DEPARTMENT OF CIVIL, ENVIRONMENTAL AND GEOMATIC ENGINEERING

Semester Project Intermediate Report

Data-driven identificiation and classification of rail surface defects

Aiyu Liu

Supervised by: Prof. Eleni Chatzi, Cyprien Hoelzl

Sunday 23rd 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 thereafter, 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 likewise been very supportive about my progress, always arranging intermediate update sessions and staying on top of the project. The intermediate sessions and gentle reminders have been a great driver in keeping me accountable and making further progress. Eleni is very approachable, kind, and great at providing critical feedback at these sessions. She is extremely active in her endeavours and one can tell 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 track-condition assessment system is expensive, time consuming and ineffective, primarily involving manual inspection. This condition assessment is evolving from a manual-labor based approach to a data-driven focused industrialized assessment. This paradigm shift from manual to data-driven approach is taking place at this current point in time, as more railway companies are making this transition and funding research within this area as seen in [14], [13], [1] and [2].

Switzerland's *Swiss Federal Railways* (SBB), is one of those companies being a part of this paradigm shift. SBB has specifically built one special *Diagnostic Vehicle* (Diagnosefahrzeug - DFZ) designed for defect detection and identification among other purposes. For this, accelerometers have been installed at the front and back of the DFZ to collect the signal responses from the wheel and the train track (see appendix A.3 for the accelerometer positions).

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 defect type will have a specific signature, which will allow for its identification and classification. This assumption is well-grounded in the litterature as seen in [5], [11] where the assumption is made for human activity recognition.

1.1 Objective

The objective of this project is to identify and classify rail surface defects. We aim to build an effective machine learning 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 toward predictive train-infrastucture maintenance, which will ultimately result in a more reliable network [12].

1.2 Defects

A defect can be seen as a deviation from a standard train track; and can be further subcategorized based on official defect documentation [9] provided by *International Union of Railways* (UIC). Upon inspection of defective tracks, SBB inspectors reports defects

with reference to the UIC-based, defect report document by SBB. An example of this can be seen in A.4.

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. A point defect is perceived as a sharp signal response, whereas a range-defect is perceived over a greater time period. Examples of point and range defects can be seen respectively in 1.1 and 1.2.

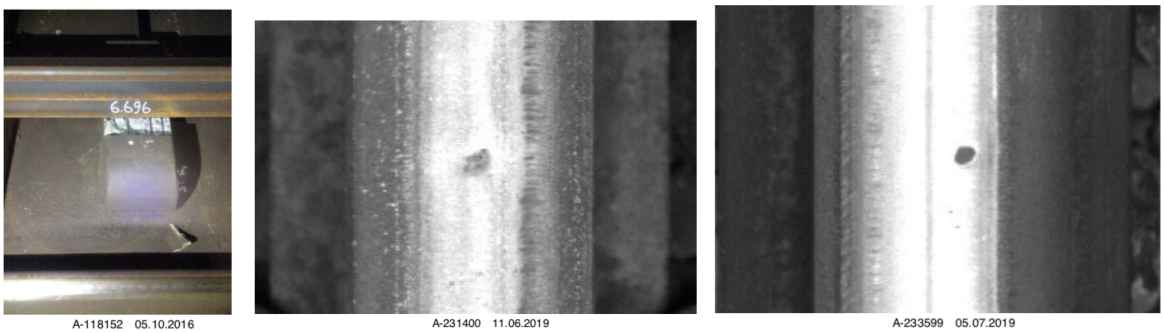


Figure 1.1: **Left**: Schwelle: *Beschadigte Betonschwellen* (A-118152); **middle:** Fahrbahn: *Verletzungen* (A-233599); **right:** Vignolschiene: *Verletzungen* (A-231400). The format is: track component: *defect type* (defect ID). These have all been reported as defects (with subcategories) with a length equals to zero, and thus have been classified as point defects using our terminology. (Source of pictures: SBB's defect report document)

Figure 1.2: **Top:** Gleis-149.8m: *Head-Checking: Periodische Rissbildung / Ausbrechungen an der Fahrkante* (A-184063); **middle:** Schienenzwischenlage-5.0m: *Weiterer Abweichungstyp* (A-146358); **bottom**: Bankett-1169.5m: *Ungenugendes Schotterprofil* (A-103213). The format is: track component-length: *defect type* (defect ID). These have all been reported as defects (with subcategories) with a length strictly greater than zero, and thus have been classified as range defects using our terminology. Most of these range defects have two pictures likely to give more detail. (Source of pictures: SBB's defect report document)

Furthermore, each defect is associated with a variety of attributes such as: associated component, defect component location, defect type etc. For this work, the focus has been on the defect component/category. That is, the track component with which the defect was identified on (see appendix A.1). This is, however, only a rough classification, as this attribute exists at a 'higher level', seeing that the lowest level of defect sub-division is the actual defect type (see appendix A.2 for a list of defect types).

The defect attribute for analysis is not a trivial choice, as the signal responses likely depend on the defect type, its associated component and its location on the component etc. In the future, it will be important to account for other attributes. Albeit, this work only considers the defect components. In the following, 'defect' will thus refer to these defect components.

Finally, only the point defects are subject to analysis, as this simplifies the problem statement; range-defects and its associated factors are thus disregarded. In the future, range defects naturally also need to be taken into account.

1.3 Switches and insulation joints

To distinguish between defects and non-defects, this work also considers *switches* and *insulation joints*' seen in the figures 1.3 and 1.4. Fact of the matter, is that, defect signals can vary a lot depending on the defect component. In addition, these sample amounts are far less compared to that of switches and insulation joints. With the incorporation of these non-defects, access to more samples is achieved, while these samples are suspected to have clearer signatures – easier to classify.

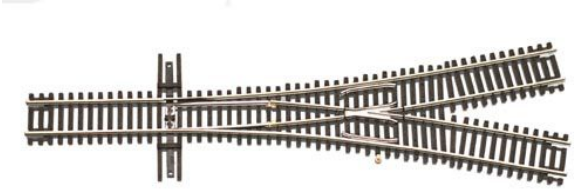


Figure 1.3: (Picture sources from right to left: [8], [6])

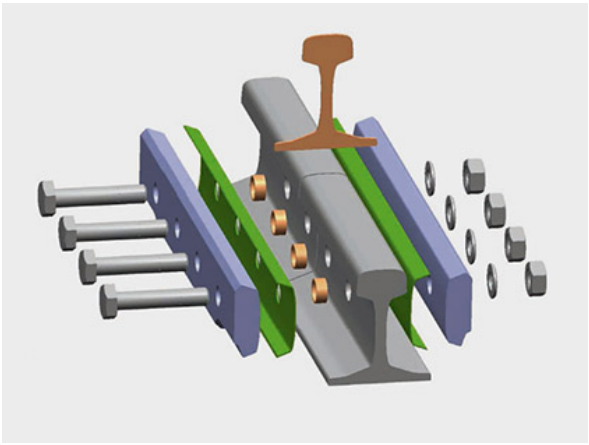


Figure 1.4: (Picture sources from right to left: [3], [4])

In order to not get confused about the different terms, the word 'entity' will be used as an umbrella term for the different track entities: switch, insulation joint and defect.

1.4 Data

The data has been collected and provided by SBB. Using their DFZ, 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 the data collection, the data is imported into a Python-VR object, which is composed of `pandas DataFrames`. `Python` objects with pandas `DataFrames` were chosen for their functional flexibility in terms of data storage and data accessibility.

The accelerometers capture the accelerations at the vertical (Z) and transversal (Y) axes (along with the timestamps at each recording), of which, only the Z axis was considered in this project, as it is assumed to be the most responsive to vertical perturbations. These accelerometers are installed on both leading (axle 1) and trailing axles (axle 4) of the measurement coach. See appendix A.3 for visualisations of the accelerometer placements on the DFZ.

In this work, the following DFZ measurement rides in table 1.1 are used for classification.

From	To	Date	Campaign ID
Bern	Olten	2019-05-27T08_55_55	819Z DFZ01
Olten	Bern	2019-05-27T10_03_59	077Z DFZ01
Bern	Olten	2019-05-27T13_05_53	330Z DFZ01
Olten	Bern	2019-05-27T14_10_51	425Z DFZ01

Table 1.1: Measurement rides for classification. ('From' and 'To' are cities in Switzerland)

Lastly, defect attributes and locations were retrieved from SBB's database and synchronized with the measurement data.

1.5 Code

The code is written purely in python. To create neural network architectures, `keras` (version `2.2.5`) along with `tensorflow` (version `2.1.0`) is used. `keras` is essentially a high-level neural networks library which runs on top of `tensorflow`. It has a consistent, simple API and provides clear and actionable feedback upon user error. Models are easily made by connecting configurable building blocks together, with few restrictions [7]. 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 latest model evaluation workflow can be found in `Colab`:
https://colab.research.google.com/drive/10lxoYyX1bMRkdQiIQsf_87_xhXDccOKJ

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

Chapter 2

Design and Implementation

For the process of defect classification the pipeline in figure 2.1 was designed. In the next sections, an overview will be given of how each step was implemented.

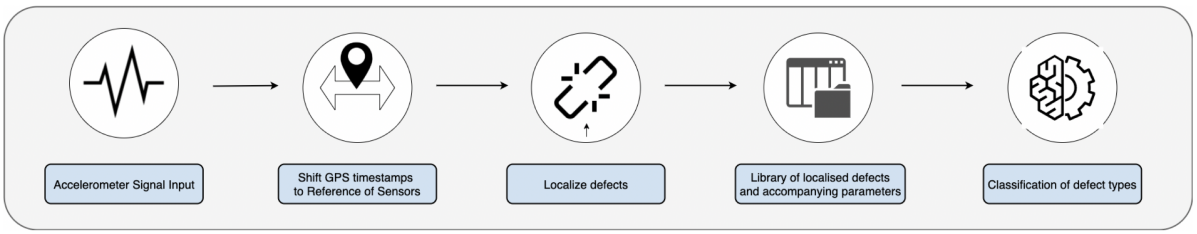


Figure 2.1: Primary pipeline

2.1 Localization of the DFZ

The vehicle is localized using the combination of multiple data streams: track transponders (Eurobalise), GPS signals, Odometers and Switch detection systems. The location is reported with respect to the GPS Antenna of the vehicle which is close to the pantograph, where the reported localization error has been determined to be quite small. It is reported along with a timestamp corresponding to the location of the vehicle at that time, at the location of the antenna.

2.2 Shift of GPS timestamps

Since the exact position (covered distance) at each accelerometer at either side of the GPS antenna is unknown, the goal in this step is to compute these from the reported location of the DFZ. The GPS sensor is sampled at a lower frequency compared to the accelerometers (every 25 cm vs 24 kHz respectively). From this, the corresponding GPS positions for each accelerometer sample necessarily need to be found. This is done by linear interpolation using the timestamps of the accelerometers and GPS.

Depending on the direction of the vehicle, the offset between the accelerometers and the GPS sensor positions on the vehicle body is then added/subtracted to achieve the accelerometer positions. An illustration of this process is shown in 2.2.

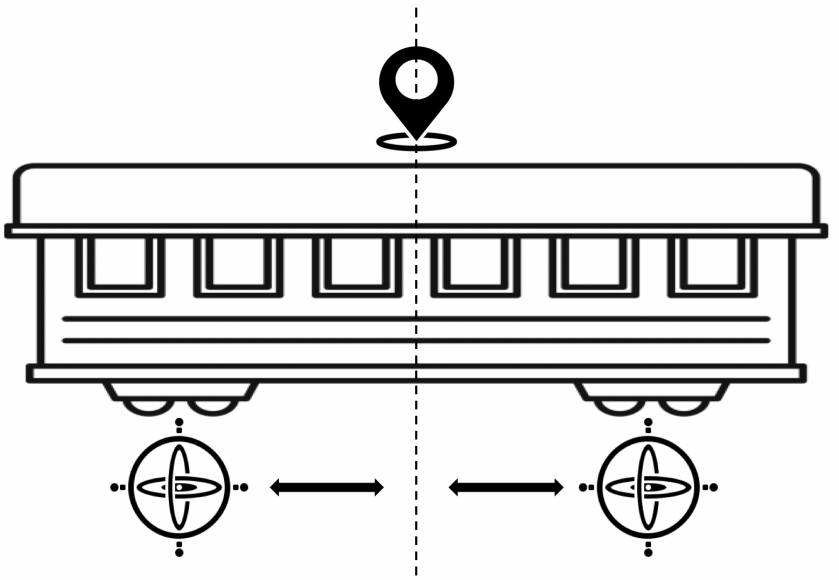


Figure 2.2: The location of each timestamp of the accelerometers have to be determined through linear interpolation

2.3 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 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 it is assumed that each track entity is identified by a well-formed peak, one first needs to find this peak within a reasonable offset from the defect location. Thereaftere, that peak is centered within a window with another reasonable window length 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, a search is performed to find the maximum, absolute acceleration recording that has occured 1 second after and 1 second before the defect timestamp. Once the peak has been found, it is centered in a 1 second window (0.5 sec on each side). An illustration of this proces can be seen in 2.3.

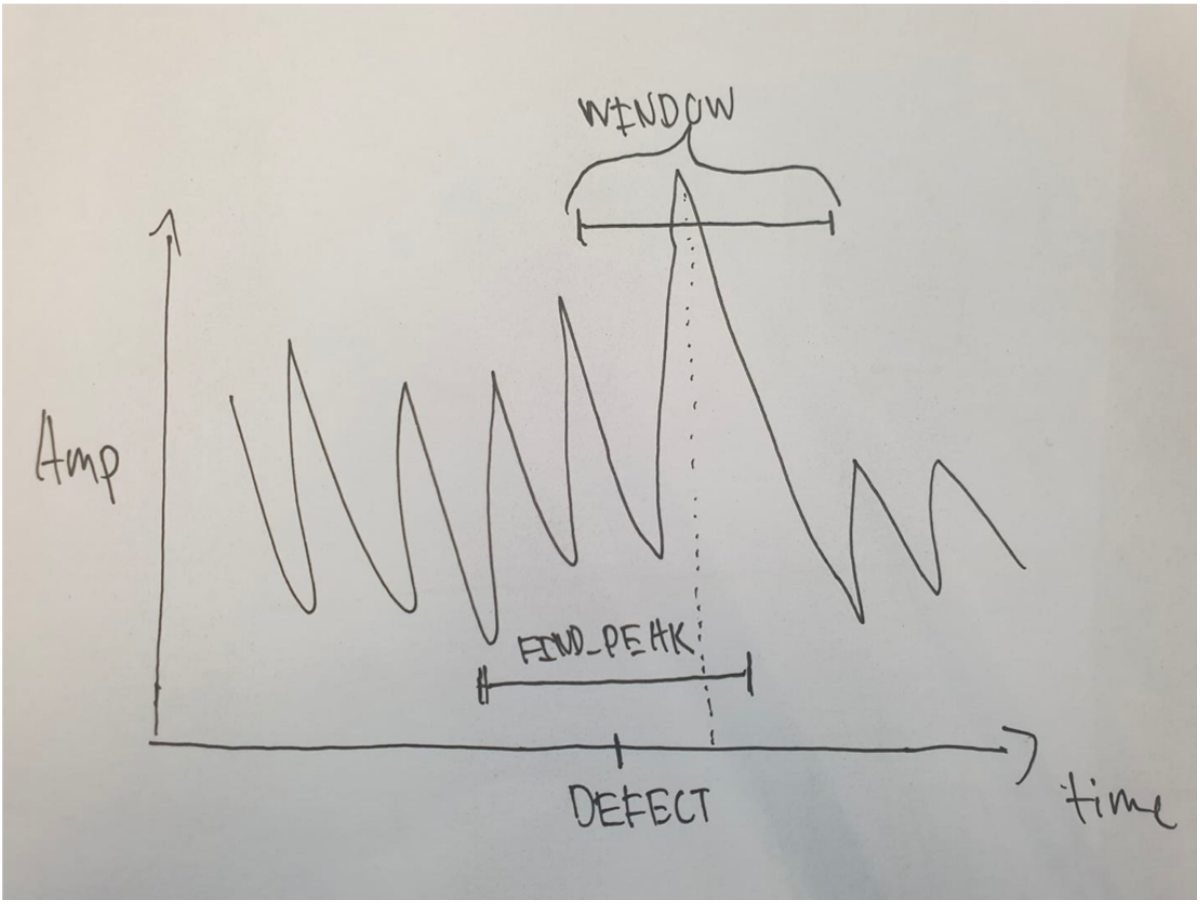


Figure 2.3: Setting peak/acceleration windows for an entity

2.4 Entity library

The peak windows arguably forms the central feature of the defect library. However, based on domain knowledge, other features like vehicle speed as well as defect severity likewise needs to be considered for our neural network:

- **Peak/acceleration windows:** accelerations of length 1 for each reported entity.
- **Vehicle speed (m/s):** vehicle speed at the closest timestamp of the reported entity.
- **Severity/urgency:** defects have an urgency label signifying how much time is left before acting; labels $1, 2, 3, 4$, where urgency decreases in ascending order. Insulation joints and switches have been self-engineered with label 5, as all entities need to have the same feature column for training.

Additional information about each entity has been retrieved as well, such as: timestamps, driving direction, corresponding entity IDs etc.

Given a specific measurement ride object, we either retrieve each feature directly from the corresponding `dataframe` or with the use of designated helper functions for those requring extra processing. Currently, have have a 2-level nested `for loop`, looping for each axle outerly, and looping for each entity entry innerly.

The implementation of this could have made more elegant by operating directly on the dataframe, which might also increase speed of the implementation as the `pandas` library has optimised their dataframe operations. However, speed and efficency was not a major concern in this project.

The resulting plots for each entity can be seen in figures 2.4, 2.5, 2.6; for the defects seen in the the introduction, refer to appendix A.5.

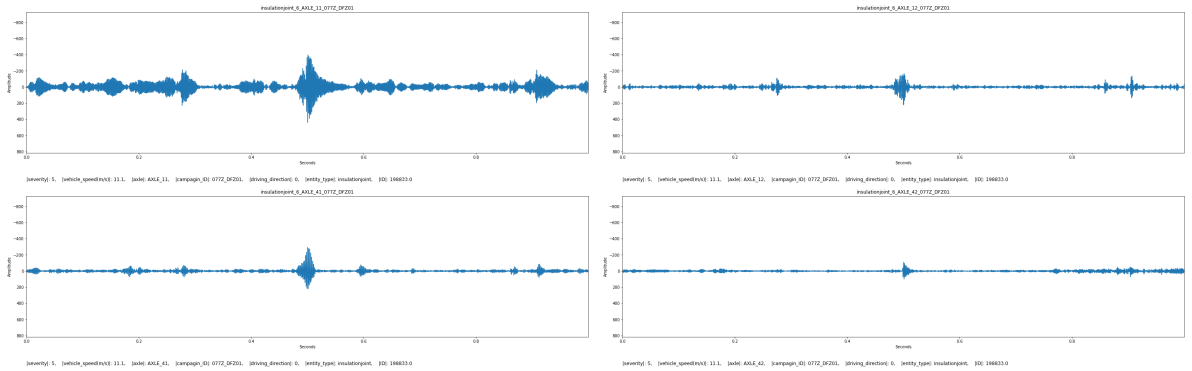


Figure 2.4: Insulation Joint

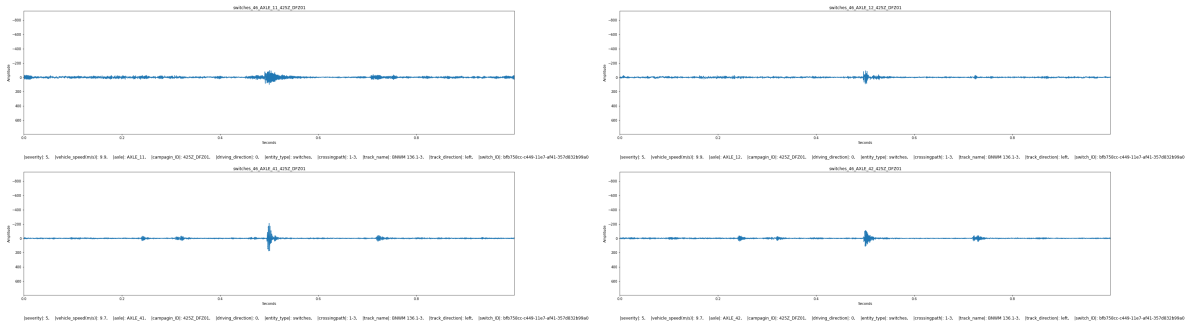


Figure 2.5: Switches

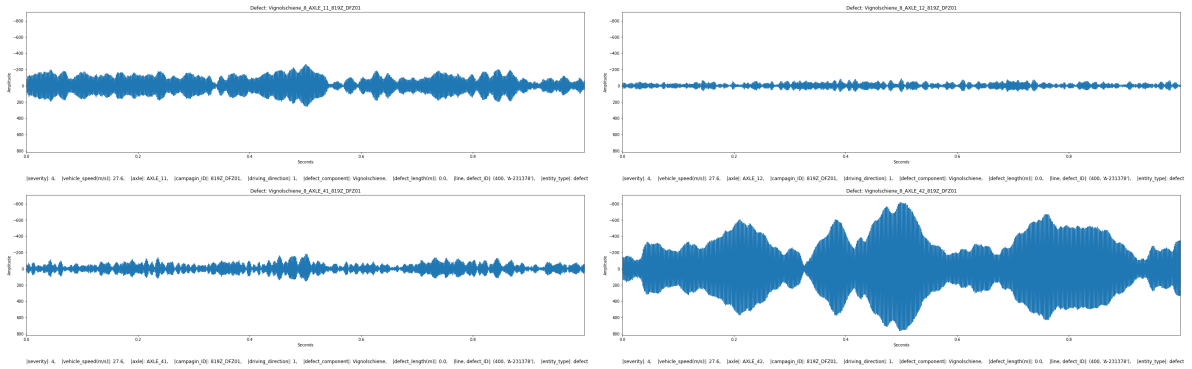


Figure 2.6: Defect: vignolschiene

2.5 Classification

A primary `NN` (neural network) class along with a `ModelMaker` class have been created. 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 one can keep track of how the models have been modified and improved.

2.5.1 NN class

To make a classification, relevant feature selection needs to be performed. The features are then simply fed 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()`	uses the trained model to classify on a test set

Other utility API functions	
`measure_performance()`	
`plot_metrics()`	
`plot_confusion_matrix()`	
`load_weights()`	
`load_model_()`	
`save_history()`	
`save_model()`	
`save_classification_to_csv()`	
`run_experiment()`	evaluates the given model for a # of repetitions

Table 2.1: To train a model, the first API functions needs to be called sequentially. Other utility functions are rather self-explanatory.

2.5.2 ModelMaker class

Below is the code for building the baseline model in **??** using `keras`.

```
def _makeModel2(self):
    model = Sequential()
    model.add(Reshape((self.seq_len, self.channels),
                      input_shape=(self.seq_len * self.channels,)))
    model.add(Conv1D(filters=20,
                     kernel_size=10,
                     input_shape=(self.seq_len, self.channels),
                     activation= 'relu'))
    model.add(MaxPooling1D(pool_size=24))
    model.add(Flatten())
    model.add(Dense(16, activation='relu'))
    model.add(Dropout(rate=0.3,seed=1))
    model.add(Dense(self.n_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
                  optimizer=Adam(learning_rate=0.0001),
                  metrics=self.metrics)
    self.model = model
```

Line 1-4
The model is firstly defined as a feed forward neural network. Secondly, the input shape has to be correctly formatted. As input, the model needs to receive the input shape of (number of samples, signal length $\times$ number of channels). Each sample is then converted into the shape: (signal length, number of channel). For instance, inputting 4 axle channels each of 24000 (24 kHZ accelerometers) acceleration readings is converted: $(24000 \times 4,) \rightarrow (24000, 4)$

Line 5-8
The input is then convolved with 20 fitlers/feature detectors of kernel size 10. `relu` is the Rectified Linear Activation, which is the standard activation function used in the majority of machine learning applications. Insert drawing

Line 9
For each filter, `MaxPooling` takes a number of accelerations defined by the `pool_size` as input and outputs the max acceleration, resulting in a smaller signal of length: original signal length divided by `pool_size` . In the general case, this would correspond to segmenting the filters into retrieving the max accelerations. Insert picture

Line 10-13
The resulting filters of the max pooling is then flattened into a 1D array, after which it is fed into 16 neurons with `relu` activation. A `Dropout` layer is then added to prevent overfitting; by randomly removing neuron connections. Finally, each neuron in the last `Dense` layer outputs a probability (using `softmax` activation) corresponding to its designated class. The final classification class of a training sample is then taken to be the maximum of these probabilities. Insert picture .

Line 14-17
The model is then compiled, where the overall training goal is to minimize the cost/loss function, `cross-entropy`, which is the common loss function used for multi-class classification.

2.6 Feature-set visualisations

After evaluating the results from the model (results can be seen in **??**), we have not achieved any significant results. Using PCA on the feature-set, the aim is to visualise the class clustering.

Chapter 3

Evaluation

In this chapter, the results are presented and the findings are discussed. Insert more describing text? More effort is need in general for this chapter

3.1 Baseline Model

Experimenting with models have been an ongoing process throughout this project; in the end a final baseline model has been determined to be the one seen in table 3.1. This very simple convolutional neural network is explained previously in 2.5.2. The hyper-parameters that are used for training can be seen in table 3.2.

Layer	Output Shape	Number of params
Conv1D	(, 23991, 20)	220
MaxPooling (24)	(, 999, 20)	-
Flatten	(, 19980)	-
Dense	(, 16)	319696
Dropout	(, 16)	-
Dense	(, num_classes)	187

Table 3.1: Baseline model

Hyper-parameters	Value
Epochs	20
Batch size	Full training set
Optimizer	Adam (0.0001)
Loss function	Cross-entropy

Table 3.2: Hyperparameters and its accompanying values

The epoch value was found to strike a good balance for the bias-variance tradeoff (basically prevent overfitting to the training data). Since memory is not of high importance, we input the full training set as the batch size. Meaning, all the training samples are seen at once for each training epoch. The chosen optimizer is the default choice in the Deep Learning community and provides adaptive learning rate with initial value

set to 0.0001. Finally, the cross-entropy loss function is commonly used for multi-class, categorical classification. Insert drawing of model

3.2 Dataset class distributions

Given the full feature-set, the model is trained on 85% of the data and validated on the remaining 15% both seen in 3.3. The model is trained, crucially, by weighing the imbalanced classes according to its percentage of the total training set. That is, minority classes are given higher importance.

Worth mentioning is the fact that this is the same validation set that was used for model selection during training, which is not best practice. It would have been ideal to use a separate un-seen test set for this evaluation, although with the limited training data, it has been deemed adequate. Otherwise not enough information would be sufficient for both training and evaluation.

		Training set		**Validation set**	
Class	**Defect Comp.**	**Count**	**Percent**	**Count**	**Percent**
0	Fahrbahn	93	4.02%	11	2.69%
1	Gleis	57	2.46%	15	3.67%
2	Herzstck	46	1.99%	10	2.44%
3	Schiene	180	7.78%	32	7.82%
4	Schienenzwischenlage	50	2.16%	6	1.47%
5	Schotter	148	6.39%	20	4.89%
6	Schwelle	51	2.20%	13	3.18%
7	Vignolschiene	161	6.95%	35	8.56%
8	Zungenschiene	34	1.47%	10	2.44%
9	Ins. joint	988	42.68%	160	39.12%
10	Switch	507	21.90%	97	23.72%

Table 3.3: Class distributions among training and validation set. The full dataset consists of **2724** samples. The training set totals **2315** samples, and the validation set totals **409** samples.

Worth mentioning is the fact that each of these were trained on different training and validation sets. This was a mistake that was only realised later, where the ideal would be to use the same sets for all the model and feature set combinations. For now, the differences are rather minor and 3.3 shows a sample distribution for evaluation of the final featureset (row) in the metric results table, 3.5.

3.3 Metrics and feature-set

As the training process is stochastic in nature, the model is trained 10 times (experiments), after which an average along with the associated standard deviation are computed. In addition, the following metrics are also used to define the model performance: loss (**L**), accuracy (**ACC**), F1-score (**F1**), area under the curve (**AUC**) and the associated confusion matrix (**CM**). These results are summarised in table 3.5.

explain metrics?

The baseline model has been utilised to determine the best feature-set with the assumption that this feature set would generalize to future NN models. The feature-set combinations can be found in table 3.4.

Feature-sets	**# feat.**	**Parameters**
All-AX	4	1s acc. window
AX	1	1s acc. window
AX-LP	1	1s acc. window filtered at 0.63 threshold
AX,SP	2	1s acc. window, vehicle speed
AX-LP, SP	2	1s acc. window filtered at 0.63 threshold, vehicle speed

Table 3.4: All feature-sets used for defect classification. **Reminder**: 1 second window corresponds to 24000 samples since each accelerometer is sampled at 24kHz

Model / Metric	L	ACC)	F1	AUC
BL (ALL-AX)	2.089 (±0.118)	0.411 (±0.144)	0.303 (±0.107)	0.508 (±0.048)
BL (AX)	2.223 (±0.050)	0.378 (±0.041)	0.271 (±0.031)	0.463 (±0.060)
BL (AX-LP)	2.223 (±0.069)	0.384 (±0.034)	0.311 (±0.030)	0.540 (±0.032)
BL (AX, SP)	1.850 (±0.112)	0.431 (±0.025)	0.337 (±0.027)	0.583 (±0.054)
BL (AX-LP, SP)	1.845 (±0.118)	0.443 (±0.021)	0.357 (±0.026)	0.593 (±0.052)

Table 3.5: Average metrics times and their standard deviations in parenthesis over 10 experiments. BL is short for baseline model. The parenthesis denotes the used feature-set. (All-AX), (AX), (AX, LP) and (AX,SP) denote that a single training sample respectively consists of: all 4 axle channels, 1 individual axle channel, 1 individual axle channel low-pass filtered and 1 individual axle channel along with its corresponding speed.

3.4 Plots

In the following, the training and validation plots for each metric (loss, acc) during each epoch are presented. The format is: **Top:** confusion matrix for the predicted vs true classes/labels. **Left:** average training/validation loss accompanied by error bars. **Right**: average training/validation accuracy accompanied by error bars.

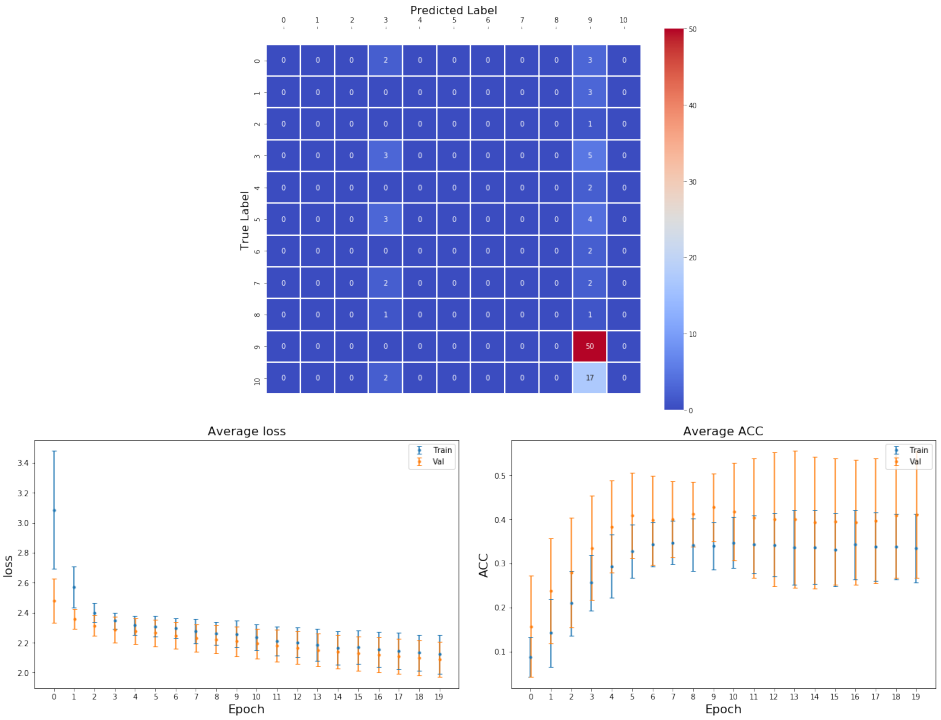


Figure 3.1: **BL (ALL-AX)**

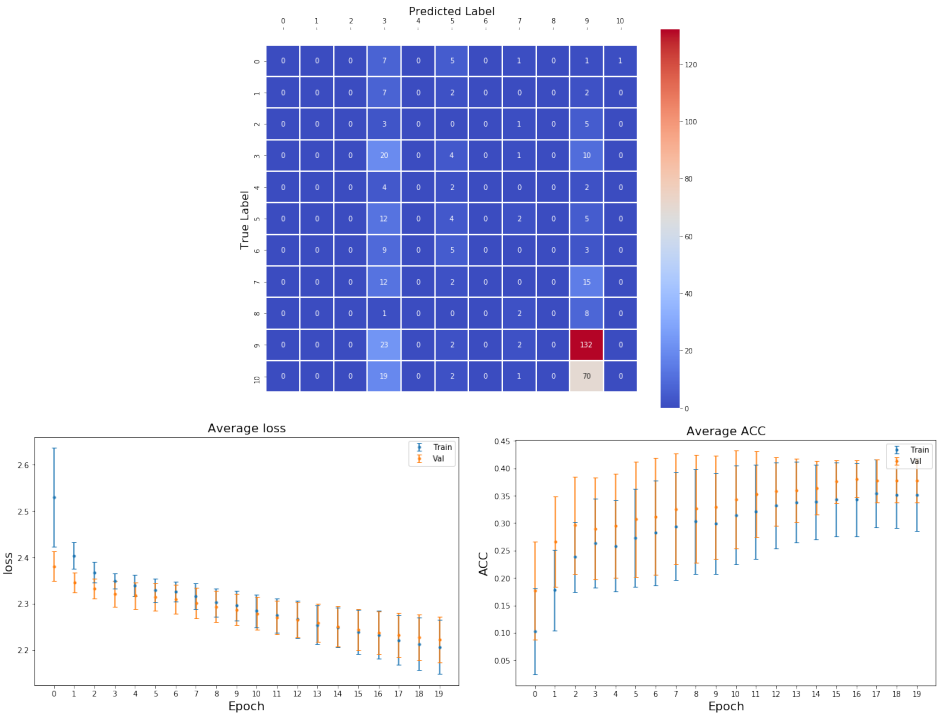


Figure 3.2: **BL (AX)**

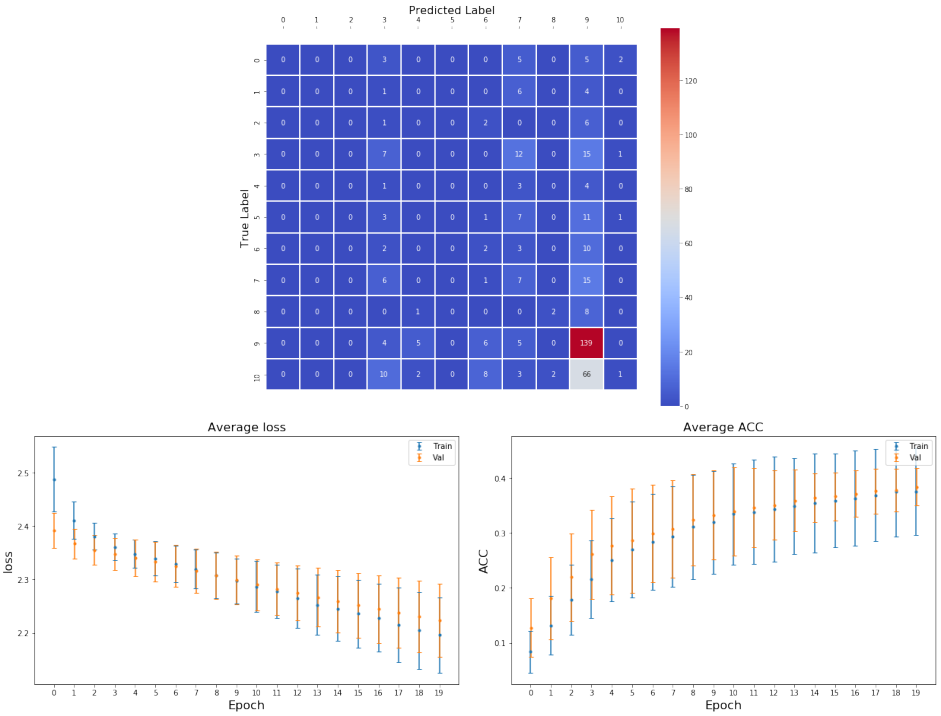


Figure 3.3: **BL (AX-LP)**

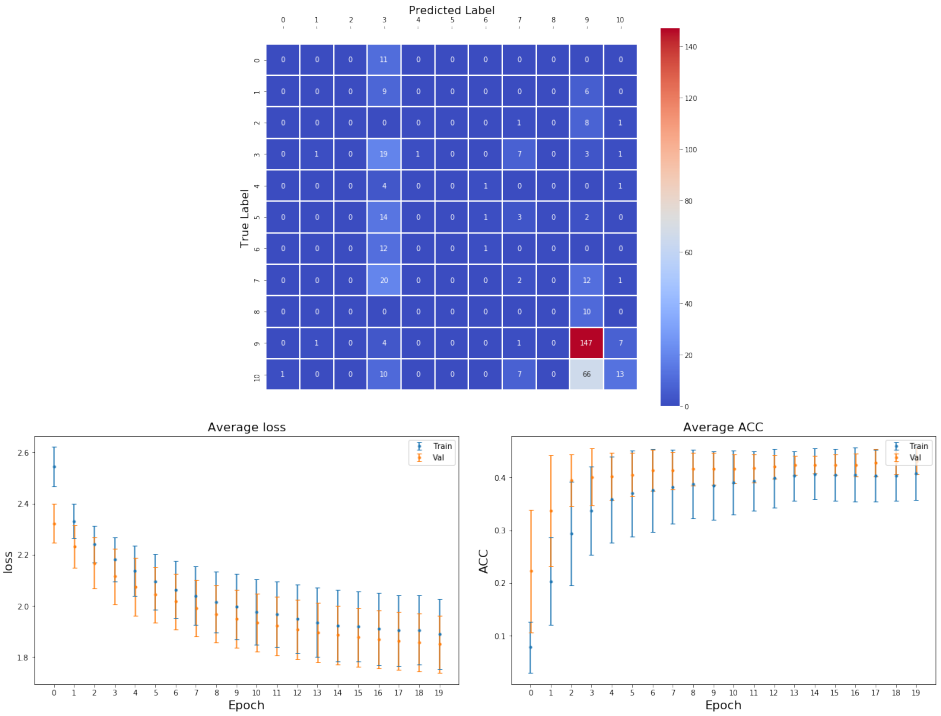


Figure 3.4: **BL (AX, SP)**

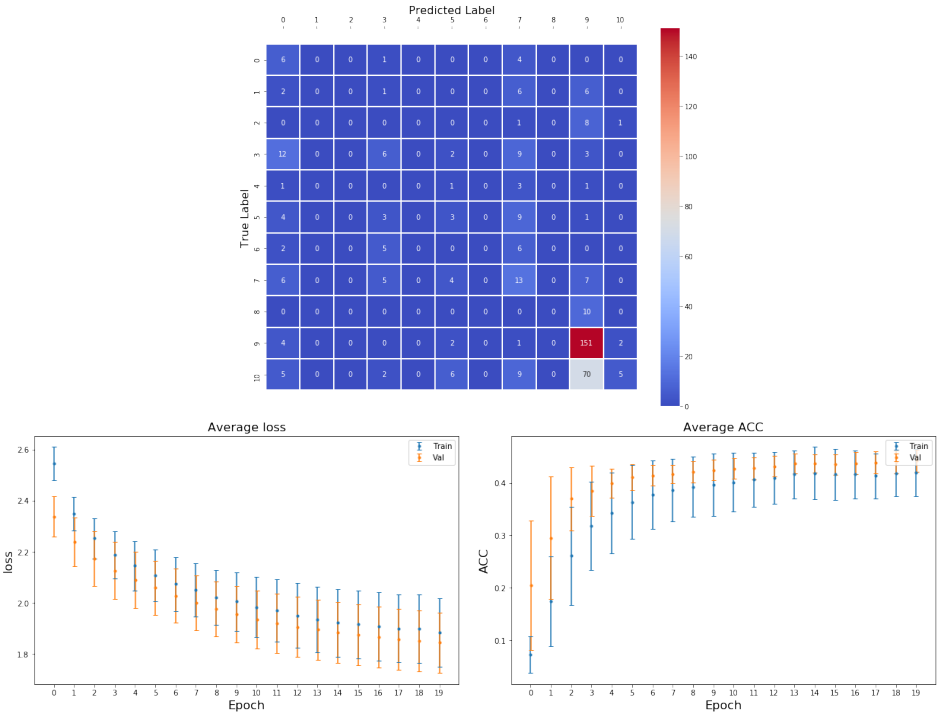


Figure 3.5: **BL (AX-LP, SP)**

The remaining metric plots could not be retrieved as `Tensorflow` has to be configured in quite a complex manner to make it account for other metrics. This could however be incorporated with more time. Albeit,he primary utility of metrics during training is for logging and for utilising suitable callbacks.

Add individual training plots for each experiment to the appendix. These are only confusion matrices taken from one experiment, although it would be better to compute the average confusion matrix from all the experiments. Perhaps, it would be better to just keep the last result plot and move the worse results to the appendix

3.5 Visualisation of class clustering

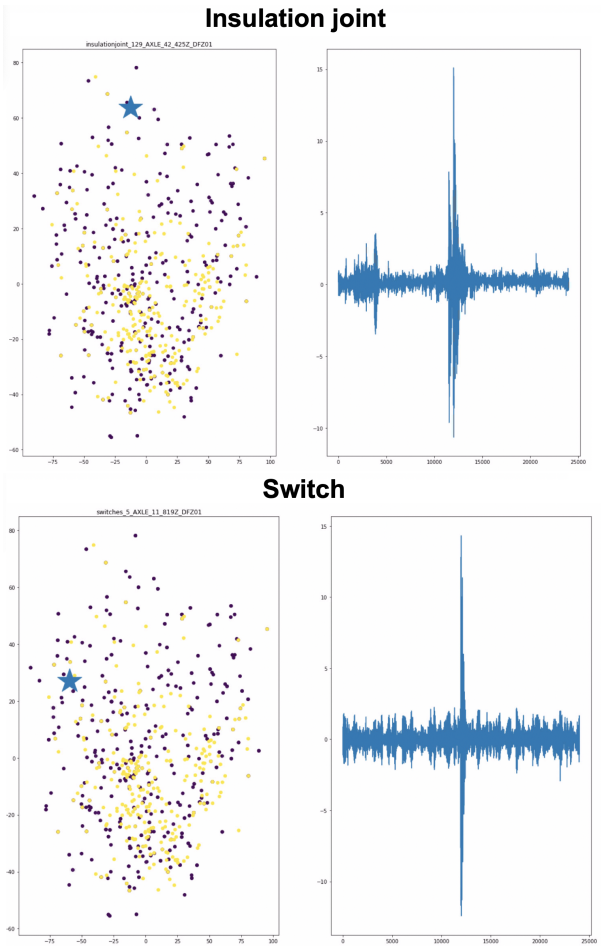


Figure 3.6: PCA class separability

Insert more plots like this, has not been done fully

3.6 Discussion

From the resulting metrics and plots, it is evident that no truly significant results were achieved. However it can be said with good confidence that using a feature-set consisting of lowpass-filtered acceleration windows in addition to vehicle speed results in better performance. These findings suggests that further pre-processing techniques and more sophisticated feature selection will achieve better classification results.

With regard to each feature-set, it can be seen from **all 4 channels**, that it has the worst confidences as the standard deviations are rather high compared to the other feature-sets. The accompanying plots also illustrate this matter. For this particular case, this can likely be explained by the four times smaller sample amount compared to the other feature-sets, as all 4 axle channels are combined into one sample. Minority-class removal has been done at the same threshold as the others (class samples above 10). Generally, it can be said that the model's poor results are very likely attributed to the small-size and class imbalance of the dataset.

For **single axle**, actual confidence in results can be seen by the decrease in error bars in the plots. Although, overfitting to the training data happens after the 14th epoch, as the training loss dips below the validation loss. To prevent this from happening, one can employ `callbacks` during training such that it stops when meeting certain user-defined requirements. For this evaluation, this has not been done as variables are kept the same throughout feature-set evaluations.

Low-pass filtering the acceleration windows generally follows the same trend as the previous feature-set. For this case it can be seen that this extra pre-processing step improves performance of metrics. When accounting for the **single axle channel with vehicle speed**, it can be seen that more classes can be predicted; resulting in more of a general classifier as well as achieveing higher overall metric scores. For this particular feature-set (and the next) it can be seen that there is still a bit more room for improvement in the training loss as it is still in the process of converging.

The last examined feature-set, **single, low-pass filtered axle channel and corresponding vehicle speed**, has the best performance metrics out of all and shows better variety in its classification predictions.

Furthermore, the confusion matrices show that there is no prediction for some of the classes. This is likely due to the segmentation of the signal when passing the input through the model as explained in 2.5.2. It is suspected that a prediction for each class could be reached by perhaps `average pooling` instead of `max pooling`.

From all confusion matrices, it can be observed that preference is given to a select-few classes. Namely, number 9 (insulation joint) as expected, number 3, 7, 5. Surprisingly enough, for the switches (number 10) few predictions exist and they primarily gets classified as insulation joints. In fact, from looking at the acceleration windows of switches, they do not seem distinct from any other signal. This is however in contrast to the insulation joints, where clearly defined maximal peaks are defined at the center.

Finally, other important factors to take into account when evaluating this model, are the age of the tracks and severity of the defects. The utilised data in this work are produced from the DFZ which drove on brand new tracks built in 2006-2018. As a result, the defects severities are all minor; 648 and 304 out of 1264 defects, are labeled 4 and 3 respectively (lowest severity labels).

Comment more on the metrics individually in the future.

Chapter 4

Future work and conclusion

This future work section sounds more like a discussion and perhaps needs to be moved to the previous chapter

4.1 Future work

There is still a lot of work that needs to be done in this project. These are summarised in the next sub-sections, in no particular order.

4.1.1 Channels

It might be interesting to consider sensor fusion of the ZY axes and the axle channels (8 total channels). The transverse axis, Y, would especially be valuable for switches as this channel can pick up sideways hits in the case of switches. These switch signals are clearly different from that of the other entities and may thus be more easily separable.

In this project, only the accelerometer accelerations at the axle-boxes were considered. However, the other accelerometer sensors higher up in the vehicle: bogie(Z, Y) and body (Y) (see appendix A.3), will also be worth analyzing.

Finally, in order to ensure that each axle channel response is uniform, it is also important to account for the varying calibrations of the accelerometers.

4.1.2 Feature-set

Another interesting aspect would be to account for the severity as a feature/input to the model; or even as the classification output.

For periodic, heartbeat electrocardiogram signals, heartbeat-specific features have been extracted as seen in [10]. Perhaps, a similar approach could have been employed for this project. Although, the entity signals are not periodic, and may thus not have the same applicability.

4.1.3 Peak windows

If the peak windows are not retrieved, sized and positioned optimally, much information about the entities are lost. Hence, the acceleration windows arguably forms the most crucial aspect of the feature-set. Evidently, the goal of these windows is to strike a fine balance between capturing the most important information vs. capturing disruptive

information. E.g. another peak at the corners of the window or merged peaks. Thus, the ideal would be to search for the peaks within an appropriate offset from the defect location and then center around the 'best' peak with the smallest possible window. For this project, a window length of 1 second was set for all entities. Although, a smaller window length could likely have been employed.

Furthermore, each entity vary a lot in terms of the signal responses. E.g. switch signals usually have two high amplitude responses. For this, it might be suitable to define entity specifc window lengths. This would however necessitate a slightly more sophisticated neural network, as the inputted signal lengths would not be equal, thus requiring multiple inputs.

4.1.4 Identification of entities

The identification of the infrastructure components has relied on a direct information retrieval from SBB's databases. Albeit, they are not always reliable. To improve the identification method, a fusion of different track entity information would provide a better measure of identification. I.e. the order and distance between entities, as well as the uncertainty in the reported location of the DFZ. Considering that insulation joints and switches might overlap in the signal responses, this would be an improvement to the current identification process. A visualisation of this could be seen in listing 4.1. S, I, C and '...' (dots) denote switch, insulation joint, distance between track component respectively.

```
Expected: "...I,....S,I,....C,....I,.............I,...S,..C,.."
Measured: "...I,..S,...I,...C,...I,......I,...,...S,..C,.."
```

Listing 4.1: Expected layout of entities vs actual layout.

Furthermore, at times the defect can only be detected from one side of the track (AX11 and AX12) as one defect crossing sample with two signatures seen from different location. This would help in cases where the defect is only on one side of the track.

4.1.5 Class separability

More emphasis should be placed on feature-set class separability to determine which set of features provide the best classification confidence; better class clustering leads to better classification.

The visualisation of class separation should arguably have been done first. Considering the fact that if a clear cluster separation exists, then applying a neural network would be a bit exaggerated. In that case, a simple multi-class support-vector machine from the `sklearn` library would suffice for this task.

4.1.6 Visualisation of neural network

The idea of this subsection is to use class activation maps to visualise how the neural network makes its decisions.

4.1.7 Real-world simulated defects

A crucial limitation to this project is the defects sample amounts. Another approach to gather these samples is to self-engineer them by implementing a small-scale model of the real-life train infrastructure. This was exactly the goal for our team in the ETH Hatchery [1], where we built such a model to collect signal responses from self-engineered defects. Even though it would be an extremely simplified model of the real-world, it would, however, provide proof of conccept for the idea of data-driven defect identification and classification.

[1] https://sph.ethz.ch/eth-week-hatchery/

4.2 Conclusion

The aim of this project has been to identify and classify rail surface defects. A machine learning pipeline that takes information about defects as input and outputs the corresponding classification confidence, has been built. The empirical space of focus has been on track defect components of zero-length, where the identification of these has relied solely the information provided in SBB's database.

The resulting classification has not shown any significant results. They have shown that using the retrieved acceleration windows are usable for this classification task, although there is much room for improvement by potentially tweaking the model as well as working with other features. It has been found that inputting the low-pass filtered acceleration windows along with corresponding vehicle speed produces notable improvement in the classification confidence

Despite the mediocre results, this work lays a good foundation to move onwards with further research. Not to mention the fact the library code is built in a modular, extensible manner for further improvements.

4.3 Todo

- insert pictures
- make the training sets uniform hroughout all the model/feature evaluations
- see if outlier removal helps?
- Do class separation visualisations
- test the auc score
- Change name of SBBproject to feature generator
- `defects['severity'].value_counts()`
- Overall model explanation - poolsize of 24, 20 filters

Bibliography

[1] Deutsche bahn establishes digital railway company — railway-news. https://railway-news.com/db-establishes-digitale-schiene-deutschland/. (Accessed on 02/16/2020).

[2] How digitalization is evolving intelligent rail infrastructure — rail stories — siemens. https://www.mobility.siemens.com/global/en/portfolio/rail/stories/how-digitalization-is-revolutionizing-rail-traffic.html. (Accessed on 02/16/2020).

[3] Insulated rail joint. http://www.railroad-fasteners.com/news/Insulated-Rail-Joint.html. (Accessed on 02/13/2020).

[4] Insulated rail joints. http://www.railroadpart.com/rail-joints/insulated-rail-joints.html. (Accessed on 02/13/2020).

[5] 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).

[6] Railroad switches at rs 300000 /piece — - pankaj steel industries, ahmedabad, ahmedabad — id: 3879200755. https://www.indiamart.com/proddetail/railroad-switches-3879200755.html. (Accessed on 02/13/2020).

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

[8] Types and uses of model train switches and turnouts. https://www.thesprucecrafts.com/model-train-switches-2382606. (Accessed on 02/13/2020).

[9] Uic code 712: Rail defects. **International Union of Railways(UIC**, January 2002.

[10] Miquel Alfaras, Miguel C. Soriano, and Silvia Ortn. A fast machine learning model for ecg-based heartbeat classification and arrhythmia detection. *Frontiers in Physics*, 7:103, 2019.

[11] J. C. Davila, A. M. Cretu, and M. Zaremba. Wearable Sensor Data Classification for Human Activity Recognition Based on an Iterative Learning Framework. *Sensors (Basel)*, 17(6), Jun 2017.

[12] Vincent Meyer Zu Wickern. Challenges and reliability of predictive maintenance, 03 2019.

[13] Sebastian Rapp, Ullrich Martin, Marius Strhle, and Moritz Scheffbuch. Track-vehicle scale model for evaluating local track defects detection methods. *Transportation Geotechnics*, 19, 06 2019.

[14] Siddhartha Sharma, Yu Cui, Qing He, Reza Mohammadi, and Zhiguo Li. Data-driven optimization of railway maintenance for track geometry. 2018.

Appendix A

Introduction

A.1 List of defect components

Bankett
Befestigung
Dienstweg
EKW / DKW
Einfache und symmetrische Weiche
Entwsserungsgraben
Fahrbahn
Gleis
Gleisbettung
Halbe Zungenvorrichtung
Herzstck
Herzstckspitze
Kleber
Rippenplatte
Schiene
Schienenklemme
Schienenzwischenlage
Schotter
Schweissverbindung
Schwelle
Stockschiene
Vignolschiene
Weiche
Zungenschiene

Insulation Joint
Switches

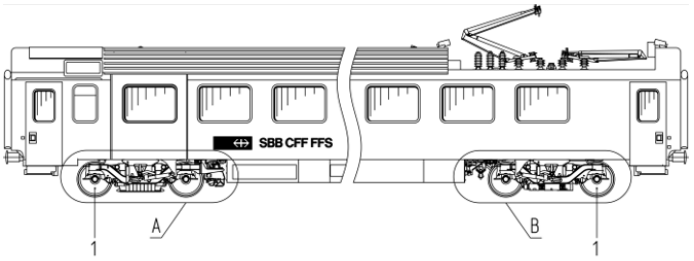
Table A.1: The entities that are subject to classification

A.2 List of defect types

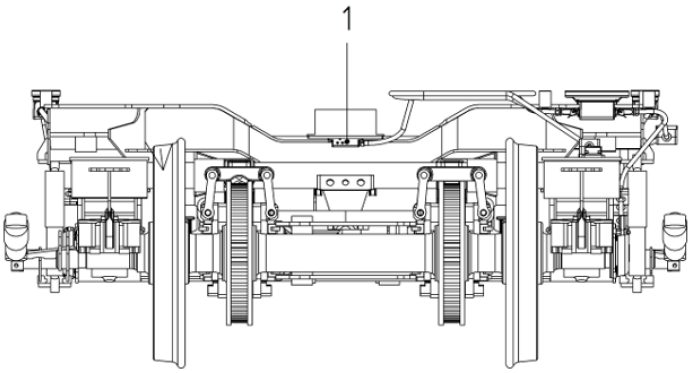
AT-Schweissung mit Squat
Ausbruch in der Zunge
Bankett
Beschdigte Betonschwellen
Entwsserung oder Wasserkanal verstopft
Gleisgeometrie
Grundparameter
Gleisgeometrie, Spurweite
Gleisgeometrie, Standardabweichungen
Head-Checking: Periodische Rissbildung / Ausbrechungen an der Fahrkante
Isolierstoss mit berwalzung
Kilometer-, Hektometer- oder Metertafeln
Lose/fehlende Schienenbefestigung
Lose/fehlende Schwellenbefestigung
Mitten in der Schienenlnge: Oberflchenfehler
Mitten in der Schienenlnge: Schleuderstellen oder Schnellbremsspuren
Mitten in der Schienenlnge: Squat / Rissbildung und rtliche Einsenkung der Laufflche
Oberflchlicher Fehler (Flgelschiene oder Herz)
Schienenende: Sprdbruch
Schienenende: Verquetschung
Schienenende: rtliche Einsenkung der Laufflche
Schotter auf Schwellen
Schotterfliessen, weisser Schotter
Ungengendes Schotterprofil
Verletzungen
Weiterer Abweichungstyp
Zungenspitze angefahren

Table A.2: Defect types

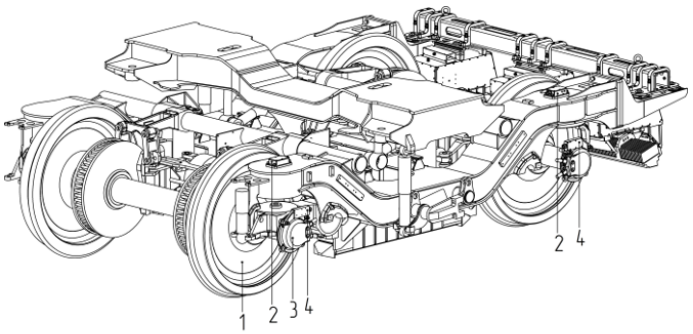
A.3 Vehicle and accelerometer placements



1 – tensometric wheel sets, equipped with sensors, converters and signal transmitters



1 – sensor for coach body acceleration



1 – tensometric wheel set
2 – sensors for bogie frame acceleration, one-axis
3 – odometer
4 – sensors for axle box accelerations, double-axis

Figure A.1: Illustrations of the accelerometer placements on the DFZ. For this project we have only considered axle number 4 in the third figure. (Source of picture: SBB documents)

A.4 SBB defect report example

A-103213 (Offen ohne Massnahme) Exportdatum: 26.09.2019 10:50

Objekt-Infos

Inspektionsobjekt:	AESP 284.2 - WANZ 112.2	Av-Region:	RME-FB-Lyss
Anlagenstrukturelement	Bankett	Massnahmen-ID:	

Position der Abweichung

Linie:	400 Löchligut - Wanzwil - Rothrist West		
km von:	17.2	km bis:	18.4
Positionierung:	Keine Positionierung		
Position:			

FS-spezifische Positionsfelder

Gleis:	
Weiche:	
Mast:	
FL-Sektor:	

Informationen zur Abweichung

Abweichungstyp (DE):	Ungenügendes Schotterprofil	Entdeckungsart:	Av - Anlagenverantwortlicher
		Erste Feststellung am:	30.04.16 07:09 durch: Christian Schärli
Schweregrad:	6	Letzte Feststellung am:	14.10.18 11:53 durch: Christian Schärli
Dringlichkeit:	4	Massnahmenidee:	
Bemerkung:	Fast die ganze Länge am tiefen Strang		

FB-spezifische Informationen zur Abweichung

SCDE Relevant:	Nein
Ultraschallfelder	
Notverlaschen:	Nein
In Garantie:	Nein
Qualität:	
Schienenprofil:	SBB VI bzw. EN 60 E1/E2
Frist:	
Radius:	3197.9
Prüfungsart:	
Hersteller:	
Jahr:	

Ausführung durch

U-Nummer:	
Datum:	
Unterschrift:	

Figure A.2

A-103213 25.03.2017

Figure A.3: A typical report for an arbitrary defect usually contains one description page followed by its picture(s)

A.5 Defects in Introduction

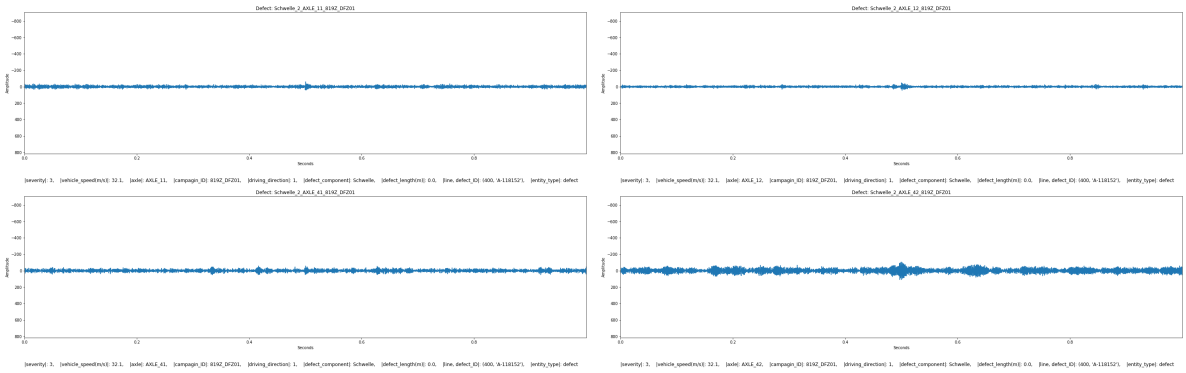


Figure A.4: Schwelle: *Beschadigte Betonschwellen* (A-118152)

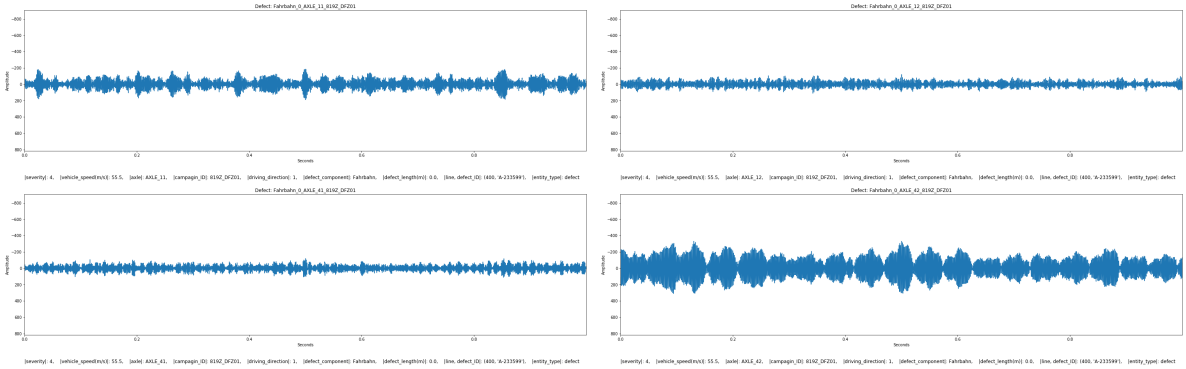


Figure A.5: Fahrbahn: *Verletzungen* (A-233599)

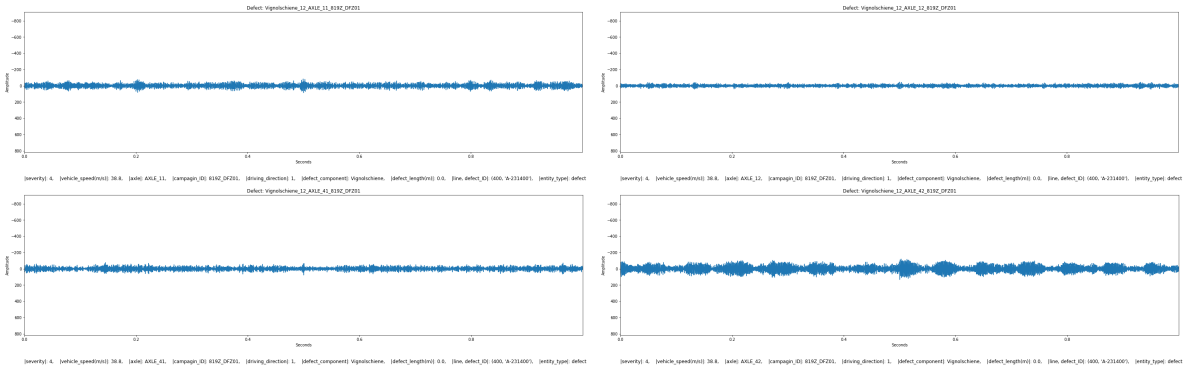


Figure A.6: Vignolschiene: *Verletzungen* (A-231400)

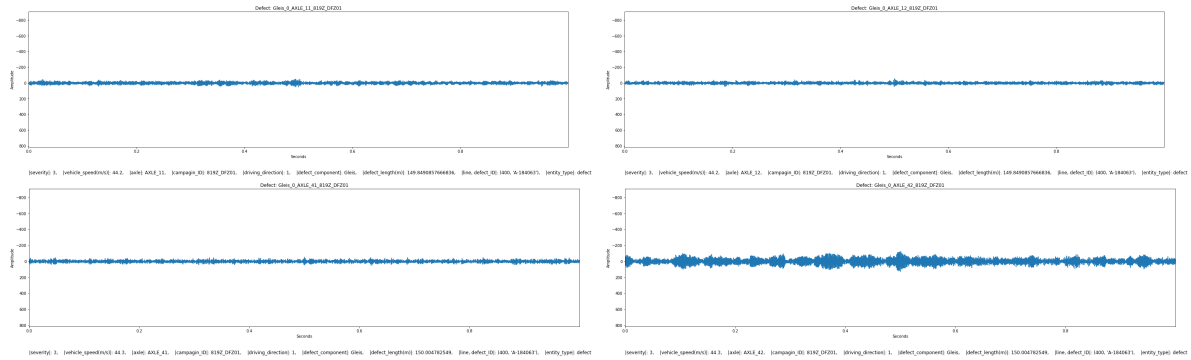


Figure A.7: Gleis-149.8m: *Head-Checking: Periodische Rissbildung / Ausbrechungen an der Fahrkante* (A-184063)

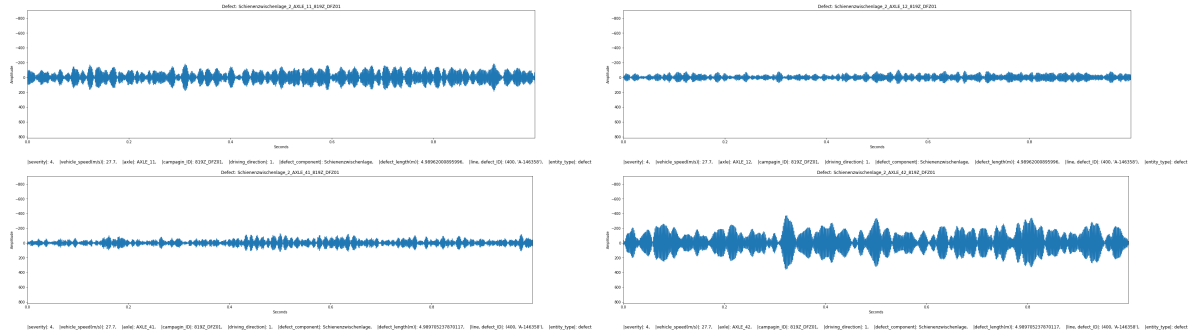


Figure A.8: Schienenzwischenlage-5.0m: *Weiterer Abweichungstyp* (A-146358)

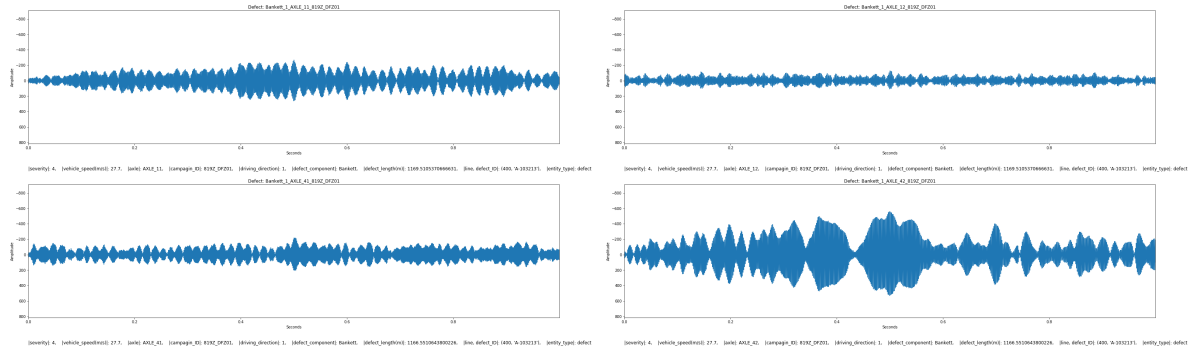
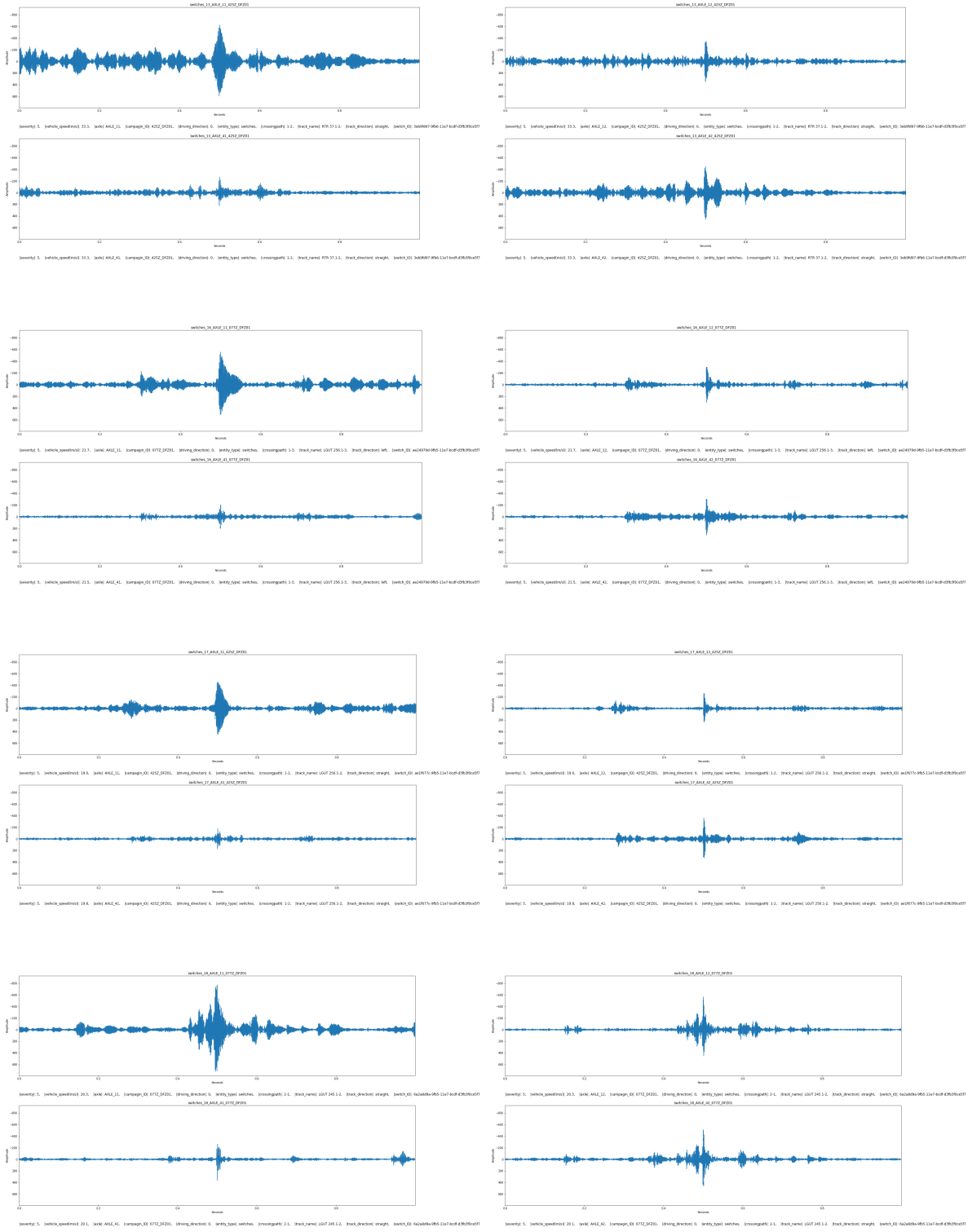


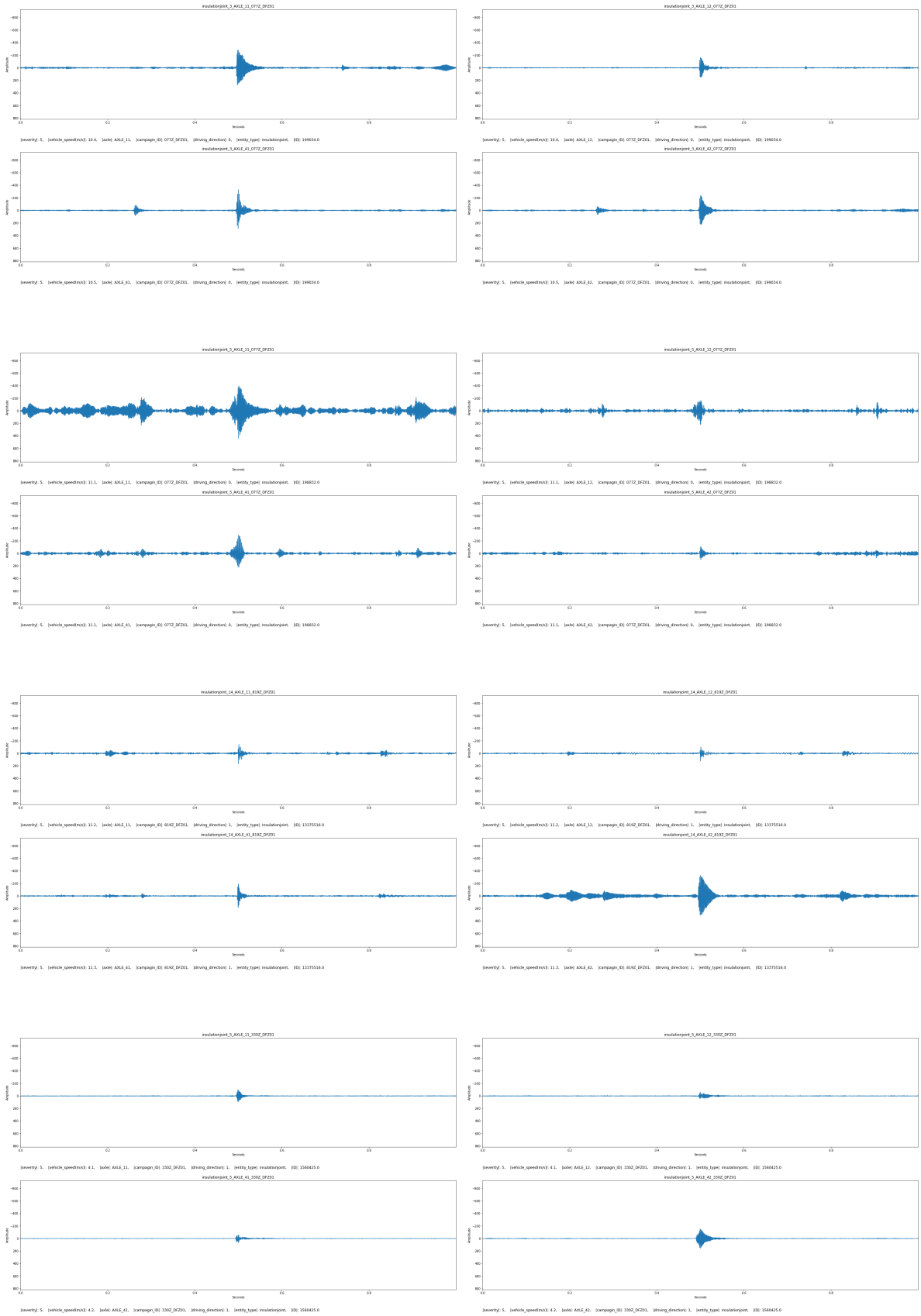
Figure A.9: Bankett-1169.5m: *Ungenugendes Schotterprofil* (A-103213)

A.6 Implementation

A.6.1 Switches



A.6.2 Insulation Joints



A.6.3 Defects

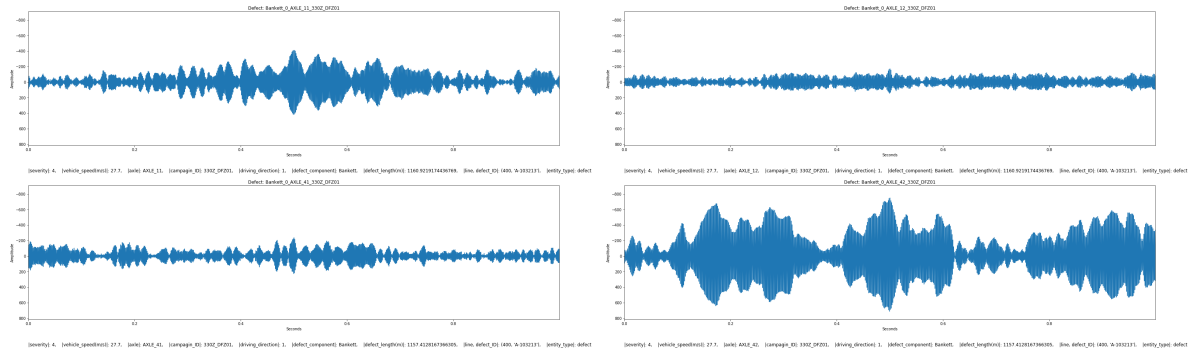


Figure A.10: Bankett

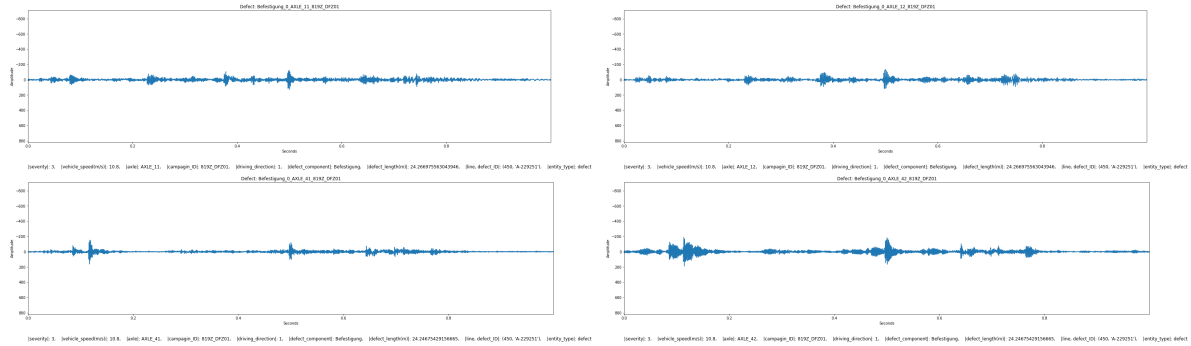


Figure A.11: Befestigung

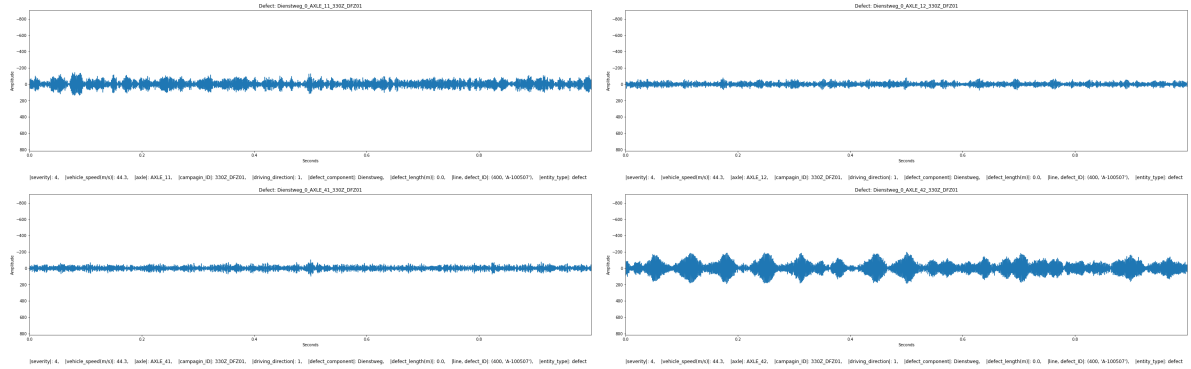


Figure A.12: Dienstweg

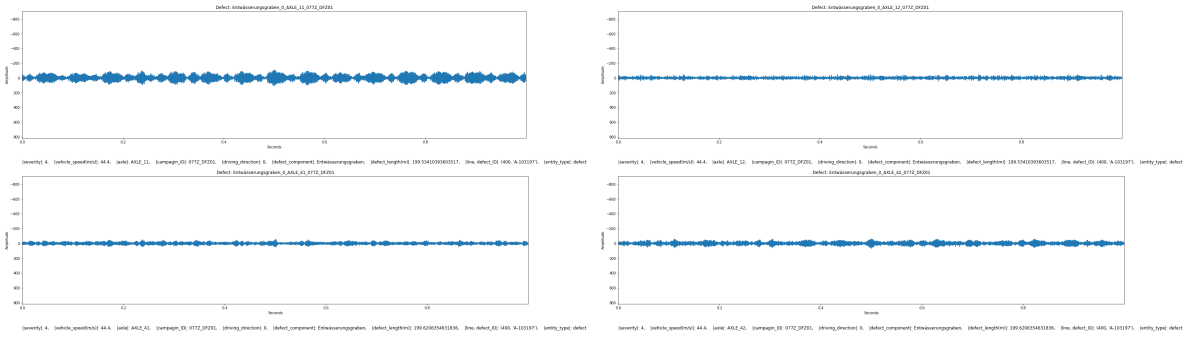


Figure A.13: Entwasserungsgraben

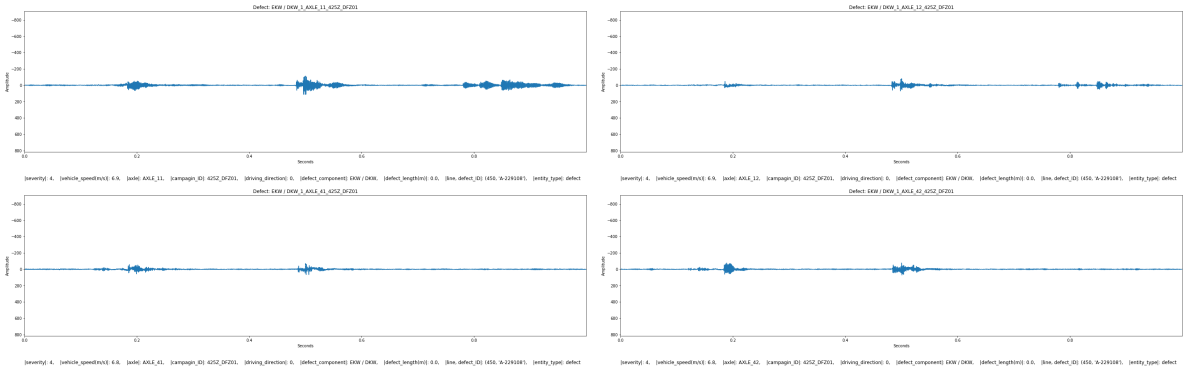


Figure A.14: EKW / DKW

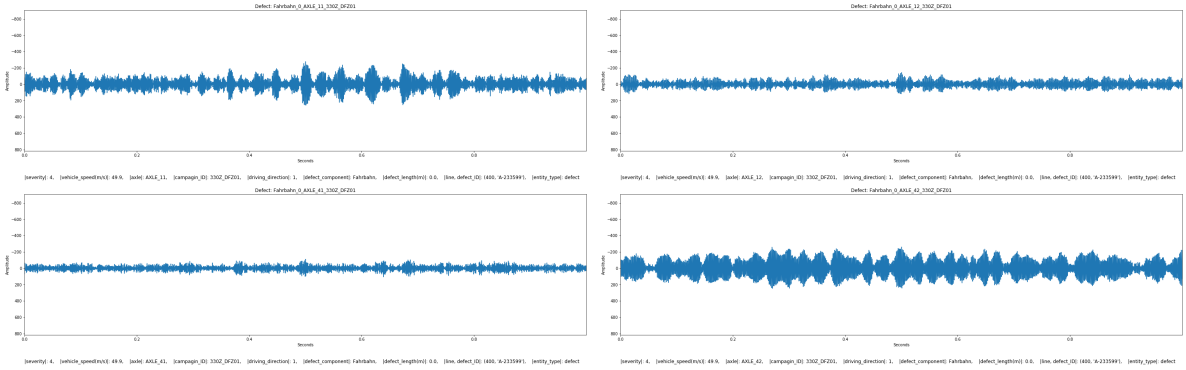


Figure A.15: Fahrbahn

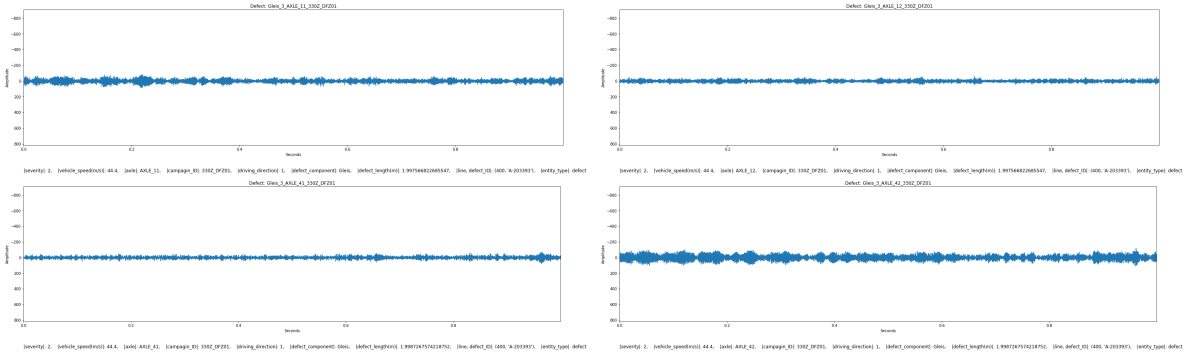


Figure A.16: Gleis

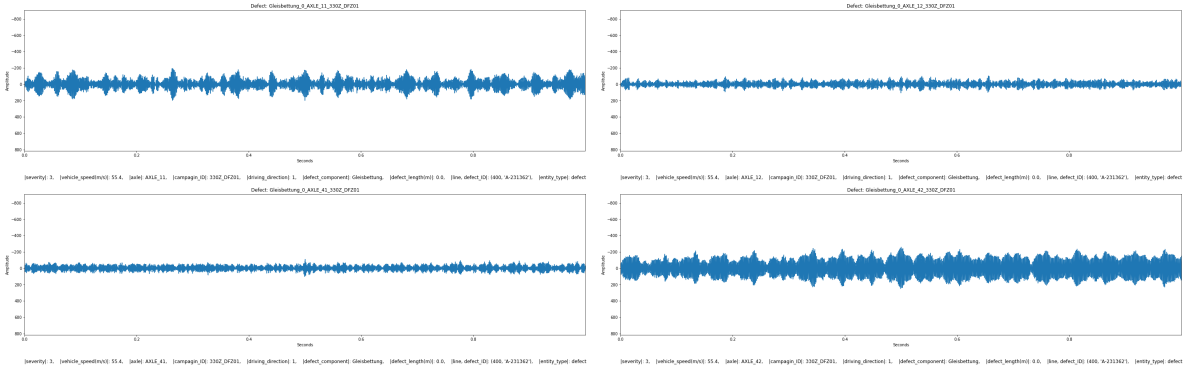


Figure A.17: Gleisbettung

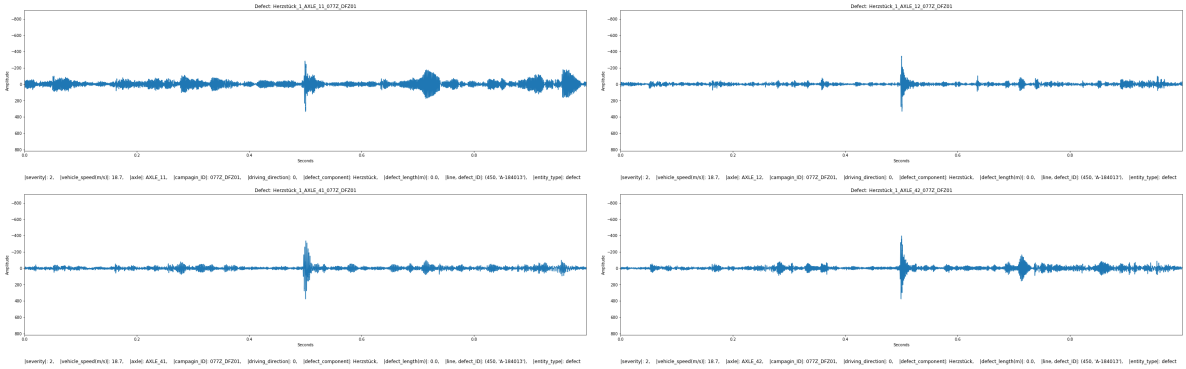


Figure A.18: Herzstuck

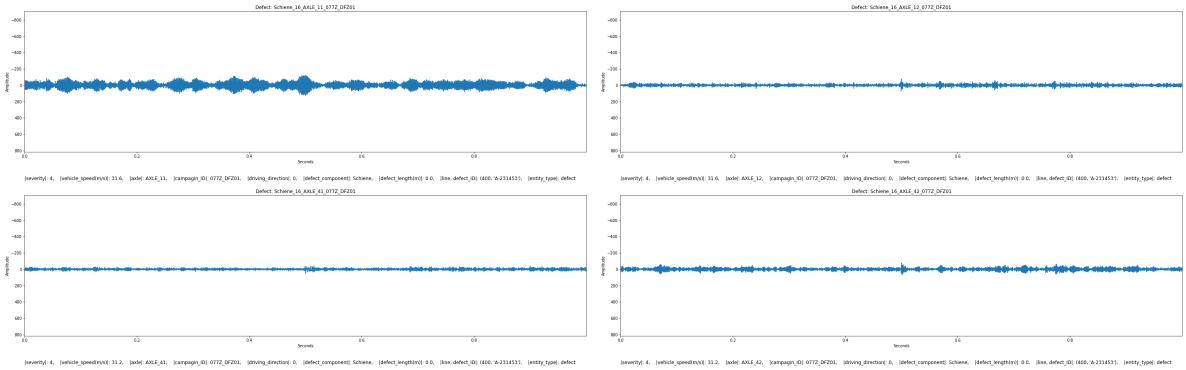


Figure A.19: Schiene

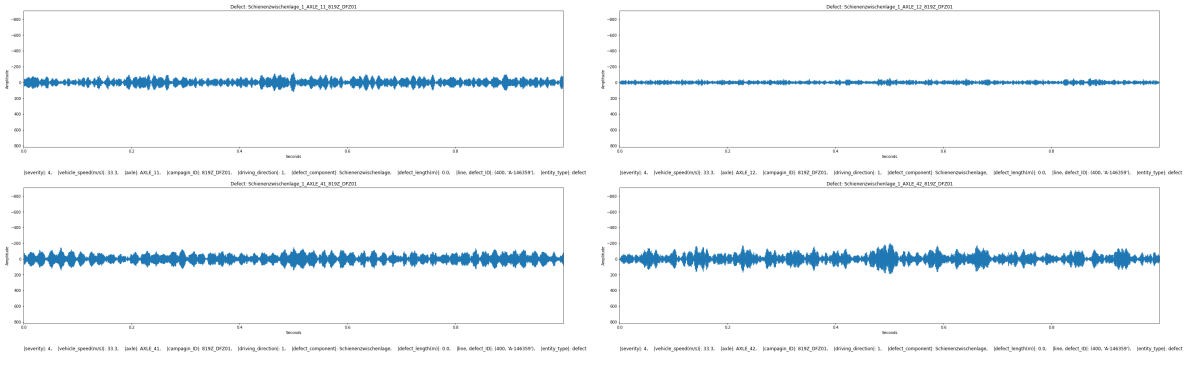


Figure A.20: Schienenzwischenlage

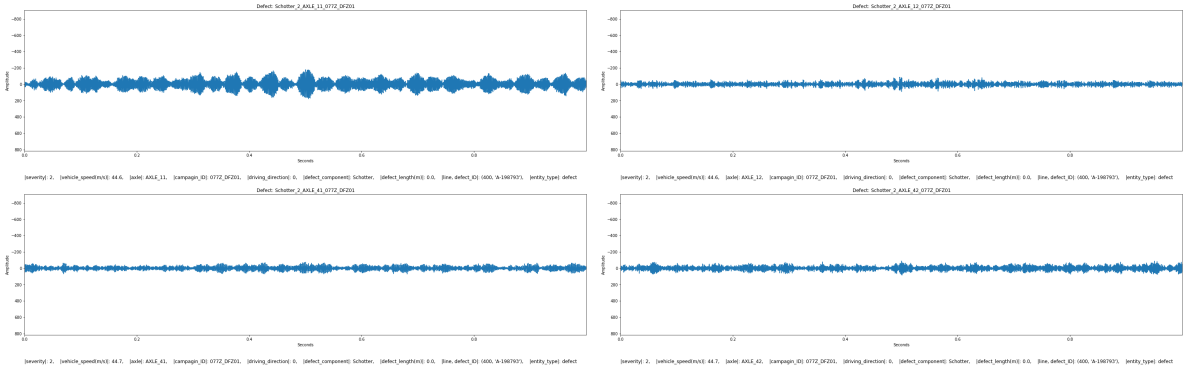


Figure A.21: Schotter

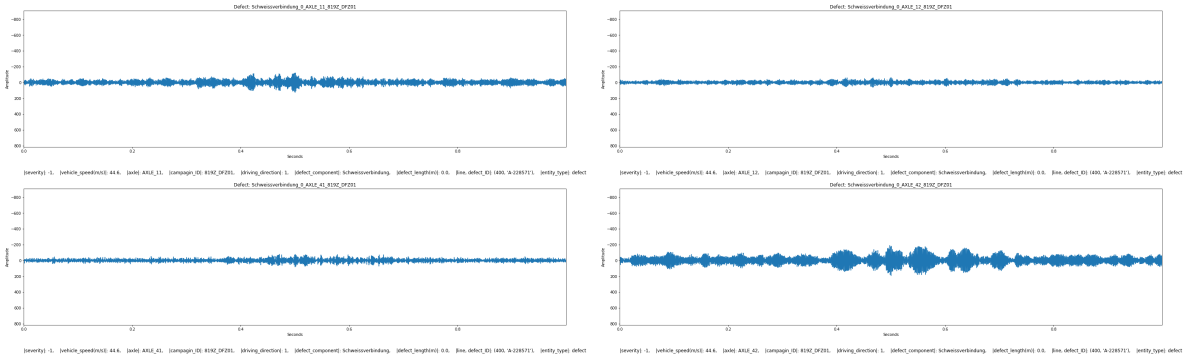


Figure A.22: Schweissverbindung

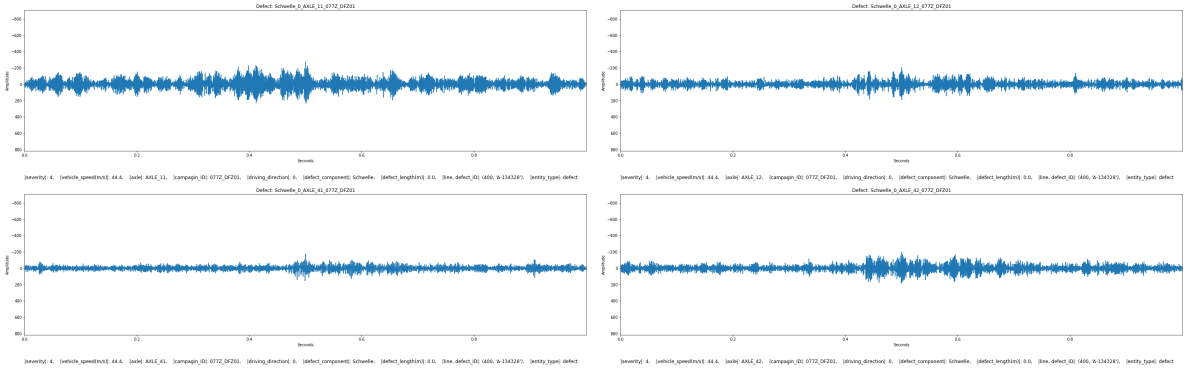


Figure A.23: Schwelle

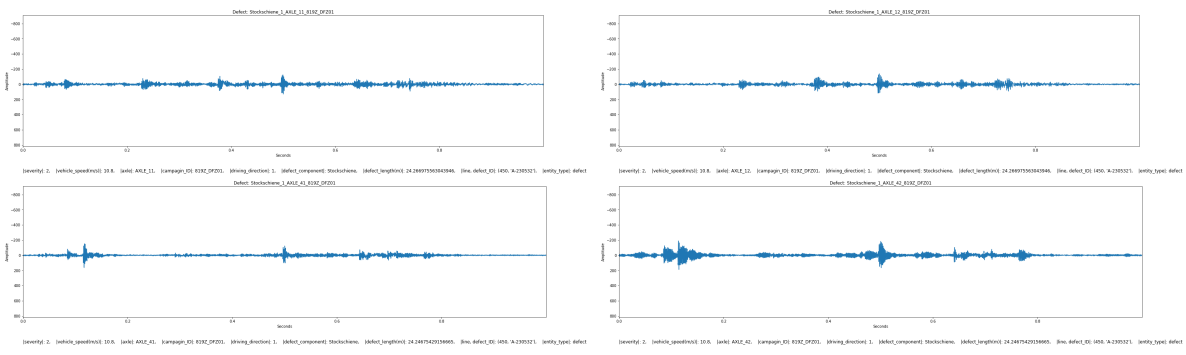


Figure A.24: Stockschiene

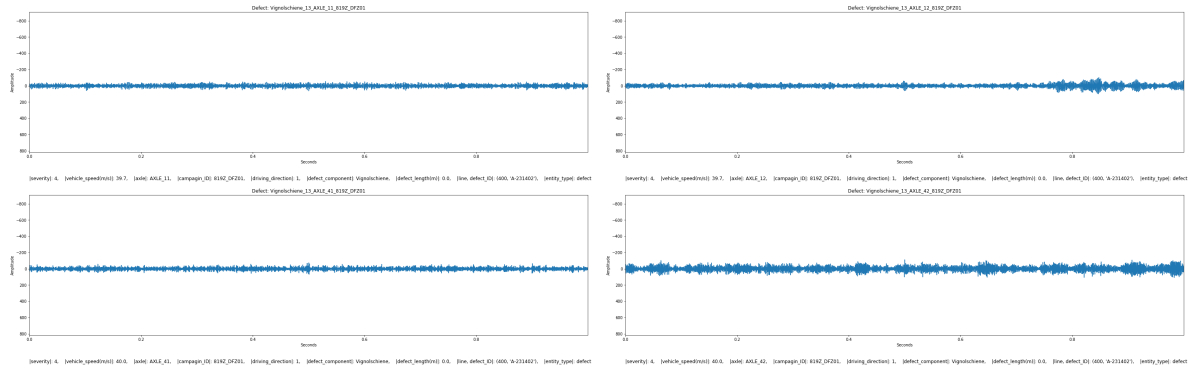


Figure A.25: Vignolschiene

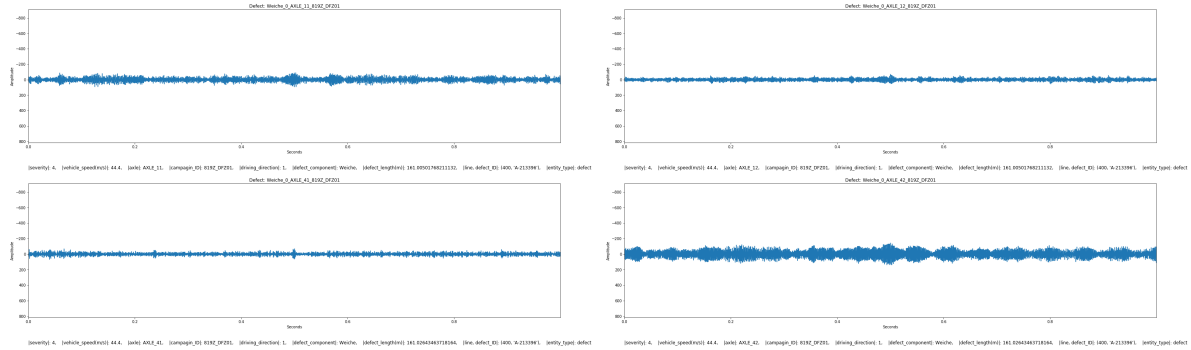


Figure A.26: Vignolschiene

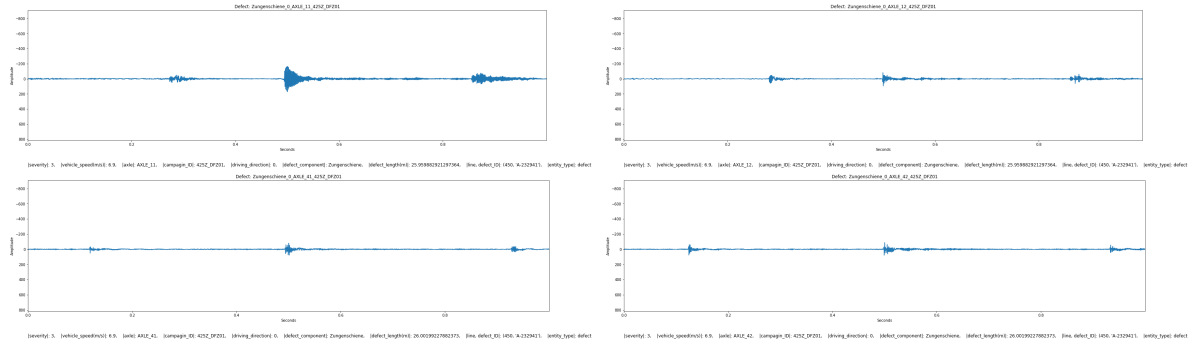


Figure A.27: Zungenschiene

Appendix B

Evaluation

B.1 All 4 axle channels

Metrics / run #	L	ACC	TP	TN
1	–	–	–	–
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	–	–
8	–	–	–	–
9	–	–	–	–
10	–	–	–	–

Table B.1: Experiment result of 10 runs

confusion matrix, epoch plots for each experiment

B.2 Single axle channel

Metrics / run #	L	ACC	TP	TN
1	–	–	–	–
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	–	–
8	–	–	–	–
9	–	–	–	–
10	–	–	–	–

Table B.2: Experiment result of 10 runs

confusion matrix, epoch plots for each experiment

B.3 Low pass-filtered single axle channel

Metrics / run #	L	ACC	TP	TN
1	–	–	–	–
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	–	–
8	–	–	–	–
9	–	–	–	–
10	–	–	–	–

Table B.3: Experiment result of 10 runs

confusion matrix, epoch plots for each experiment

B.4 Single axle channel with speed

Metrics / run #	L	ACC	TP	TN
1	–	–	–	–
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	–	–
8	–	–	–	–
9	–	–	–	–
10	–	–	–	–

Table B.4: Experiment result of 10 runs

confusion matrix, epoch plots for each experiment

B.5 Lowpass filtered single axle channel with speed

Metrics / run #	L	ACC	TP	TN
1	–	–	–	–
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	–	–
8	–	–	–	–
9	–	–	–	–
10	–	–	–	–

Table B.5: Experiment result of 10 runs

confusion matrix, epoch plots for each experiment