DEPARTMENT OF CIVIL, ENVIRONMENTAL AND GEOMATIC ENGINEERING

Semester Project Report

Data-driven identificiation and classification of rail surface defects

Aiyu Liu

Supervised by: Prof. Eleni Chatzi, Cyprien Hoelzl

Saturday 15th 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.

"The condition assessment is evolving from a manual labour based approach to an industrialized assessment. This paradigm shift,.... (don't mention 40 inspectors, their job will in the future still be very important but at the office, checking false positives.)

In order to cope with this issue, *Swiss Federal Railways* (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.2).

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 [3] about 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 towards reducing delays and making trains more punctual and reliable. In this development, the first step consists of identification and classification, while the second step ultimately consists of future defect prediction.

toward a predictive maintenance which will result in a more reliable network [cite article about predictive maintenance]

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 [7] 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.3.

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.

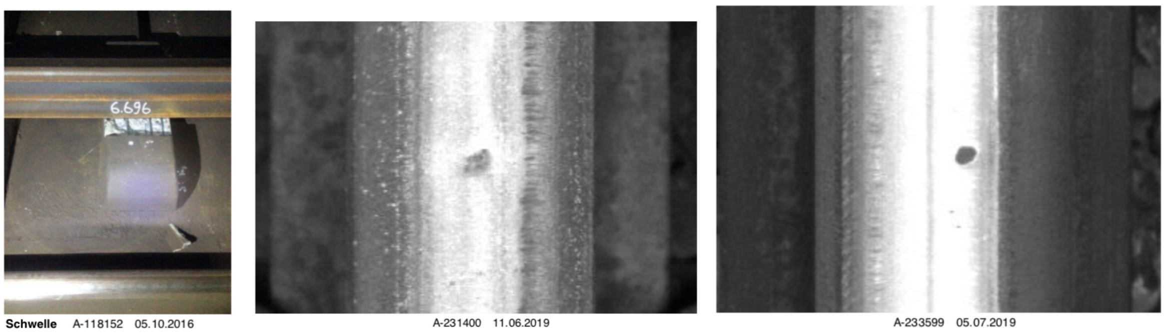


Figure 1.1: Left, middle, right: Schwelle (A-118152), Fahrbahn (A-233599), Vignolschiene (A-231400). The format is: track component (defect ID). These have all been reported as defects (with subcategories) with a length equals to zero, and thus have been classified as point defects using our terminology.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

Figure 1.2: Top, middle, bottom: Gleis-149.8m (A-184063), Schienenzwischenlage-5.0m (A-146358), Bankett-1169.5m (A-231400). The format is: track component-length (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)

For this project, we have solely focused on the point defects for analysis, as this simplifies the problem statement. We thus disregard range-defects such that we do not have to deal with the extra, associated factors.

1.3 Switches and insulation joints

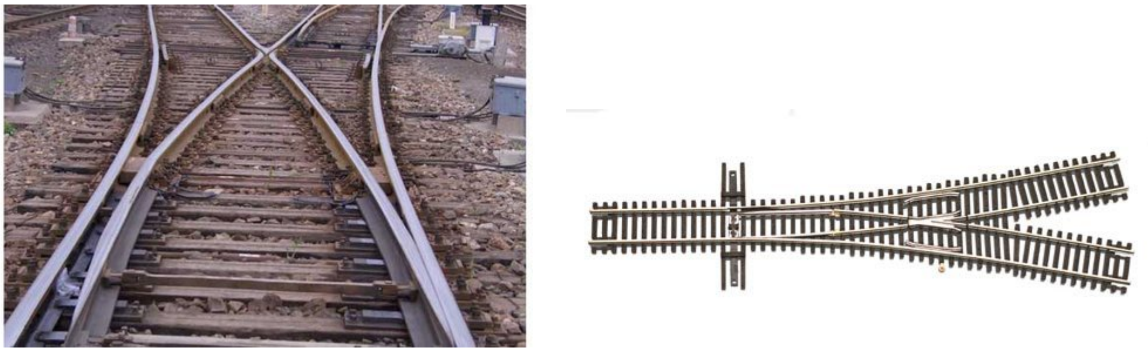


Figure 1.3: (Picture sources from right to left: [6], [4])

Figure 1.4: (Picture sources from right to left: [1], [2])

In this report, we will use the word 'entity' 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 (Y) and transversal (Z) axes (along with the timestamps at each recording), of which, only the Z axis was considered, 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.2 for visualisations of the accelerometer placements on the DFZ.

In this work, we are using the DFZ measurement rides in table 1.1 for classification.

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

Table 1.1: Measurement rides for classification

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, we are using: `keras` along with `tensorflow`. `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 [5]. 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, an overview will be given of how each step was implemented.

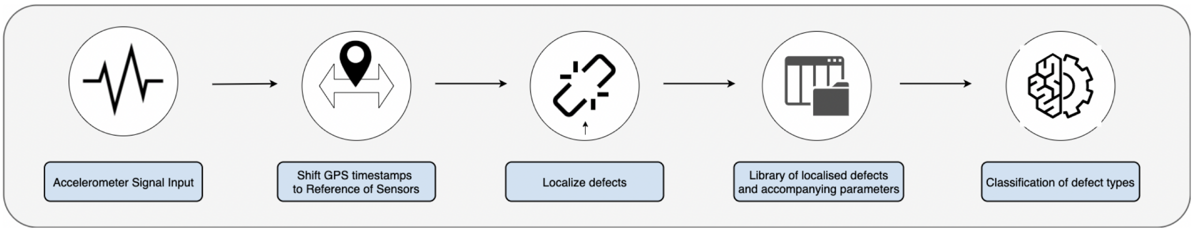


Figure 2.1: Primary pipeline

2.0 Localization of the vehicle:

The vehicle is localized using the combination of multiple data streams: track transponders (Eurobalise) GPS signals, Odometers and Switch detection systems. The reported localization error has been determined to be quite small.

2.1 Shift of GPS timestamps

The location is reported with respect to the GPS Antenna of the vehicle which is close to the pantograph. The location is reported along with a timestamp corresponding to the location of the vehicle at that time at the locatio nof the antenna. todo

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 (every 25 cm vs 24 kHz respectively), we first need to get the corresponding positions for each accelerometer sample. 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 added/subtracted. todo

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).

todo

the window offset can be very small (as small as possible without losing the peaks). Otherwise you will merge too many peaks using the peakfinder. 0.5 of even 0.1 should be ok.

2.3 Entity library

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

- **Timestamps:** timestamps for the sampled acceleration
- **Acceleration:** sampled acceleration at axle box.
- **Vehicle speed (m/s):** vehicle speed at the closest timestamp
- **Severity:** Urgency of the entity; 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 – all entities need to have the same feature column for training.

Additional information about each entity has been retrieved as well, such as: driving direction and corresponding entity IDs. For each entity, we crucially set a true, class label such that we are able to do supervised learning.

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, we 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.

show a few entity signals and their features, refer to the defects presented in introduction, appendix for more signals?

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()`	classifies on an eventual test set

Other utility API functions	
`measure_performance()`	currently only on validation data
`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.

2.4.2 ModelMaker class

As mentioned in the introduction 1.5, this is where we make use of `keras`.

See example of this in next chapter.

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

Layer	Output Shape	Number of params
-	-	-

Table 3.1: Model 1

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

do one model at a time

3.2 Model evaluations

Defect	2	%

Table 3.2: Entity distribution, class distributions

In this section we evaluate our model with regard to a variety of metrics: loss (**L**), accuracy (**ACC**), true positives (**TP**), false positives (**FP**), true negatives (**TN**), false negatives (**FN**), precision (**P**), recall (**R**), area under the curve (**AUC**).

Model / Metric	Model 1			
L	–	–	–	–
ACC	–	–	–	–
TP	–	–	–	–
FP	–	–	–	–
TN	–	–	–	–
FN	–	–	–	–
P	–	–	–	–
R	–	–	–	–
AUC	–	–	–	–
Relative diff?	–	–	–	–

Table 3.3: Average metrics times and their standard deviations in parenthesis - rel diff?

average epoch plot?

intermediate results

```
Results model 1, only defects, new dataset
>>> Summarize results <<<
Overall loss: 2.969% (+/-0.009)
Overall acc: 0.074% (+/-0.014)
Overall true_positives_10: 0.300% (+/-0.458)
Overall fp: 0.600% (+/-0.917)
Overall tn: 3609.400% (+/-0.917)
Overall fn: 189.700% (+/-0.458)
Overall precision: 0.175% (+/-0.317)
Overall recall: 0.002% (+/-0.002)
Overall auc: 0.568% (+/-0.007)

Results model 1, only ins joints and switches, new dataset
>>> Summarize results <<<
Overall loss: 0.691% (+/-0.001)
Overall acc: 0.587% (+/-0.000)
Overall true_positives_11: 155.000% (+/-0.000)
Overall fp: 109.000% (+/-0.000)
Overall tn: 155.000% (+/-0.000)
Overall fn: 109.000% (+/-0.000)
Overall precision: 0.587% (+/-0.000)
Overall recall: 0.587% (+/-0.000)
Overall auc: 0.572% (+/-0.031)
```

3.3 Visualisation of class clustering

insert the pca plots

3.4 Discussion

Data amount, circumvent: could self-engineer data.

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.

ensure that data is uniform. That is, some of the data has calibration and some hasnt.

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.

how good is the foundation to move onwards with further research

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
- 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

- add speed as a feature
- 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?
- Which type of defects are we actually working with, we can see that it does okay at the switches and ins joints, but no chance with the defects
- concat all the axle channels and train on them
- try to use low pass filter - (would make everything faster)
- 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
- account for severity
- unique bincount of each defect severity
- insert signal plots, uniform axes. dont show peaks
- better epoch plots
- You can put more citations, for example about some articles focusing on recognizing speech/sound, signals which have also very similar signatures
- Look more into defect types, its specifics
- (drawings)
- too many self in classifier?

Normal. You are on tracks which are "brand new- built in 2006-2018". so the defects are all minor. The new measurement data we recently got would work much much better since they drove on pretty bad bad bad stuff :)

numpy array is faster for operating on data. Pandas s more convenient for storing the data

Bibliography

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

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

[3] 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).

[4] 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).

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

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

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

Appendix A

Introduction

A.1 List of defect categories

herstuck

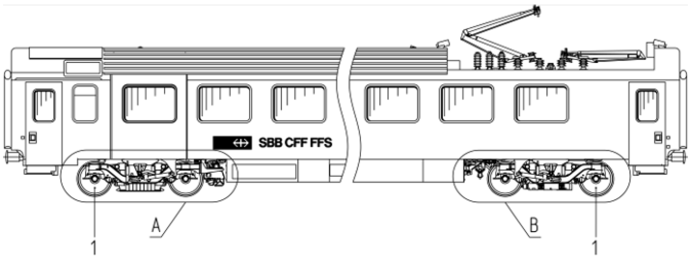
schiene

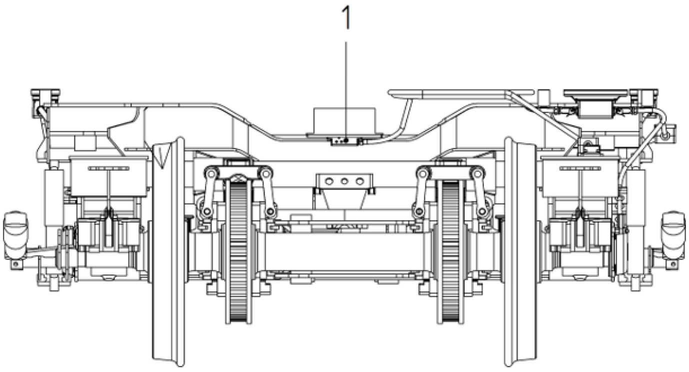
etc

maybe just retrieve from the SBB reports

yupp, just SBB reports, or the SBB defect catalog (should be in the first files I gave you)

A.2 Vehicle and accelerometer placements





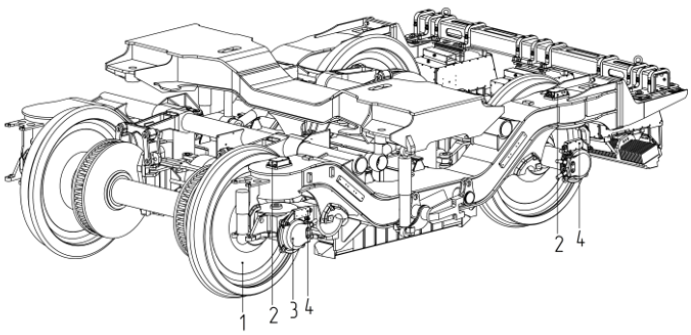


Figure A.1: Illustrations of the accelerometer placements on the DFZ. For this project we have only considered axle number 4 in the third figure. (Source of picture: SBB documents)

A.3 SBB defect report example

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

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

Position der Abweichung		**FS-spezifische Positionsfelder**	
Linie:	400 Löchligut - Wanzwil - Rothrist West	Gleis:	
km von:	17.2 km bis: 18.4	Weiche:	
Positionierung:	Keine Positionierung	Mast:	
Position:		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		**Ausführung durch**	
SCDE Relevant:	Nein	U-Nummer:	
Ultraschallfelder		Datum:	
Notverlaschen:	Nein	Unterschrift:	
In Garantie:	Nein		
Qualität:			
Schienenprofil:	SBB VI bzw. EN 60 E1/E2		
Frist:			
Radius:	3197.9		
Prüfungsart:			
Hersteller:			
Jahr:			

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)

Appendix B

Evaluation

Metrics / run #	L	ACC	TP	TN	FP	FN	P	R	AUC
1	–	–	–	–	–	–	–	–	–

Table B.1: Experiment result of 10 runs

confusion matrix, epoch plots