Automated Track Infrastructure Recognition Using Vibration Analysis

Name: Daniel Fathi

Introduction

In this project, the focuse is to build an automatic method that can detect railway infrastructure events like bridges, turnouts, and rail joints based on vibration signals recorded on the train. To do this, we have a sensor system that was installed on a train, it included two vibration sensors on the left side and on the rightside and a GPS unit. The vibration sensors recorded rail vibrations at a high sampling rate (500 Hz), and the GPS unit simultaneously tracked the geographical coordinates latitude and longitude, and the speed of the train.

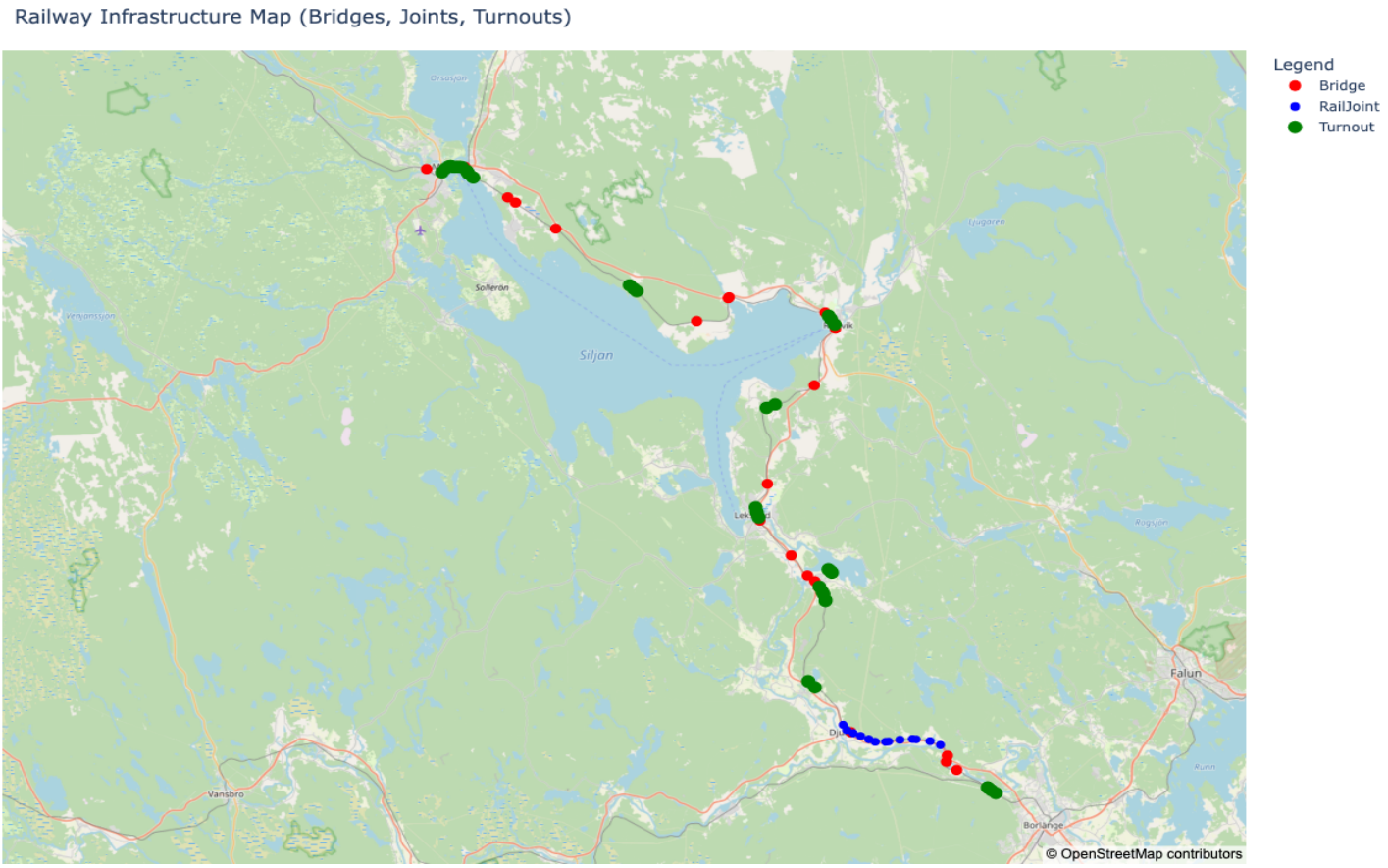
Also, the information about the infrastructure events (bridges, turnouts, and rail joints) was collected from the central control system. But not all of these events actually happened on the exact track the train was moving on, it was so that some of them were on nearby side tracks. So, I add a filtering step to keep only the events that the train actually passed over. When we are able to automatically detect these events by analyzing the vibration signals we can make it possible to monitor the track condition in real time. This helps improve safety and makes it easier to plan and optimize railway maintenance.

Project Steps

The data we used has two parts: Data 1 is location info about infrastructure events a list of bridges, turnouts, and rail joints. Data 2 has the vibration sensor and GPS data recorded while the train actually moved along that same track.
Here we can see what I did step by step:

1) Loading Infrastructure Data and Drawing the Static Map:

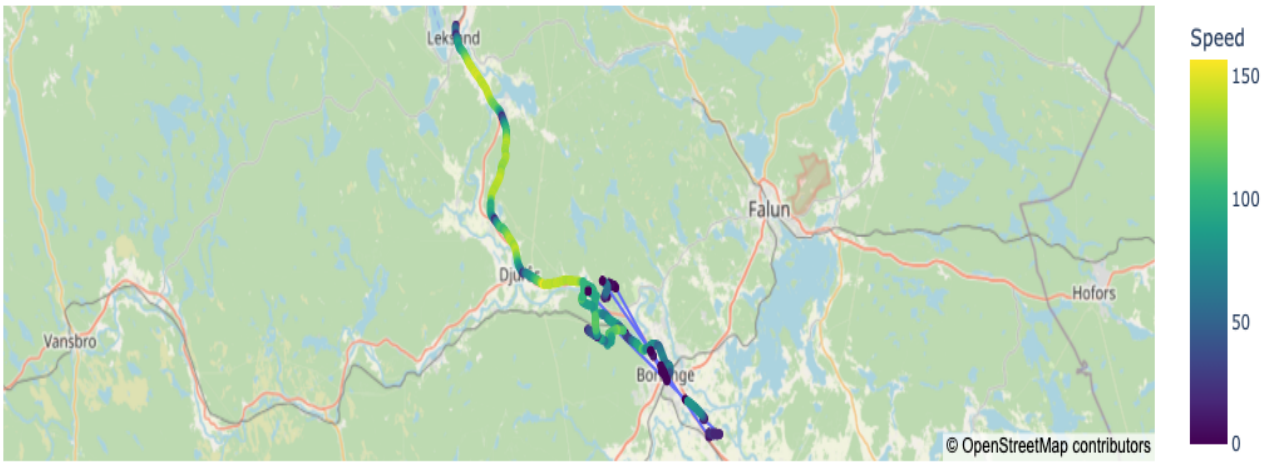
First, I loaded the CSV files for bridges, turnouts, and rail joints from the Data 1 folder and then I used Code 1. Each file had the GPS coordinates (latitude and longitude) of one type of infrastructure. Then, I plotted these points on a static map to see how the infrastructure is spread out along the track. For example, the figure below shows the location of all bridges, turnouts, and joints based on their coordinates. Each type of infrastructure is marked in a different color, red for bridges, green for turnouts, and blue for joints.



2) Selecting the Right Track Data and Plotting the Speed Map:

From the Data 2 folder, I picked the GPS and vibration files that belonged to the same track as before, in this case it was the Borlänge–Mora track in Sweden. I used Code 2, loaded and merged the files GPS data coordinates and speed and the two vibration channels, then I drew the full path of the train on a map and used color to show the trains speed at each point along the route.
On this map, the color of each point shows the trains speed, the blue and purple for low speeds and yellow for higher speeds. This image helps us understand how the speed changes during the route.

GPS Track (Data 2)



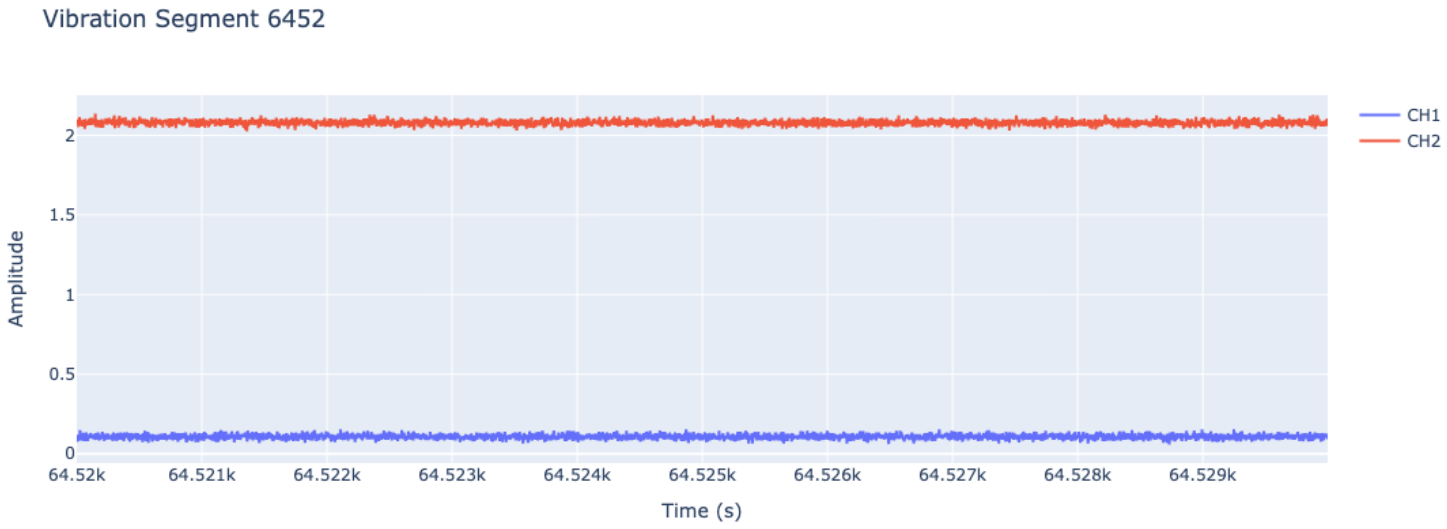
3) Filtering Only the Events Actually Passed by the Train:

I had both the infrastructure data from step 1and the actual GPS path of the train from step 2 I filtered out the events that the train did not really pass over. Or I can say that I matched the coordinates of the bridges, turnouts, and joints with the GPS path, and removed any events that was not close to the trains route. After this filtering, I got a final list of infrastructure events that the train actually encountered during the trip .

4) Segmenting Vibration Signals and Labeling Events:

The vibration signals recorded by each sensor were split into short time segments. In this project, I used 10-second segments which equals 5000 samples at a 500 Hz sampling rate. Then I checked if each vibration segment matched an infrastructure event from before, and gave it a label based on that. It was like If the time and location of a 10-second segment lined up with an event I labeled it with the type of that event like bridges for bridges, turnouts for turnouts, and Rail joints joints and if there was no infrastructure event during that segment it as other.

The figure below shows one 10-second segment of vibration data. Both channels CH1 and CH2 are plotted over time. This segment shows a distinct vibration pattern like a spike or specific oscillation, so it was labeled as an infrastructure event for example crossing over a rail joint.



5) Extracting Numerical Features from Each Vibration Segment:

In this part I extracted a set of numerical features from each one. The goal was to summarize the important info from the signal to feed into machine learning models. Some of the features was basic time domain stats like mean, standard deviation, maximum, and skewness of the acceleration signal in each segment.

For example, a rail joint might cause a short, sharp spike in the signal which would show up in features like max and standard deviation. On the other hand, crossing a bridge might produce a more complex vibration pattern with specific frequency content something you'd catch in frequency based features. Choosing the right features is very important in how accurate the detection models are.

6) Training Classical ML Models and a Deep Learning Model:

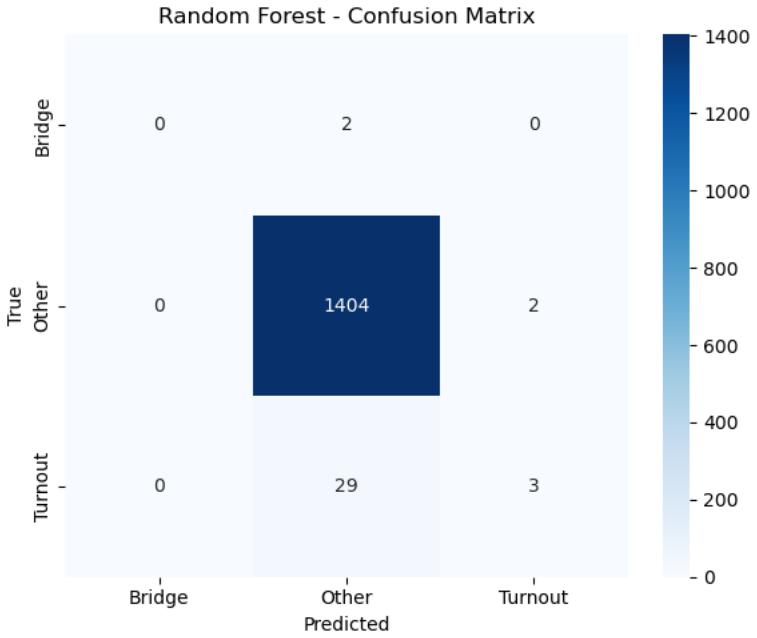
At this stage, I used the extracted features and their labels to train several machine learning models.

First, I trained three classic ML models: **K-Nearest Neighbors (KNN)**, **Support Vector Machine (SVM)**, and **Random Forest**, using the labeled dataset. Each model learned to predict what kind of event (bridge, turnout, joint, or other) each vibration segment belongs to based on its features.

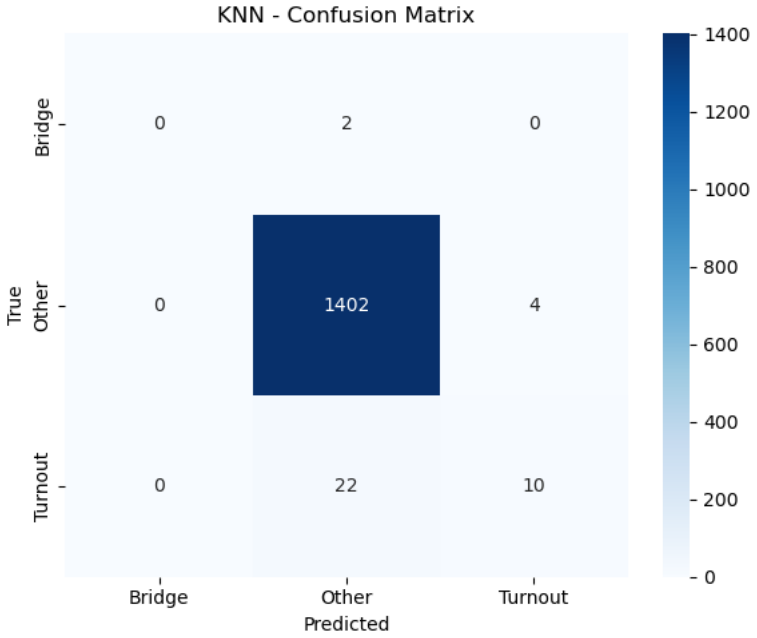
I also trained a deep learning model using the **Keras** library. This was either a multi layer neural network (MLP) or a convolutional network (CNN), which could learn directly from the raw data or more complex features to pick up vibration patterns.

The results from all three classical models are shown as confusion matrices. By looking at these matrices, we can see how well each model performed for each class and where they struggled.

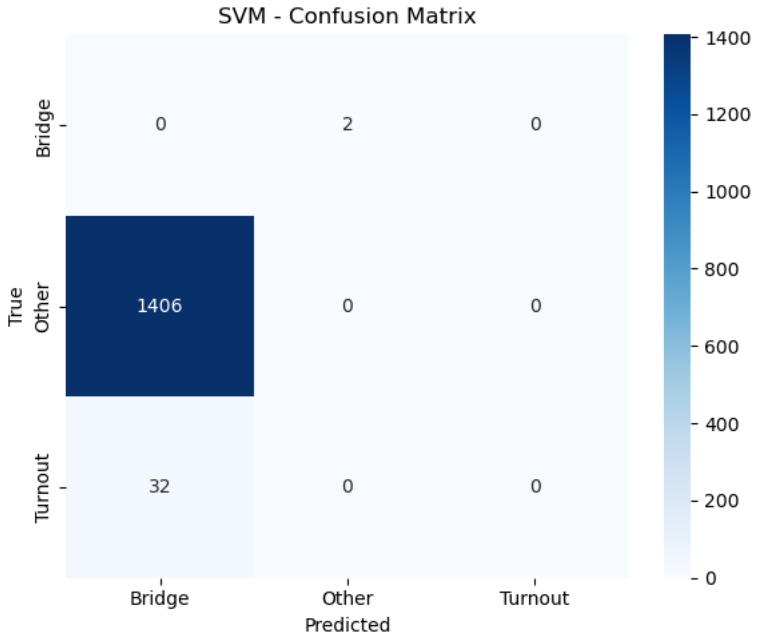
Random Forest: Very good at classifying 'Other', limited success with Turnout



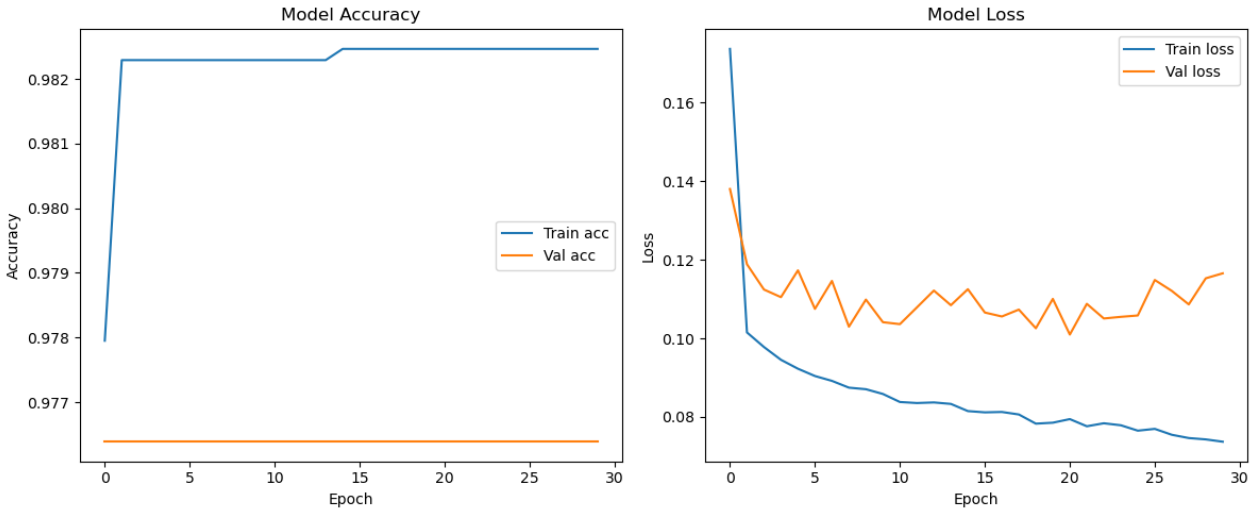
KNN: Best performance among classic models for Turnout, weak on Bridge



SVM & Logistic Regression: Poor performance on minority classes



Deep model generalized better, despite imbalance.



Conclusion

This project showed that by combining vibration and GPS data with machine learning, it will be possible to automatically detect key infrastructure elements on a railway track. Among all the models, **Random Forest** gave us the best performance with high accuracy.
Of course, there were still some issues like signal noise and precise synchronization but with better sensors and smarter preprocessing, the system's accuracy can be pushed even further.
This kind of approach could be scaled up for real world rail monitoring, helping detect faults and weak points automatically a solid step toward smarter and more modern railway maintenance.