Neural network based condition monitoring for track circuits

T. de Bruin

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Neural network based condition monitoring for track circuits

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

T. de Bruin

May 25, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Neural network based condition monitoring for track circuits

by

T. de Bruin

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: May 25, 2015

Supervisor(s):

Prof.dr. R. Babuška

Ir. K.A.J. Verbert

Reader(s):

Prof.dr.ir. B.H.K. de Schutter

Dr. D.M.J. Tax

Summary

Railway track circuits are electrical systems that are used for train detection. For the safety and availability of railway networks it crucial that the track circuits function properly. In order to plan the maintenance of the track circuits in a way that ensures they continue to function properly, it is important to detect the presence of faults in the track circuits as early as possible. Additionally, the cause of the fault and the severity of the fault should be determined. In this thesis a method is proposed to accomplish these tasks based on the commonly available measurement signals.

In this thesis, artificial neural networks are used to perform the fault diagnosis directly from the measured signals from several track circuits in a geographic area. Although only a small amount of measurement data was available during the writing of this thesis, it is most likely that reasonable amounts of (unlabeled) measurement data will become available at a later time, as the required data logging equipment has already been installed. To train the networks, the small available dataset is analyzed and used together with the currently available qualitative understanding of the effects of faults on the system to construct a generative model. The synthetic data produced by this model are then used to train and test the neural networks.

Artificial neural networks have recently achieved state of the art performance on difficult pattern recognition problems in several different fields such as image recognition and speech recognition. These recent successes can be largely attributed to the combination of large networks and large labeled datasets. In the condition monitoring domain, large labeled datasets are generally not available. This prevents the use of the large neural networks that have become so successful in other fields. In-spite of this, some of the ideas that have become popular in other domains might still have value in the condition monitoring domain. This thesis focuses on bringing the Long Short-Term Memory architecture and the concept of end-to-end learning to the condition monitoring domain. In order to use the large fraction of the data that is expected to be unlabeled in practice, an unsupervised learning strategy is investigated. This strategy uses unlabeled data to pre-train a network so that it can more efficiently learn from the scarce labeled data.

The results show that a neural network trained with the end-to-end learning strategy can detect faults in track circuits and determine their cause very accurately from the synthetic

data. The fault performance of the neural network strategy, which uses no prior knowledge of the system, is comparable to that of a hand crafted method that is based on the same prior knowledge as the generative model. This proves that a neural network trained with end-to-end learning can detect faults from the data very accurately and on real data, it would probably outperform methods based on prior knowledge. It is also shown that a neural network can estimate the severity of a fault. When only a small amount of labeled data is available, it is shown that using the unsupervised pre-training strategy gives better results than using a network that is trained directly with end-to-end learning.

Based on the results in this thesis it seems likely that a Long Short-Term Memory Recurrent Neural Network could outperform fault isolation methods based on prior knowledge on the track circuit case. However, further improvements in using the unsupervised pre-training strategy are necessary to ensure that the amount of labeled data that is expected to be available in practice is sufficient to achieve this performance.

Table of Contents

Acknowledgements

I would like to thank my supervisors Robert Babuška and Kim Verbert for the time they have spent giving me the feedback that helped improve both this thesis and myself. It has been a pleasure to learn from you both.

Delft, University of Technology
May 25, 2015

T. de Bruin

Chapter 1

Introduction

To enable the safe and efficient operation of a railway network, it is crucial to detect the presence of trains in the sections of a railway track. The railway track circuit is worldwide the most commonly used component for train detection [1]. A railway track circuit detects trains by sending an electrical current from a transmitter at one end of a section, through the rails and a receiver at the other end of the section, back to the transmitter. When no train is present in the section this will cause a high current flow through the receiver. When a train does enter the section, the wheel-sets of the train short this circuit, leading to a low current flow through the receiver. This principle is illustrated in Figure 1-1. Based on the current level in the receiver, the section is reported free or occupied.

There are several factors that influence the current flow through the receiver, both when a train is present in the section and when no train is present. During normal operation these current levels vary due to influences from the environmental effects and the properties of the trains passing through the section. In addition to this, faults can develop in the system and affect the current levels as well. When these *faults* reach a certain *severity*, they can influence

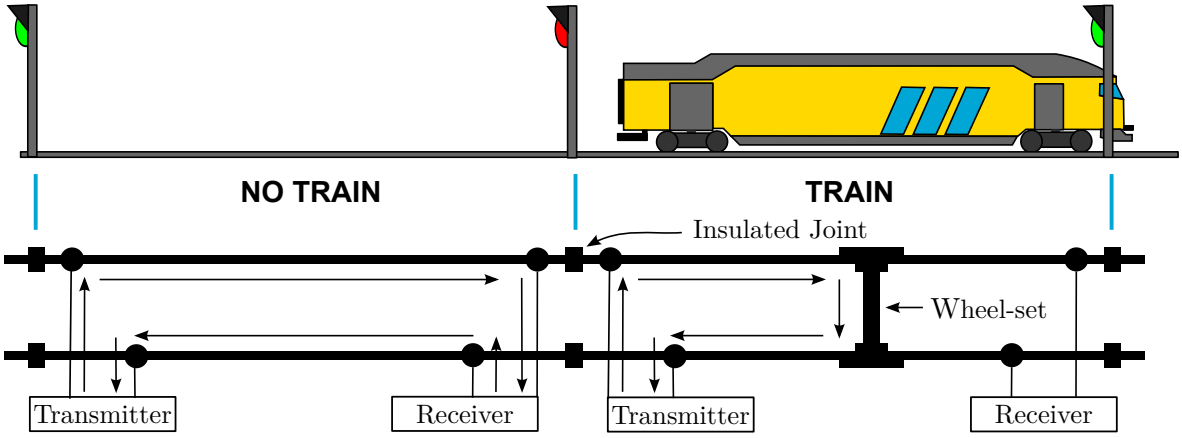


Figure 1-1: Basic working principle of a track circuit.

the current levels to the extent that it causes a *failure*; where the track circuit is no longer able to correctly classify the section as occupied or free.

To prevent accidents, the track circuit system is designed to be fail-safe. As such, when a fault in the system leads to a failure, the track circuit should report the presence of a train even when the track is not occupied. However, when this happens, train delays occur. Moreover, in-spite of the fail-safe design of the track circuit, there are situations in which the railway section can be incorrectly reported as free, which can potentially lead to dangerous situations. Both from the availability standpoint and the safety standpoint it is therefore very important to prevent track circuit failures. This requires a preventive maintenance strategy to ensure that components are repaired or replaced before a fault develops into a failure. To schedule the maintenance of the track circuits in the most efficient and effective manner, it is necessary to detect faults as soon as possible and to determine the cause and severity of the fault.

In this thesis the following terminology is used: *fault detection* is defined as determining whether a fault is or is not present in the system. *Fault isolation* is defined as the combined problem of the detection of faults and the determination of the cause of the fault when one is present. In this thesis fault isolation is a *classification* problem with seven *classes*; one for the healthy state and six for the different *fault types* considered in this thesis. *Fault diagnosis* is defined as performing both the fault isolation task and providing an estimate of the severity of the fault when one is present. The eventual goal of this research is *condition monitoring*, which is defined in this thesis as performing fault diagnosis on a system frequently enough to enable condition based maintenance.

The monitored variable in the track circuits is the root mean squared value of the magnitude of the electrical current in the receiver. Using only this information, a condition monitoring system needs to be able to distinguish between six different fault types and the normal variation of the track circuit currents. This is made possible by combining signals from several track circuits in a small area, since the fault types can be identified from their spatial and temporal dependencies [2]. Additionally, although the variables that determine the normal variation in the current levels (e.g. train properties, weather, etc) are not measured, they are latent variables that influence not only the track circuit of interest, but also other track circuits in the neighborhood. This makes it possible to infer these variables from the current measurements of a group of track circuits, which allows separating the current variations caused by normal variation factors from the effects caused by faults.

The six fault types considered in this thesis are:

- Rail contamination
- Insulated joint defect
- Conductive object
- Mechanical rail defect
- Electrical disturbances
- Ballast degradation

An explanation of how these faults affect the electrical current in a railway track circuit and the definition of their spatial and temporal dependencies used in this thesis is given in Section 2-5-7.

Qualitative knowledge about the spatial and temporal dependencies of the faults considered is given in [2]. This knowledge can be used to create a condition monitoring system. In this thesis however, the view is taken that a more precise relationship between the measured signals and the condition of a track circuit can be learned from historic measurement data. To this end, an Artificial Recurrent Neural Network called the Long Short-Term Memory (LSTM) network [3] will be used.

Artificial Neural Networks have recently achieved state of the art performance on a range of challenging pattern recognition tasks, such image classification [4] and speech recognition [5]. Although neural networks have been used for condition monitoring for some time, the ideas that have enabled state of the art performance in image recognition and speech recognition do not appear to be applied in this domain. This can be partly explained by the fact that these methods require large amounts of labeled data, something that is generally not available in the condition monitoring domain, especially of faulty systems. However, this thesis takes the view that some of the ideas that allow neural networks to achieve state of the art performance in other domains *are* applicable to the condition monitoring domain. Therefore the feasibility of applying these concepts to the track circuit case is investigated.

Currently, not enough measurement data are available to train the network and to verify its performance. Therefore, the available data are combined with qualitative knowledge of the fault behaviors [2] to construct a generative model. The neural network is trained and tested with synthetic data produced by this model. Since the relevant data logging equipment is already installed, it seems reasonable to assume that enough measurement data will become available at some future time for the training of the networks. Since this measurement data will be mostly unlabeled, the unsupervised pre-training of the network is investigated to minimize the amount of labeled data that will be needed.

The rest of this thesis is organized as follows. The generative model used to generate the synthetic data to train and test the neural networks is discussed in Chapter 2. Chapter 3 discusses the theoretical foundation of the neural networks used in this thesis and motivates the choices for the structure of the networks. In Chapter 4 the results of training and testing the networks with the synthetic data are presented. A summary of the results and the conclusions of this thesis are given in Chapter 5.

Chapter 2

Generative Model

In this chapter, a generative model is constructed to produce sufficiently realistic synthetic data samples representing both healthy and faulty track circuit behavior. Note that the purpose of the synthetic data is to test the ability of an artificial neural network to separate the effects of faults on the currents in the receiver of a track circuit from the normal variations of those currents. As a consequence, it is more important that the synthetic data exhibits comparable variations to real data than that the model accurately calculates the currents.

In [2] the nominal working of a track circuit as well as the qualitative effects of faults on the behavior of the system have been described. In Section 2-1 a summary of the working of track circuits is given. Based on this knowledge and a dataset with measurements, the model is constructed. A description of the dataset used is given in Section 2-2.

Initially, the measured data are fitted to an exponential curves model, which is described in Section 2-3. In Section 2-4, the values of the parameters of this model that have been fitted to the data are analyzed. The dependencies of these values on the environmental conditions during the measurements are investigated and a simpler model is constructed based on the parameter distributions from the initial model. In Section 2-5, the structure of this simplified model is summarized and the value distributions and dependencies of the parameters are discussed for both nominal and faulty cases. In Section 2-6, the sampling strategy for the data is discussed.

2-1 Track circuit working

The generative model described in this chapter is based on the double rail alternating current track circuits used in the Netherlands. A detailed description of the working of track circuits and the effects of the faults considered in this thesis can be found in [2]. This section gives a brief summary.

An electric model of a track circuit is given in Figure 2-1. This model represents the part of a railway track that is monitored by one track circuit. This part is called a section. The

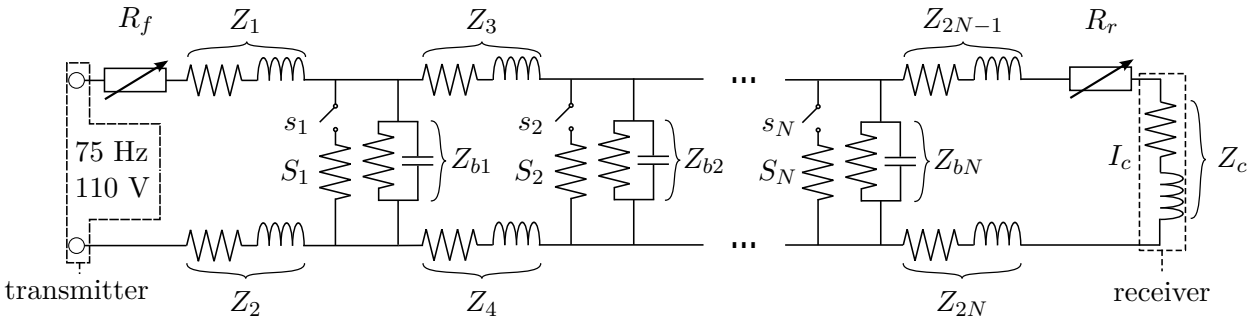


Figure 2-1: Electrical model of a track circuit. The model describes one track section which is split into N parts in the model.

transmitter on the left of Figure 2-1 sends an alternating electric current signal through the rails. When the track circuit is working properly, this signal travels through one of the rails to the opposite end of the track circuit and through the receiver. It then flows back through the other rail to the transmitter. In this case, the amplitude of the current passing through the receiver I_c will be high. As soon as a train arrives in the section, the path from the rails through the wheels and axle of the train to the other rail will form a short circuit. When the system is functioning correctly, the resistance of this short circuit will be sufficiently low for the amplitude of the current through the receiver I_c to immediately drop to a very low level. The amplitude of the current should ideally stay at this very low level until the last wheel set leaves the section, at which it should quickly return to the high level.

In the electric model given in Figure 2-1, the section of track that is monitored by a single track circuit is divided into N subsections along the track. For the n-th subsection, Z_{2n-1} and Z_{2n} are the impedances of the two rails. When a wheel set is present in subsection n the switch s_n is closed. The resistance S_n represents the short circuit path through a wheel set of the train. The impedance Z_{bn} represents the short circuit path through the ballast between the rails in the n-th subsection.

Track circuits are installed in sections of track of different lengths. This will influence the total impedance of the rails. Furthermore, when rails are joined this creates a locally higher impedance. Finally, the impedance of the ballast is influenced by factors such as the length of the track, the cleanliness of the ballast and the weather. It is important to ensure that the current I_c is sufficiently high when no train is present in the section and sufficiently low as soon as a train enters, in spite of these variations. To achieve this, the adjustable resistances R_r and R_f are tuned manually for each track circuit.

2-2 Measurement dataset

The dataset that is used in this research to construct a model of the normal (healthy) behavior of track circuits contains measurements of the current flowing through 3 track circuits spread over 2 neighboring tracks that are located at station Beilen in the Netherlands. These track circuits are indicated with TC_A, TC_B and TC_C. Their relative locations are shown in Figure 2-2. Track circuits TC_A and TC_B are located in consecutive sections on the same track and TC_C is on a parallel track next to TC_A. All sections are next to platforms in the station.

Figure 2-2: Relative locations of the track circuits in the dataset.

Measured data

During a period of 1 year, the following 2 types of measurements have been taken on the 3 track circuits:

1. **Train Passing:** When the root mean squared value of the receiver current I_c drops below a certain threshold, a train is presumed to be present in the section. When the root mean squared current rises above the threshold again the train is assumed to have left the section. Between these times the train is presumed to be in the section and a *train passing event* is occurring. The detection of this event from the measured current triggers the storage of measurement data. Note that the event is registered when the current level drops to a sufficiently low level. This means that the data are only stored correctly when the track circuit does not suffer from a fault that is so severe that it causes a failure. A fault that has not yet developed into a failure does not affect the data storage. The following data are stored for each registered train passing event:
   (a) The date and time of the train passing event
   (b) The root mean squared value of the receiver current $I_c(t)$ from 2 seconds before the train arrives until 2 seconds after the train leaves the section, measured every 40 milliseconds.
2. **Empty track:** When no train is present, every 15 minutes a measurement is saved with the following data:
   (a) The date and time of the measurement.
   (b) The maximum, average and minimum values of the current during the 15 minutes.

The dataset that is available will be used to model the normal (fault free) behavior of a track circuit. It is assumed that the three track circuits in the data set do not suffer from faults.

2-3 Generative model structure

Although the current I_c could be calculated from the electrical model of Figure 2-1, the values of the impedances and their *quantitative* dependencies on faults are not known. Additionally, the aim of the generative model is not to exactly model the electrical current in the receiver, but rather to produce synthetic data that are similar to those from the data logging

equipment connected to the receivers. This measuring equipment has dynamics of its own that are not included in the electrical model. Therefore, instead of constructing a model from first principles, a simple model is used instead which is made of several exponential curves with a structure that was empirically found to fit the real measurements well. This model is then fitted to a large dataset of measured current values during train passings and it is attempted to find the quantitative dependencies of the model parameters on the known influences present during the historical measurements. Based on the parameters found for the relatively expressive model discussed in this section a simplified model is constructed in Section 2-4.

During a train passing event, four phases can be distinguished. These phases are indicated in Figure 2-3:

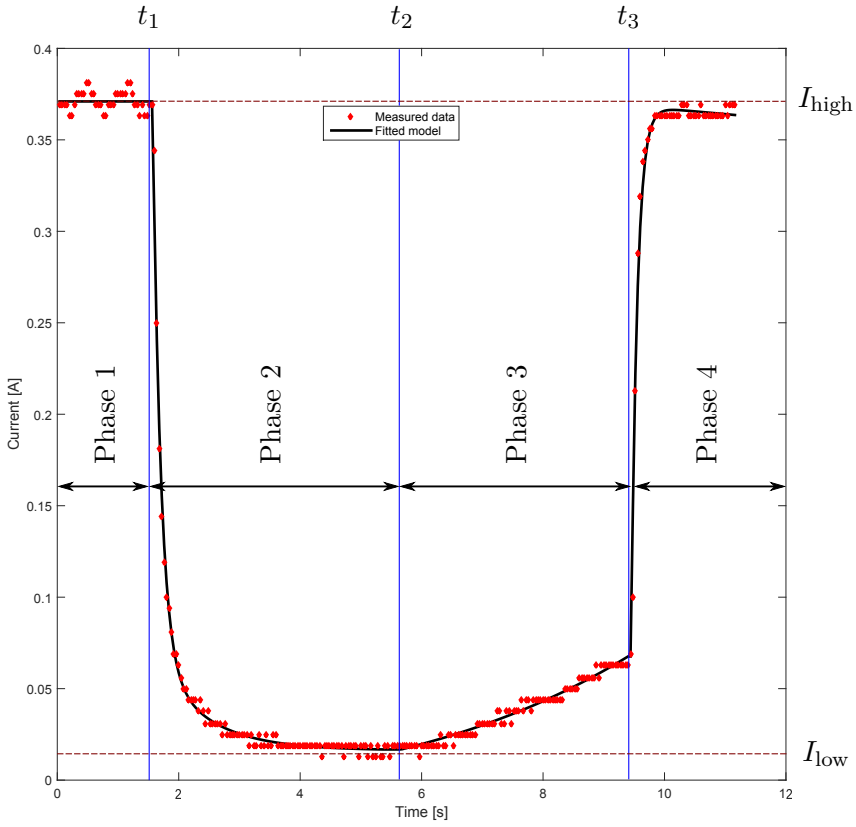


Figure 2-3: Measured current in a track circuit while a train passes and a model of the form of (2-1) fitted to it.

- **Phase 1:** Between t_0 and t_1 the train has not yet arrived in the section. During this phase the current $I_c(t)$ through the receiver should therefore be at the high level: $I_c(t) = I_{\text{high}}$.
- **Phase 2:** At $t = t_1$ the first wheel-set of the train enters the section. In Figure 2-1 this can be represented as closing s_1. If the resistance S_1 of the wheel set short circuit

connection is low enough this should result in a very quick drop of $I_c(t)$ to its low value I_{low}. However, in a large portion of the samples in the dataset the current drop is more gradual. By fitting a number of samples from three different track circuits to several equations for step responses it was found that this phase could be accurately and robustly described by a second-order model with the transient response of the form: $I_c(t) = \alpha_1 e^{-(t-t_1)/\tau_{\alpha_1}} + \beta_1 e^{-(t-t_1)/\tau_{\beta_1}}$.

- **Phase 3:** Although ideally $I_c(t) = I_{\text{low}}$ should hold until the last wheel-set of the train leaves the section, in the majority of the samples in the dataset the current starts to increase before this time. The curve between $t = t_2$ where the current is at the lowest level and $t = t_3$ where the last wheel-set leaves the section can take different forms. These are discussed in section Section 2-4. In almost all cases this curve can be accurately described by a function of the form: $I_c(t) = \alpha_2 e^{(t-t_2)/\tau_{\alpha_2}} + \beta_2 e^{(t-t_2)/\tau_{\beta_2}}$.
- **Phase 4:** After the last wheel-set leaves the section at $t = t_3$ the current $I_c(t)$ quickly increases to a value near I_{high}. On some of the samples some overshoot is observed and on some samples a trend after the step is observed. Although a first order step response was found to accurately describe many of the samples, a function of the form $I_c(t) = \alpha_3 e^{(t-t_3)/\tau_{\alpha_3}} - \beta_3 e^{-(t-t_3)/\tau_{\beta_3}}$ was found to represent these less common cases as well and is therefore chosen for the initial expressive model.

The initial expressive model that is fitted to the measured data has the following form, with $\Delta I_{\max} = I_{\text{high}} - I_{\text{low}}$:

$$I_c(t) = \begin{cases} I_{\text{high}} & \text{for} \quad t < t_1 \quad \text{(Phase 1)} \\ I_{\text{low}} + \Delta I_{\max}\left(\alpha_1 e^{-(t-t_1)/\tau_{\alpha_1}} + \beta_1 e^{-(t-t_1)/\tau_{\beta_1}}\right) & \text{for} \quad t_1 \leq t < t_2 \quad \text{(Phase 2)} \\ I_{\text{low}} + \Delta I_{\max}\left(\alpha_2 e^{(t-t_2)/\tau_{\alpha_2}} + \beta_2 e^{(t-t_2)/\tau_{\beta_2}}\right) & \text{for} \quad t_2 \leq t < t_3 \quad \text{(Phase 3)} \\ I_{\text{low}} + \Delta I_{\max}\left(\alpha_3 e^{(t-t_3)/\tau_{\alpha_3}} - \beta_3 e^{-(t-t_3)/\tau_{\beta_3}}\right) & \text{for} \quad t \geq t_3 \quad \text{(Phase 4)} \end{cases} \tag{2-1}$$

2-3-1 Model fitting

To fit all of the available data to the model given in (2-1), for each of the measurement sequences from the data logging equipment the times t_1, t_2 and t_3 need to be determined. This was done by defining the times in the following way:

- t_1: the time of the measured sample before the sample with the largest negative one step difference to the next sample.
- t_2: the last index of the 10 sample average with the lowest value.
- t_3: the time of the measured sample before the sample with the largest positive one step difference to the next sample.

After this, the value of I_{high} is taken as the average of the samples up to t_1. The value of I_{low} is taken as the lowest 10 sample average. Then, the data from the subsequences representing phase 2, phase 3 and phase 4 are fitted to the equations given in (2-1) with the `lsqnonlin` matlab function.

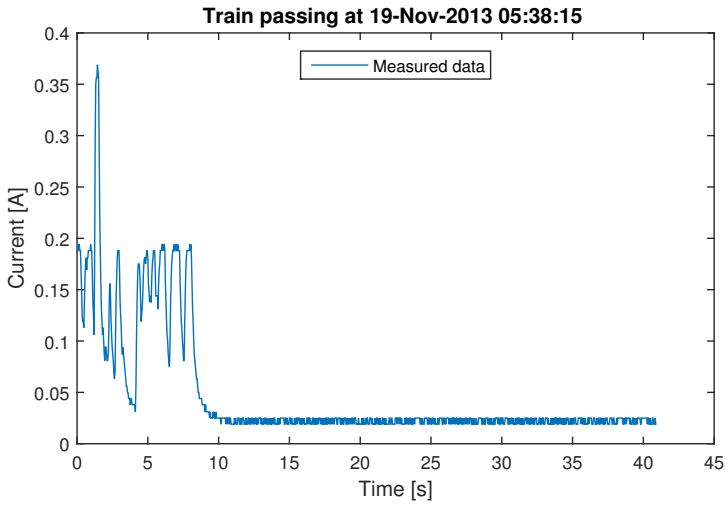


Figure 2-4: Train passing measurement with two types of data corruption. The first 8 seconds are corrupted by a disturbance signal. After about 8 seconds the train is present in the section but the measurement reaches its maximum length of 40.96 seconds with the train still in the section.

2-3-2 Dataset problems

Problems exist with the dataset. One issue is that the train passing measurements are at most 40 seconds long. Since all 3 of the track circuits are located next to a platform in a station, trains quite often stay in the section for longer than this time period. The train passing measurements that suffer from this type of data corruption have a current level at the end of the measurement period that is far too low and can be automatically identified. An additional problem is that when an empty track measurement is taken after one of these train passing measurements is finished, the minimum and average values of the empty track measurement are too low as the train is still in the section when the measurement starts. This type of data corruption was automatically detected in about half of the samples. This matches up well with the time table of trains scheduled to pass station Beilen. About half of these trains are scheduled to stop at the station and the other half is not.

Aside from these problems stemming from the limited maximum measurement length, a significant portion of the data is corrupted by a disturbance signal with a large amplitude. The origin of this signal is unknown. One possibility is that it is related to the Automatische Trein Beinvloeding (automatic train influencing) (ATB) signal, which is an electric signal that is part of a safety system which is transmitted through the same cables. Not all measurements are affected by the disturbance signal and not all track circuits are affected equally. Most of the samples with this type of data corruption come from TC_B. The samples that are corrupted by this disturbance signal can be detected automatically fairly reliably from their one step difference signal. However, this does not ensure that the data are uncorrupted. An example of a measurement suffering from both the disturbance signal and the limited measurement length problems is shown in Figure 2-4

Table 2-1: Dataset properties.

Track circuit	TC_A	TC_B	TC_C
Total train passing events measured	25349	30402	32361
Usable models extracted	12502	6570	10470

Retained data

To ensure that most of the used data is uncorrupted and therefore representative of the normal behavior of a track circuit, the data of a train passing are fitted to a model which was found to describe a large number of the train passings accurately. This model is described in the next section. By only retaining the measurements that are described accurately by the model, it is ensured that there are no large anomalies in the retained data samples. The criteria that are used to determine that a fitted model describes the measured data during a train passing event well enough are:

1. The mean squared error between the measurements and the fitted model is below $6 \cdot 10^{-5}$
2. The absolute value of the mean difference between the measurements and the fitted model is below $2 \cdot 10^{-3}$

These values have been chosen by visually comparing samples of fitted models to the measured data with increasing errors and determining the minimal error values for which bad samples started to appear. In Table 2-1 the number of fitted models that have been retained are given for the three track circuits in the data-set.

2-4 Parameter dependencies and model simplification

In this section the values of the parameters of (2-1) that were found by fitting this model to the retained measured data are investigated. Based on the parameter distributions found, a simplified model is proposed.

The influences of the presumed factors of variation on the parameters of interest are verified using the Analysis of Variance (ANOVA) test. In the figures presented in this section that investigate these dependencies, the parameters of interest will be given along the vertical axis. The measurements were grouped according to the suspected factors of variation, given along the horizontal axis. If the parameter of interest in a figure is indeed dependent on the factor of variation used for the grouping of the measurements, the means of the different groups should differ significantly. If the presumed dependency is not present, the different groups should all be realizations of the same distribution. The ANOVA test gives the probability p_0 that all groups contain realizations from the same distribution. Therefore, lower values of p_0 indicate that it is more likely that there is a dependency between the suspected factor of variation and the parameter of interest. The value of p_0 is included with all figures in this section that investigate these dependencies.

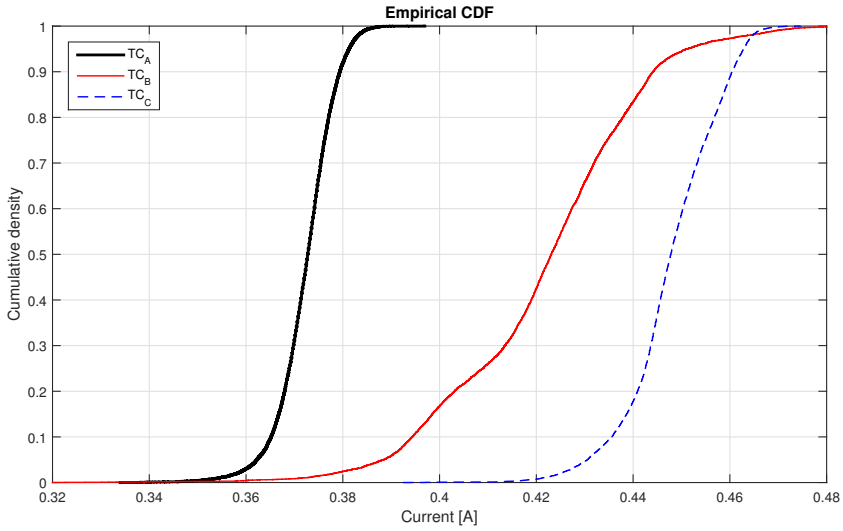


Figure 2-5: Cumulative density distributions of I_{high} per track circuit.

Table 2-2: Current values per track circuit based on 80% probability density.

Track circuit	TC_A	TC_B	TC_C
I_{high} lower bound [A]	0.365	0.394	0.436
I_{high} upper bound [A]	0.379	0.444	0.460
I_{low} lower bound [A]	0.015	0.019	0.014
I_{low} upper bound [A]	0.022	0.023	0.021

2-4-1 High and low current values

The most important parameters in the model are the values of the current with no train in the section I_{high} and the value when the current is at its lowest I_{low}. To investigate the natural variation of these values, the fitted models of the form of (2-1) have been investigated. Some extreme values occur that are likely the result of undetected data corruptions or abnormal behavior of the track circuits. To get an idea of the normal behavior, only the 80% most common values are considered. For example, in Figure 2-5 the cumulative densities of the values of I_{high} are given. The lower and upper bounds that will be considered normal behavior are the values with a cumulative density of 0.1 and 0.9 respectively. Using this method the ranges of I_{high} and I_{low} are collected for all three track circuits and given in Table 2-2.

The values of I_{low} are very consistent for all three track circuits. Furthermore, over the period of a year 80% of the measurements differed at most by $7 \cdot 10^{-2}$ A. Since the measurement resolution is $6 \cdot 10^{-2}$ A, these values are presumed constant during normal behavior.

The values of I_{high} show more variation. There is a difference between the different track circuits which can be the result of the differently tuned resistances R_f and R_r. For each track circuit there is also variation between the different measurements. An explanation for these variations can be sought in Figure 2-1. The voltage source, the tuning of the track circuit and the impedance of the rails are presumed to be constant. Therefore, when no train is present this variation should be caused by the varying impedance of the ballast Zb. This value is

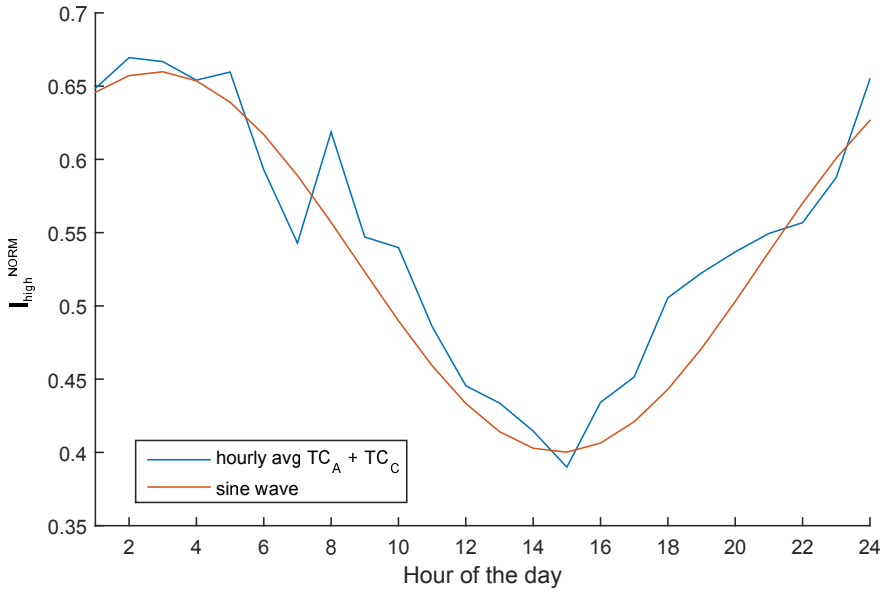


Figure 2-6: Dependency of I_{high} on the time of the day [$p_0 = 1.5 \cdot 10^{-2}$].

dependent on the environmental conditions.

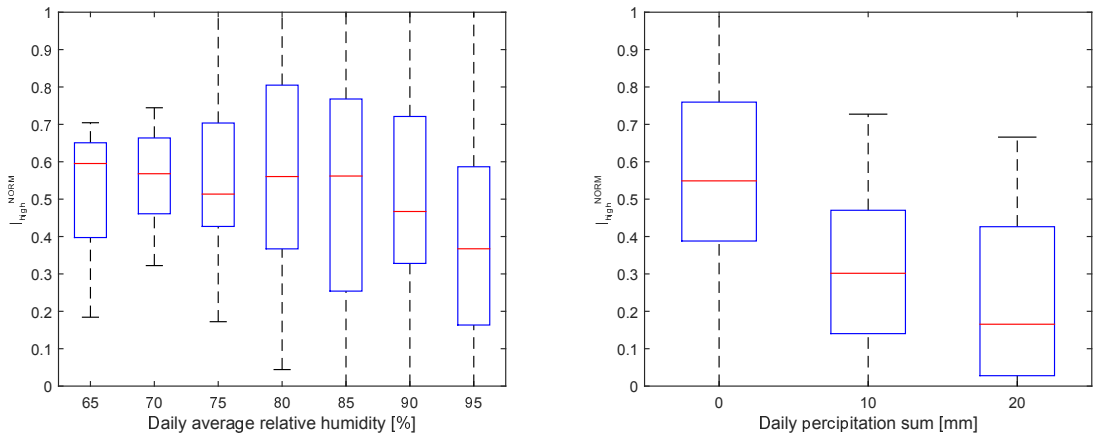
To investigate the dependencies that can explain the variations of I_{high} independently of the tuning of a particular track circuit the values have been normalized through:

$$I_{\text{high}}^{NORM} = \frac{I_{\text{high}} - I_{\text{high}}^{LOW}}{I_{\text{high}}^{HIGH} - I_{\text{high}}^{LOW}} \tag{2-2}$$

Where I_{high}^{LOW} and I_{high}^{HIGH} are the lower and upper bounds of the 80% most common values for the same track circuit, as given in Table 2-2. This way, the effects can be more easily compared and the extent to which a dependency explains the variation becomes more clear. Track circuits TC_A and TC_C showed similar responses to different environmental conditions but track circuit TC_B did not show any dependency on environmental conditions. This fact will be reflected in the generative model by giving each separate track circuit a stochastic sensitivity to environmental effects. In this section only the behavior of track circuits TC_A and TC_C is considered.

Since environmental conditions tend to change throughout the day, the normalized values of I_{high} are averaged per hour in Figure 2-6. It can be seen that the values of I_{high} follow a trend that can be approximated by a sine wave. The current with no train present is highest around 3 at night and lowest around 3 in the afternoon.

The environmental conditions at the times of the measurements can be approximated. For the two weather stations closest to the location of the track circuit, the daily weather data were downloaded for all days during the measurement period. Through linear interpolation based on the distance between the weather stations and the location of station Beilen, the daily environmental conditions that influenced the condition of the ballast have been approximated. In Figure 2-7 these conditions are compared to the daily average values of I_{high} for track



(a) Dependency on humidity [$p_0 = 2.4 \cdot 10^{-2}$]. **(b)** Dependency on precipitation [$p_0 = 4.1 \cdot 10^{-8}$].

Figure 2-7: Dependency of the normalized values of I_{high} on the environmental conditions. The values for I_{high} are normalized according to (2-2).

circuits A and C. It would seem likely that wet conditions would lead to a lower impedance of the ballast, leading to a lower value of I_{high} as more current flows between the rails through the wet ballast instead of through the receiver. This effect can indeed be seen from Figure 2-7. For high relative humidities the value of I_{high} becomes lower. When it rains this effect is even stronger.

2-4-2 Phase 2 parameters

The second part of (2-1) describes the current through the receiver from the train entering the section at t_1 until the current reaches its lowest point at t_2. This phase of the train passing events is fitted to the model:

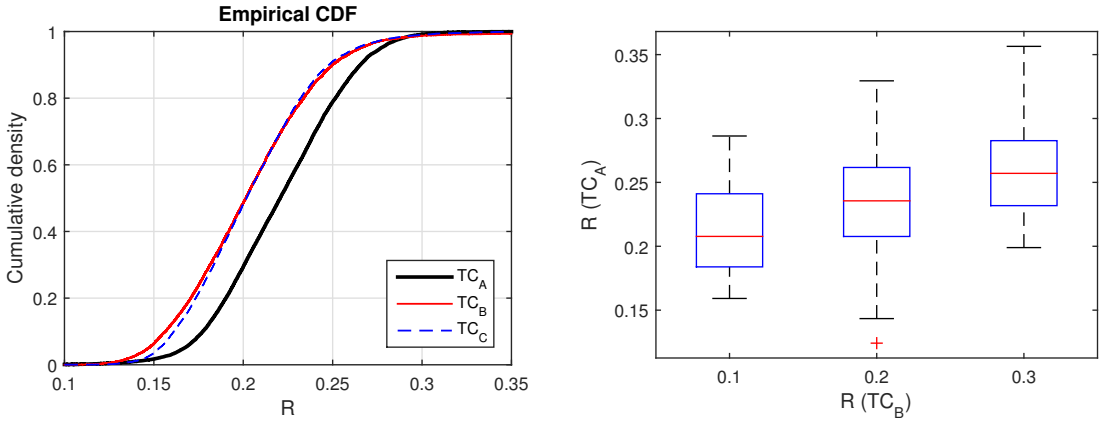
$$I_c(t) = I_{\text{low}} + \Delta I_{\text{max}} \left(\alpha_1 e^{-(t-t_1)/\tau_{\alpha_1}} + \beta_1 e^{-(t-t_1)/\tau_{\beta_1}} \right) \quad \text{for} \quad t_1 \leq t < t_2$$

When examining the fitted values for $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$, it was found that these parameters took similar values for most measurements in all three track circuits. To simplify the model these parameters are therefore taken as constants. Their values are taken as the average values for all 3 track circuits:

- $\tau_{\alpha 1} = 0.108\ s$
- $\tau_{\beta 1} = 0.588\ s$

Given the form of (2-1) it should hold that $\alpha_1 + \beta_1 = 1$. With $\tau_{\alpha 1}$ and $\tau_{\beta 1}$ fixed this leaves just one free variable for this phase. This variable will be called R and leads to a simplified model for this phase:

$$I_c(t) = I_{\text{low}} + \Delta I_{\text{max}} \left((1-R) e^{-(t-t_1)/\tau_{\alpha 1}} + R e^{-(t-t_1)/\tau_{\beta 1}} \right) \text{for} \quad t_1 \leq t < t_2 \tag{2-3}$$



(a) Cumulative density of R values per track circuit.

(b) R per train [$p_0 = 1.0 \cdot 10^{-9}$].

Figure 2-8: Distribution and dependencies of the R value in (2-3).

The quick step response of a properly working track circuit that shorts well once the first wheel set enters is represented by $\tau_{\alpha 1}$. The part represented by $\tau_{\beta 1}$ is much slower and presumably models the lowering of the train shunt resistance when more wheel sets gradually enter the section over time. This can be seen in Figure 2-1 as closing subsequent switches $s1, s2, s3, ...$ over time. With these two explanations of the time constants, the parameter R can be seen as a measure of the relative resistance of the train shunt S_x compared to the resistance of the path through the receiver. Higher values of R correspond to a slower drop in the current and thus a higher resistance of the path through the train.

The measured data have been refitted to equation (2-3). From Figure 2-8a the distributions of the R values for the three track circuits can be seen. It is clear that the values are track circuit dependent. This can be explained by the fact that R depends on the ratio between the resistance of the train shunt and the impedance of the path through the rails and the receiver. This second path includes the tunable resistance R_r.

Apart from the dependence on the track circuit tuning, R should also be dependent on the train properties. In theory, lightweight trains can have a higher resistance due to the fact that the contact between the wheels and the rails is less good than for heavy trains. To investigate this effect, in Figure 2-8b the R value in TC_A is compared to the value of R in TC_B when the same train passes. This is achieved by considering train passing events less that 60 seconds apart. The value of $p_0 = 1.0 \cdot 10^{-9}$ that is the result of the ANOVA test indicates that there is a dependency of R on the specific train passing event and therefore most likely on the train. However, this dependency fails to fully explain the variation of the parameter R.

2-4-3 Phase 3 parameters

The third part of (2-1) describes the current through the receiver from the lowest point at t_2 until the last wheel-set leaves the section at t_3. To the measured data samples of this phase a model of the following form has been fitted:

$$I_c(t) = I_{\text{low}} + \Delta I_{\max} \left(\alpha_2 e^{(t-t_2)/\tau_{\alpha 2}} + \beta_2 e^{(t-t_2)/\tau_{\beta 2}} \right) \quad \text{for} \quad t_2 \leq t < t_3$$

For a healthy track circuit the current could be expected to remain at the low level I_{low} during this time. However, as can be seen from Figure 2-3, this is not always the case. By analyzing the parameters of the curves that were fitted to the data of this phase, it is found that there are curves with positive, negative and constant derivatives. In track circuits TC_A and TC_B most of the measurements showed a mostly linear trend. This does not hold for most of the measurements of track circuit TC_C, which show an asymptotic step response like trend. In Figure 2-9 representative samples are shown from the three track circuits.

In spite of the fact that it does not accurately describe the current development in track circuit TC_C, a linear function was chosen as a simplified model for this phase. Using a linear function of time makes it possible to describe the behavior of the current with just two parameters. These are:

- t_2: The time at which the current starts to rise.
- ΔI_3: The fraction of $\Delta I_{\max}$ that the current has risen between t_2 and t_3, where t_3 is the moment the last wheel-set leaves the section.

Since with a good wheel-shunt ΔI_3 should be near 0, this value should be descriptive of the quality of the wheel-shunt. The simplified model of the current for this phase is now:

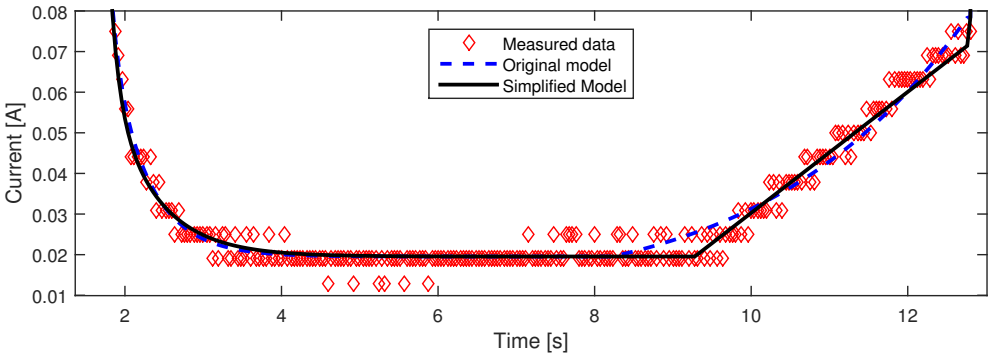
$$I_c(t) = I_{\text{low}} + \Delta I_{\max}\left((t - t_2)\frac{\Delta I_3}{(t_3 - t_2)}\right) \quad \text{for} \quad t_2 \leq t < t_3 \tag{2-4}$$

The cumulative densities of the values for ΔI_3 that were found by refitting the data to this simplified model are given in Figure 2-10.

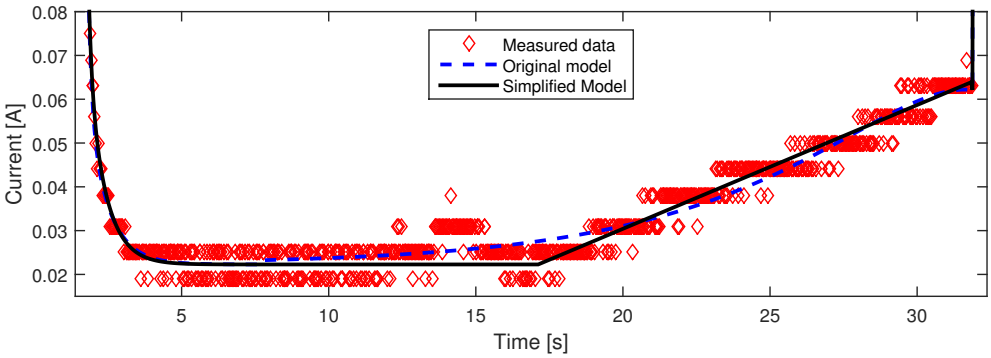
From Figure 2-1 it can be seen that it would only make sense for the current to start rising significantly when the front of the train has left the section and the rear of the train is completely in the section. If the length of the train is equal to the length of the section this point would be about halfway between the first wheel-set entering at $t = t_1$ and the last wheel-set leaving the section at $t = t_3$. For trains that are shorter or longer than the section this moment would be later. In Figure 2-11 the cumulative distributions of the normalized moment the current starts rising $(t_2 - t_1)/(t_3 - t_1)$ are given. For track circuits TC_B and TC_C the starting moment t_2 is indeed about halfway between t_1 and t_3 and for TC_A it is somewhat later. This makes it likely that t_2 can indeed be defined as the moment the last wheel set is in the section and the first wheel set leaves the section.

Since ΔI_3 should be related to the relative resistance of the wheel shunt compared to the path through the receiver, it should be dependent on the train. In Figure 2-12 the normalized values of ΔI_3 are given for TC_A compared to the values associated with the same train in TC_B. It can be seen that most of the variation of ΔI_3 is indeed dependent on the train.

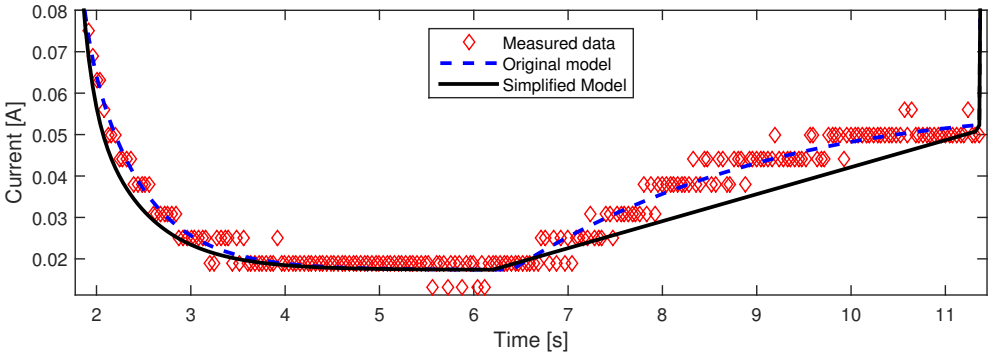
Since both R and ΔI_3 should be higher for trains with a high shunt resistance it is expected that there is a strong correlation between these parameters. In Figure 2-13 the normalized values of R are compared to the normalized values of ΔI_3 for the same train passings in TC_A. From this it appears that these parameters are completely *unrelated*. The ANOVA value of $p_0 = 0.54$ confirms this. For TC_B and TC_C the same lack of correlation was found. When comparing Figure 2-12 to Figure 2-8 it can be seen that ΔI_3 is more strongly related to a specific train than R. This makes it likely that ΔI_3 is descriptive of the train shunt resistance and the variation in R is related to other factors.



(a) Example from track circuit TC_A.



(b) Example from track circuit TC_B.



(c) Example from track circuit TC_C.

Figure 2-9: Fit of the simplified model for phase 2 and 3.

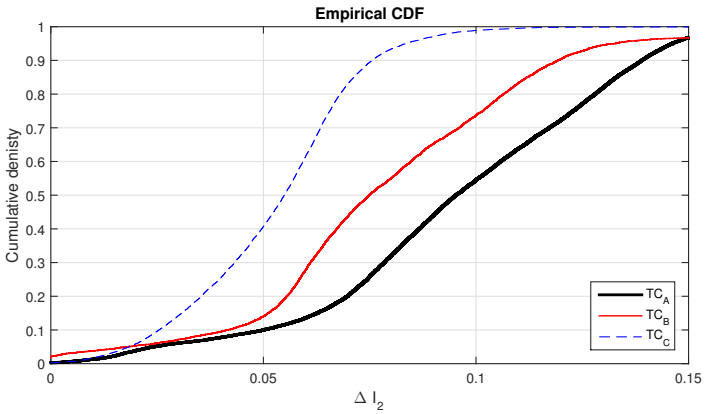


Figure 2-10: Cumulative densities of ΔI_3 per track circuit.

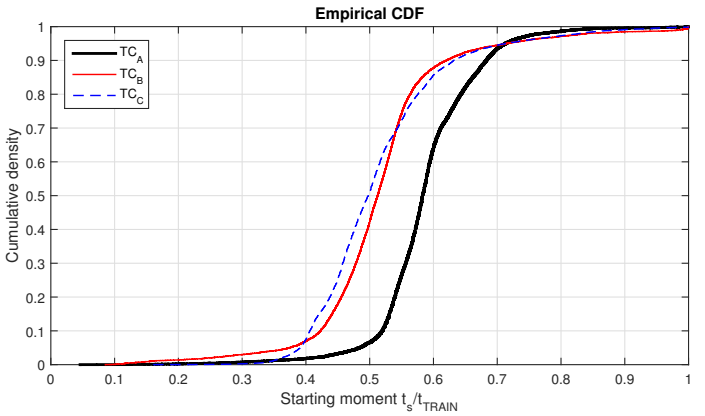


Figure 2-11: Normalized starting moment of the rising current $\frac{t_2-t_1}{t_3-t_1}$.

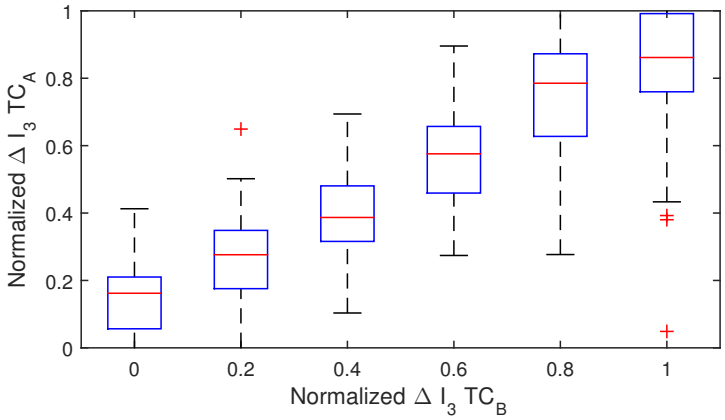


Figure 2-12: Dependency of ΔI_3 on the train [$p_0 = 5.3 \cdot 10^{-94}$].

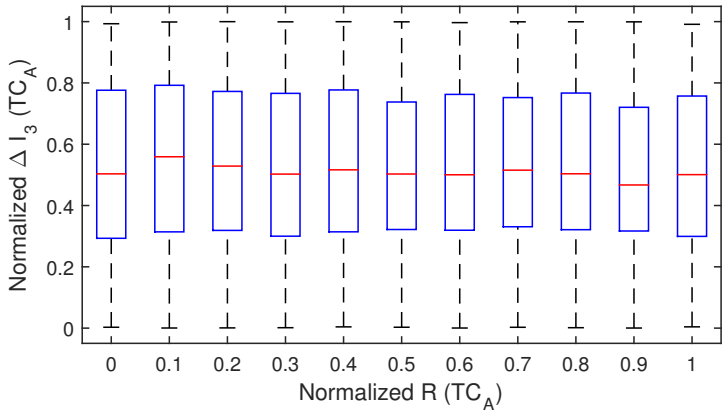
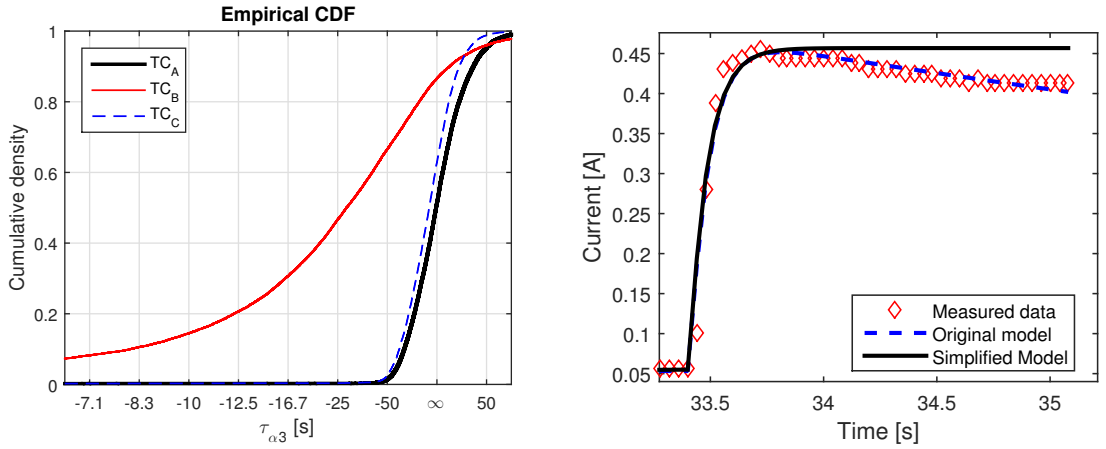


Figure 2-13: Relationship between ΔI_3 and R [$p_0 = 5.4 \cdot 10^{-1}$].



(a) Values of $\tau_{\alpha 3}$ in the original model.

(b) Simplified model misfit for $\tau_{\alpha 3} = -8\ s$.

Figure 2-14: Modeling of the fourth phase of the train passing event.

2-4-4 Phase 4 parameters

The fourth section of the model is the current step that takes place after the last wheel set of the train leaves the section. A model of the form:

$$I_C(t) = I_{\text{low}} + \Delta I_{\max}\left(\alpha_3 e^{(t-t_3)/\tau_{\alpha 3}} - \beta_3 e^{-(t-t_3)/\tau_{\beta 3}}\right) \quad \text{for} \quad t \geq t_3$$

has been fitted to all the data samples of this phase. The time constant $\tau_{\beta 3}$ represents the step and the time constant $\tau_{\alpha 3}$ represents the behavior after the step.

In Figure 2-14a the cumulative densities of the time constants $\tau_{\alpha 3}$ are given for the three track circuits. For track circuits A and B these time constants represent approximately a slight linear upwards or downwards trend after the initial step. Since the time constants are very large and distributed evenly around infinity (representing a constant) this part of the model is replaced with:

$$I_c(t) = I_{\text{low}} + \Delta I_{\max}\left(1 - e^{-(t-t_3)/\tau_3}\right) \quad \text{for} \quad t \geq t_3 \tag{2-5}$$

With $\tau_3 = \tau_{\beta 3}$. The values of $\tau_{\beta 3}$ for the fitted models of the three track circuits were examined. This revealed that for 80% of the values from track circuits TC_A and TC_C the 95% rise-time differed less than 0.1 seconds. For track circuit TC_B 80% of the 95% rise-time values were within 0.3 seconds. Additionally, none of the considered faults is expected to influence this parameter. Therefore it is likely that no relevant information about the condition of the track circuit is conveyed in this phase. As a result the time constant τ_3 is fixed at the average value:

- $\tau_3 = 0.080\ s$

For track circuit B it can be seen from Figure 2-14a that the assumption that $\tau_{\alpha 3} = \infty$ is a worse approximation than it is for the other two track circuits. The current in track circuit TC_B tends to drop down after the initial step up. In Figure 2-14b the mismatch between the simplified model and the measurements is given for $\tau_{\alpha 3} = -8\ s$. For 90% of the measurements from track circuit TC_B and almost all of the measurements from track circuits TC_A and TC_C the mismatch is less than that represented in Figure 2-14b. Based on this observation it seems that the simplified model of (2-5) fits the data well enough.

2-5 Generating synthetic data

So far, real measured data has been used to create a simple model that describes the behavior of the measurements from a railway track circuit. In this section it will be discussed how this model will be used to create synthetic data. The generative model will be used to generate data sets that contain many input *sequences* for neural networks, where each sequence contains current measurements $I_c(t)$ from many train passing events $\mathcal{T}$ in several track circuits in a geographic area. For each of the *sequences* the properties of the track circuits will be different, as it has been found in the previous section that each track circuit behaves differently and the eventual condition monitoring system should work irrespective of which specific track circuit it is monitoring.

The simplified model that describes the current $I_c(t)$ during a single train passing event in a single track circuit is:

$$I_c(t) = I_{\text{low}} + \Delta I_{\max} \begin{cases} 1 & \text{for} \quad t < t_1 \\ (1-R)e^{-(t-t_1)/\tau_{\alpha 1}} + Re^{-(t-t_1)/\tau_{\beta 1}} & \text{for} \quad t_1 \leq t < t_2 \\ (t-t_2)\frac{\Delta I_3}{(t_3-t_2)} & \text{for} \quad t_2 \leq t < t_3 \\ 1 - e^{-(t-t_3)/\tau_3} & \text{for} \quad t \geq t_3 \end{cases} \tag{2-6}$$

With the following values for the time-constants:

- $\tau_{\alpha 1} = 0.108\ s$
- $\tau_{\beta 1} = 0.588\ s$
- $\tau_3 = 0.080\ s$

And $\Delta I_{\max} = I_{\text{high}} - I_{\text{low}}$. It is assumed that the value of $I_c(t)$ between two train passings will follow a linear trend from the value of I_{high} during the first train passing to the value of I_{high} during the second train passing. The difference between these values is expected to be minimal. As a result, for each track circuit, the (synthetic) current value $I_c(t)$ can be calculated at any time if the values of $I_{\text{low}}, I_{\text{high}}, R, \Delta I_3, t_1, t_2$ and t_3 are known for the train passing events. The time values t_1, t_2 and t_3 are defined in the generative model as:

- t_1: the moment the first wheel set enters the section.
- t_2: the moment the first wheel set has left the section and the last wheel set is already in the section.
- t_3: the moment the last wheel set leaves the section.

The speed of the trains is presumed constant while they pass through the section, which makes the calculation of the times t_1, t_2, t_3 relative to the start time of a train passing event trivial:

$$t_1 = 0 \tag{2-7}$$

$$t_2 = \begin{cases} L_S/V_T & \text{if} \quad L_T \leq L_S \\ L_T/V_T & \text{if} \quad L_T > L_S \end{cases} \tag{2-8}$$

$$t_3 = (L_T + L_S)/V_T \tag{2-9}$$

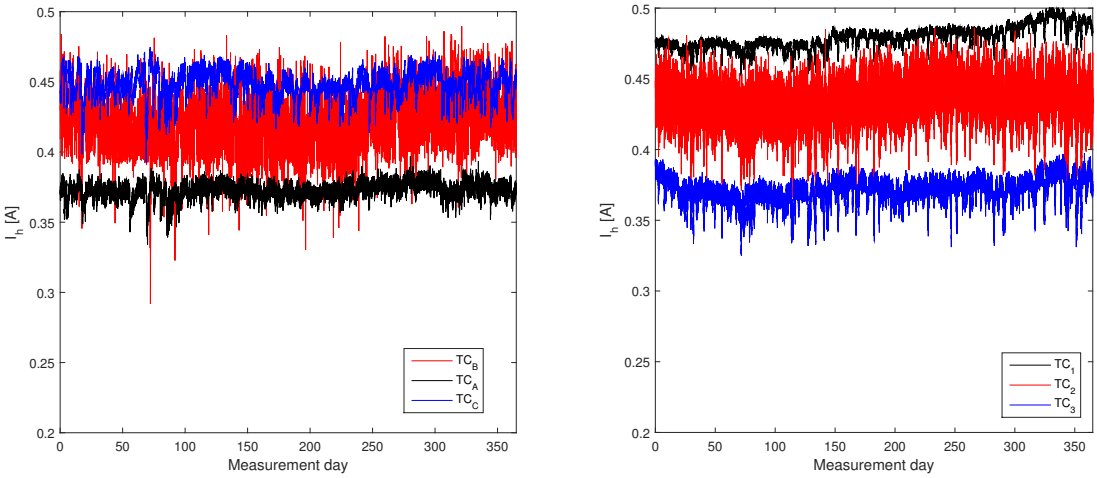
Where L_S is the length of the section, L_T is the length of the train and V_T is the velocity of the train. The generation of the values of the other variables is discussed in the next section.

2-5-1 Probabilistic parameter values

In Section 2-4 the dependencies of the model variables I_{high}, I_{low}, R and ΔI_3 on measurable quantities have been investigated. These dependencies will be used in the generative model. However, the dependencies found only partly explain the variation over time of these parameters that is observed in the measured data. The aim of the generative model is to generate synthetic data with similar variation as the measured data. This way, the ability of a neural network to learn to distinguish trends in the data caused by faults from normal variations in the data can be investigated. To test the learning ability of the network, generating data that have a realistic amount of variation is more important than making sure the values are based on well understood dependencies, or even the correctness of the current values generated by the model. Therefore, several sources of random variation will be added to the model.

2-5-2 Generating I_{high}

By inspecting Figure 2-15a it can be seen that there are a number of properties that determine the value of I_{high} that differ per track circuit. Not only does the average value of I_{high} differ per track circuit, but also the amount of variation of I_{high} over short- and over long time-periods is track circuit dependent. Finally, it can be seen that there are trends that occur in all track circuits with different intensities. To replicate this behavior several sources of



(a) Measurement data. **(b)** Synthetic data sample.

Figure 2-15: Values of I_{high} over the period of a year with no faults.

random variations are added and the sensitivity of the track circuits to these sources are drawn from probability distributions.

The probability distributions that the track circuit parameters are sampled from are chosen based on the assumptions that the three track circuits in the available dataset all represent normal behavior and together span most of the range that defines normal behavior. Therefore, the track circuit parameters are drawn from distributions that are chosen in such a way that the behaviors of the three track circuits in the dataset are likely realizations of those distributions.

The value of I_{high} is generated by the model for every train passing event $\mathcal{T}$and every track circuit TC according to the following formula:

$$I_{\text{high}}(TC,\mathcal{T}) = I^N_{\text{high}}(\text{TC}) + I^V_{\text{high}}\Big(W(t)\mathcal{S}_w(TC) + T(\mathcal{T})\mathcal{S}_T(TC) + \mathcal{R}_1(\mathcal{T})\mathcal{S}_{R1} + \mathcal{R}_2(\mathcal{T})\mathcal{S}_{R2}\Big) + \mathcal{S}_W\Big(\mathcal{R}_3(t) + \mathcal{R}_4(\mathcal{T})\Big) + \Delta I^F_{\text{high}} \quad (2\text{-}10)$$

With the following parameters:

- $I^N_{\text{high}}(TC)$: The nominal value of I_{high} for the given track circuit. This value is determined when the track circuit is created and does not change over time.
- $I^V_{\text{high}}(TC)$: The amount of short-term variation of the particular track circuit per train passing. In Figure 2-15a it can be seen that in the given dataset the value of I_{high} varies a lot more per data point for TC_B than it does for the other two track circuits. It can also be seen that the amount of short-term variation is not necessarily related to the amount of long-term variation.
- $W(\mathcal{T})$: The short-term environmental influences. Primarily the wetness of the ballast. This value is calculated as described in Section 2-5-3.

- $T(\mathcal{T})$: The sinusoidal influence of the time of the day on the value of I_{high} as shown in Figure 2-6.
- $\mathcal{S}_x(TC)$: Denotes the sensitivity of the track circuit to changes in variable x. For example, the sensitivity of the current I_{high} on the weather conditions $\mathcal{S}_W(TC)$ might be dependent on the properties of the ballast between the rails. The sensitivities to the different sources of variation are all presumed independent for simplicity. In reality this is not likely to be the case.
- $\mathcal{R}_1(\mathcal{T}), \mathcal{R}_2(\mathcal{T})$: Short term random effects. These are drawn from their probability density functions for every train passing event. The value of $\mathcal{R}_1(\mathcal{T})$ is unique for every track circuit and the value of $\mathcal{R}_2(\mathcal{T})$ is shared among all track circuits in the area.
- $\mathcal{R}_3(\mathcal{T}), \mathcal{R}_4(\mathcal{T})$: Long term (seasonal) random effects. These values are used to imitate the long-term trends found in Figure 2-15a. The value of $\mathcal{R}_3(\mathcal{T})$ is track circuit specific and that of $\mathcal{R}_4(\mathcal{T})$ is shared among all track circuits in the area. Since these long-term effects are presumed to be seasonal, their influence is determined by the track circuits' sensitivity to the environmental conditions $\mathcal{S}_W(TC)$. The values are updated during every train passing by adding a new random number to the bias.
- ΔI^F_{high}: The bias of I_{high} due to the presence of a fault in the track circuit. A failure is defined in the model as I_{high} being lower than 0.2 A. The severity of a fault $F(\mathcal{T})$ ranges from 0 (no fault) to 1 when the fault causes a failure. ΔI^F_{high} is calculated as: $\Delta I^F_{\text{high}}(\mathcal{T}) = F(\mathcal{T})(0.2 - I^N_{\text{high}}(TC))$. The development of $F(\mathcal{T})$ for different faults is discussed in Section 2-5-7

The track circuit properties are sampled from the distributions given in Table 2-3. The values of the short-term random variation sources are drawn from the distributions given in Table 2-4. The values of the updates to the long-term variance biases are drawn from the distributions described in Table 2-5.

2-5-3 Generating $W(t)$

The (short-term) environmental influence $W(t)$ represents the influence of the weather on the ballast impedance. In Section 2-4-1 the relative humidity and precipitation were identified as causes of this variation. However, the relative humidity is highest during the night and lowest during the day which would lead to a value of I_{high} that is lowest during the night and highest during the day. This directly contradicts Figure 2-6. Therefore, in the generative model only precipitation is taken into account.

Based on Figure 2-7b light and heavy rain are defined. Whether there is rain is decided per day. Based on data from the Dutch meteorological institute the chance of a day with rain is defined as: 140/365. The chance of a rainy day being a day with heavy rain is 25/140.

For every train passing event the variable $W(\mathcal{T})$ is updated according to:

$$W(\mathcal{T}) = 0.85W(\mathcal{T}-1) + \min(P(\mathcal{T}), 0) \tag{2-11}$$

Where $P(\mathcal{T})$ is the added precipitation influence at train passing event $\mathcal{T}$, which is drawn from the probability function given in Table 2-4 based on the presence of light or heavy rain

Table 2-3: Track circuit properties. For each created sequence of train passing events the properties of the considered track circuits are drawn once from these uniform or normal distributions.

	Uniform distribution		Normal distribution	
Variable	min	max	μ	σ
I^N_{high}			$4.2 \cdot 10^{-1}$	$2.5 \cdot 10^{-2}$
I^V_{high}	$2.5 \cdot 10^{-2}$	$7 \cdot 10^{-2}$		
I^N_{low}			$2 \cdot 10^{-2}$	$1.7 \cdot 10^{-3}$
I^V_{low}	$1 \cdot 10^{-3}$	$3 \cdot 10^{-3}$		
R^N			$2 \cdot 10^{-1}$	$1 \cdot 10^{-2}$
R^V			$1.8 \cdot 10^{-1}$	$1 \cdot 10^{-2}$
ΔI^N_3	$1 \cdot 10^{-1}$	$2.5 \cdot 10^{-1}$		
ΔI^V_3			$4 \cdot 10^{-2}$	$5 \cdot 10^{-3}$
$\mathcal{S}_W$			$8 \cdot 10^{-1}$	$1.67 \cdot 10^{-1}$
$\mathcal{S}_{\text{train}}$			$1 \cdot 10^{-1}$	$1.67 \cdot 10^{-1}$
$\mathcal{S}_{\text{T}}$			$1 \cdot 10^{-1}$	$2 \cdot 10^{-2}$
$\mathcal{S}_{\mathcal{R}1}$			$1.5 \cdot 10^{-1}$	$1 \cdot 10^{-1}$
$\mathcal{S}_{\mathcal{R}2}$			$5 \cdot 10^{-2}$	$8.3 \cdot 10^{-2}$

Table 2-4: Short term random variation parameter distributions. For each train passing event $\mathcal{T}$the values of the variation sources are drawn from these normal distributions.

Variable	μ	σ
$\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_5, \mathcal{R}_6$	0	$5 \cdot 10^{-1}$
$\mathcal{R}_{10}$	0	$6 \cdot 10^{-2}$
$\mathcal{R}_{14}$	0	$1 \cdot 10^{-3}$
$P_{\text{light rain}}$	$-1 \cdot 10^{-2}$	$8 \cdot 10^{-2}$
$P_{\text{heavy rain}}$	$-6 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$

Table 2-5: Long term random variation parameter distributions. for each train passing the values of the bias terms are updated by adding a sample from these uniform distributions.

Variable	min	max
$\mathcal{R}_3, \mathcal{R}_4$	$-1.5 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$
$\mathcal{R}_7, \mathcal{R}_8$	$-3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$
$\mathcal{R}_9$	$-1.5 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$
$\mathcal{R}_{11}, \mathcal{R}_{12}$	$-2.5 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$
$\mathcal{R}_{13}$	$-1.5 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$
$\mathcal{R}_{15}, \mathcal{R}_{16}$	$-2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$

on the given day. Note that a 'train passing event' $\mathcal{T}$is used as a measure of time in the model. The reason for this is given is Section 2-6.

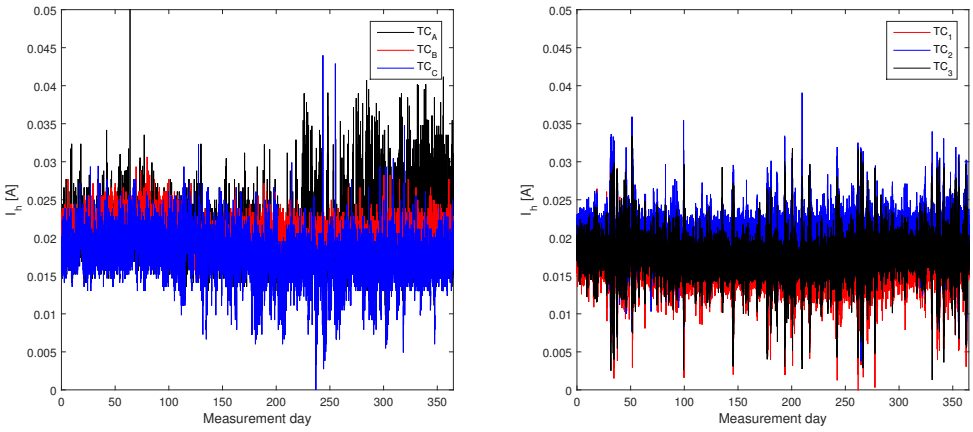
2-5-4 Generating I_{low}

The value for I_{low} is determined per train passing event for every track circuit according to:

$$I_{\text{low}}(\text{TC, t, train}) = I^N_{\text{low}}(\text{TC}) + I^V_{\text{low}}(TC)\Big(\mathcal{S}_{\text{train}}(TC)L(\text{train}) + \mathcal{R}_5(t) + \mathcal{R}_6(t)\Big) + \Big(\mathcal{R}_7(t) + \mathcal{R}_8(t) + W(t)\mathcal{R}_9(t)\Big) + \Delta I^F_{\text{low}} \quad (2\text{-}12)$$

The first part of this equation deals with the short term variance. The variable $L(\text{train})$ represents the lightness of the train, with lighter trains causing higher values of I_{low} due to the corresponding higher value of the train shunt resistance S. The weight of the train is drawn from a probability distribution for every train. The probability distribution has a different mean value for every hour of the day, with the mean of the weight being lower at night and higher during rush hours. The variables $\mathcal{R}_5(t)$ and $\mathcal{R}_6(t)$ are the short term variation random variables for the track circuit and the area respectively. Similarly, $\mathcal{R}_7(t)$ and $\mathcal{R}_8(t)$ are the long term random variation variables.

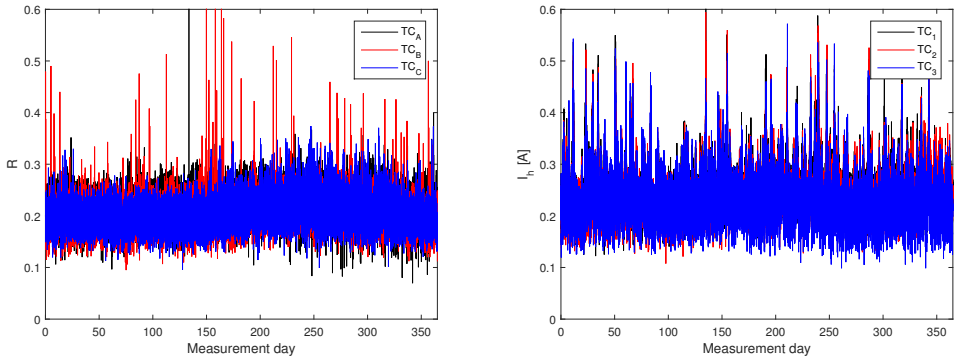
The term $W(t)\mathcal{R}_9(t)$ imitates the spikes visible in Figure 2-16a. These spikes happen only on occasion, roughly as often as there is rain. Therefore the soil dampness term is multiplied by a random number from a uniform distribution centered around zero, causing both positive as well as negative spikes. Here again replicating the variation in the data is favored over the psychical correctness of the model. In reality the condition of the ballast should have little effect on I_{low} as the electrical path through the wheel sets has a significantly lower resistance than the path through the ballast.



(a) Measurement data. **(b)** Synthetic data sample.

Figure 2-16: Values of I_{low} over the period of a year with no faults.

The term ΔI^F_{low} represents the bias of the current due to the presence of a fault in the track circuit. A failure is defined in the model as I_{low} being higher than 0.2 A. The severity of a



(a) Measurement data. **(b)** Synthetic data sample.

Figure 2-17: Values of R over the period of a year with no faults.

fault $F(t)$ ranges from 0 (no fault) to 1 when the fault causes a failure. ΔI^F_{low} is calculated as: $\Delta I^F_{\text{low}}(t) = F(t)(0.2 - I^N_{\text{low}}(TC))$.

2-5-5 Generating R

The value for R is determined per train passing event for every track circuit according to:

$$R(\text{TC, t, train}) = R^N(\text{TC}) + R^V(TC)\Big(\mathcal{S}_{\text{train}}(TC)L(\text{train}) + \mathcal{R}_{10}(t)\Big) + \Big(\mathcal{R}_{11}(t) + \mathcal{R}_{12}(t) + W(t)\mathcal{R}_{13}(t)\Big) + 2\Delta I^F_{\text{low}} \quad (2\text{-}13)$$

With $\mathcal{R}_{10}$ the short term track circuit specific random variation and $\mathcal{R}_{11}$ and $\mathcal{R}_{12}$ the long term track circuit specific and area variation respectively. The parameter $\mathcal{R}_{13}$ has been added for the same reason as $\mathcal{R}_9$ in (2-12). Here $\mathcal{R}_{13}$ is always positive to replicate the upwards spikes seen in Figure 2-17a.

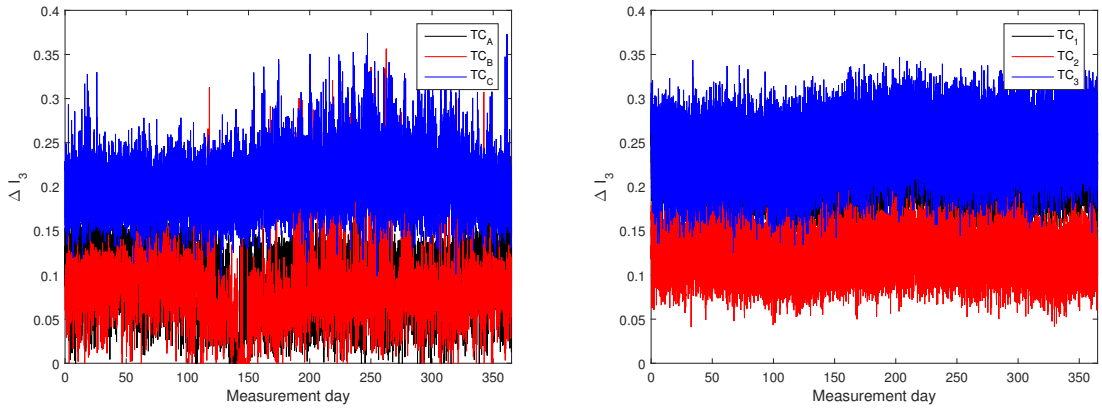
The value of ΔI^F_{low} will be around 0.2 when a fault is present in the track circuit that prevents a good train shunt. When this is the case R should be higher as well. Therefore the term $2\Delta I^F_{\text{low}}$ is added to ensure that the value of R is around 0.6 for faulty track circuits.

2-5-6 Generating ΔI_3

The value for ΔI_3 is determined per train passing for every track circuit according to:

$$\Delta I_3(\text{TC, t, train}) = \Delta I_3^N(\text{TC}) + \Delta I_3^V(TC)\Big(\mathcal{S}_{\text{train}}(TC)\ 4L(\text{train}) + \mathcal{R}_{14}(t)\Big) + \Big(\mathcal{R}_{15}(t) + \mathcal{R}_{16}(t)\Big) + 2\Delta I^F_{\text{low}} \quad (2\text{-}14)$$

This equation has a similar structure to (2-13) apart from the larger emphasis on the weight of the train. This is in accordance with Section 2-4-3 where it was found that ΔI_3 is more strongly related to a specific train than R.



(a) Measurement data. **(b)** Synthetic data sample.

Figure 2-18: Values of ΔI_3 over the period of a year with no faults.

2-5-7 Generating faults

The fault intensity $F(t)$ is generated based on a slightly changed version of the behaviors described in [2]. For the generated time sequences the track circuits are always healthy at the start of the experiment. There is at most 1 fault present in the sequence, which starts developing after an initial fault free period has passed. The length of this fault free period is sampled from a uniform distribution of between 0.1 and 0.3 times the length of the complete sequence. This is done to introduce variation in the generated data while ensuring that most faults will be noticeable before the end of the sequence. There are three types of spatial dependencies considered in the model. These are:

- D_1: Only one track circuit is affected by the fault.
- D_2: The fault affects all track circuits on one of the two tracks.
- D_3: The fault affects all track circuits on both tracks.

When a fault has more than one possible spatial or temporal dependency listed, one of those is chosen at random for every sequence generated with that fault. The following fault types are implemented in the model:

- **Rail contamination:** In the model it is assumed that the rail contamination is caused by leaves falling on the rails during autumn. The intensity of the problem is presumed to follow the first half of a sine wave, with the problem starting when the first leaves start to fall, intensifying until the most leaves per day fall in the middle of autumn and then decreasing again until there are no more leaves on the trees to fall. Since the leaves on the tracks form an insulating layer that increases the resistance of the electrical path through the wheel sets of the trains this problem influences $\Delta I^F_{\text{low}}(t)$.
  - Spatial dependency: D_3.
  - Length of autumn period in days: drawn from a normal distribution with: $\mu = 50, \sigma = 5$.

  - Maximum fault intensity: drawn from a normal distribution with: $\mu = 0.3, \sigma = 0.1$.

- **Insulated joint defect:** When the joints that ensure the electrical insulation between different track circuits wear out, the track circuit current from one track circuit could leak to a neighboring track circuit. The track circuits have been designed in such a way that this would not cause the measured current in the track circuit that it leaks into to increase. However, it would cause the current in the track circuit that it leaks out of to decrease. As such, this fault influences $\Delta I^F_{\text{high}}(t)$ for that one track circuit. As this problem is caused by the wear of the joints it is presumed to increase either linearly or exponentially with every passing train. The amount of trains that need to pass over the track circuit before the fault becomes a failure is drawn from a probability distribution.
  - Spatial dependency: D_1.
  - Temporal dependency: linear $\vee$ exponential.
  - Train passings before the fault leads to a failure: drawn from a normal distribution with $\mu = 3500, \sigma = 250$.

- **Conductive object:** A conductive object that is placed on the tracks can create a partial bypass of the insulated joints. Therefore the effect of this fault is similar to that of the insulated joint defect. This fault case differs from the last in the sense that the fault happens abruptly and the intensity does not change over time once the fault has occurred.
  - Spatial dependency: D_1.
  - Temporal dependency: abrupt.
  - Fault intensity: drawn from a normal distribution with: $\mu = 0.5, \sigma = 0.2$.

- **Mechanical rail defect:** When a mechanical defect occurs in one of the rails this will increase the electrical resistance of that rail. Therefore this fault case influences $\Delta I^F_{\text{high}}(t)$ for that particular track circuit. The damage to the rail gets worse with every passing train. Additionally, the damage done to the rail will increase as the rail becomes more damaged. As a result the fault intensity $F(t)$ progresses exponentially. This fault is presumed to lead to a failure more quickly than an insulated joint defect will. This ensures that a fault diagnosis method could at least theoretically distinguish between these faults.
  - Spatial dependency: D_1.
  - Temporal dependency: exponential.
  - Train passings before the fault leads to a failure: drawn from a normal distribution with $\mu = 1500, \sigma = 200$.

- **Electrical disturbances:** Electrical disturbances are caused by the fact that the traction currents from the trains also flow through the rails. This can saturate the rails which leads to a higher impedance for the track circuit currents and thus influences $\Delta I^F_{\text{high}}(t)$. This problem affects several track circuits along the same track as the traction currents are not hindered by the electrical insulation that separates the track circuits. The problem is intermittent as it would only occur when a train is drawing a lot of power.

  - Spatial dependency: D_2
  - Temporal dependency: intermittent; for every train passing event $\mathcal{T}$the probability of the problem occurring is: $5 \cdot 10^{-2}$.
  - Each time the problem occurs the fault intensity is drawn from a normal distribution with $\mu = 0.5, \sigma = 0.2$.

- **Ballast degradation:** The impedance of the ballast between the rails influences the value of I_{high} as more or less current will leak through the ballast instead of flow through the receiver. This impedance is influenced by the weather and varies slowly over time. Both of these effects are not considered faults. It is however possible that the ballast will continue to degrade slowly over time. This is considered a fault which affects $\Delta I^F_{\text{high}}(t)$.
  - Spatial dependency: $D_1 \vee D_2$.
  - Temporal dependency: linear $\vee$ exponential, with the number of days between the fault leads to a failure drawn from these normal distributions:
    - Linear: $\mu = 400, \sigma = 10$.
    - Exponential: $\mu = 380, \sigma = 10$.

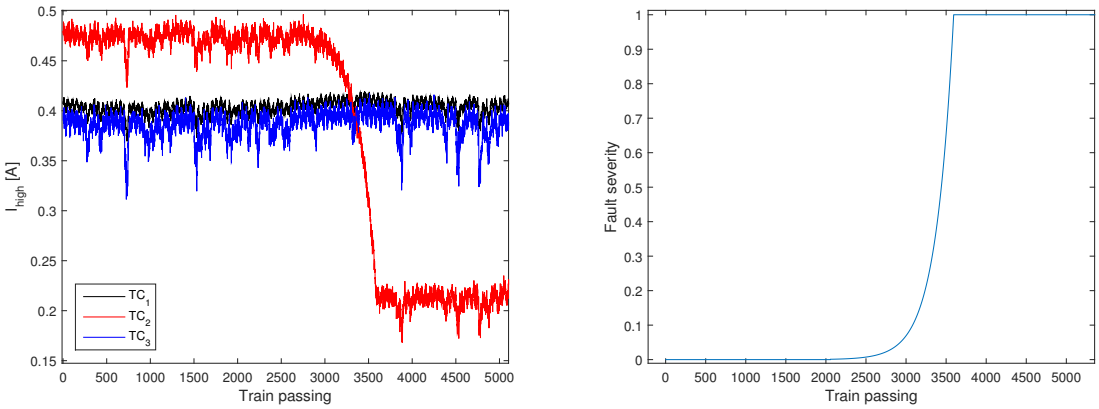
In contrast to [2], lightweight trains and ballast variation are not considered to be faults. These sources of variation are always present. In Figure 2-19 an example is given of a synthetic data sample with three track circuits in which one has a mechanical rail defect that exponentially deteriorates until it leads to a failure. A summary of the different fault types and their spatial and temporal dependencies is given in Table 2-6.

Table 2-6: Fault types and their spatial and temporal dependencies.

Fault type	Spatial dependency	Temporal dependency	Fault rate
Rail contamination	D_3	half sine	-
Insulated joint defect	D_1	linear $\vee$ exponential	intermediate
Conductive object	D_1	abrupt	-
Mechanical rail defect	D_1	exponential	high
Electrical disturbance	D_2	intermittent	-
Ballast degradation	$D_1 \vee D_2$	linear $\vee$ exponential	low

2-6 Sampling strategy

The development of faults is a process that occurs slowly. This means that there are very-long-term temporal dependencies in the data. Learning algorithms can struggle with these long-term dependencies so it is important to sample in an effective way to increase the information density of the signal that is the input of the learning algorithm. Although the current $I_c(t)$ is the only available input, the variables $I_{\text{high}}, I_{\text{low}}, R$ and ΔI_3 are more directly influenced by the fault intensity. The current $I_c(t)$ will therefore be sampled such that it is maximally descriptive of these model variables. The relationship between these variables and the current $I_c(t)$ is given in (2-6). Based on this, the current $I_c(t)$ is sampled at the following times during a simulated train passing:



(a) Unoccupied track currents I_{high}.

(b) Fault severity $F(\mathcal{T})$.

Figure 2-19: Syntactic data sample with a exponentially developing mechanical defect in TC_2.

- $t = t_1$: As the train is about to enter the section $I_c(t) = I_{\text{high}}$.
- $t = t_1 + 0.35s$: At about 0.35 seconds after the first wheel set of the train enters the section the value of the current $I_c(t)$ is influenced most strongly by the variable R.
- $t = t_2$: By sampling at the moment the current $I_c(t)$ is at its lowest the value of I_{low} can be determined.
- $t = t_3$: Just as the last wheel set of the train is about to leave the section the current $I_c(t)$ will have become equal to $I_{\text{low}} + \Delta I_{\text{max}} \Delta I_3$.

Note that only sampling the current at these four time instances will mean that the information that is gathered on one train passing is susceptible to noise. However, since the temporal dependencies will develop over a period of many train passing events, this noise can be compensated for when considering a time series of many train passing events.

To further optimize the way the synthetic data are used as the input of a neural network the following simplification is made: Each day there are N_P train passing events. These train passing events are spread out evenly over the day and during such an event there are two trains that pass the considered track circuits; one for each track. This results in the four current samples discussed above in all track circuits for one combined train passing event $\mathcal{T}$. By sampling in this way a neural network can be used that has $4 \times N_C$ inputs, where N_C is the number of considered track circuits in the geographic area. Having the whole train passing event as one input time-step will improve the chances of the network to learn the long-term temporal dependencies.

Chapter 3

Artificial Neural Network theory

An Artificial Neural Network (ANN) is a network of computing units (*artificial neurons*), the design of which was inspired by the working of the brain [6]. The similar parallel computation style that the ANNs share with their biological counterparts make them good at tasks that brains are also good at, such as pattern recognition. Neural networks are very expressive models that in theory and given enough data, can learn any complex non-linear mapping from their inputs to their outputs. This ability makes them an attractive option for the condition monitoring of track circuits, since they might be able to learn to represent the relationship between the measurement signals and the condition of a track circuit more accurately than it is currently understood.

In this chapter, the theory behind artificial neural networks is discussed. The focus of this discussion is on the neural network theory that is relevant to the fault diagnosis problem considered in this thesis. Additionally, the reasoning behind the choices for the general network structure and training algorithms is discussed. This chapter starts in Section 3-1 with a description of the artificial neuron, which is the basic building block of a neural network. In Section 3-2 the ways in which artificial neurons can be combined into a network are given and the motivation is given for the use of recurrent neural networks in this thesis. In Section 3-3 the procedure by which neural networks learn from data is explained. Additionally, this section explains the reason why recurrent neural networks have traditionally struggled to learn the longterm temporal dependencies that are present in the track circuit case. The solution to this problem that is used in this thesis is the use of the Long Short-Term Memory network architecture, which is detailed in Section 3-4. In Section 3-5 the concept of end-to-end learning is discussed. This concept has enabled state of the art performance in the object recognition and speech recognition domains. In this section it will be discussed how and why this concept could also be beneficial in the condition monitoring domain.

In Chapter 4, the exact form of the networks used in this thesis is described and the results of using these networks with the data from the generative model is presented.

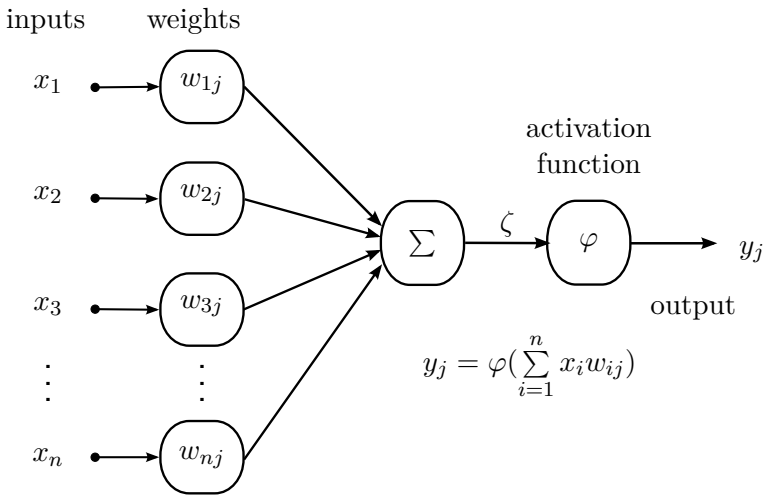


Figure 3-1: Artificial neuron.

3-1 The artificial neuron

The basic building block of an ANN is the *artificial neuron*. In Figure 3-1 the structure of such an artificial neuron *unit* is shown. The unit has n inputs $x_1, ..., x_n$. These inputs x_i are multiplied by a weight w_{ij}, where i is the input index and j is the index of the destination unit. The output (or *activation*) y_j of unit j is then calculated by applying an activation function φ to the sum of these weighted inputs:

$$y_j = \varphi(\sum_{i=1}^{n} w_{ij} x_i) \tag{3-1}$$

The argument of the activation function can also include a bias term. However, in much of the literature (e.g. [6]) this bias term is taken into account by adding an input to the unit that is always equal to 1. The bias term is then simply replaced by the weight on the connection from this input.

Activation functions

In the networks used in this thesis, artificial neuron units with different activation functions $\varphi(\zeta)$ are used. These functions are:

Linear

This activation function is simply a linear function of the inputs. Since the offset and the scaling can all be determined by the weights to the unit, the activation function is just the identity in this case.

$$\varphi(\zeta) = \zeta \tag{3-2}$$

Sigmoidal

A commonly used activation function is the (logistic) sigmoidal activation function which saturates to 0 or 1 for low and high values of ζ respectively and has non-linear behavior in between.

$$\varphi(\zeta) = \frac{1}{1 + e^{-\zeta}} \tag{3-3}$$

Hyperbolic tangent

The hyperbolic tangent activation function is similar to the sigmoidal activation function, but with the output ranging from -1 to 1.

$$\varphi(\zeta) = \frac{e^{2\zeta} - 1}{e^{2\zeta} + 1} \tag{3-4}$$

Softmax

The output of a softmax unit j is normalized with other units k in a group such that the cumulative output of the group is 1. This is useful when the group of units represents probabilities of mutually exclusive classes, which is the case in the fault isolation problem.

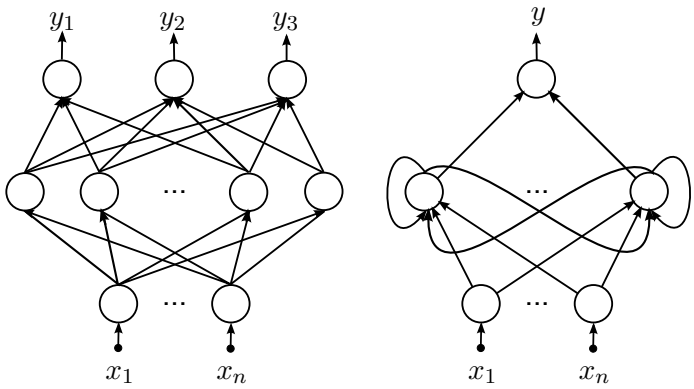
$$\varphi_j(\zeta) = \frac{e^{\zeta_j}}{\sum_k e^{\zeta_k}} \tag{3-5}$$

3-2 Network structure

A (large) number of the artificial neuron units can be combined into an ANN. A network consists of a layer with one or more inputs, often one or more *hidden* layers with each one or more units and usually an output layer with one or more units. Based on the connections between the units in the layers, two main types of networks can be distinguished [6].

Feed-forward Neural Networks

A Feed-forward Neural Network (FNN) is a network in which the outputs of one layer are used as the inputs of the next layer. There are no connections between the units of the same layer and no output connections to the inputs of a previous layer. An example of a FNN can be seen in Figure 3-2a. A network in which all outputs are connected to all inputs of the next layer is called *fully connected*. Even though networks are often fully connected they can still have many of their connection weights equal to zero, effectively breaking the connection between two neurons.



(a) Feed-forward. **(b)** Recurrent.

Figure 3-2: ANN structure types. Each circle represents an artificial neuron unit as depicted in Figure 3-1.

Recurrent Neural Networks

In a Recurrent Neural Network (RNN), the restrictions on the connections that are present in a FNN do not apply. The output of any unit can be an input to any other unit (or itself). In a fully connected RNN all outputs of all units are connected to all inputs of all units. In Figure 3-2b a RNN is shown in which only the hidden layer is fully connected.

RNNs have, at least in theory, much more potential than FNNs, since their recurrent connections allow them to store memories. However, exploiting this potential presents problems [6] which will be discussed in Section 3-3-5. These problems have prevented RNNs from being widely adopted. Recently however, many successful applications of RNNs have been published [7]. The ability of a recurrent network to maintain an internal state that is descriptive of past inputs makes them very suitable for problems involving time-series. In this thesis a RNN will therefore be used. This choice is further motivated in Section 3-5.

3-3 Network training

A neural network with a properly chosen structure has the potential to solve very complex problems. However, unlocking this potential requires finding an adequate set of the weights W on all the input connections to all the neurons. This can be very difficult and the right strategy depends on the network architecture, the desired output of the network and the available data to train the network [6].

The networks in this thesis are trained with the Stochastic Gradient Descent (SGD) optimization procedure described in Section 3-3-1. The optimization procedure will change the parameters W of the network to minimize a *loss function* $\mathcal{L}(W)$. The loss functions used in this thesis are given in Section 3-3-2. To calculate the first order derivatives of the loss function with respect to the network parameters for a given input, the Back Propagation (BP) method given in Section 3-3-3 can be used. In this thesis, RNNs are used, which require an extension of the BP method called Back Propagation Through Time (BPTT). This method is described in Section 3-3-4.

3-3-1 Stochastic Gradient Descent

When a neural network is presented with an input x, it will produce an output y. This output y is a function of both the input x and the tunable parameters W of the network: $y = f(x, W)$. These tunable parameters are the weights w on the input connections to the neuron units in the network (See Figure 3-1). The aim of training the neural network is to find a set of weights W that will make the output y of the network, given any relevant input x, as close as possible to the corresponding desired output or *target* y^t. For each input and correct output pair $\{x, y^t\}$ the loss function $\mathcal{L}(y, y^t)$ gives the cost involved with predicting y instead of y^t. This cost is a measure of the dissimilarity of y and y^t. The loss functions used in this thesis are given in Section 3-3-2.

To train the networks in this thesis, a large *training dataset* is used with n input and target tuples $\{x_1, y_1^t\}...\{x_n, y_n^t\}$. The *empirical risk* E [8] is defined as the average loss over all the training tuples for a given network:

$$E = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\Big(y_i^t, f(x_i, W)\Big) \tag{3-6}$$

When the training dataset is large enough and the class of possible networks $f(W)$ is restricted enough, minimizing the empirical risk will also minimize the loss on tuples that are not in the training dataset. That is, the network is able to *generalize* [8]. By monitoring the performance according to (3-6) on a separate validation dataset that the network is not trained on, it is ensured that this is indeed the case and that the network does not *over-fit* on the training data.

The Gradient Descent (GD) algorithm minimizes (3-6) by iteratively using the following update equation for the weights W in the network:

$$W_{t+1} = W_t - \epsilon\frac{1}{n}\sum_{i=1}^{n}\nabla_W\mathcal{L}\Big(y_i^t, f(x_i, W)\Big) \tag{3-7}$$

where ϵ is the learning rate and $\nabla_W\mathcal{L}(y_i^t, f(x_i, W))$ is a vector containing the first order derivatives of the loss $\mathcal{L}(y_i^t, f(x_i, W))$ with respect to all of the weights w in the network.

Due to the large size of the training dataset, the computational cost of calculating the derivatives for all data points in the dataset for each update of the weights is prohibitively high. Therefore, instead of averaging gradients over the whole dataset, only a few randomly chosen data points are used for each weight update. This technique is called Stochastic Gradient Descent (SGD). Although the theoretical convergence guarantees are not as good as those of full GD, in practice this technique gives better results for large neural networks and large data sets and is therefore used in this thesis. The weight update equation now becomes:

$$W_{t+1} = W_t - \epsilon\frac{1}{n}\sum_{i=1}^{B}\nabla_W\mathcal{L}\Big(y_i^t, f(x_i, W)\Big) \tag{3-8}$$

where B is the number of data points that are considered in each weight update. The set of data points used for a weight update is called a *mini batch*.

3-3-2 Loss functions

In this thesis two different loss functions are used for the two different tasks the networks need to perform; *regression* and *classification*.

Regression

For the fault severity estimation sub-problem, the neural network output is a real number $y \in \mathbb{R}$. The loss function used in this case is the squared difference between the predicted fault severity y and the true fault severity y^t at that time-step:

$$\mathcal{L} = \frac{1}{2}(y - y^t)^2 \tag{3-9}$$

Classification

The fault isolation problem is a classification problem in which it is assumed that there is, at each time-step, only one correct answer. The loss function used is the negative log likelihood (cross-entropy) function:

$$\mathcal{L} = -\sum_j y_j^t \log y_j \tag{3-10}$$

where $j \in J$ are the considered classes and y_j^t is 1 for the correct class and 0 for the other classes. This loss function is used in combination with a softmax output layer (3-5), which ensures that the outputs for all classes sum up to 1. Therefore, the loss function can only be minimized by outputting 0 for all the incorrect classes and 1 for the correct class.

3-3-3 Back propagation

To calculate the derivative of the loss function with respect to the weights $\nabla_W \mathcal{L}$ in a FNN, the BP technique can be applied. The simple structure of a FNN makes it possible to repeatedly apply the chain rule to propagate the loss derivative from the output of the network backwards towards the inputs.

Initially, the derivatives of the loss function with respect to the outputs of the network are calculated. For the regression loss function (3-9) this derivative is:

$$\frac{\partial \mathcal{L}}{\partial y} = -(y_j^t - y_j) \tag{3-11}$$

For the classification loss function (3-10) this derivative is:

$$\frac{\partial \mathcal{L}}{\partial y_j} = -y_j^t \frac{1}{y_j} \tag{3-12}$$

Now, the gradient of the network loss with respect to the total input ζ_j to the activation function of unit j can be computed (see Figure 3-1):

$$\frac{\partial \mathcal{L}}{\partial \zeta_j} = \frac{dy_j}{d\zeta_j} \frac{\partial \mathcal{L}}{\partial y_j} \tag{3-13}$$

where the derivative of the output y of a unit with respect to its summed input ζ is the derivative of its activation function (Section 3-1) with respect to ζ.

Since the influence of a weight w_{ij} (between a unit i in the previous layer and unit j) on the total summed input to unit j is simply proportional to the activation of unit i, the influence of the weights on the inputs of unit j on the total network loss can now be calculated as:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \zeta_j}{\partial w_{ij}} \frac{\partial \mathcal{L}}{\partial \zeta_j} = y_i \frac{\partial \mathcal{L}}{\partial \zeta_j} \tag{3-14}$$

The influence of the output of a unit i in the previous layer on the total network loss derivative can be calculated by summing the effect it has on the loss derivatives produced by all of the output units j:

$$\frac{\partial \mathcal{L}}{\partial y_i} = \sum_j \frac{d\zeta_j}{dy_i} \frac{\partial \mathcal{L}}{\partial \zeta_j} = \sum_j w_{ij} \frac{\partial \mathcal{L}}{\partial \zeta_j} \tag{3-15}$$

Now the procedure can be repeated and the partial derivatives of the total network loss with respect to the input weights of this layer can be calculated. Then, the loss derivatives can again be back-propagated to the preceding layer and the procedure is repeated until the derivatives of the total network loss with respect to all of the weights in the network are known.

Pairing of the loss function and output-layer units

The combination of the chosen loss function and the activation functions of the units in the output layer needs to be chosen properly. For the classification problems in this thesis, a layer of softmax units is paired with the negative log likelihood loss function. For the regression problems, a linear output layer is paired with a squared error loss function. In both cases, the derivative of the network loss with respect to the input to the activation function in the output layer units is:

$$\frac{\partial \mathcal{L}}{\partial \zeta_j} = \frac{dy_j}{d\zeta_j} \frac{\partial \mathcal{L}}{\partial y_j} = y_j - y_j^t$$

This ensures a reasonable loss gradient irrespective of the output of the unit. On the contrary, if for example a softmax output layer is trained with a mean squared error loss function, this derivative would be:

$$\frac{\partial \mathcal{L}}{\partial \zeta_j} = \frac{dy_j}{d\zeta_j} \frac{\partial \mathcal{L}}{\partial y_j} = y_j(1 - y_j)(y_j - y_j^t)$$

Note that in this case when $y_j^t = 1$ (that is, when j is the correct class) and y_j is near zero, the loss derivative is very small even though the loss is as large as possible. This will slow down the learning considerably.

3-3-4 Back propagation through time

In this thesis, RNNs are used. The recurrent connections require an extension of the back-propagation procedure called Back Propagation Through Time (BPTT) [9]. This method works by *unfolding* the network; for every time-step in a (sub)sequence, a copy of the network

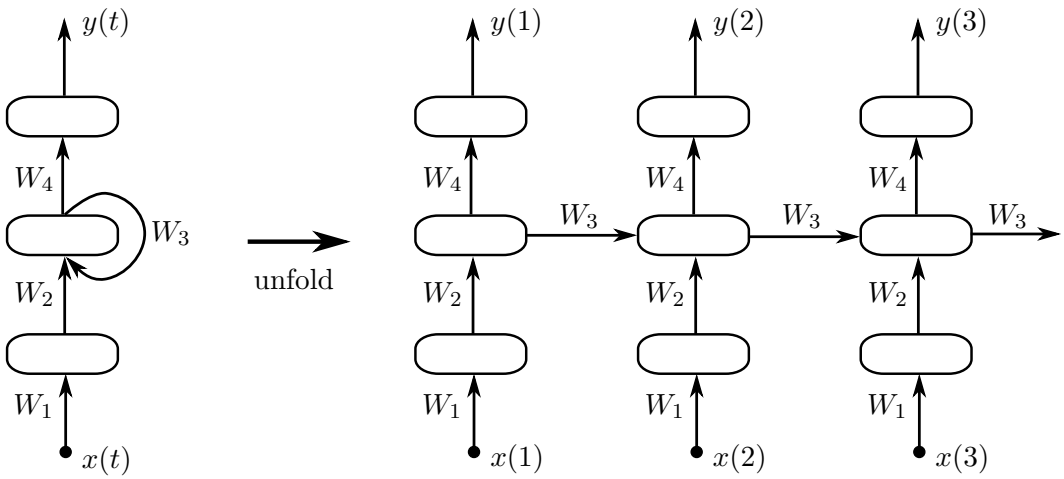


Figure 3-3: Training a RNN with back-propagation through time by unfolding it.

is made. The recurrent connections lead to the corresponding units in the network for the next time-step. This is shown in Figure 3-3. The weights of all the copies of the network are kept equal.

A time-sequence is processed in the *forward pass* by starting with the inputs at the first time-step $x(1)$ and calculating the output $y(1)$ and the outputs on the recurrent connections. Then, these recurrent outputs and the inputs of the second time-step are used to calculate the outputs in the second network copy. For each time-step this process is repeated. When the entire sequence is processed, the loss at every time-step can be calculated.

Starting from the final time-step, these losses are back-propagated through the entire sequence in the *backward pass*. Finally, the gradients with respect to the weights at all time-steps are averaged and used in the optimization procedure.

If the sequence that is used for the BPTT procedure is part of a longer sequence, the recurrent activations at the final time-step are saved and used as the inputs to the corresponding units in the first copy of the network for the next subsequence. When a new sequence is started the initial activations are all reset to 0.

3-3-5 The exploding / vanishing gradient problem

During the forward pass, inputs are applied to an artificial neural network and the resulting outputs are calculated. The outputs (or *activations*) of the individual artificial neurons with non-linear activation functions are always within a certain range. This is a consequence of the activation functions given in Section 3-1. For this reason, the activation functions are sometimes also referred to as *squashing functions*.

During the backward pass, the loss is back-propagated through the network from the outputs to the inputs. This phase is completely linear, which can cause problems [6]. When back-propagating the loss derivative through many artificial neurons, the loss derivative signal is multiplied each time by the weights on the connections between the neurons. When all of these weights are small, this will cause the loss derivative signal to become very small, leading to slow learning. When the weights are large, the loss derivative signal will become very large,

leading to unstable learning. This problem is known as the exploding or vanishing gradient problem of deep learning. *Deep* learning refers to the fact that the signals travel through many neurons on their way between the input and output [7].

When learning the weights of a FNN with a few layers the problem is not too severe, but when learning the weights of a RNN with long-term time dependencies (e.g. where the correct output of the network can depend on inputs that occurred 100 time-steps earlier) the problem is much worse. Careful initialization of the weights can help somewhat, but generally does not solve the problem [6]. In the track circuit case, very long-term time-dependencies are present.

There are several possible solutions to this problem for RNNs that have to learn long-term time dependencies. One solution is the use of the Echo State Network (ESN) [10], which uses very carefully chosen initial connection weights. Only the weights of the output layer are learned in this method. Alternatively, the network can be initialized like an ESN and all the weights can be learned using back-propagation with momentum [6]. Another possibility is to use a second order gradient descent optimization method, such as Hessian Free Optimization [11].

The solution used in this thesis is the use of the Long Short-Term Memory (LSTM) network architecture. This method has recently been successfully applied to several long-term time dependency problems [12]. Examples include handwriting recognition and generation [13], speech recognition [14], machine translation [15], caption generation for images [16], predicting the output of computer programs [17] and recognizing emotions expressed in music [18]. The LSTM method will be described in the next section.

3-4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a RNN method based on a special network architecture that is capable of capturing very long-term time dependencies [3]. This makes it an interesting choice for the condition monitoring of track circuits, since very long term time dependencies are expected to be characteristic for some faults. In contrast to other methods it can learn relationships between events spaced out in time even when there is no local regularity. It is also robust to noisy sequences [7].

LSTM architecture

A neural network with LSTM units is able to store information for long periods of time by introducing *memory cells* and *gate units* into the network architecture. The memory cells are linear neurons that have a recurrent self connection with weight 1. This enables them to remember a value over subsequent time-steps. To prevent the memory in the memory cell from becoming overwritten by irrelevant inputs, LSTM uses an input gate unit that can determine when inputs are passed to the memory cell and when they are not. Additionally, there is a forget gate that can erase the contents of the memory cell and an output gate that controls when the LSTM unit outputs its memory to the rest of the network. An LSTM unit can be thought of as a differentiable bit of computer memory with read, write and reset

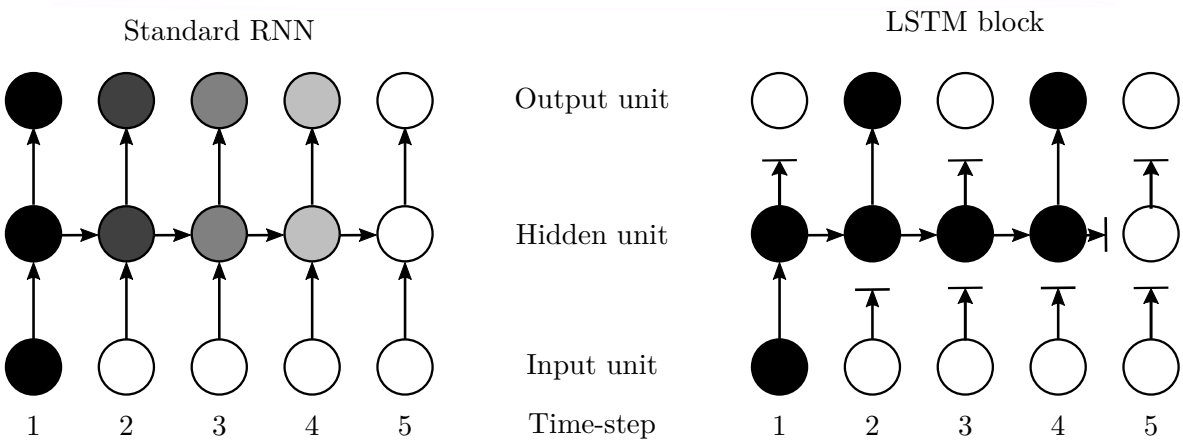


Figure 3-4: The vanishing gradient problem explained from the *forward pass*. The shade of the units indicates their sensitivity to a relevant input at time-step 1. Although recurrent connections enable the storage of a memory, a standard RNN has problems learning long-term time dependencies as new irrelevant inputs will overwrite the original memory. An LSTM memory block mitigates this problem by introducing *gate units* that can prevent irrelevant inputs from entering the memory cell. The forget gate furthermore allows the memory to be forgotten when it is no longer relevant and the output gate will only send the memory to subsequent units when the memory is relevant.

inputs [19]. The benefit of controlling the flow of information to and from the memory cell in this way with the gate units is illustrated in Figure 3-4.

The gate units, which are neurons with a sigmoidal activation function (3-3), work by multiplying the signals going to and from the memory cell with their outputs. When the output of the gate unit is close to zero this effectively blocks the signal. When the output of the gate is close to 1 the signal is passed on unaltered. The architecture of an LSTM unit is shown in Figure 3-5, where the multiplications are indicated by the black dots.

The equations describing an LSTM unit are:

$$i(t) = \text{sigm}\left(W_{xi}x(t) + W_{hi}h(t-1) + b_i\right) \tag{3-16}$$
$$f(t) = \text{sigm}\left(W_{xf}x(t) + W_{hf}h(t-1) + b_f\right) \tag{3-17}$$
$$a(t) = \tanh\left(W_{xa}x(t) + W_{ha}h(t-1) + b_a\right) \tag{3-18}$$
$$o(t) = \text{sigm}\left(W_{xo}x(t) + W_{ho}h(t-1) + b_o\right) \tag{3-19}$$
$$M(t) = f(t)M(t-1) + i(t)a(t) \tag{3-20}$$
$$b(t) = \tanh(M(t)) \tag{3-21}$$
$$y(t) = o(t)b(t) \tag{3-22}$$

In these equations, $i(t), f(t),$ and $o(t)$ are the outputs of the input, forget and output gates at time-step t respectively. The input and output nonlinearities are $a(t)$ and $b(t)$. $M(t)$ is the output of the memory cell at time-step t and $y(t)$ is the output of the LSTM unit. The inputs to the unit consist of two vectors of signals. The first is $x(t)$, which contains the outputs from the previous layer at the same time-step. The second vector is $h(t-1)$, which contains the outputs from the units in the same layer at the previous time-step. Note that in the networks used in this thesis, the layers with LSTM units are fully recurrently connected.

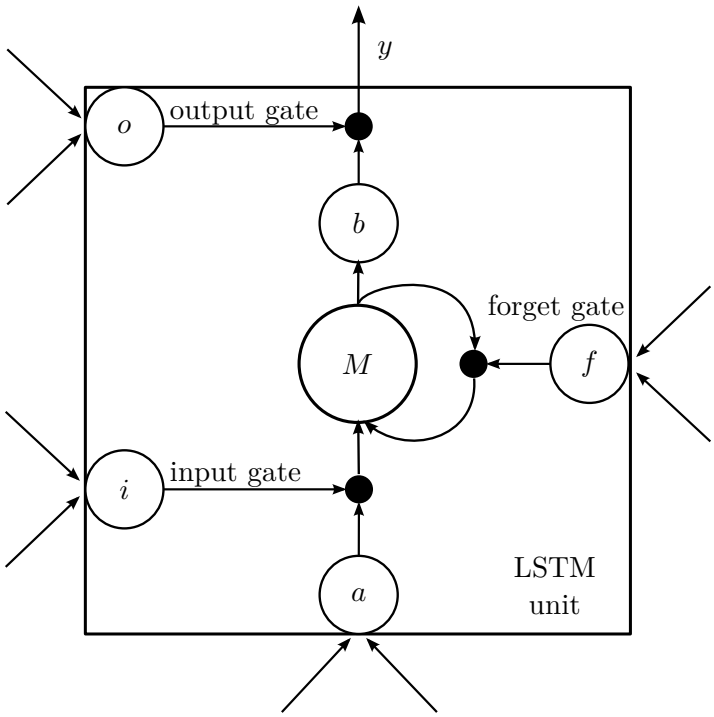


Figure 3-5: Architecture of the LSTM unit. The units a and b are hyperbolic tangent $(3\text{-}4)$ input and output neurons. M is the memory cell, which is a linear $(3\text{-}2)$ neuron with a recurrent self connection with weight 1. The black dots indicate a multiplication of the outputs of the sigmoidal $(3\text{-}3)$ *gate neurons* denoted by $i, o$and f with the outputs of the other units. This allows the input gate neuron i and output gate neuron o to selectively block or pass the signals from and to the memory unit and it allows the forget gate neuron f to reduce or reset the value of the memory unit.

A RNN with LSTM units can be trained with the BPTT algorithm discussed in Section 3-3-4 [19]. The architecture of the LSTM unit helps solve the exploding / vanishing gradient problem discussed in Section 3-3-5. This can be seen when examining equations for the backwards pass through the unit. Let q^ζ denote the input sum to neuron q (see Figure 3-1). For example: $i^\zeta(t) = W_{xi}x(t) + W_{hi}h(t-1) + b_i$ (see (3-16)). Then the following equations describe the backward pass through the LSTM unit:

$$\frac{\partial \mathcal{L}}{\partial o^\zeta}(t) = \text{sigm}'\Big(o^\zeta(t)\Big)b(t)\frac{\partial \mathcal{L}}{\partial y}(t) \tag{3-23}$$

$$\frac{\partial \mathcal{L}}{\partial M}(t) = o(t)\tanh'\Big(M(t)\Big)\frac{\partial \mathcal{L}}{\partial y}(t) + f(t+1)\frac{\partial \mathcal{L}}{\partial y}(t+1) \tag{3-24}$$

$$\frac{\partial \mathcal{L}}{a^\zeta}(t) = i(t)\tanh'\Big(a^\zeta(t)\Big)\frac{\partial \mathcal{L}}{\partial M}(t) \tag{3-25}$$

$$\frac{\partial \mathcal{L}}{f^\zeta}(t) = \text{sigm}'\Big(f^\zeta(t)\Big)M(t-1)\frac{\partial \mathcal{L}}{\partial M}(t) \tag{3-26}$$

$$\frac{\partial \mathcal{L}}{i^\zeta}(t) = \text{sigm}'\Big(i^\zeta(t)\Big)a(t)\frac{\partial \mathcal{L}}{\partial M}(t) \tag{3-27}$$

Note from (3-24) that if the output gate is closed ($o(t) = 0$) and the forget gate stays open ($f(t+1) = 1$), the loss derivative in the memory cell stays unchanged: $\frac{\partial \mathcal{L}}{\partial M}(t) = \frac{\partial \mathcal{L}}{\partial M}(t+1)$. The loss gradient no longer explodes or vanishes. This allows the network to learn dependencies between outputs and inputs that are spaced many time-steps apart by learning to 'trap' the loss gradient in the memory cell.

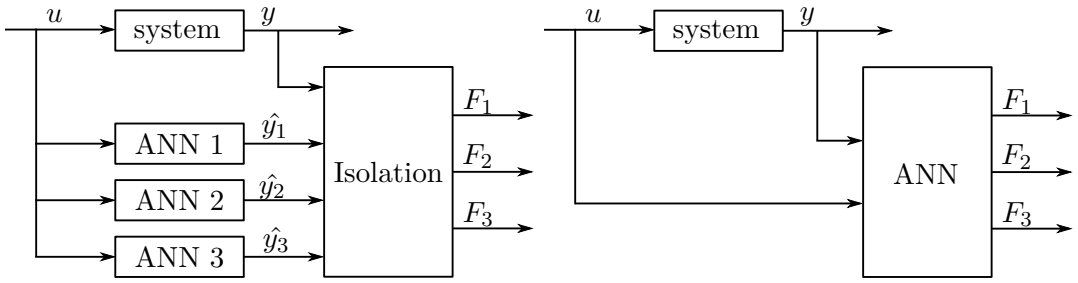
3-5 End-to-end learning

Artificial Neural Networks have recently achieved state of the art performance on several pattern recognition tasks. One reason for these successes is the use of a strategy called 'end-to-end learning'. This strategy is based on moving away from hand-crafted feature detectors and manually integrating prior knowledge into the network. Instead, networks are trained to produce their end results directly from the raw input data. To use end-to-end learning, a large labeled data set is required. When this requirement is met, the benefits of a holistic learning approach tend to be larger than the benefits of explicitly using prior knowledge [14].

One example of a field in which this strategy has been successfully applied is image recognition. For this problem, convolutional networks achieve state of the art performance by using raw pixel values, instead of using hand-crafted feature detectors as inputs [4]. Another example is speech recognition, in which methods using phonemes as an intermediate representation are being replaced by methods transcribing sound data directly into letters [5, 14].

This shift towards end-to-end learning seems to be a logical result of the increasing availability of training data and larger computational budgets in these fields. As it becomes possible to train larger neural networks on more data, less restrictions to the problem definition are needed for the problem to be solvable. As a result, the neural network can move from solving only a sub-problem towards solving the whole problem directly.

In the fault isolation literature, neural networks are often still solving sub-problems. One popular method is to use a multiple model strategy. This strategy involves training a separate



(a) Multiple model strategy **(b)** End-to-end learning

Figure 3-6: The multiple model strategy that is common in the fault isolation literature compared to the concept of end-to-end learning. Note that for the track-circuit case, the output of the system is the electrical current of the track circuit of interest. The inputs are latent variables (e.g. train properties, weather conditions and seasonal changes) that can be inferred from the current values measured in the other track circuits.

neural network for each fault mode. These networks then predict the output of the system for that fault mode given the inputs. A separate module compares these output predictions to the output of the real system. The fault isolation is based on which network output best explains the true system output. This principle is illustrated in Figure 3-6a and used in recent papers such as [20] and [21].

The neural network multiple-model strategy might be sub-optimal for several reasons. Firstly, when training a separate neural network as a model of the system for each fault case, only labeled data of that fault case can be used to train that specific model. However, apart from the faults, all models describe the same system and should therefore have a lot in common. The data that describes the other fault modes should therefore also be relevant. This means that having completely separate models for each fault case presumably leads to an inefficient use of the available training data. Secondly, as seen from the papers in other fields that use end-to-end learning, dividing the problem into sub-problems can hurt the eventual performance. It might therefore be beneficial not to separate the different fault models and the fault isolation step, but rather to train one network to produce the fault isolation directly from all the available measurements. This proposed end-to-end neural network fault isolation strategy is depicted in Figure 3-6b.

In the literature where fault detection or fault isolation is done directly from an input time-sequence, e.g. [22], a FNN is often used. Since a FNN has no memory, the complete time-sequence to be considered needs to be presented to the network at the same time. This requires the network to have separate input units for every considered input time-step, resulting in a fault isolation output that is always based on a fixed number of time-steps. In this thesis a RNN is used instead. This preference can also be explained in the context of end-to-end learning. Firstly, the choice of the number of time-steps to be considered for the fault isolation in a FNN is based on prior knowledge of the system dynamics. The idea of end-to-end learning is not to use prior knowledge, but instead have the network learn the dependencies. By using an LSTM RNN the network can learn how many previous time-steps to consider by storing information in its memory cells.

Another way in which using a RNN is more in line with the idea behind end-to-end learning is that it moves away from (over) simplifying the problem. Historically, although they are

in theory more appropriate for time-series problems, RNNs were found too difficult to train on problems with medium to long-term time dependencies. This led to the use of FNNs on these problems, since these networks are less complex and thus easier to train. Now that the LSTM structure has been shown to solve the learning problem, the simplification of using a FNN is no longer required and better performance is expected from a RNN.

Chapter 4

Network implementation and results

In this chapter, the implementation of the Artificial Neural Network (ANN) strategy outlined in Chapter 3 is discussed. The resulting networks are trained and tested with synthetic data from the generative model described in Chapter 2.

In Section 4-1 the basics of the implementations are discussed. In Section 4-2 the results of the network for fault isolation are discussed. Section 4-3 discusses the results for the fault severity estimation network. To gain some insights into how the ANN is solving the fault isolation problem, the t-SNE technique is used in Section 4-4. To better understand the performance of the ANN method, the results of a fault isolation network are compared to the results of a hand crafted multiple model fault isolation method in Section 4-5.

The results discussed in Section 4-2 until Section 4-5 are based on the assumption of the availability of large amounts of labeled data. Since this assumption is not realistic, the consequences of using less labeled data are investigated in Section 4-6. Unsupervised pre-training is investigated as a way to reduce the need for labeled data.

4-1 Implementation and datasets

The neural networks in this thesis are implemented in the scientific computing framework `Torch` [23]. Parts of the code are adapted from the implementation of [17].

In order to take the spatial dependencies of the faults into account, the network input consist of the electrical current signals from five separate track circuits. This flow of information in shown in Figure 4-1 for the fault isolation network. The signals come from the track circuit that is being diagnosed $I_A(t)$, as well as two other track circuits on the same track $\{I_{B1}(t), I_{B2}(t)\}$ and two track circuits on an adjacent track $\{I_{C1}(t), I_{C2}(t)\}$. The input to the networks at each time-step $\mathcal{T}$consists of the four current measurements discussed in Section 2-6 from each of the five considered track circuits, resulting in 20 current values per time-step .

To keep the temporal dependencies from becoming too long-term, 20 time-steps $\mathcal{T}$are considered per day. All *sequences* that are generated have a length of 100 days. This means that

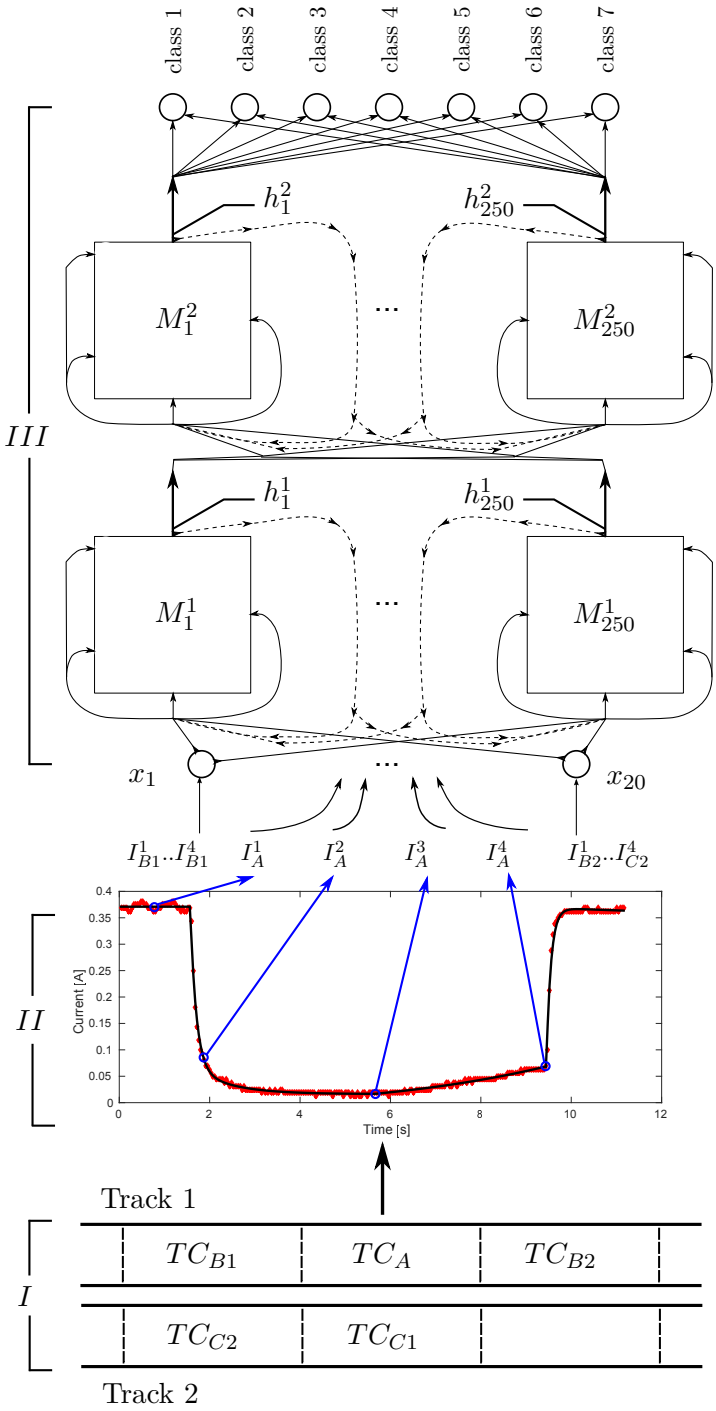


Figure 4-1: Fault identification process overview. For each time-step $\mathcal{T}$ the currents of five track circuits (I) are sampled (II) when a train passes. These samples are the input to the neural network (III) which uses them to update the likelihood of the seven different fault isolation classes.

every input sequence contains 2000 time-steps with 20 measurement values per time-step. To speed up the learning process the input data are normalized to have a mean of zero and a standard deviation of 1 [24].

To train and validate the neural networks, three separate data sets are generated. The first one is a *training data set*, the second is a *validation data set* and the last is a *test data set*. For each sequence in the datasets the properties of the track circuits and the properties of the fault are stochastically determined.

The networks are trained only with data from the training data set. During training, the network performance on the validation data set is checked periodically. This is done for two reasons. The first is to guard against over-fitting. When the performance on the training data set keeps improving while the performance on the validation dataset deteriorates, the network is over-fitting. This would hurt its ability to generalize. Therefore the weights of the network that produced the best performance on the validation dataset are used after the training has stopped. The second reason is to lower the learning rate when the performance on the validation dataset stops improving. This was found to make the learning procedure more robust with respect to the chosen hyper parameters. Training is stopped once network performance no longer changes significantly. To prevent too large changes to the weights, the norm of the gradients is constrained through [17] :

$$\left(\frac{\partial \mathcal{L}}{\partial W}\right)_C = \begin{cases} \frac{M}{\left|\frac{\partial \mathcal{L}}{\partial W}\right|}\frac{\partial \mathcal{L}}{\partial W} & \text{if } \left|\frac{\partial \mathcal{L}}{\partial W}\right| > M \\ \frac{\partial \mathcal{L}}{\partial W} & \text{if } \left|\frac{\partial \mathcal{L}}{\partial W}\right| \leq M \end{cases} \tag{4-1}$$

Where $\left(\frac{\partial \mathcal{L}}{\partial W}\right)_C$ is the constrained version of the vector with the partial derivatives of the loss with respect to all the weights in the network and M is the maximum gradient norm.

The test data set is only used after the training of the networks has stopped. The results of testing the networks on this data set are given in this chapter.

4-2 Fault isolation

The fault isolation network has two LSTM layers with 250 units in each layer. It has a softmax output layer with 7 units; 1 for the healthy state and 6 for the fault types that are considered in this thesis.

The network is trained to give a classification of the sequence at every time-step $\mathcal{T}$. The target for this classification $y^t(\mathcal{T})$ is the healthy state, unless the sequence contains a fault for which the *severity* at that time-step $\mathcal{T}$is above 0.15. The severity of the fault is between 0 and 1. A fault with a severity of 0 will have no influence on the electrical current levels and a fault with a severity of 1 will influence the current enough to cause a failure, where the track circuit is no longer able to function correctly. The value of 0.15 is chosen to detect the faults as early as possible without having any false positive fault detections.

4-2-1 Fault detection

For the *fault detection* part of the fault isolation problem, it is important to detect the presence of a fault long before it leads to a failure, while not classifying the normal variations in the

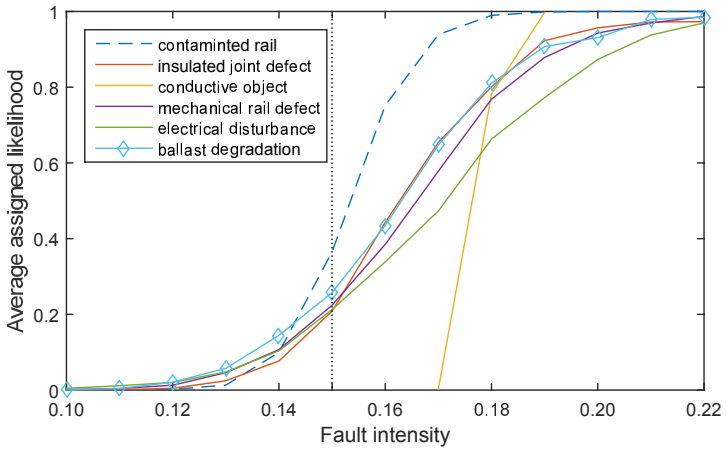


Figure 4-2: Fault isolation performance.

healthy data sequences as faulty. The test dataset contains 100 sequences that were generated to be fault free. The highest likelihood that the ANN has assigned to the presence of a fault for the 200.000 time-steps in these sequences is 0.15. The average likelihood the network assigns to the presence of a fault during the 200.000 time-steps in the healthy sequences was $2.1 \cdot 10^{-5}$. This shows that the network does not suffer from false positive fault detections.

When the detection of a fault is defined as assigning a likelihood of less than 0.5 to the healthy state, the network detects *all* faults where the fault intensity is above 0.28; long before it leads to a failure. The average likelihood that the network assigns to faults of a certain severity is shown per fault type in Figure 4-2.

4-2-2 Fault cause determination

For the fault isolation problem it is not only necessary to determine when a fault is present in the system, but also to determine the cause of the fault. In Table 4-1 the ability of the network to distinguish between different fault types is reported. In this table, the average likelihood over all time steps of all sequences in the test data set that the network assigns to the different classes is given per true class. The rows of the table represent the true class and the columns represent the estimated class. It can be seen that the network has no problems

Table 4-1: Fault isolation confusion matrix for the network trained with end-to-end learning.

True class / assigned likelihood	H	RC	IJ	CO	MRD	ED	BD
Healthy	0.99	0.00	0.00	0.00	0.00	0.00	0.00
Rail Contamination	0.01	0.99	0.00	0.00	0.00	0.00	0.00
Insulated Joint	0.03	0.00	0.96	0.00	0.00	0.00	0.01
Conductive Object	0.01	0.00	0.01	0.99	0.00	0.00	0.00
Mechanical Rail Defect	0.03	0.00	0.00	0.00	0.97	0.00	0.00
Electrical Disturbance	0.09	0.00	0.00	0.00	0.00	0.91	0.00
Ballast Degradation	0.19	0.00	0.00	0.00	0.00	0.00	0.81

with distinguishing the different fault types. The highest likelihood that is assigned to a wrong fault type occurs when an insulated joint defect is present. On average a likelihood of 0.0096 is then assigned to the ballast degradation class. The reason for this is that the insulated joint defects and ballast degradation faults both can have the same spatial and temporal dependencies. However, since it is assumed in the generative model that ballast degradation develops more slowly, the ANN is still able to distinguish between these different fault types very successfully, since the average likelihood that the network assigns to the correct class is 0.9593. Note that from Table 4-1 it might appear that the network is not good at detecting

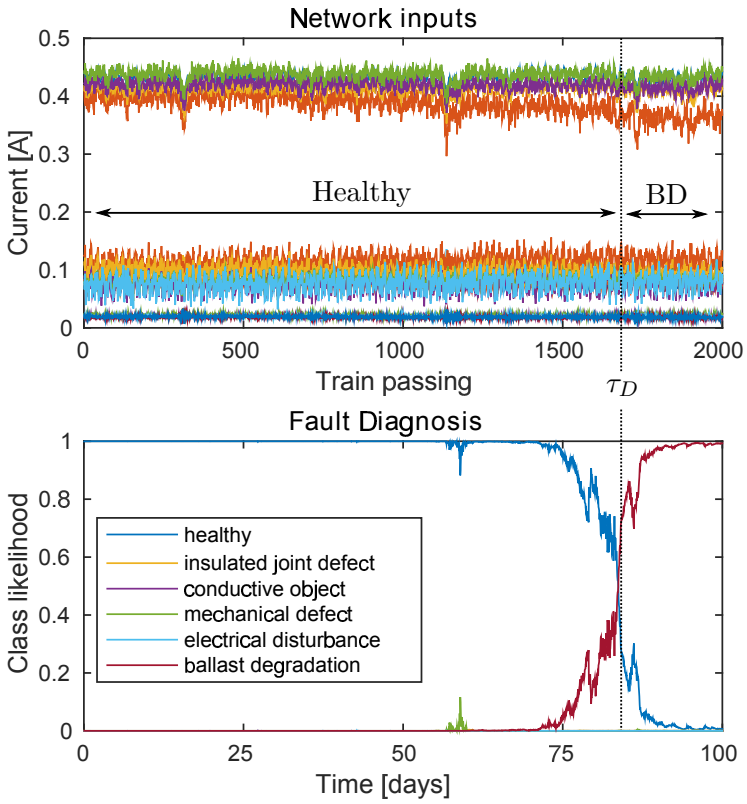
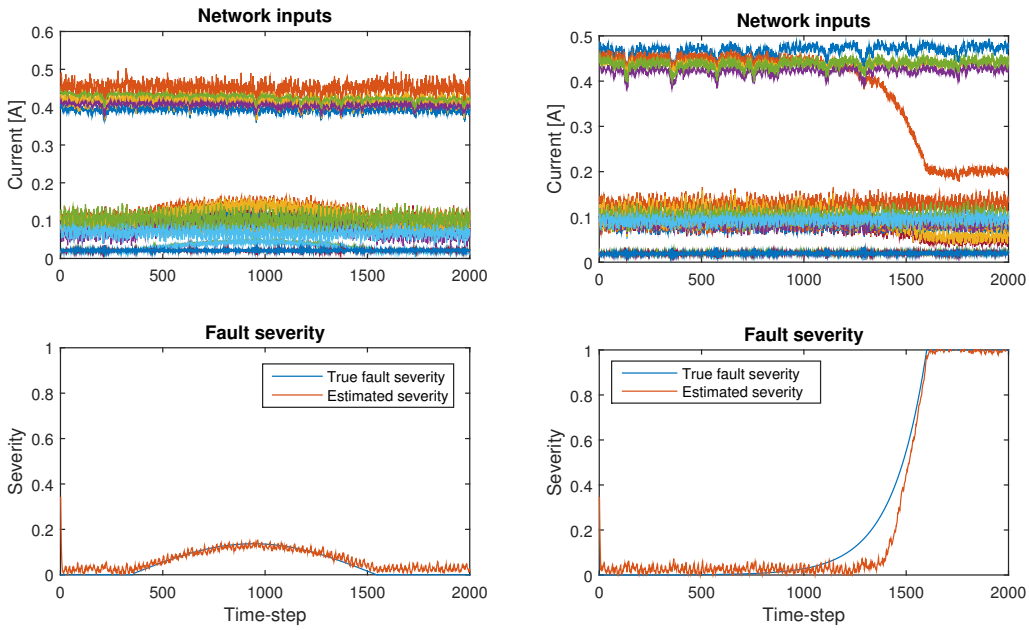


Figure 4-3: Network inputs and output for one realization of a ballast degradation fault sequence. The detection time $\mathcal{T}_D$ marks the detection threshold. Before this point the correct classification is healthy and after this point the correct classification is ballast degradation (BD).

ballast degradation faults. As can be seen from Figure 4-2 however, this is not the case. The lower likelihood that is on average assigned to the correct class stems from the fact that the ballast degradation faults develop very slowly. This results in more time-steps where the fault severity is close to the detection threshold of 0.15. In Figure 4-3 an example is given of a sequence of inputs to and outputs from the fault isolation network for a ballast degradation sequence. The detection threshold is shown in the figure.



(a) Rail contamination example.

(b) Mechanical defect example.

Figure 4-4: Examples of the output of the fault severity estimation network.

4-3 Fault severity estimation

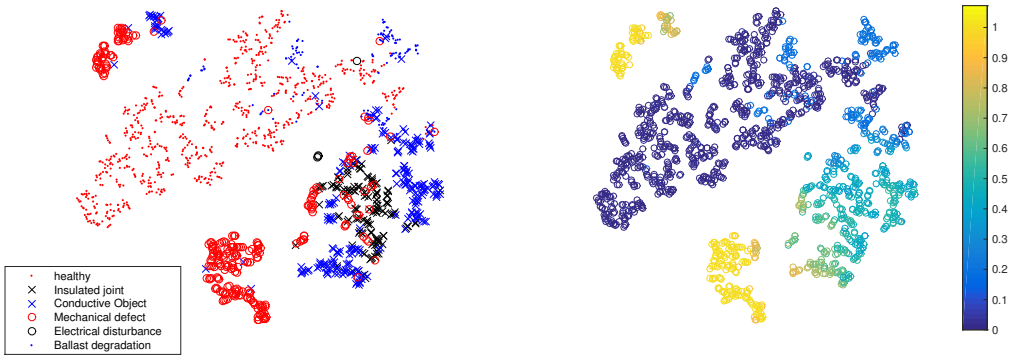
The fault severity estimation network has two LSTM layers with 250 units in each layer. It has a linear output layer with one unit. The target output for the network at each time-step is the true fault severity at that time-step.

In Figure 4-4 examples are shown of the outputs of the fault severity network on sequences from the test data set. It can be seen that the network has learned to give an estimate of the severity. However, from Figure 4-4b it can be seen that the network sometimes is slow in recognizing the presence of a fault, which limits its usefulness. In addition to this, the networks are trained with the true fault severity as a target at every time-step. It does not seem reasonable that this information will be available when training with real data. For these reasons, the rest of this thesis will focus on the fault isolation network.

4-4 Investigating network properties using t-SNE

To gain more insight into what the fault identification network has learned, the internal representations of the network after presenting the whole input sequences to it will be investigated. After presenting 1500 sequences to the network the state of the memory units in the LSTM cells and the activations of the output units of the two recurrent layers in the network are stored. These activations are the network's internal representation of the sequence of events that has preceded the final time-step and of the last input.

To compare these unit activation vectors, t-SNE [25] is used. This technique makes it possible



(a) Labeled by true class. **(b)** Labeled by fault severity.

Figure 4-5: t-SNE representation of the activity vectors of the output units in the *first* recurrent layer at the last time-step of the sequences in the test data set ($h^1(2000)$).

to embed these 250-dimensional vectors in a 2-dimensional image in such a way that the vectors that are close together in the 250 dimensional space are also close together in the 2 dimensional plot. Therefore, input sequences that are similar according to the network will occur close together in the plots.

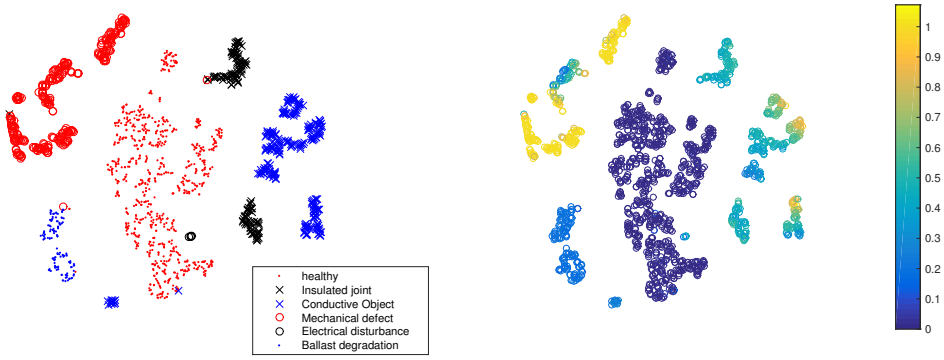
Role of the layers

The network has two recurrent layers. The inputs to the first layer are the current measurements from the track circuits. The outputs of this layer are the inputs to the second layer. The outputs of the second layer are used as the inputs to the *softmax* classification layer. The idea behind having multiple layers is that each subsequent layer might use the outputs of the previous layer and form higher level abstractions of the data. To investigate whether this has happened, the activation vectors of the output units of both layers are plotted. Figure 4-5 shows the activations of the output units in the first recurrent layer at the last time-step for all 1500 sequences in the test set. Figure 4-6 shows the same for the second layer.

From Figure 4-5a it can be seen that the outputs of the *first* recurrent layer of the classification network are not too sensitive to the temporal dependencies in the data, as sequences from different classes are close together in the plot. From Figure 4-5b it can be seen that the similarity of the outputs of the first layer seem to be based mostly on the fault severity at the final time-step as sequences with similar fault intensities are close together.

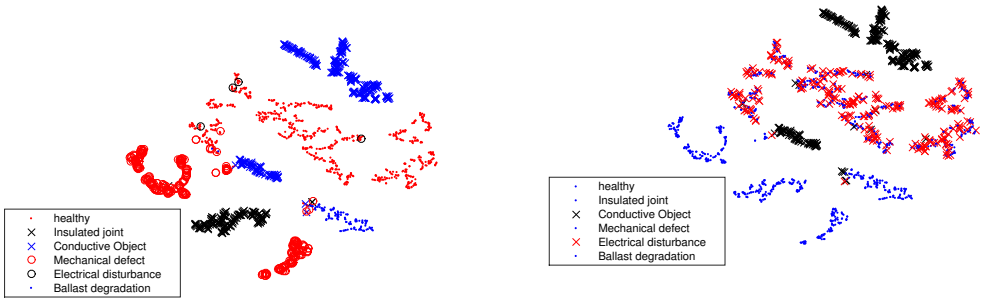
The activation vectors of the output units in the *second* layer are labeled by the true fault isolation class in Figure 4-6a. The grouping here seems based mostly on the true class and therefore on the underlying dependencies that define these classes.

In Figure 4-7a the state of the memory units in the second layer can be seen in the final time-step of the sequences. It is interesting to note that the classes are less clearly separated here than they are in the output units of this layer. Presumably the information about the fault severity coming from the first layer at the same time-step is used to improve the classification. Alternatively it might mean that the network remembers more information about the sequence than what is passed on at any given time to the *softmax* layer.



(a) Labeled by true class. **(b)** Labeled by fault severity.

Figure 4-6: t-SNE representation of the activity vectors of the output units in the *second* recurrent layer at the last time-step of the sequences in the test data set ($h^2(2000)$).



(a) Labeled by true class.

(b) Separating conductive objects from electrical disturbances .

Figure 4-7: t-SNE representation of the state of the memory units in the second recurrent layer at the last time-step of the sequences in the test data set ($M^2(2000)$).

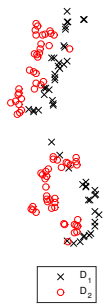


Figure 4-8: Ballast degradation sequences labeled per spatial dependence.

To gain more insights into how the network learns to isolate faults, it can also be attempted to deduce how the network distinguishes between the conductive object- and the electrical disturbance fault types. Both faults abruptly lower the value of the current when a train is not present in the section. But while the current subsequently stays low for the conductive object fault, it is only intermittently low for the electrical disturbance. Furthermore, an electrical disturbance affects multiple track circuits along the same track where a conductive object impacts only one. From Figure 4-7b it can be seen that the network keeps a memory of a conductive object being present in the network. It does not however keep a memory of the fact that electrical disturbances have been observed earlier in the sequence, as the sequences for which this is the case are not separated from those of the healthy sequences. In fact, in Figure 4-7a it can be seen that also for the sequences that are at that time-step undergoing an electrical disturbance, the state of the memory is similar to those in the healthy state.

Spatial dependencies

As discussed in Section 3-5, the prior knowledge of the spatial and temporal fault dependencies is not explicitly used in the network. Doing so on real data could introduce a bias if the prior knowledge turns out to be inaccurate. Since the model is here trained and tested with synthetic data that is generated by a model that is based on the prior knowledge it is interesting to see to what extent the network has learned to identify these dependencies by itself.

Clearly, since the fault types differ only based on their spatial and temporal dependencies and the network manages to correctly classify them, it has learned to distinguish between these dependencies. However, the spatial dependencies are not strictly necessary to distinguish between these 5 faults. Therefore it is interesting to see if the network has learned these dependencies or not. One example is the degradation of the ballast, which can affect either one track circuit or several along the same track. These spatial dependencies are identified with D_1 and D_2 respectively. For each sequence with a ballast degradation fault one of these options is picked with equal probability. In Figure 4-8 the sequences suffering from the ballast degradation fault are shown. It appears from the plot that, although these sequences are very similar, the network does distinguish between these spatial dependencies.

4-5 Method comparison

In Section 4-2 it has been shown that the fault isolation network has successfully learned the tasks of detecting faults and determining their cause. However, it is difficult to judge the performance of the network on this task without comparing it to an alternative method. Fortunately, another method is being developed [26] to solve the same fault isolation problem. This method is developed to work with the data from the generative model described in Chapter 2. The method uses a multiple model strategy. The different models are based on the prior knowledge from [2] that the generative model of Chapter 2 is also based on. Since the fault models are not learned from data, the method does not suffer from the problems mentioned in Section 3-5.

When the comparison was made, the multiple model strategy used only the electrical current measurements from three unoccupied track circuits and a precipitation measurement to

perform the fault isolation of three fault types. A new neural network was trained to use these same inputs to isolate the same faults. Since the neural network does not use any prior knowledge this change did not require any significant change to the code.

In Figure 4-9 a comparison is given of the mean squared errors between the true classification of all considered time-steps and the output of the two methods. The figure shows that the neural network achieves the best performance on the sequences representing healthy track circuit behavior and sequences representing ballast degradation. The multiple model method achieved the best performance in isolating faults caused by conductive objects and electrical disturbances. It was observed that the performance of both methods is very similar. A slight change in the parameters in either method could change which method outperformed the other on all classes, with neither method ever outperforming the other on all classes simultaneously.

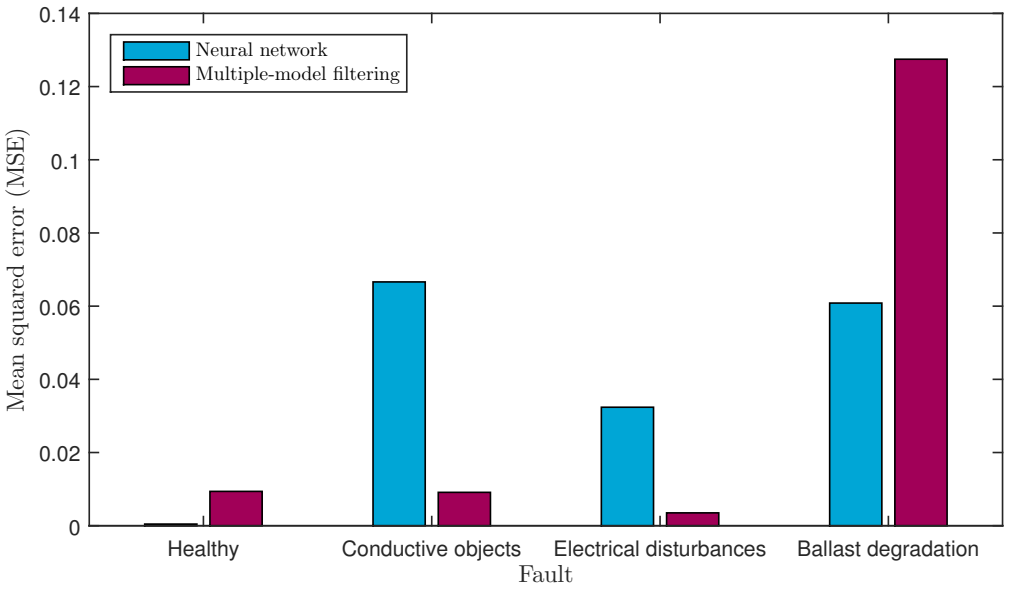


Figure 4-9: Comparison between the method proposed in this thesis and the hybrid multiple model approach proposed in [26].

Since the multiple model method is based on the same prior knowledge of the fault dependencies that the generative model is, it is good to see that the neural network has comparable performance on the synthetic data from the generative model. This proves the ability of the neural network to, given enough labeled data, learn these dependencies accurately.

The performance of the multiple model method on real track circuits will depend on the accuracy of the prior knowledge that this method is based on. The multiple model method is therefore likely to work less good in reality than it does on the synthetic data. Similarly, the neural network will loose performance on real track circuit data as it is unlikely that enough labeled data are available to train it with the same accuracy as the model used in this section. Which method works best on real track circuits is therefore likely dependent on the amount of available training data and the accuracy of the prior knowledge from [2].

4-6 Unsupervised pre-training

In the preceding sections of this chapter it has been shown that, given a large amount of labeled data, a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) can learn to identify faults from the provided measurement data. Unfortunately, the assumption that large amounts of labeled measurement data will be available is not realistic. Therefore, in this section, unsupervised pre-training will be investigated as a way to reduce the amount of labeled data needed to train the networks.

In this section it is assumed that for each of the seven fault isolation classes, there are only five labeled sequences available for training. It is also assumed that a large amount of unlabeled measurement data is available.

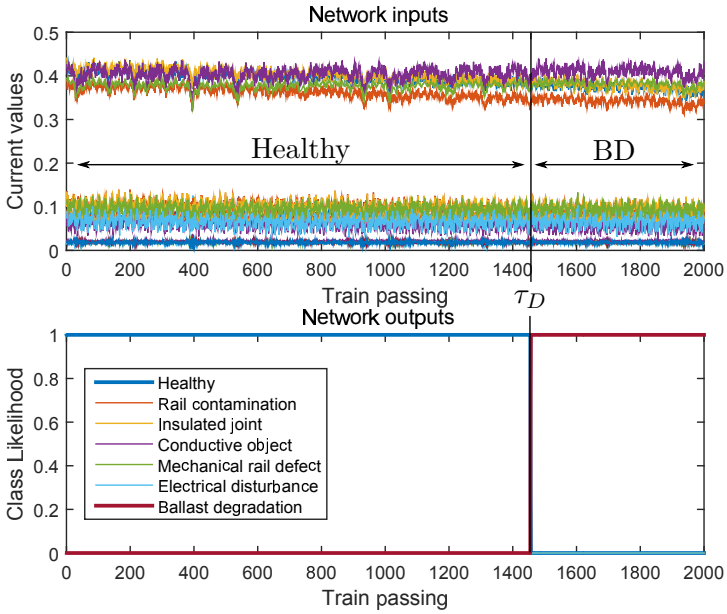
There are two main problems with having only a small amount of data available to train a neural network. The first is that the underlying dependencies are not accurately represented in the available data. This problem can unfortunately only be solved by acquiring more data. The second is that a large neural network can over-fit to the training data. The network will find a way to minimize the error on the training data that does not minimize the error on data that is not in the training data set.

While training the network described in Section 4-2 on a large amount of labeled data the loss on the training and test data sets are about equal, with the loss on the test set being 0.0335. When training the same network on only the 35 labeled sequences considered in this section, the loss on the training set becomes 0.0001 with the losses on the test set deteriorating to 1.76; clearly the network is over-fitting rather badly. This can also be seen from Figure 4-10, which shows two sequences with a ballast degradation fault. Both sequences are very similar and yet the output of the network on the sequence from the training set is perfect, while the output of the network when tested on a sequence that it was not trained on is very bad. It appears the network has managed to memorize the sequences in the training set, which badly hurts its ability to generalize.

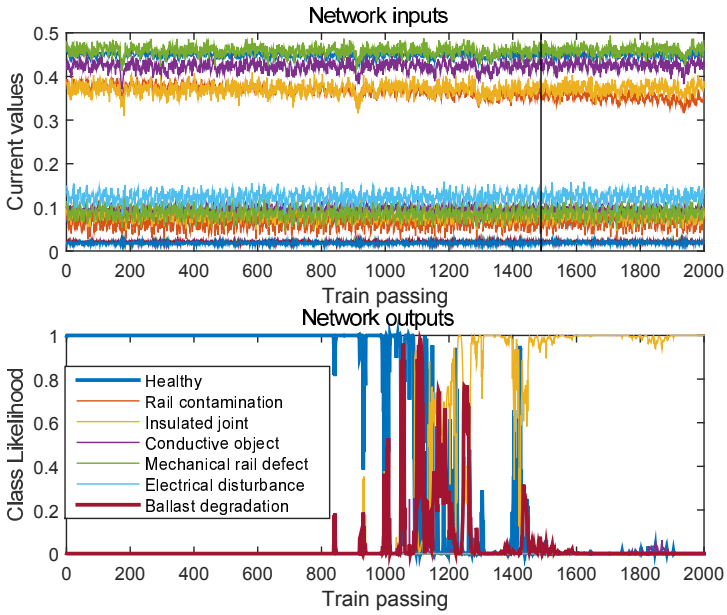
To solve the over-fitting problem several methods are available. The most simple solution is to use a smaller network and stop the training of the network as soon as the performance on the validation data stops improving. Another method is to use a regularization technique such as dropout [27]. The best solution however would be to simply have more data. Therefore in this thesis it will be attempted to learn from unlabeled data as well as labeled data.

Since the equipment that measures the track circuit current has already been installed, it seems reasonable to assume the availability of large amounts of *unlabeled* measurement data. The presumed lack of *labeled* measurement data stems from the fact that since no fault isolation method is currently available, the labeling would have to be done by hand.

Even though the unlabeled data can not be used directly to train the network to perform the fault isolation task, it does contain valuable information about the system. Especially since most of the complexity of the system is in the variation of the current values as a consequence of factors like the weather. These variations are present in data from all fault isolation classes. It might therefore be beneficial to try and learn about the behavior of the system from the unlabeled data. The understanding of the system that is acquired in this way can then be used to learn more efficiently from the labeled data.



(a) Sequence from the training data set.



(b) Sequence from the test data set.

Figure 4-10: The over-fitting problem. The correct classification for both sequences is *healthy* until the vertical line in the inputs and *ballast degradation*(BD) afterwards.

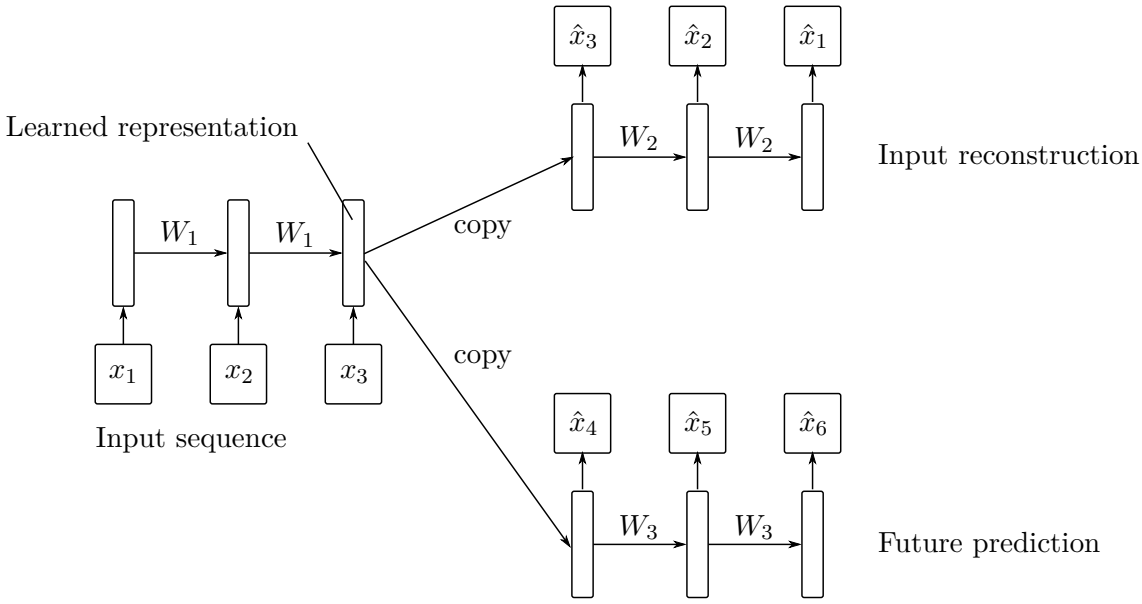


Figure 4-11: Unsupervised pre-training (adapted from [28]).

A recent paper that used LSTM RNN networks to learn from unlabeled data is [28]. This paper used an *encoder* network that was presented with a number of frames from videos. After every input time-step the internal state of the encoder network - the values in the memory neurons of the LSTM units - was copied to two different *decoder* networks. The first decoder network is trained to reconstruct the inputs that were presented to the encoder. The second decoder needs to use the same initial internal state to predict a number of future inputs. This idea is shown in Figure 4-11. The idea behind both reconstructing the past and predicting the future based on the same initial internal state is that in order to do this, the state should contain useful information about the trends in the data. The derivatives of the losses that the two decoder networks make with respect to their internal state are back-propagated through time and passed to the encoder network. This way the encoder network is trained to read a sequence of inputs and to produce a descriptive representation of the state of the system.

For the track circuit case this idea is implemented with an encoder network that has two recurrent layers with 250 LSTM cells in each layer. For each subsequent time-step that is presented to the encoder, the state of its 500 memory cells passed to the two decoders which use these 500 values to reconstruct the inputs of the last 100 time-steps and predict the inputs of the next 100 time-steps. In Figure 4-12 the outputs of the decoders are shown compared to the true inputs. It appears that the encoder network has successfully learned to use the 500 memory neurons to store information about concepts such as the place in the day-night cycle (Figure 2-6), the nominal current levels and the amount of variation of current levels for the different track circuits. Most importantly, it has stored information that allows the decoders to accurately recall and predict the negative trend that can be seen for one of the track circuits in Figure 4-12.

The representation that the encoder network extracts from an input sequence could be useful for the fault isolation task. To use this knowledge, a classification layer with softmax units is added to the network, where the inputs to the softmax units are a linear combination of the

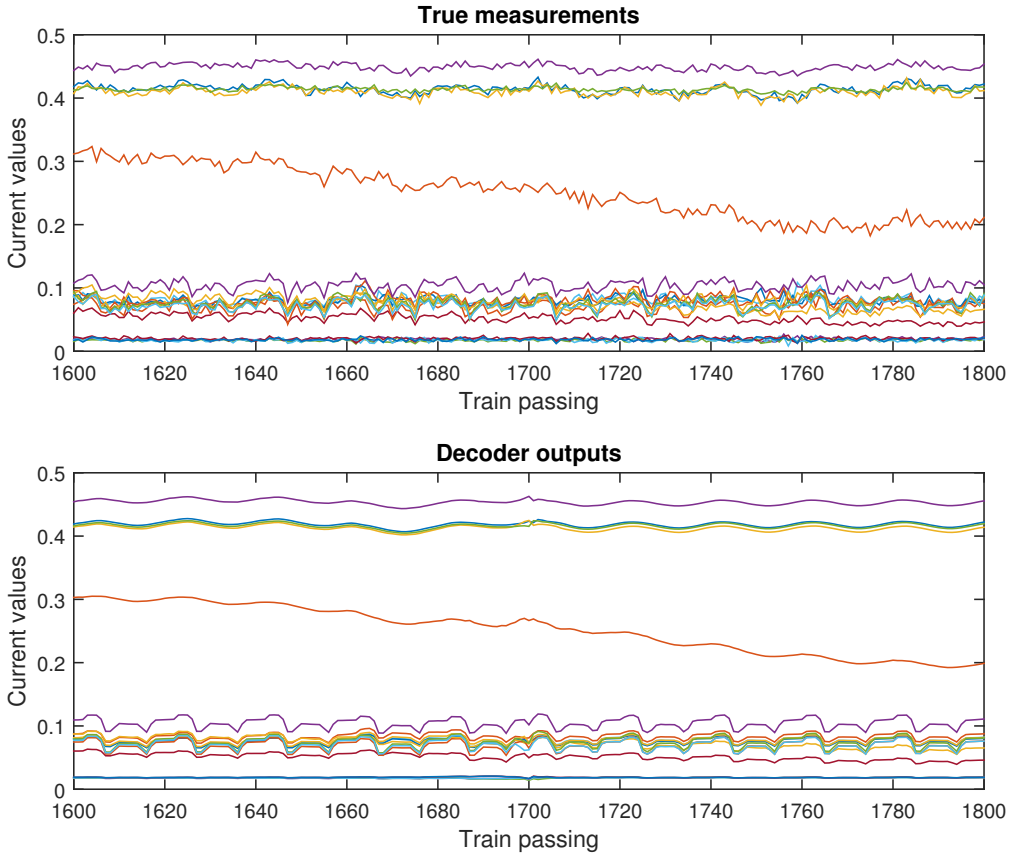


Figure 4-12: Encoder Decoder performance. In this example, the true network inputs have been presented to the encoder network until time-step 1700. The state of the encoder is then used by the decoders to reconstruct the last 100 input time-steps and predict the next 100 input time-steps.

Table 4-2: Fault isolation confusion matrix for the small network trained directly with end-to-end learning.

True class / assigned likelihood	H	RC	IJ	CO	MRD	ED	BD
Healthy	0.95	0.01	0.01	0.01	0.01	0.00	0.02
Rail Contamination	0.09	0.91	0.00	0.00	0.00	0.00	0.00
Insulated Joint	0.56	0.00	0.17	0.19	0.03	0.00	0.04
Conductive Object	0.22	0.00	0.22	0.44	0.10	0.00	0.01
Mechanical Rail Defect	0.31	0.00	0.11	0.26	0.29	0.00	0.02
Electrical Disturbance	0.91	0.00	0.01	0.01	0.02	0.00	0.03
Ballast Degradation	0.83	0.00	0.05	0.05	0.02	0.00	0.04

Table 4-3: Fault isolation confusion matrix for the unsupervised pre-training strategy.

True class / assigned likelihood	H	RC	IJ	CO	MRD	ED	BD
Healthy	0.92	0.01	0.01	0.02	0.02	0.00	0.01
Rail Contamination	0.06	0.82	0.00	0.01	0.01	0.00	0.01
Insulated Joint	0.31	0.00	0.21	0.30	0.12	0.01	0.05
Conductive Object	0.11	0.00	0.15	0.51	0.19	0.01	0.03
Mechanical Rail Defect	0.13	0.00	0.11	0.34	0.38	0.01	0.02
Electrical Disturbance	0.77	0.01	0.04	0.05	0.08	0.02	0.03
Ballast Degradation	0.62	0.00	0.11	0.14	0.04	0.01	0.08

outputs of the encoder. To prevent over-fitting, only the weights on these linear connections are trained, using the 35 sequences in the labeled training data set.

The results of using the unsupervised pre-training strategy can be compared to those of using a small network that is trained exclusively on the labeled data. For the training of a small network on the 35 labeled sequences, the best results were achieved with a network that had 1 recurrent layer with 50 LSTM units. The averaged loss on the test data set was 0.414 for this network. The unsupervised pre-training strategy achieved an average loss of 0.362. In Table 4-2 the confusion matrix of the small network strategy is given. In Table 4-3 the confusion matrix is given for the unsupervised pre-training strategy. It can be seen by comparing these tables that although the pre-training does improve the performance somewhat, the results are still quite poor. It is likely that the results can be improved further by adding dropout [27] and experimenting more with the size of the encoder network and the length of the reconstruction and prediction sequences. It does not seem likely however that the results will improve enough to make a neural network based approach preferable above other approaches (e.g. [26]) without the availability of a larger amount of labeled data than was presumed in this section.

Chapter 5

Conclusions

The goal of this thesis was to develop a condition monitoring method that could detect faults in railway track circuits and determine the cause and severity of the faults, based only on commonly available measurement signals. In this chapter, the main results of this thesis are summarized and recommendations for further research are given.

5-1 Summary and results

In this thesis, a neural network based approach has been proposed for fault diagnosis of railway track circuits. Artificial neural networks have recently achieved state of the art performance on difficult pattern recognition problems in several different fields such as image recognition and speech recognition. These recent successes can be largely attributed to the combination of large networks and large datasets. In the condition monitoring domain large datasets are generally not available. This prevents the use of the large neural networks that have become so successful in other fields. In-spite of this, some of the ideas that have become popular in other domains might still have value in the condition monitoring domain. This thesis has focused primarily on the use of the Long Short-Term Memory (LSTM) architecture and the concept of end-to-end learning. A recent method for unsupervised pre-training has also been applied to the condition monitoring problem to make use of the unlabeled data when not enough labeled data are yet available.

Since the required data logging equipment has already been installed on several track circuits, it is presumed in this thesis that a large amount of unlabeled data will become available. In addition to this, when maintenance is performed on a track circuit in which a fault is developing, the measurement data from that track circuit might be labeled by the maintenance crew. This would lead to the availability of some labeled data. To train and test a machine learning method before these data become available, a generative model has been made.

The results of creating the generative model, training an LSTM network in an end-to-end fashion on a large set of labeled data and the results of pre-training an LSTM network on unlabeled data to make better use of a small labeled data set are given below.

5-1-1 Generative model

By analyzing the historic measurements from 29.542 trains that have passed over 3 track circuits, a generative model has been made that describes the transient behavior of the current in the receiver of a track circuit during the passing of a train. The effects of faults on these currents that are described in [2] have been implemented in the model. Some dependencies of the currents on external factors have also been quantified. These external factors include precipitation, the time of day and the specific train that passes through a section. As these dependencies only explained part of the variations in the historical measurement data, several stochastic variables have been added to introduce random variations with different temporal and spatial properties. This ensured that the synthetic data from the generative model exhibited comparable variation to the real measurement data. This way, the ability of the artificial neural network to separate the effects of faults on the currents from the normal variations of the currents can be better examined.

5-1-2 End-to-end learning

It has been shown in several domains that training a neural network to produce an end result directly from raw input data can work better than explicitly integrating prior knowledge into the network or trying to simplify the problem. In this thesis, this end-to-end learning strategy has been applied to the track circuit condition monitoring case using Long Short-Term Memory networks.

It has been shown that a neural network trained with this strategy can detect faults accurately. A network that was trained to detect faults with a fault severity of above 0.15 (on the scale from 0 to 1) never made a false positive fault detection on the test data set. All faults in the data set with a severity of 0.28 or above were detected, with most faults detected when their severity was around 0.18. The network also correctly identified the fault causes, with on average at least 98.6% of the likelihood assigned to the correct fault type. The performance of the neural network strategy, which uses no prior knowledge of the system, is comparable to that of a hand-crafted method that is based on the same prior knowledge as the generative model. This proves that a neural network trained with end-to-end learning can learn to isolate faults from the data very accurately and would probably outperform methods based on prior knowledge on real data. It has also been shown that using a separate neural network, an estimation of the fault severity can be made.

5-1-3 Unsupervised pre-training

It can be assumed that the majority of the data that will be available in practice will be unlabeled. To extract the knowledge of the system that is embedded in this data, an unsupervised pre-training strategy has been used. It has been shown that when only a small amount of labeled data is available, using a pre-trained network works better than using end-to-end learning. However, when trained with only 5 labeled example sequences of each fault type, the pre-trained network only detected faults from four out of the six fault types. Of these, only two fault types were correctly identified with more than 50% confidence.

5-2 Future work and recommendations

It is likely that significant further improvements can be made to the unsupervised pre-training strategy. One useful addition would be a regularization technique such as dropout [27] to further prevent over-fitting. In addition to this, there is a number of hyper parameters that can be optimized to achieve better results. The high computational cost of training a large neural network has made it infeasible to properly optimize the hyper parameters of the networks and the training algorithm in this thesis. Especially for the unsupervised pre-training strategy it might be beneficial to try out networks with a different number of LSTM units and layers and to try different lengths for the sequences that the decoders are required to reconstruct and predict.

By training and testing the neural networks with the synthetic data from the generative model, it has been shown that these networks can learn the type of fault dependencies that are expected to be present in the data from real track circuits. It was also shown that they can learn to distinguish the trends in the data caused by faults from some of the normal variation that is expected to occur. However, these proofs were based on some simplifications that need to be addressed before the method can be applied in practice. One of these simplifications is the fact that in the generative model, train passings always occur with constant time intervals. As a result, the model can not distinguish the effect of the passing of a train on the development of a fault from the effect of the passing of time on the development of a fault. Another simplification that has been made in this thesis is that for the pre-training strategy, unlabeled data was used which contained a fault in around 26% of the samples. In reality the amount of samples that represent faults will be much lower.

Based on the results in this thesis it seems likely that a Long Short-Term Memory Recurrent Neural Network could outperform fault isolation methods based on prior knowledge on the track circuit case. However, further improvements in using the unsupervised pre-training strategy are necessary to ensure that the amount of labeled data that is expected to be available in practice is sufficient to achieve this performance.

Appendix A

PAPER: Railway Track Circuit Fault Identification using Recurrent Neural Networks

Bibliography

[1] J. Chen, C. Roberts, and P. Weston, "Fault detection and diagnosis for railway track circuits using neuro-fuzzy systems," *Control Engineering Practice*, vol. 16, no. 5, pp. 585–596, 2008.

[2] K. Verbert, B. de Schutter, and R. Babuska, "Exploiting spatial and temporal dependencies to enhance fault diagnosis: Application to track circuits," in *(accepted for publication) in Proceedings of the 14th European Control Conference*, (Linz, Austria,), 2015.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," *arXiv preprint arXiv:1501.02876*, 2015.

[5] A. Y. Hannun, C. Case, J. Casper, B. C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014.

[6] G. Hinton, "Coursera course: Neural networks for machine learning," 2014.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *arXiv preprint arXiv:1404.7828*, 2014.

[8] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, pp. 421–436, Springer, 2012.

[9] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[10] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[11] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1033–1040, 2011.

[12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[13] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[14] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[16] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *arXiv preprint arXiv:1411.4555*, 2014.

[17] W. Zaremba and I. Sutskever, "Learning to execute," *arXiv preprint arXiv:1410.4615*, 2014.

[18] E. Coutinho, F. Weninger, B. Schuller, and K. R. Scherer, "The munich lstm-rnn approach to the mediaeval 2014 emotion in music task," 2014.

[19] A. Graves, *Supervised sequence labelling with recurrent neural networks*, vol. 385. Springer, 2012.

[20] M. Shao, X.-J. Zhu, H.-F. Cao, and H.-F. Shen, "An artificial neural network ensemble method for fault diagnosis of proton exchange membrane fuel cell system," *Energy*, vol. 67, pp. 268–275, 2014.

[21] S. S. Tayarani-Bathaie, Z. S. Vanini, and K. Khorasani, "Dynamic neural network-based fault diagnosis of gas turbine engines," *Neurocomputing*, vol. 125, pp. 153–165, 2014.

[22] J. Sun, R. Wyss, A. Steinecker, and P. Glocker, "Automated fault detection using deep belief networks for the quality inspection of electromotors," *tm-Technisches Messen*, vol. 81, no. 5, pp. 255–263, 2014.

[23] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," tech. rep., IDIAP, 2002.

[24] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[25] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[26] W. Verbeek, "Condition monitoring for track-circuits: A hybrid approach," Master's thesis, Delft University of Technology, the Netherlands, 2015.

[27] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[28] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," *arXiv preprint arXiv:1502.04681*, 2015.

Glossary

List of Acronyms

ANN	Artificial Neural Network
ATB	Automatische Trein Beinvloeding (automatic train influencing)
ANOVA	Analysis of Variance
FNN	Feed-forward Neural Network
RNN	Recurrent Neural Network
BP	Back Propagation
BPTT	Back Propagation Through Time
SGD	Stochastic Gradient Descent
GD	Gradient Descent
LSTM	Long Short-Term Memory
ESN	Echo State Network