DEPARTMENT OF CIVIL, ENVIRONMENTAL AND GEOMATIC ENGINEERING

Semester Project Report

Data-driven identificiation and classification of rail surface defectse

Aiyu Liu

Supervised by: Cyprien Hoelzl, Prof. Eleni Chatzi

Tuesday 11$^{\text{th}}$ February, 2020

Acknowledgements

This semester project would not be possible without the help of my supervisors, Cyprien Hoelzl and professor Eleni Chatzi. I first reached out to Eleni for a semester project opportunity during ETH week – for the purpose of improving my skills in doing research. Shortly therafter, I was introduced to her PhD student, Cyprien Hoelzl, about a project revolving around identifying and classiying defects on train tracks.

Cyprien has been an exceptional mentor throughout this entire project. I received all the acdemic guidance necessary and whenever I had issues, I could always drop by his office or text him, after which helpful answers would promptly ensue. From the beginning, I could tell that he is down-to-earth, hard-working, very intelligent and possess great specialization in the field of train maintenance and monitoring. He is very good at explaining difficult concepts (with his quick and intuitive hand-drawings) and provided me with many informative resources. Furthermore, he truly cared about my progress, goes out of his way to aid me, and provides constructive feedback for everything I present to him.

Eleni has also been very supportive about my progress, always arranging intermediate update sessions and staying on top of the project. These have been a great driver in keeping me accountable and making further progress. Eleni is very approachable, kind, and great at providing feedback at the intermediate update sessions. She is extremely active in her endeavours and one can that she is an expert in the field of Structural Health Monitoring (among others).

Finally, I greatly appreciate the help that I have received from Eleni's other PhD student, Harry (Mylonas Charilaos). He has provided me with very informative tools/feedback for my work with neural network architectures. Although interactions were few, one can immediately tell that he is very knowledgeable about the field of machine learning.

I have great gratitude for this opportunity working alongside Eleni and her multi-talented team. It has been a pleasant and educational experience. I sincerely could not ask for better supervisors.

Contents

Chapter 1

Introduction

Railway companies need to continuously and sufficiently maintain the train tracks and optimally detect defects in order to have a more punctual and more effective train system. However, the current system is expensive, time consuming and ineffective. That is, maintenance agents need to walk along tracks and check them for defects. For visualisation purposes, there is roughly 5200 km of rails in Switzerland which needs to be inspected by 40 experienced inspectors.

In order to cope with this issue, Swiss Federal Railways (SBB) has specifically built two new special diagnostic vehicles (SDV) designed for defect identification among other purposes. For this, two accelerometers have been installed at the front and back of the vehicle to collect the signal responses from the wheel and the train track.

insert picture?, mention boogey?

A defect in train tracks can be seen as a discontinuity. As a train passes over this discontinuity, it will result in a perturbation that can be detected by sensors. It is our main assumption that each type of defect will have a specific signature that will allow its identification and classification. This is similar to the idea presented in [1] about human activity recognition.

1.1 Objective

As the title implies, the objective of this project is to identify and classify rail surface defects. To do this, we aim to build an effective pipeline that takes information about defects as input and outputs a classification confidence for these defects. By succesfully identifying and classifying the defects, we take one step further towards reducing delays and making trains more punctual and reliable. The first step in this development consists of identification and classification, while the second step ultimately consists of future defect prediction.

1.2 Defects

Evidently, a defect can be seen as a deviation from the standard train track. For the exact defect type, SBB has self-constructed a database for the individual defect definitions . Here is a few examples:

is this a recognized system?

insert pictures

Generally, a defect is separated into two overarching types: range- and point-defects. I.e. a defect that is detected at a single point versus a defect that is detected at varying lengths.

comment on the inserted pictures about which is point vs range

For this project, we have solely focused on the point defects for analysis, as this simplifies the problem statement. A point defect is perceived as a sharp signal response, whereas a range-defect is perceived over a greater time period. We thus disregard range-defects such that we do not have to deal with the extra, associated factors.

See list of defect types in appendix?

1.3 Data

The data has been collected and provided by SBB. Using their SDV, SBB has made trips back and forth to different cities in Switzerland in order to collect various data including but not limited to accelerometer data. After getting the data from SBB, it then goes through a processing pipeline (designed by Cyprien), after which the data can be manipulated with `python` dataframes (from `pd.DataFrame`). The accelerometer captures the accelerations at the XYZ-axes (along with the timestamps at each recording), of which we are only concerned with the Y-axis for the vertical pertubations for the accelerometer at the axle. See appendix A for visualisations of the accelerometer placements on the SDV.

Furhtermore, the locations of the defects have to be retrieved from SBB's database. which were retrieved by Cyprien.

Which data was worked on, put in tables, switches, ins joints and defects

We need to define terminlogy of these: defects, ins joints, switches. In the following we will use entities as an umbrella term for the aforementioned track phenomenons.

which track entities are we actually analysing

1.4 Code

The code is written purely in python. To create neural network architectures, we are using: `keras` along with `tensorflow`. `keras` is essentially a high-level neural networks library which runs on top of `tensorflow`. It has consistent, a simple API and provides clear and actionable feedback upon user error. Models are easily made by connecting configurable building blocks together, with few restrictions [2]. The models were trained in `Google Colab`, which is a web application provided by Google that enables users to run python code in the web browser with access to GPUs[1]. It is very similar to `Anaconda's Jupyter Notebooks`, except that `Colab` runs in the browser, is collaborative and provides free usage of GPUs (meaning model training goes faster).

All the code can be found on github:
https://github.com/Aiyualive/SemesterProject2.0.

The specific model execution workflow can be found `Colab`:
https://colab.research.google.com/drive/12VBz_KrJxeyR_pjpkC87fewv5aMSEI5_

[1] https://colab.research.google.com/notebooks/intro.ipynb

Chapter 2

Design and Implementation

For the process of defect classification we designed the pipeline in 2.1. In the next sections, I would like to give an overview of how each step was implemented.

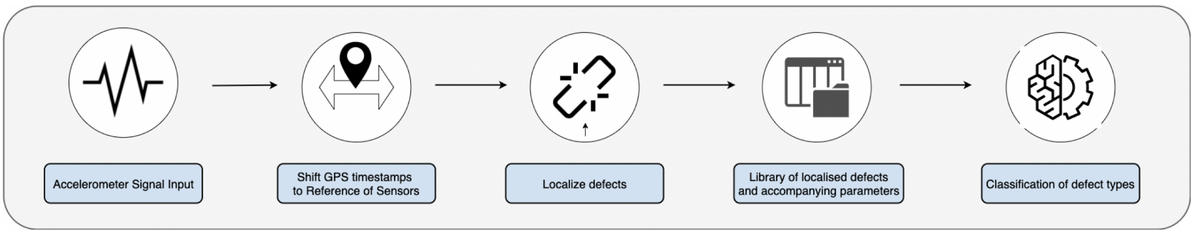

Figure 2.1: Primary pipeline

2.1 Shift of GPS timestamps

The SDV has its GPS sensor installed at a specified location on the vehicle body. However, what we need to achieve is the position (covered distance) at each accelerometer at either sides of the GPS. Since the GPS sensor is sampled at a lower frequency compared to the accelerometers (20kHz vs 24kHz respectively), we first need to get the corresponding positions for each accelerometer sample. This is done by interpolation using the timestamps of the accelerometers and GPS.

Depending on the direction of the vehicle we then subtract/add the offset between the accelerometers and the GPS sensor with regard to the position of these sensors on the vehicle body.

insert drawing of how it is calculated?

2.2 Peak windows

Retrieving the signal response around the defect location forms a crucial aspect in the overarching pipeline. The goal of this step is to, around each defect, create a ”window” containing accelerometer accelerations of a specified time length – wherein Within the highest acceleration recording around is found in the center. As a result, all of these windows would be uniform in the sense that they are all centered according to the highest recording of a defect. It is then assumed that each window forms the signature of each track entity.

Since we are assuming that each track entity is identified by a well-formed peak, we first need to find this peak within a reasonable offset from the defect location, after which we center around that within another reasonable offset.

In the code, this is done by defining two parameters: `find_peak_offset = 1` and `window_offset = 0.5`. I.e. given a defect timestamp, we search for the highest acceleration recording that has occured 1 second after and 1 second before the defect timestamp. Once the peak has been found, we then center it in a 1 second window (0.5 sec on each side).

insert a drawing of how this works?

2.3 Entity library

The peak windows arguably forms the central feature of the defect library. However, from domain knowledge, other features like speed also needs to be considered for our neural network. Apart from the peak windows, we have also extracted a variety of relevant features that might be useful for classifcation. These include .

list, and how was it implemented

some can be extracted directly by the original dataframe, some needs some processing.
talk about only analysing point defects
show a few entity signals and their features, appendix for more signals?
refer to the defects presented in introduction

2.4 Classification

We have created a primary `NN` class (short for neural network) along with a `ModelMaker` class. The former does everything from pre-processing the data to evaluating the used model. The latter, as the name suggests, is utlised for creating and using different models, which is useful as we can keep track of how the models have been modified and improved.

2.4.1 NN class

To make a classification, we first need to select the relevant features. Then we simply feed the features into an `NN` object, where the API of the `NN` class can be called for classification. The usage of the `NN` class is demonstrated below in 2.1.

API of NN class	
`__init__()`	initialises a `NN` object
`prepare_data()`	pre-process data, this includes standardisation of data
`make_model()`	uses `ModelMaker` class to select a model
`fit()`	trains the model
`classify()`	trains the model

Table 2.1: To train a model, these functions needs to be called sequentially

Other useful API functions	
`measure_performance()`	currently only done on validation data
`plot_metrics()`	plots the metrics
`plot_confusion_matrix()`	plots the confusion matrix
`load_weights()`	
`load_model_()`	
`save_history()`	
`save_model()`	
`save_classification_to_csv()`	
`run_experiment()`	evaluates the given model for a number of repetitions

Table 2.2: f

insert into appendix?

2.4.2 ModelMaker class

See example of this in next chapter.

Explain each layer?

2.5 Visualisation

Finally, after evaluating the results (results can be seen in the next section) from the neural network, we have not achieved any significant results. Arguably, the visualisations of class separability should have been handled first. However, the previous steps took the majority of the time.

Chapter 3

Evaluation

Here we will present the results and discuss the findings herein.

3.1 Models

Table 3.1: Table Title

Layer	**Output Shape**	**Number of params**
Synthetic data	2000	9

Dataset formats

http://alexlenail.me/NN-SVG/AlexNet.html

3.2 Model evaluation

In this section we evaluate our model with regard to a variety of metrics: loss, true
Insert explanation of each metric

Metrics / run #	Model1	ClaveVectors	MNIST-bin-digits	MNIST-all-digits
1		61.355	3212.497	2454.585
2	16.568	62.396	3346.047	2782.271
3	14.362	59.282	2908.446	2618.519
4	13.85	62.193	3043.189	2600.233
5	13.755	61.024	3309.422	2470.139
6	12.218	58.531	3192.178	2823.685
7	12.232	60.187	3287.289	2452.566
8	13.346	59.398	3026.486	2810.943
9	11.433	61.892	3127.583	2528.202
10	11.63	60.925	3018.402	2746.729

Table 3.2: Experiment result

Defect Type	2

Table 3.3: Entity distribution, different defect types and class distributions

just do one model at a time and show their metrics

3.3 Visualisation of class clustering

insert the pca plots

3.4 Discussion

Data amount

should have done visualisation first, if we have clear cluster separation, applying a neural network would be a bit exaggerated. And in that case, we could opt for a simple multi class support vector machine from the `sklearn` library. However, using `tensorflow` was the plan from the get-go as it is more industrially-applicable, so we disregarded simpler methods.

Chapter 4

Conclusion and future work

4.1 Conclusion

Results were quite mediocre, but has a lot of room for improvement. I am sure that given more time I would be able to explore and evaluate the results further.

4.2 Future work

- Might be interesting to also consider the XZ-axes.
- range defects
- tune the peak finding parameters
- track entity dependent/specific window offsets
- we must not set the findpeakoffset too high
- Questions, what if you want to use multiple features with 1 CNN
- I have participated in ETH Hatchery (ref), where our team build a prototype with a model train. We let the model train drive on the track with self-engineered defects.

4.3 TODO

- get percentage of each class in the validation set
- create an average of the model ie run model for more times
- add speed as a feature
- what does each filter do? what is kernel size?
- very fast speed, overlap between switch and ins, old vs new rail, ax1 arrow 2 arrow 3 arrow 4
- 3D plots?
- change the defect library to use pandas instead?
- visualise what the network is doing using Harry's code
- use speed as a feature also
- be consistent with function naming and variable names: function names with underscore and variable names with camelcase
- Which type of defects are we actually working with, we can see that it does good at the switches and ins joints, but no chance with the defects
- we should instead call it an entity library – make up your mind
- try only with defects and no ins and switches
- separate all the axle channels and train on them
- try to use low pass filter
- get better results
- we dont need non-defects for defect classification, we could just input something else and make sure that it is not a defect
- a specifc defect type vs ins joint vs switch

Bibliography

[1] Introduction to 1d convolutional neural networks in keras for time sequences. https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf. (Accessed on 02/11/2020).

[2] Tensorflow vs keras: Which one should you choose. https://analyticsindiamag.com/tensorflow-vs-keras-which-one-should-you-choose/. (Accessed on 02/10/2020).

Appendix A

Introduction

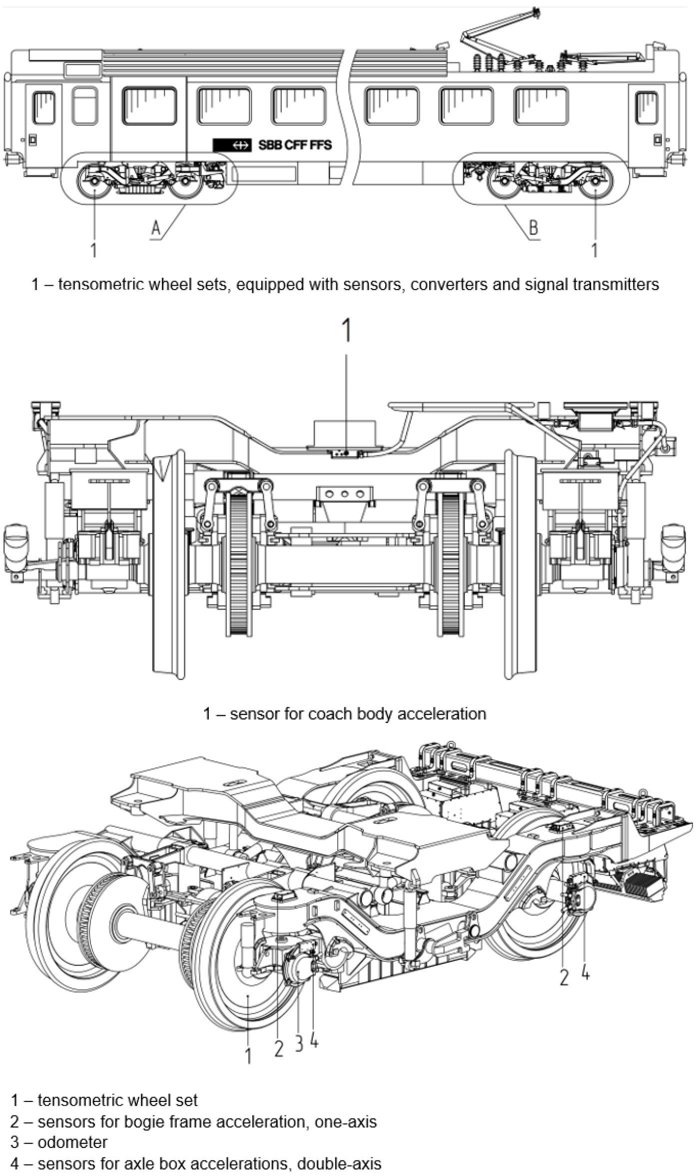

Figure A.1: These figures have been provided by Cyprien's folder of SBB documents. They show the accelerometer placements. For this project we have only considered axle number 4 in the third figure.

Appendix B

Implementation

Functions in NN class
prepare data

Appendix C

Results

Latent Dimension	2
Shape	8×8
Latent Points	64
Dist. Mixtures	Bernoulli
Regularisation α	0.001

RBF Dimension	2
RBF grid	4×4
RBF centres	16
Basis function	Gaussian
Widths $\boldsymbol{\sigma}$	1.0

Table C.1: The parameters for the latent model and the RBF neural network. Distribution mixtures and basis functions cannot be altered as our implementation is specific for the Bernoulli GTM version