making for spacecraft operations. For our project, we have decided to remove all the null values as that seemed the most appropriate choice based on the datasets.

3.2 Downsampling the data

To enhance the manageability and computational efficiency of our spacecraft telemetry datasets, we implemented a downsampling strategy across all time series. This approach involved reducing the frequency of data collection, thereby condensing millions of data points into more actionable insights. This method allows us to maintain a clear overview of spacecraft health, highlighting key changes and potential issues without the computational burden of handling millions of data points continuously.

3.2.1 Bus Voltage

The dataset represents a univariate time series containing approximately 1.8 million data points. Each data point reflects a bus voltage reading captured at a 5-minute interval. To enhance computational efficiency, the data was down-sampled to include readings captured every 8 hours, resulting in a dataset of 18,807 rows. Rows with missing values were excluded from the analysis. The data exhibits a predictable annual voltage dip. Additionally, the trend remains constant until encountering a change point around the beginning of 2008. This change point is characterized by a spike in voltage followed by a subsequent downward trend. *Figure 1* in the appendix visualizes this data.

3.2.2 Total Spacecraft Bus Current

This dataset, comprising roughly 1.5 million entries, represents a univariate time series of bus current readings captured every 5 minutes. For analysis efficiency, the data was down-sampled to include only readings every 8 hours, resulting in a more manageable dataset. Missing values and zero readings, which may represent inactive periods, have been excluded to focus on active operation times. The data reflects a stable pattern of current usage with occasional surges that could signify operational adjustments or system anomalies.
Figure 2 in the appendix visualizes this data.

3.2.3 Wheel Temperature

This dataset represents a univariate time series dataset comprising approximately 1.5 million data points, each representing the temperature measured at a 5-minute cadence following mean resampling from an original 1-second cadence. After cleaning to exclude missing values and removing data points where the temperature was recorded as zero (indicating inactivity of the reaction wheels), the dataset still presents a substantial amount of data for analysis. The temperature readings are highly variable and show no apparent signs of a predictable pattern or seasonality. *Figure 3* in the appendix visualizes this data.

3.2.4 Battery Temperature

This dataset encapsulates a univariate time series with over 1.5 million temperature readings, gathered from January 2004 to January 2018, at 5-minute intervals. It focuses on the temperature of the spacecraft's batteries, which are crucial for powering the satellite's systems and ensuring operational integrity. Fluctuations in battery temperature can be critical, as deviations from the normal range may signal potential anomalies or necessary maintenance actions. Conversely, some of these deviations might be normal under certain conditions but still appear anomalous due to their rarity. This dataset is vital for monitoring the health and efficiency of the spacecraft, aiding in timely decision-making for safety and maintenance.
Figure 4 in the appendix visualizes this data.

3.2.5 Reaction Wheel RPM

This dataset constitutes a comprehensive univariate time series consisting of approximately 48,865,494 RPM readings, observed over a span of roughly 7 years from January 2011 to January 2018. Each entry captures the rotation per minute of a reaction wheel within a spacecraft, a critical component for attitude control. The dataset showcases a high- resolution data collection with a very dense sampling rate. *Figure 5* in the appendix visualizes this data.in the appendix visualizes this data.

3.3 Scaling the data and Time steps

To standardize the range of the data and enhance model training, we employed a standard scaler. This scaling method is crucial for reducing the risk of overfitting by normalizing the data, ensuring that each feature contributes proportionately to the model's predictions. Additionally, we have incorporated time steps of 150 data points in our analysis. This approach allows our models to use sequences of 150 past data points to predict future values, facilitating the identification of patterns over time and improving the accuracy of our predictions in the spacecraft telemetry data. This method is especially effective in capturing temporal dependencies, which is vital for forecasting and anomaly detection in time series data.

4. Methodology

4.1 Convolutional Autoencoder (CAE)

4.1.1 What is CAE?

Convolutional Autoencoders is a type of neural network that brings together the strength of convolutional layers and autoencoders to learn the underlying structure of the input data. Convolutional Neural Network (CNN) is a deep network that can learn to identify patterns in input data and can understand spatial dependencies. For our data set of univariate time series numerical input, we want to adopt a network that enables this inherent quality of dependencies between instances. Autoencoder is a neural network where the input is the same as output; which is great for learning about unlabelled data.

4.1.2 How does CAE work?

CAE employs an autoencoder, which in turn uses convolutional layers during the encoding and the decoding processes. The input is encoded to a lower-dimensional representation in the embedded latent space and then reconstructed through the decoder with goals of creating the input data itself as the output of the network. We used mean squared error (MSE) as our evaluation metric, given by the average of the difference between the predicted and the true values. We dynamically measure our threshold value at each epoch by getting the average of $\bar{x} + 2 * \sigma(x)$, where x is the MSE and σ(x) is the standard deviation.

4.1.3 Structure of our CAE

As shown in the figure 6, the network consists of the encoder and the decoder. The former includes convolutional layers that are used to extract the features from sequences, and then the batch normalization is employed to stabilize and accelerate the training. We added dropout layers as a regularization technique to prevent overfitting by randomly setting a fraction of input to 0 at each update during training time. The decoder reverses the transformations done by the encoder to reconstruct the original input through transpose convolutional layers. The global average reduces the output of the final convolutional layer by taking the average over the time dimension, resulting in a single scalar prediction per sample.

4.2 Temporal Convolutional Network (TCN)

4.2.1 What is TCN?

Temporal Convolutional Networks (TCNs) are a type of convolutional neural network architecture used to model sequential data, making it suitable for time series data. They address some limitations of Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) networks, particularly in terms of computational efficiency and the ability to handle long-range dependencies with the help of dilated convolutions.

TCNs can handle input sequences of varying lengths without the need for padding or truncation, thanks to their convolutional nature. This characteristic is beneficial when dealing with real-world data that may not conform to a fixed size. RNNs can also handle variable-length sequences but often require additional mechanisms such as padding and masking to do so effectively.

4.2.2 How does TCN work?

As mentioned before, TCN is a 1-D convolutional network. Unlike traditional convolutional networks used in image processing, TCNs are designed to handle one-dimensional sequences. Each convolutional layer applies a set of filters to the sequence, extracting features at different timescales. It takes in a 3-dimensional tensor as an input (batch size, input shape, input size) and gives another 3-dimensional tensor as output (batch size, input shape, output size).

A key property of TCNs is causality, meaning the model should not violate the temporal order of data. The output at a certain time step should not be influenced by future input data. TCNs achieve this through causal convolutions, where the convolution operation is masked or structured such that the model can only see current and past data, never the future. This is crucial for tasks like time-series forecasting, where future data points are not available during inference.[5]

Two of the most crucial hyperparameters in TCN are **kernel size** and **dilation rate.**

Kernel Size (K): This parameter affects how many time steps in the past the network looks at with each convolutional operation. A larger kernel size increases the receptive field of each convolutional layer, allowing the network to consider more of the input sequence when generating a feature map. For our TCN model, we have chosen the kernel size to be 1, as we are already incorporating time steps in the input data.

Dilation Rate (D): This dictates the spacing between the values in the kernel. As the dilation rate increases, the receptive field becomes wider, enabling the network to integrate information over longer time spans without increasing the computational cost. For instance, a

dilation rate of 2 means the convolution skips every other input point. This allows the network to cover more time steps and capture longer-term dependencies with fewer layers.[6]

Receptive Field (R): The receptive field in the context of neural networks, particularly convolutional neural networks (CNNs) and Temporal Convolutional Networks (TCNs), refers to the region of the input data that a particular feature or output is looking at or influenced by. It essentially describes how much of the input data affects or contributes to the computation of a specific output in the network.

Calculating receptive field size using kernel size and dilation rate:

Receptive field size for one layer: R = (K-1) * D + 1
Receptive field size across all layers: R = 1 + ∑(K-1) * D
Receptive field size if dilation rate doubles each layer: R = 1 + (K-1) * ∑2^i

4.2.3 Structure of our TCN

The TCN we have used consists of 4 layers - 1 input layer, 2 hidden layers and 1 output layer. The input layer consists of the shape of the input data, kernel size, dilation rate and activation function. As mentioned earlier, the kernel size remains 1 for all the layers. Dilation rate is 1 for the input layer and it doubles each layer (2 for the 1st hidden layer and 4 for the 2nd hidden layer). Activation function used for the input and hidden layers is “rectified linear unit” (reLU). We have a dropout of 0.2, which means that 20% of the data is discarded after each layer. This helps us to prevent overfitting of the model. We also use the flatten() function to convert the 3-dimensional output that TCN gives into a 1-dimensional output. The output layer consists of 1 neuron which uses the “linear” activation function to give linear function of the input as output. It uses the “Dense” function which means that it's a fully connected layer. *Figure 9* in the appendix shows the architecture of our TCN.

5. Results

The models classify a data point as an anomaly if the error in predicting that point is greater than a certain threshold. This threshold is calculated using the following formula:

Threshold = $\bar{x} + 2 * \sigma(x)$, where x is the mean squared error (MSE).

5.1 Results of CAE

The model predicted around 19.1% of the total data points as anomalies for the **Bus Voltage** dataset with a MSE of 0.678 in predicting the data. The threshold was calculated to be 0.638. *Figure 10* gives a visual representation of the anomalies.

For the **Bus Current** dataset, our CAE model had a mean squared error of 0.929 while predicting the data. It classified around 13% of the data anomalies considering the error threshold to be 0.885. *Figure 11* gives a visual representation of the anomalies.

For the **Battery Temperature** dataset, the model had an MSE of 0.539. 48.95% of the total data were detected as an anomaly with a threshold of 0.532. *Figure 12* gives a visual representation of the anomalies.

In the **Reaction Wheel Temperature** dataset, the model had an MSE of 0.603. 24.7% of the data was classified as anomalous points with a threshold of 0.697. *Figure 13* gives a visual representation of the anomalies.

Finally, the model had an error of 0.857 while predicting the values for the **Wheel Speed** dataset. It classified 1.2% of the total data point as anomalies, which is about 0.4% of the total dataset. The threshold here was 1.019. *Figure 14* gives a visual representation of the anomalies.

5.2 Results of TCN

For the **Bus Voltage** dataset, our TCN model had a mean squared error of 0.163 while predicting the data. It classified 1221 data points (around 6% of the total data) as anomalies considering the error threshold to be 0.188. *Figure 15* gives a visual representation of the anomalies.

The model predicted 398 (around 2% of the data) data points as anomalies for the **Bus Current** dataset with a MSE of 0.191 in predicting the data. The threshold was calculated to be 0.267. *Figure 16* gives a visual representation of the anomalies.

In the case of the **Battery Temperature** dataset, the model had an error of 0.084 and predicted 1253 data points, which is around 8% of the total data, as an anomaly with a threshold of 0.11. *Figure 17* gives a visual representation of the anomalies.

For the **Reaction Wheel Temperature** dataset, the model had an MSE of 0.061. 474 anomalous points (around 10% of the total data) were detected with a threshold of 0.091. *Figure 18* gives a visual representation of the anomalies.

Finally, the model had an error of 0.013 while predicting the values for the **Wheel Speed** dataset. It classified 213 points as anomalies, which is about 0.4% of the total dataset. The threshold here was 0.092. *Figure 19* gives a visual representation of the anomalies.

5.3 Comparing the results

We can see that the TCN model performs much better than the CAE model. It had a much better MSE compared to CAE and was also able to better detect anomalies as a result.

6. Limitations

Anomaly detection with time series data particularly creates a challenge of the inherent dependencies between data instances, when most of machine learning has the underlying assumption of the opposite before employing various algorithms. With that, the literature on networks and algorithms on such a dataset is more limited than a more straightforward numerical data with independent instances. Another limitation we faced was something that all unsupervised learning faces - the lack of labeled data. This in turn made it hard to cross-validate our results, even with existing research work [2]. Moreover, the case of anomaly detection in spacecraft telemetry in particular is a very unique scenario. This means there is no existing pre-trained network that is open-sourced that stores the pattern such as that of our dataset for us to employ. The rarity also showcases as an opportunity - less literature means more exploration and more space for discovery. In this study, we held onto that as we experimented with our varied architectures on our dataset.

References

[1] What is Anomaly Detection?

[2] Unsupervised Machine Learning for Spacecraft Anomaly Detection in WebTCAD.pdf

[3] A Better Autoencoder for Image: Convolutional Autoencoder

[4] Chen, Shuangshuang, and Wei Guo. 2023. "Auto-Encoders in Deep Learning—A Review with New Perspectives" Mathematics 11, no. 8: 1777. https://doi.org/10.3390/math11081777

[5] Luke Guerdan | Diving Into Temporal Convolutional Networks

[6] Dilated Convolution - GeeksforGeeks

Github Repository: tenzin-kunsang648/FinalProject_DS5230_Spring24 (github.com)

Appendix

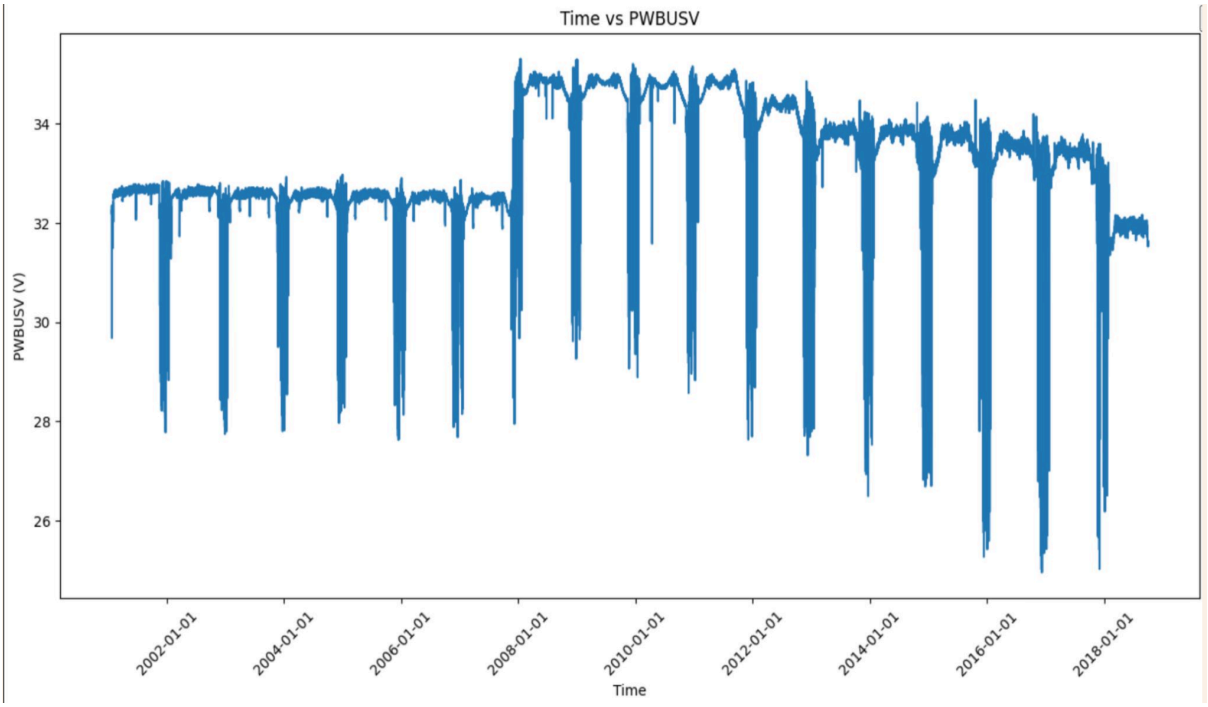

Figure 1: Plot of Time vs Bus Voltage

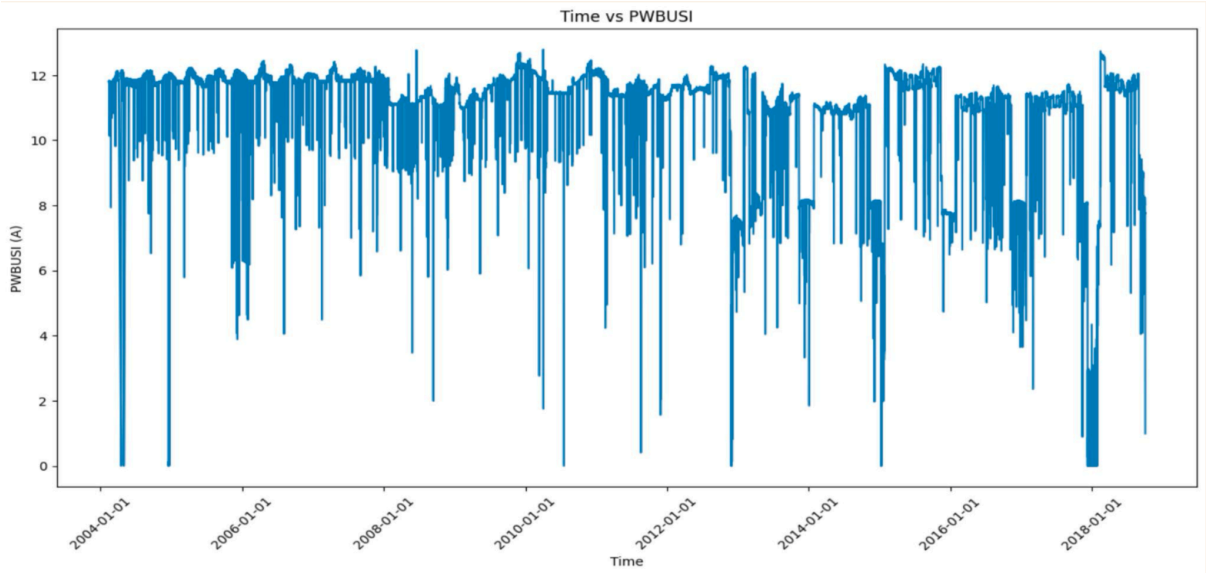

Figure 2: Plot of Time vs Bus Current

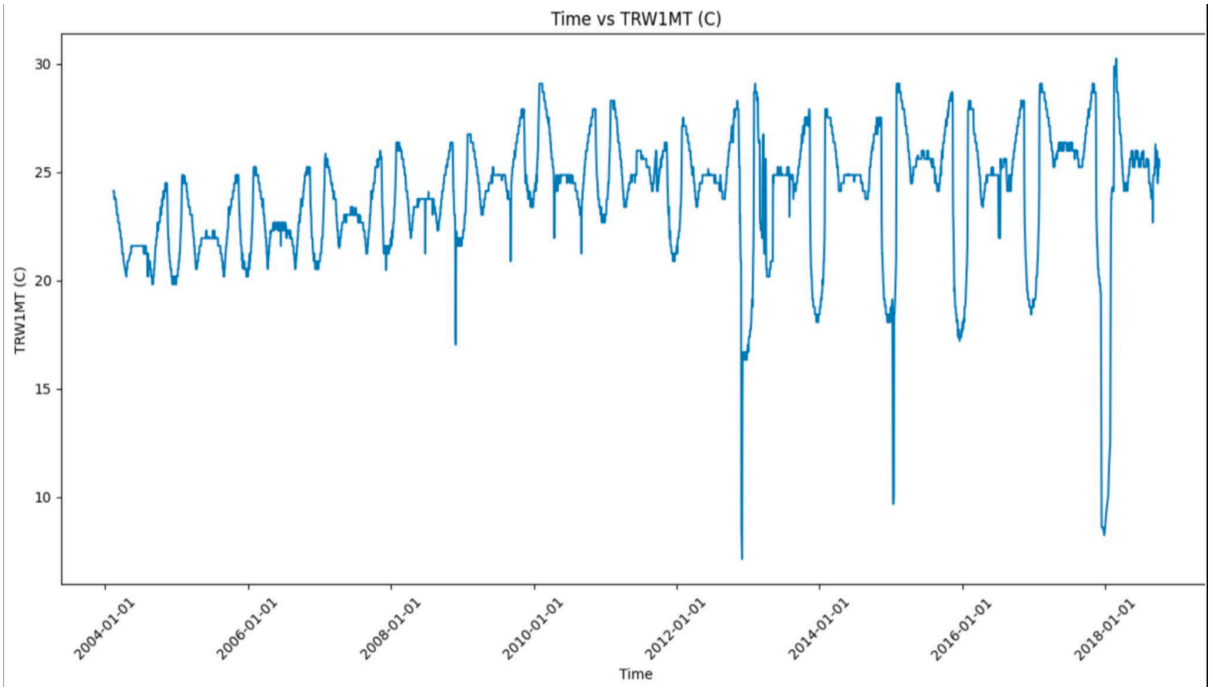

Figure 3: Plot of Time vs Wheel Temperature

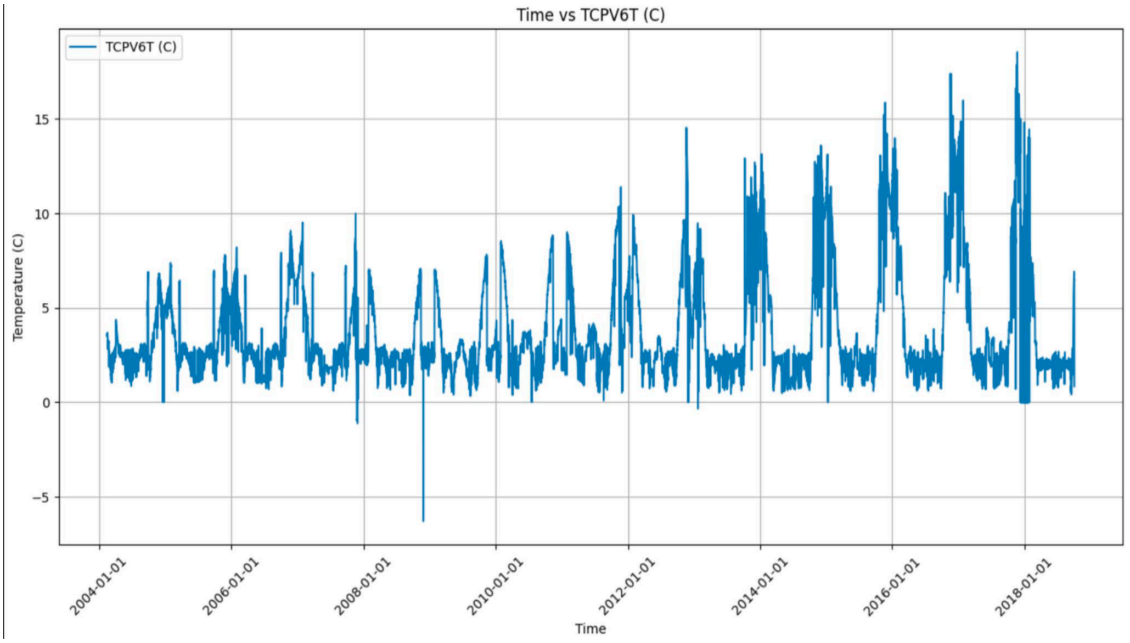

Figure 4: Plot of Time vs Battery Temperature

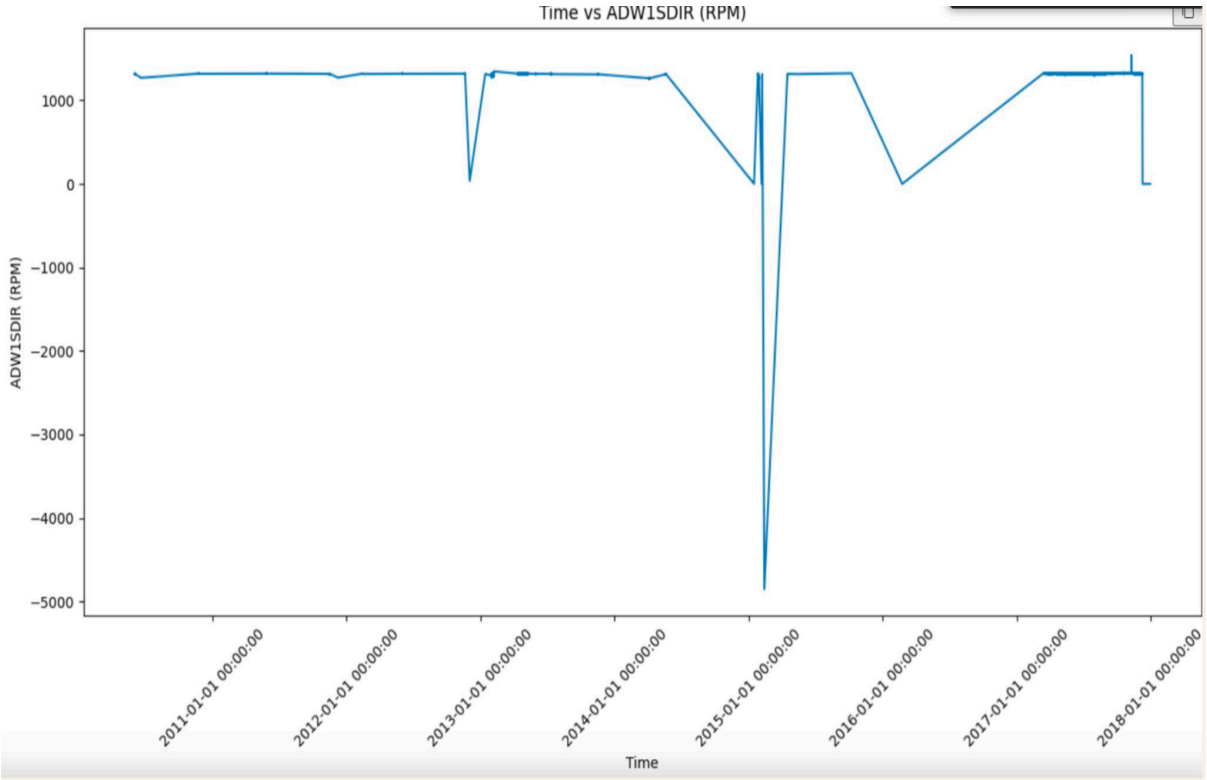

Figure 5: Plot of Time vs Wheel RPM

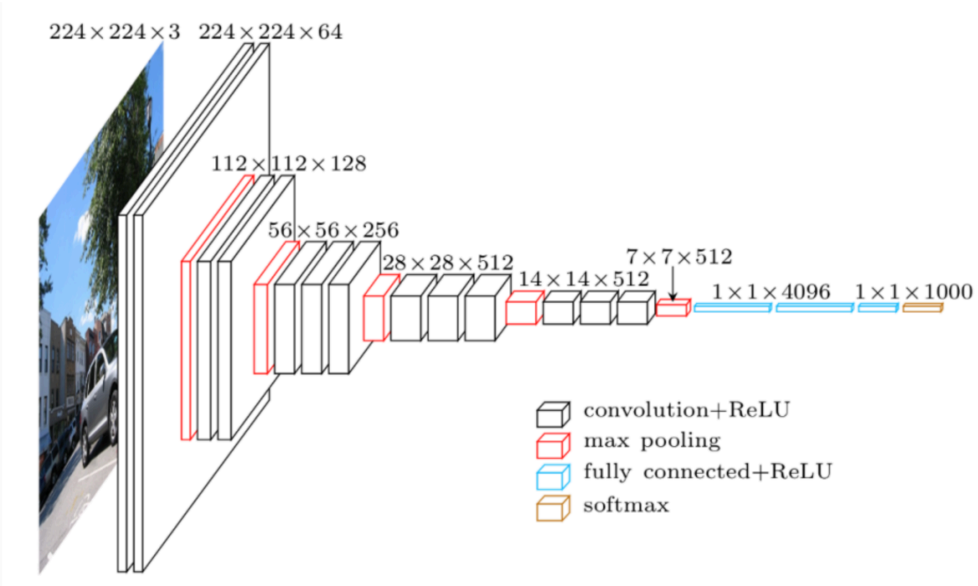

Figure 6: Structure of a typical convolutional neural network [1]

[1] https://www.jeremyjordan.me/convnet-architectures/

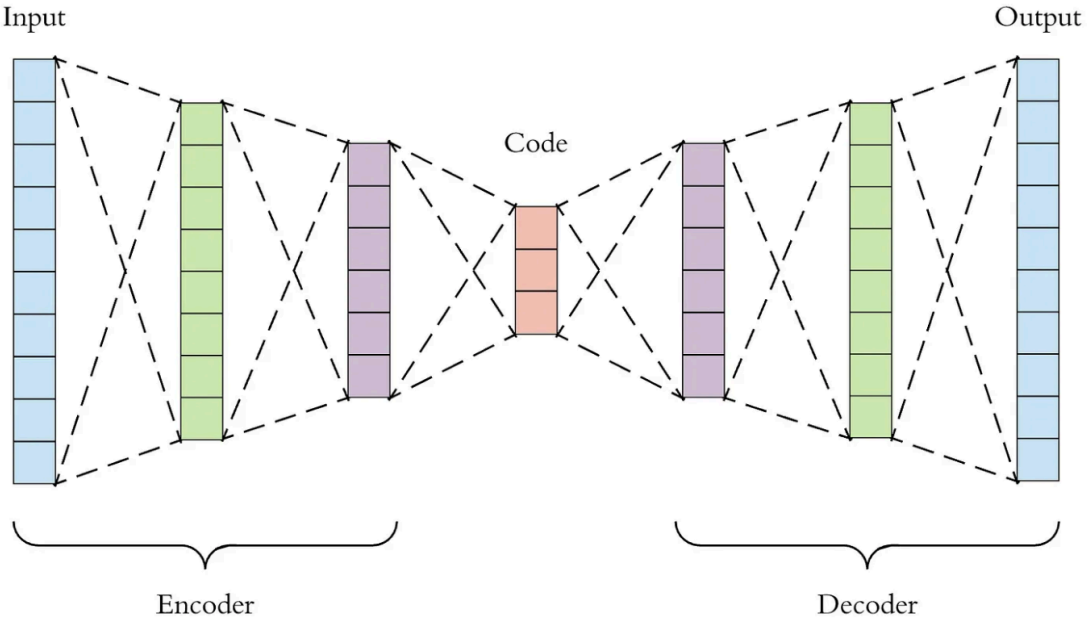

Figure 7: Structure of an autoencoder[2]

Layer (type)	Output Shape	Param #
input_layer_15 (InputLayer)	(None, 5, 1)	0
conv1d_45 (Conv1D)	(None, 5, 16)	64
batch_normalization_30 (BatchNormalization)	(None, 5, 16)	64
dropout_30 (Dropout)	(None, 5, 16)	0
conv1d_46 (Conv1D)	(None, 5, 8)	392
conv1d_47 (Conv1D)	(None, 5, 4)	100
conv1d_transpose_45 (Conv1DTranspose)	(None, 5, 8)	104
batch_normalization_31 (BatchNormalization)	(None, 5, 8)	32
dropout_31 (Dropout)	(None, 5, 8)	0
conv1d_transpose_46 (Conv1DTranspose)	(None, 5, 16)	400
conv1d_transpose_47 (Conv1DTranspose)	(None, 5, 1)	49
global_average_pooling1d_15 (GlobalAveragePooling1D)	(None, 1)	0

Figure 8: Architecture and configuration of the CAE model

[2] https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 1)	0
conv1d (Conv1D)	(None, 150, 64)	128
dropout (Dropout)	(None, 150, 64)	0
conv1d_1 (Conv1D)	(None, 150, 64)	4,160
dropout_1 (Dropout)	(None, 150, 64)	0
conv1d_2 (Conv1D)	(None, 150, 64)	4,160
dropout_2 (Dropout)	(None, 150, 64)	0
flatten (Flatten)	(None, 9600)	0
dense (Dense)	(None, 1)	9,601

Figure 9: TCN architecture

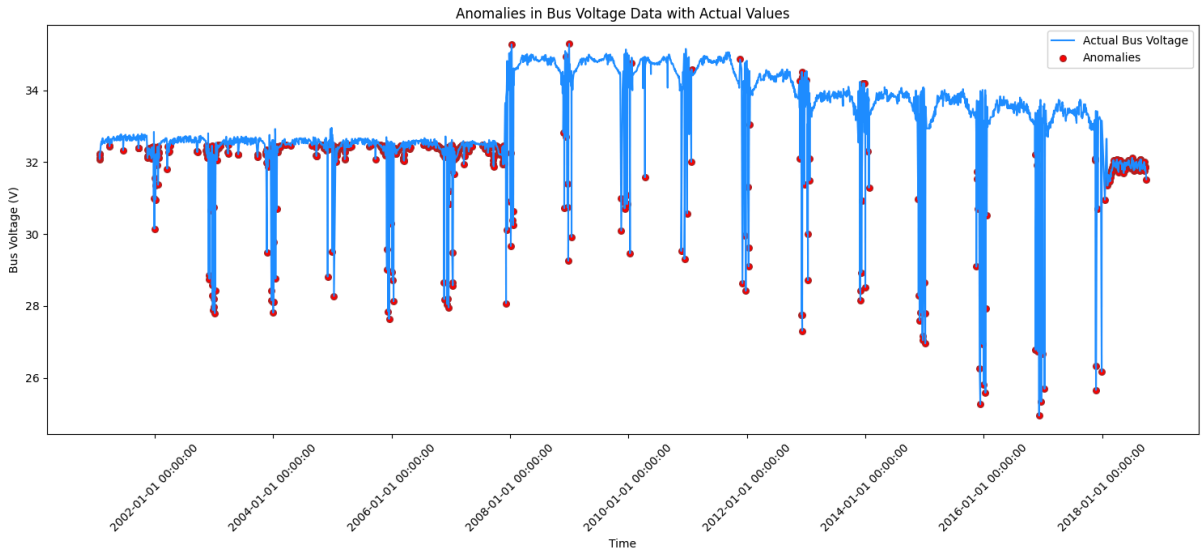

Figure 10: Anomalies in bus voltage data using CAE

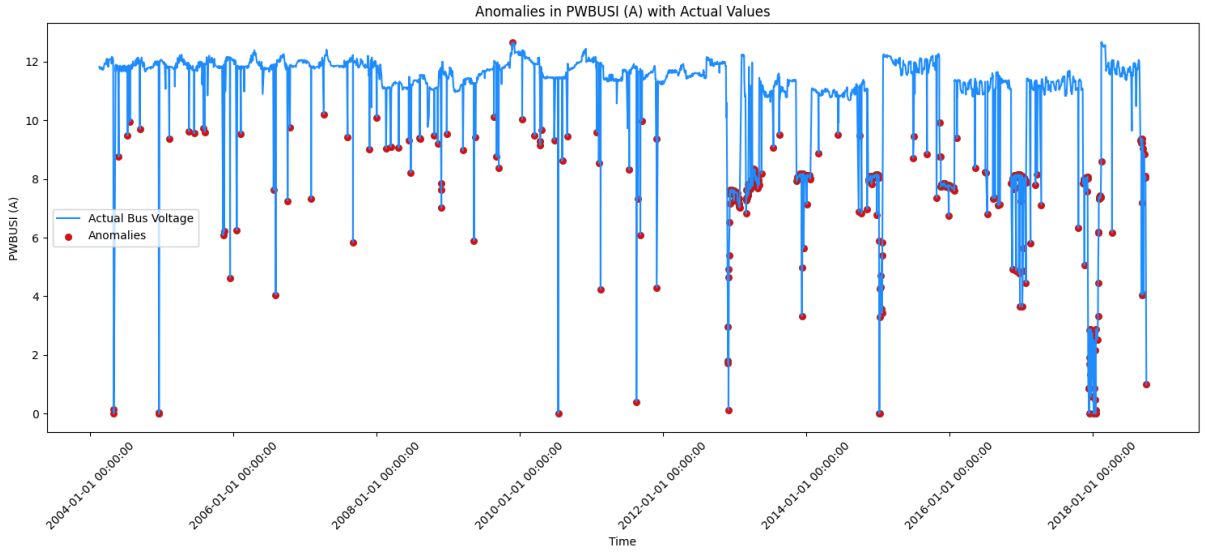

Figure 11: Anomalies in Bus current data using CAE

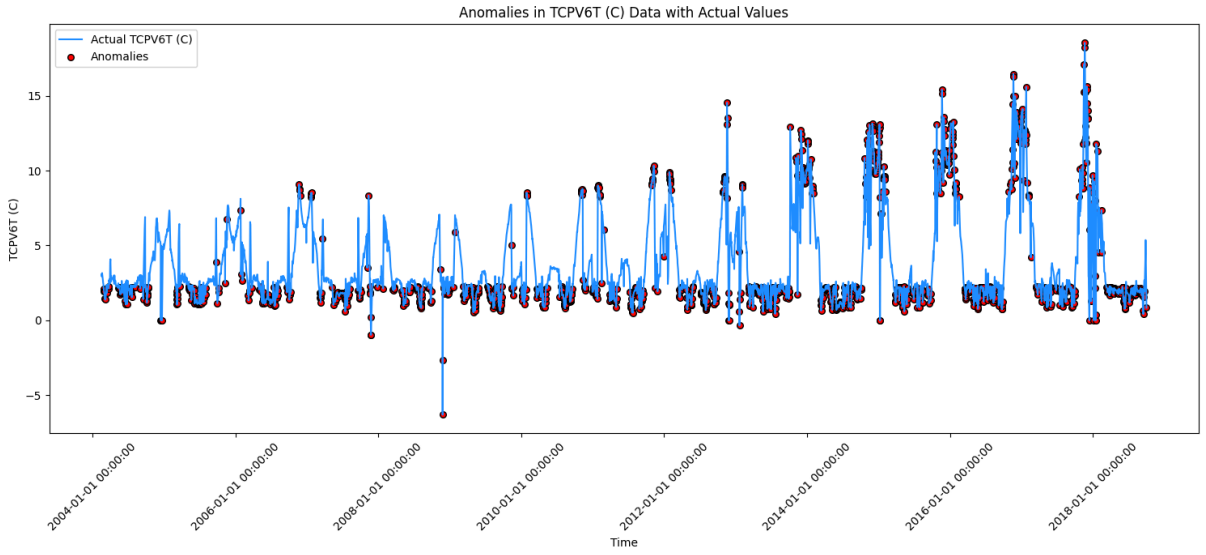

Figure 12: Anomalies in Battery temperature data using CAE

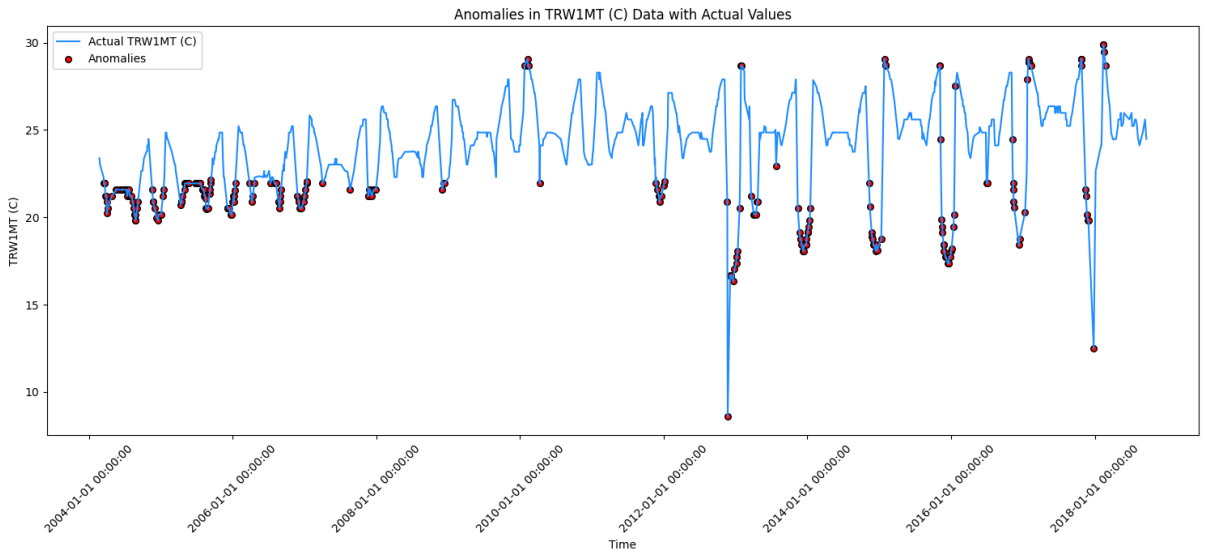

Figure 13: Anomalies in Reaction wheel temperature dataset using CAE

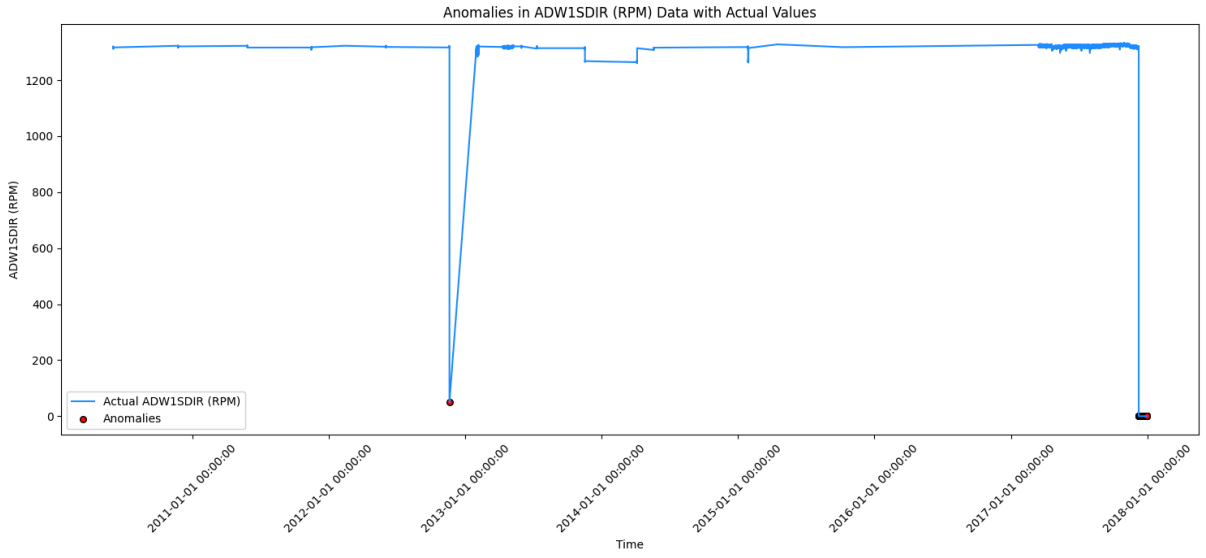

Figure 14: Anomalies in wheel speed data using CAE

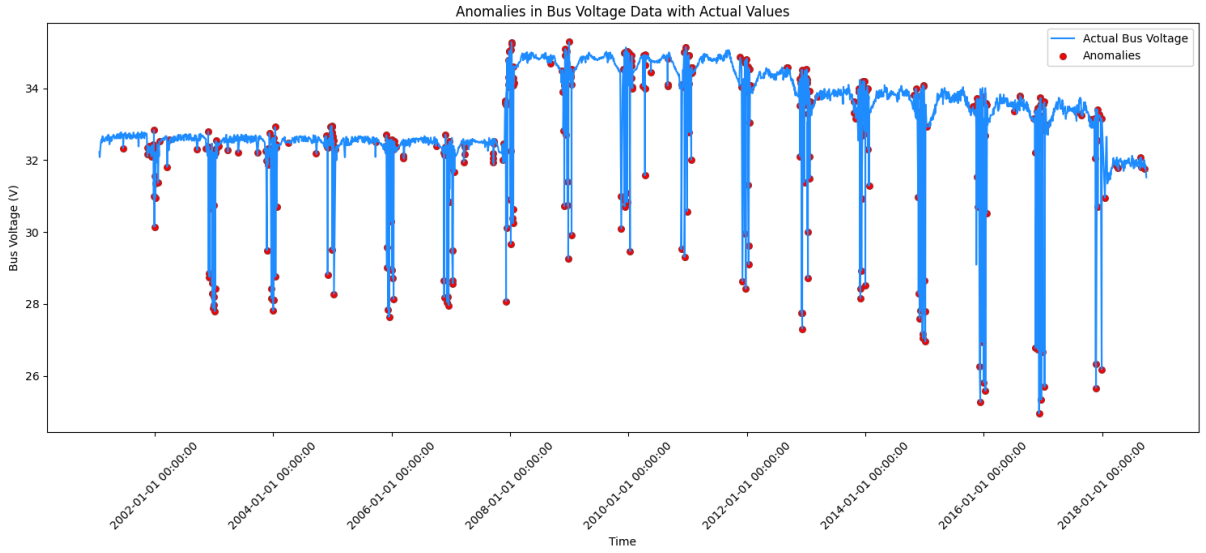

Figure 15: Anomalies in bus voltage data using TCN

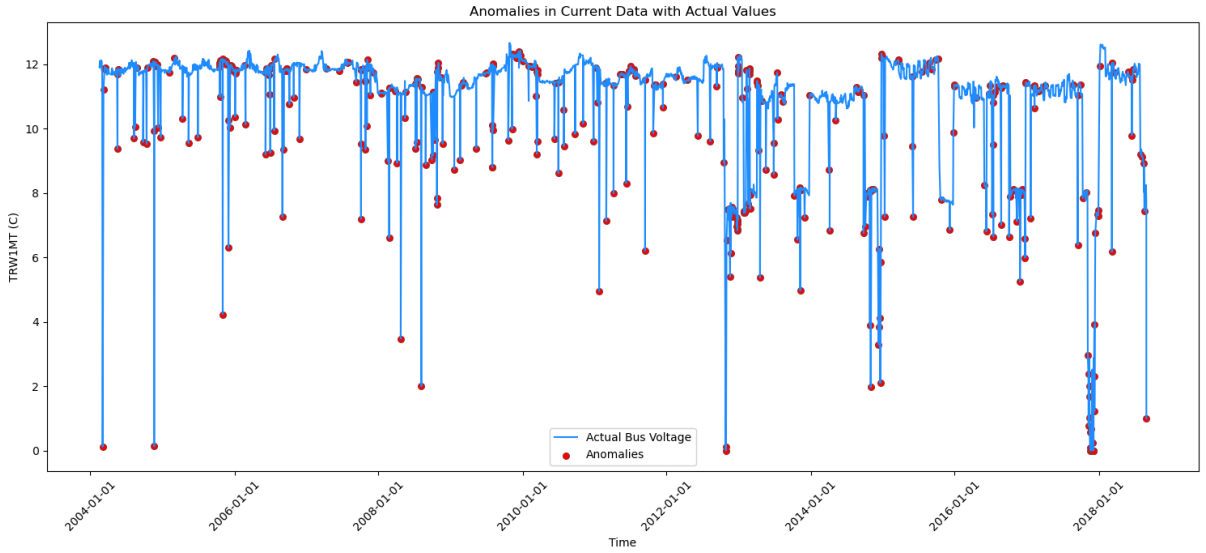

Figure 16: Anomalies in Bus current data using TCN

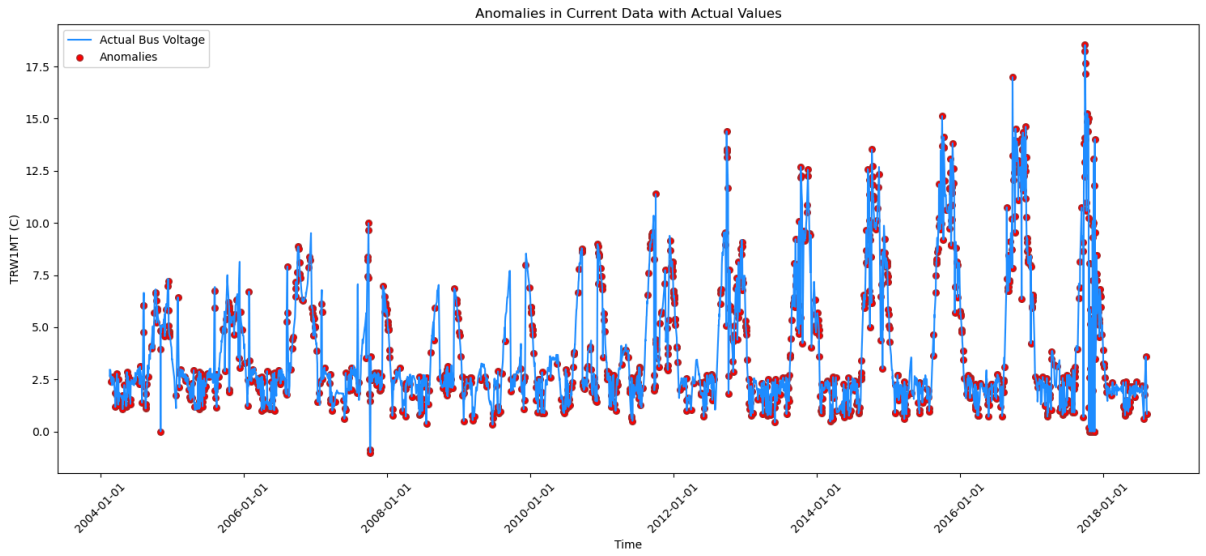

Figure 17: Anomalies in battery temperature using TCN

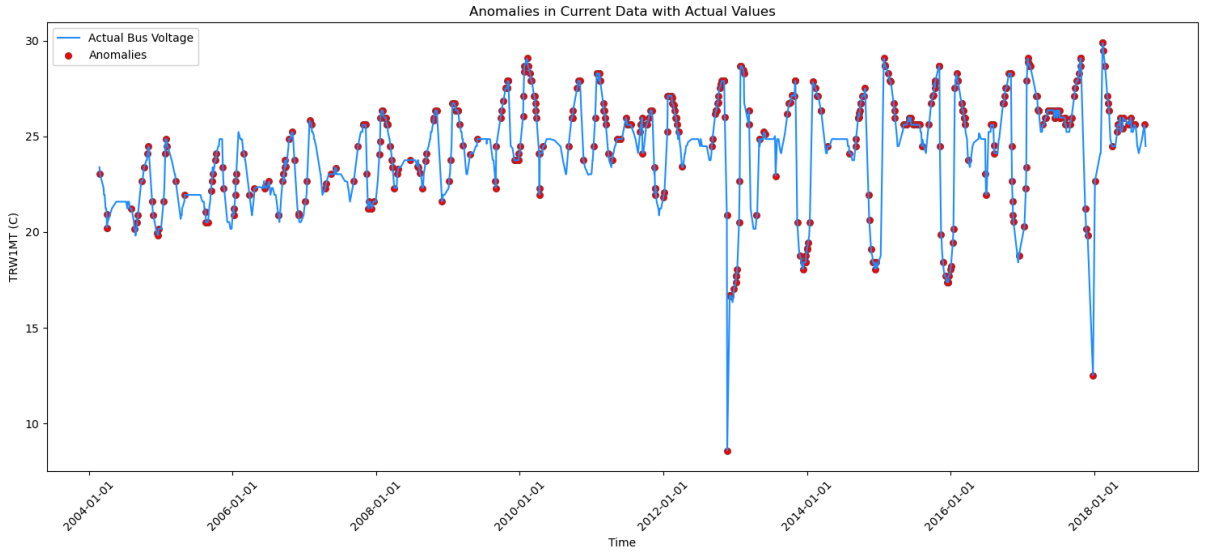

Figure 18: Anomalies in Reaction wheel temperature using TCN

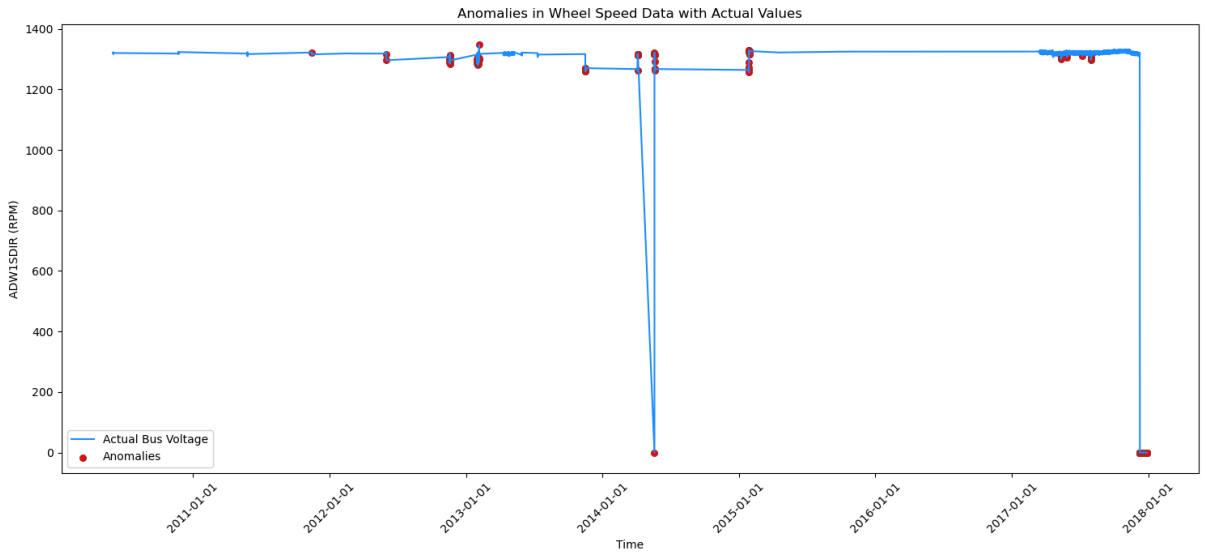

Figure 19: Anomalies in wheel speed data using TCN