Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Data driven anomaly detection for rails using in-service railway vehicles

Bachelor's Thesis

Felix Beckers

fbeckers@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Andreas Plesner
Prof. Dr. Roger Wattenhofer

July 24, 2024

Acknowledgements

I want to thank Andreas Plesner for being my supervisor and for his invaluable help and guidance while working on this project. I want to thank Prof. Dr. Roger Wattenhofer for giving me the opportunity to write this thesis. I am also grateful for the access to the computer cluster and the computing resources of the Distributed Computing Group.

Abstract

This project evaluates machine learning (ML) driven methods of anomaly detection for rails. It is analysed to what extent rail irregularities can be predicted based on the dynamics of different trains using convolutional neural networks (CNNs). The overall structure consists of the data generation for rails, the train dynamics simulation, followed by the training and subsequent testing and assessment of the ML models used. More precisely, the project tries to find an answer to the question to what level a CNN that has been trained with the dynamics of a reference train, can predict the rail condition based on the dynamics of a new train that is defined by slightly different parameters.

Although the model training output is inferior to what was demonstrated in previous work, suitably trained CNNs are capable of predicting the state of the rails from a new and unknown train's dynamics to the same precision if the parameter adjustment is small. Tests using trains having large parameter modifications show an increase in the prediction error.

Two enhancement strategies are analysed. Generalised training uses accumulated dynamics from multiple trains as training data and results in a small overall improvement in the predictive capability compared to the standard case. Fine-tuning with varying amounts of training data shows that a pre-trained model can adapt to the dynamics of a new train easily, even to those of a train having big parameter changes, but indicates the presence of model overfitting.

Contents

Chapter 1

Introduction

1.1 Background

The railway network infrastructure constitutes the basis of one of the biggest sectors of public transportation. Switzerland has one of the best railway networks worldwide [1] and the Swiss Federal Railways (SBB) operate on tracks having a total length of 3,266 km [2]. The rails are the roadbed for trains. Railway vehicles are heavily used because there are different types and they have advantages over buses, planes and ships. There are for example normal passenger trains, high-speed trains, night trains with sleeping opportunities and freight trains. These vehicles travel long distances and carry many passengers or a large amount of freight. They are environmentally friendly, cost-effective and most importantly, safe. One of the reasons for the high safety level is the nature of its locomotion. The vehicles are subject to a rolling movement on a pair of specifically designed rails. Therefore, and due to varying weather conditions, the chassis of the train, including the springs and dampeners, and the wheels experience wear and tear. The rails are subject to deterioration too, which leads to deformations as time passes. These rail deformations, also called irregularities, can cause trains to shake and in extreme cases to derail. Thus, they need to be detected and eventually repaired.

1.2 Project structure

This project is divided into multiple parts. Chapter 1 provides a comprehensive introduction while Chapter 2 describes the track data that is needed for the simulation of the train dynamics. Chapter 3 analyses the actual simulation of railway vehicles on a synthetic track and outputs the train dynamics calculated by solving a set of ordinary differential equations (ODEs). Moreover, this simulation is done for multiple trains, each of which is defined by slightly different parameters. In Chapter 4, the train dynamics are used to train convolutional neural networks (CNNs) that are able to predict the track irregularities. Upon having a suitably trained CNN, it is then evaluated on new and unknown trains

to assess the robustness of the predictive capabilities to a parameter change in the train specification. In the final part, some optimisation strategies are discussed.

1.3 Theoretical problem

This project begins by analysing the feasibility of predicting rail irregularities based on train dynamics. Therefore, the railway vehicle system is conceptualised as a transfer function H, mapping track irregularities to vehicle dynamics. The calculation of the train dynamics as described in Chapter 3 can be abstracted by the drawing in Figure 1.1. The system H loads the generated track information described in Chapter 2, defines a vehicle based on a set of parameters, adds more information like the wheel and rail profile which are not further explained, and yields the calculated dynamics.

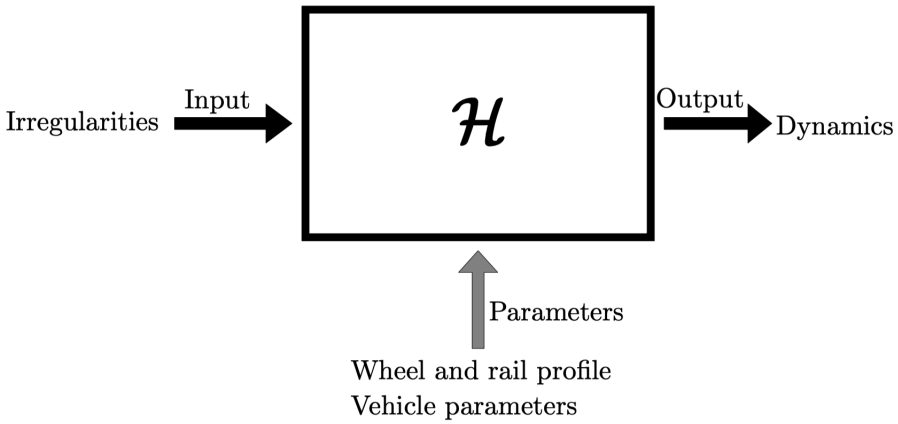

Figure 1.1: The system H loads rail irregularities as input and outputs train dynamics. The output depends on additional parameters, such as the wheel and rail profile and the train specification. This figure was taken from [3].

Chapter 4 aims to approximate the inverse mapping H^{-1} that retrieves the rail irregularities from the train dynamics of a particular train. Consequently, this inverse mapping changes from train specification to train specification.

Challenges arise due to the non-injective nature of the mapping, as mentioned in [3, 4], suggesting that multiple irregularities could yield identical dynamic responses, complicating the inverse solution. Despite these challenges, the presence of noise in real-world data enables the construction of a useful approximation of the inverse mapping, especially when leveraging data-driven methods that can autonomously learn the relevant features.

1.4 Related work

Lasse Engbo Christiansen described in his master's thesis [4] a mathematical model of a bogie and explores the connection between its lateral motion and the lateral irregularities of a track. The model uses real wheel and rail profiles and includes vertical degrees of freedom to make the model realistic. This model is used in Chapter 3 to simulate a train running over a track by outputting its wheel displacements and dynamics.

Andreas Plesner explored in his master's thesis [3] data-driven machine learning methods to predict track disturbances from the dynamics of in-service railway vehicles. His vehicle model is implemented in Julia, benefiting from built-in ODE solvers. He uses CNNs to predict the irregularities of the rails. Furthermore, in a semester project [5], he compared the usage of observed accelerations to the usage of observed positions and velocities in combination with CNNs to predict the track disturbances. It turns out that using observed position and velocity data provides better results.

Detecting and predicting rail irregularities is not the only goal in the context of anomaly detection of trains running over a track. One can also try to identify anomalies in the springs in the chassis of a train vehicle. Lin Xiao approached this problem in her bachelor's thesis [6] by using CNNs with long short-term memory (LSTM) layers.

Chapter 2

Track data

2.1 Overview

Since the motivation for this project is of theoretical nature, there is no real track data available. Consequently, this data needs to be generated. This generated data should have the same statistical properties as the data originating from a real railway track in order to minimise the gap between the theoretical concept and a possible practical application. To achieve the required level of authenticity, a vector autoregression (VAR) model is applied to a small data set of around 20,000 samples which represents a real track segment. It is ensured the synthetic track is sufficiently long, hence providing enough unique data for the train simulation. The mentioned real track information and the code skeleton were provided. The data generation was performed using Matlab [7], version 23 (R2023b).

2.2 Track description

In the real world, a railway track consists of a pair of parallel steel rails mounted on wooden or concrete ties. These ties are placed perpendicularly at regular intervals along the track and are embedded in ballast. An abstraction is necessary for the implementation in computer programs, enabling it to perform calculations. Therefore, the rails are modelled by seven metrics, namely the longitudinal position (in metres), the curvature (in radians/metre), the inclination (in radians) as well as the lateral and vertical positions of the left and right rail (in metres). When mapping the rails onto a standard three-dimensional coordinate system, the longitudinal axis is designated as the x-direction, the lateral as the y-direction and the vertical as the z-direction. The curvature represents the rate at which the rails deviate in the xy plane while the inclination is defined as the angle formed by the height difference of the left and right rail. For simplicity reasons, the simulated trains in this project drive with a constant velocity over a straight track which has no inclination. The velocity is therefore set to 120 $\frac{km}{h}$, the curvature to 0 $\frac{rad}{m}$ and the inclination to 0 rad. This thesis' focus lies on the

rails' deviations from their nominal position which are depicted in Figure 2.1 for clarity. Note that the generated track is actually straight, despite what is shown in the illustration.

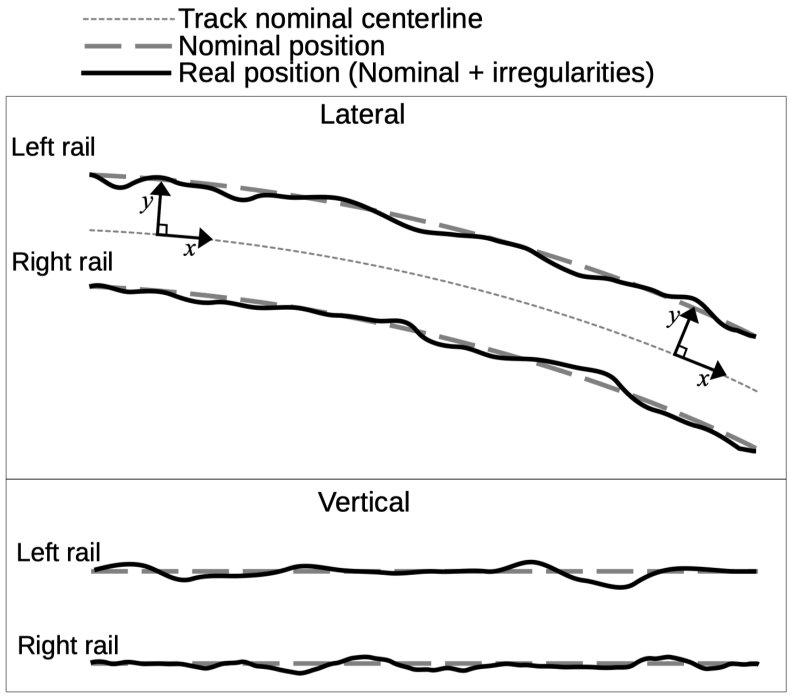

Figure 2.1: The sketch illustrates the lateral rail irregularities from a top view perspective and the vertical rail irregularities as two single curves describing the elevation profile. This figure was taken from [3].

Given this sequential and ordered model of the rails with constant spacing, it can be viewed as a multivariate time series. This enables the use of statistical models such as VAR models to infer knowledge from the underlying data. The four-dimensional data points representing the displacements in the yz plane can therefore be calculated by such a model.

2.3 Vector autoregression model

A vector autoregression (VAR) model is a statistical model used to capture the linear dependencies in time series data. This enables the generation of a new time series data set having the same statistical properties as the original data. Specifically, in a model of order p, the current sample t depends on a linear combination of the previously seen p samples and some additional normally distributed noise [8]. In the multivariate case, this process is described by equation 2.1 where y

is the k-dimensional time series vector, A_i is a time-invariant k by k coefficient matrix and e is a k-dimensional noise vector.

$$y_t = A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t \tag{2.1}$$

2.4 Implementation details

As already mentioned, the rails are defined by seven metrics. Every position on the rails is defined by its coordinates in the xyz space. Since the railway track consists of two single parallel rails, the resulting unique coordinate points are five-dimensional. These metrics are depicted in Table 2.1. The generated track segment comprises 10^6 data points, each separated by 0.16 m in positive x direction. This leads to a total distance of 160 km.

x	Longitudinal position
y_{left}	Lateral displacement of the left rail
y_{right}	Lateral displacement of the right rail
z_{left}	Vertical displacement of the left rail
z_{right}	Vertical displacement of the right rail

Table 2.1: Important metrics of the rails

The VAR model has an order of 13, meaning the previous 13 data points are used for the iterative generation of new points in the time series. First, the process starts by estimating the model, which involves determining the coefficients of its matrices using the real track data set. Then, it simulates a new multivariate time series consisting of the lateral and vertical irregularities of the rails, namely y_{left}, y_{right}, z_{left}, and z_{right} for linearly increasing x.

Once the irregularities are generated, they are post-processed. This is conducted by scaling all four dimensions independently to attain global lateral displacement maxima of 12 mm and global vertical displacement maxima of 8 mm. The scaled and non-scaled irregularities of a 100-metre-long slice of the track are shown in Figure 2.2.

Instead of generating an even larger track segment, eight different track segments, each comprising 10^6 samples and 160 km of length, are generated by setting a different seed. This leads to a repertoire of eight track data files, amounting to a total of $8 \cdot 10^6$ samples and describing 1,280 km of track length.

The order of the VAR model was selected based on a study of the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). Both criteria are used to assess the quality of a statistical model [9]. Moreover, it is assumed that the generated data accurately reflects real track data, with similar distribution and statistical properties. The distribution and quality of the

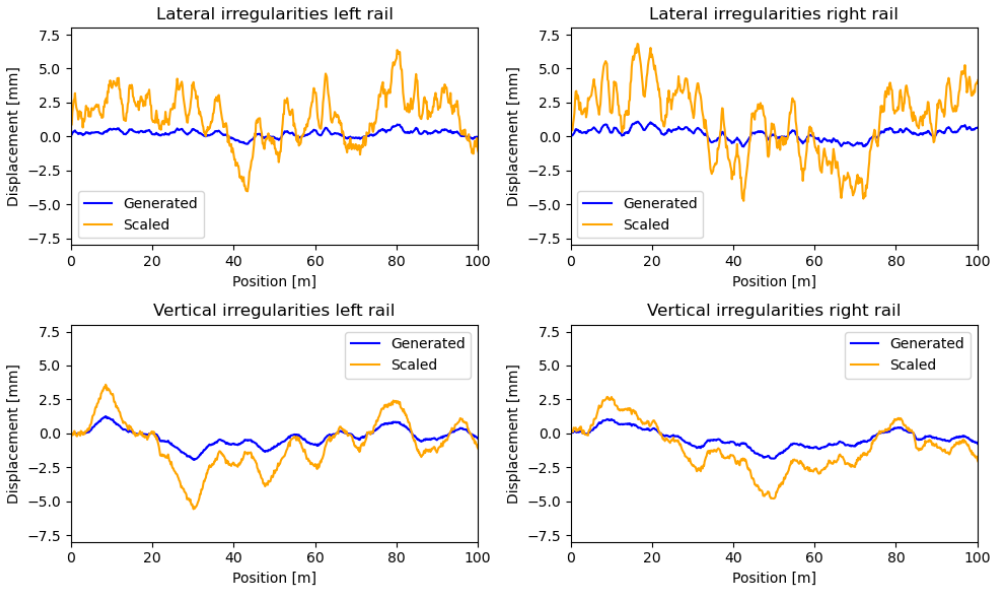

Figure 2.2: Rail irregularities, scaled and non-scaled, shown along a 100-metre-long track segment

generated irregularities can be assessed by conducting a power spectrum density analysis, as it is demonstrated in [5]. However, both of these studies are omitted due to the scope limitations of this project.

Chapter 3

Train dynamics

3.1 Overview

With the availability of synthetic track segments, the next step is to focus on the simulation of the train's dynamics. Since the generated train dynamics data is used for machine learning model training in the latter part of the project, it is essential to generate a substantial amount of training data to ensure robust model performance. The simulation program is a C++ program called 'code_sal_impulse' and was developed as part of the research presented in [4]. The compilation and execution of said program was run on CPU nodes on the computer cluster of the Distributed Computing Group.

3.2 Train description

The train vehicle is composed of a car body and two bogie frames. Every bogie has a front and a rear wheel set which connect the vehicle to the rails. These components are connected by springs and dampeners. The set of springs connecting the bogie frame and the wheel set is called primary suspension while the secondary suspension constitutes the springs and dampeners on the bogie level that support the car body. As a visual aid, a sketch of the train vehicle is drawn in Figure 3.1. The vehicle is modelled as a rigid body that is subject to the translation and rotation motions listed in Table 3.1. In the simulation, only the leading bogie frame is considered for efficiency reasons. The location of the springs and dampeners is depicted in Figure 3.2 where the springs and dampeners are labeled with k_i and D_j, respectively.

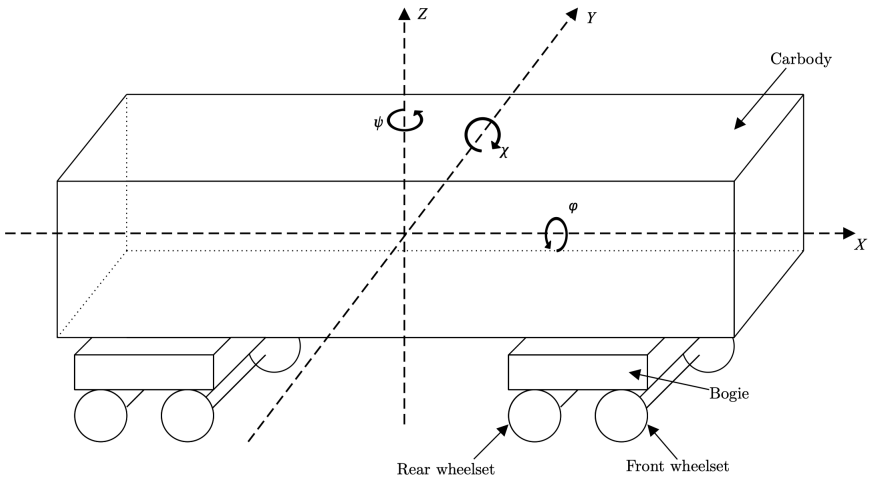

Figure 3.1: Train vehicle sketch showing how a carbody and its bogies are centred in the xyz coordinate system including the relative motions. This figure was taken from [5].

Relative motions	**Symbol**	**Notation**
Translation in direction of travel	X	Longitudinal
Translation in transverse direction, parallel to the track plane	Y	Lateral
Translation perpendicular to the track plane	Z	Vertical
Rotation about a longitudinal axis	φ	Roll
Rotation about a transverse axis, parallel to the track plane	χ	Pitch
Rotation about an axis perpendicular to the track plane	ψ	Yaw

Table 3.1: Relative motions. This table was adapted from [5].

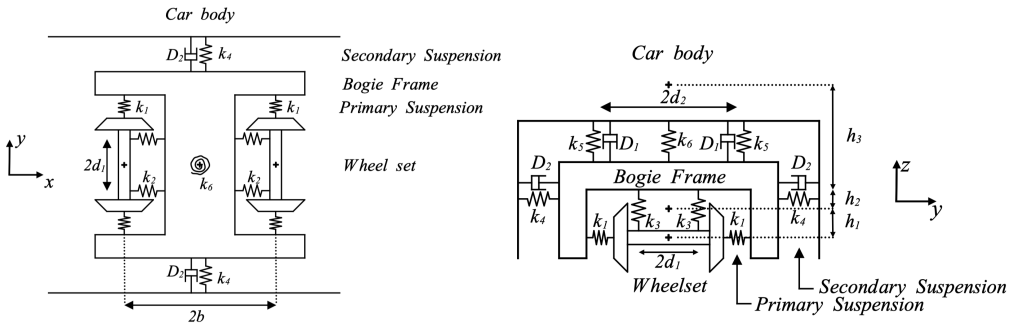

Figure 3.2: The left sketch shows the bogie from a top view perspective. The right sketch shows the bogie from a front view perspective. The primary suspension system consists of the springs k_1, k_2 and k_3 while the secondary suspension system includes the springs k_4, k_5, k_6 and dampeners D_1 and D_2. These figures were taken from [3].

The mathematical model of the train vehicle employs the so-called Cooper-rider's bogie. This means the springs and dampeners are considered linear and obey to Hooke's Law. The locomotion of a train can be described by the analysis of all internal and external forces, especially the wheel-rail interactions. The acceleration is found using Newton-Euler equations and, in this way, also the position and velocity by integration. The train's dynamics are then simulated by numerically solving a system of ordinary differential equations (ODEs). The definition of the ODEs is more thoroughly explained in [3, 4]. The final dynamics consist of 30 metrics describing among other the motions brought up in Table 3.1. The dynamics are listed as y_{1-30} in Table 3.2.

y_1	Front wheel set lateral position
y_2	Front wheel set lateral velocity
y_3	Front wheel set yaw angle
y_4	Front wheel set yaw circular velocity
y_5	Rear wheel set lateral position
y_6	Rear wheel set lateral velocity
y_7	Rear wheel set yaw angle
y_8	Rear wheel set yaw circular velocity
y_9	Bogie frame lateral position
y_{10}	Bogie frame lateral velocity
y_{11}	Bogie frame yaw angle
y_{12}	Bogie frame yaw circular velocity
y_{13}	Bogie frame roll angle
y_{14}	Bogie frame roll circular velocity
y_{15}	Car body roll angle
y_{16}	Car body roll circular velocity
y_{17}	Front wheel set vertical position
y_{18}	Front wheel set vertical velocity
y_{19}	Rear wheel set vertical position
y_{20}	Rear wheel set vertical velocity
y_{21}	Front wheel set roll angle
y_{22}	Front wheel set roll circular velocity
y_{23}	Rear wheel set roll angle
y_{24}	Rear wheel set roll circular velocity
y_{25}	Bogie frame vertical position
y_{26}	Bogie frame vertical velocity
y_{27}	Bogie frame pitch angle
y_{28}	Bogie frame pitch circular velocity
y_{29}	Rolling constraint β_1 (front wheel set)
y_{30}	Rolling constraint β_2 (rear wheel set)
y_{31}	Distance driven

Table 3.2: Train dynamics as generated by the simulation program. This table was adapted from [3] and the source code of the program.

3.3 The reference train

From a physical perspective, every train vehicle in the real world is unique and has therefore a different driving behaviour. The driving conduct depends on the exerting forces from the environment and the vehicle components, as well as on the masses and moments of inertia of the bogie elements. Overall, a train is modelled as a set of 24 different parameters. The important parameters regarding the springs and dampeners, as well as the corresponding initialisation values of the reference train train_0, are listed in Table 3.3. The origin of the values and more parameter information can be found in [4].

$k_1 = 1823$ kN/m	$k_5 = 333.3$ kN/m
$k_2 = 3646$ kN/m	$k_6 = 2710$ kN/m
$k_3 = 3646$ kN/m	$D_1 = 20$ kNs/m
$k_4 = 182.3$ kN/m	$D_2 = 29.2$ kNs/m

Table 3.3: String and dampening constants

3.4 The train variations

Since this project focuses on the strings and dampeners of the bogie frame, multiple train instances with slightly different parameter initialisations are needed. These train instances can then be compared. Therefore, six variants of the reference train are created. train_{1-4} have a single spring parameter increased or decreased by 5 %. train_{5-6} have all six spring constants and the two dampening coefficients reduced and augmented by 5 %, respectively. As a result, train_{1-4} constitute the group of the trains with small parameter changes and train_{5-6} adhere to the trains with big parameter alterations. The exact parameter adjustments are shown in Table 3.4. Given the reference train, this leads to a collection of seven distinct trains.

Train instance	**Parameter change**
train_0	N/A
train_1	$k_1 \to k_1 \cdot 0.95$
train_2	$k_1 \to k_1 \cdot 1.05$
train_3	$k_2 \to k_2 \cdot 0.95$
train_4	$k_2 \to k_2 \cdot 1.05$
train_5	$k_{1-6}, D_{1-2} \to k_{1-6} \cdot 0.95, D_{1-2} \cdot 0.95$
train_6	$k_{1-6}, D_{1-2} \to k_{1-6} \cdot 1.05, D_{1-2} \cdot 1.05$

Table 3.4: Train parameter changes

3.5 Implementation details

All seven trains are simulated on the previously generated eight synthetic track segments. This gives a job count of 56. The segments are numerated from seed_1 through seed_8. The simulation program inputs the respective track data files. The simulation output includes on the one hand the 30-dimensional dynamics data and on the other hand the eight lateral and vertical displacements of the left and right front and rear wheels. It contains additional information, which is not pertinent to this project. Every single train is scheduled to run for 32 km while outputting the dynamics and irregularities at a frequency of 200 Hz [10]. The sampling interval and number of samples are given by equations 3.1 and 3.2. As the simulation commences at 50 m, the scheduled work leads to 191,700 samples with a density of 6 data points per metre.

$$\text{sampling interval} = \frac{\text{velocity}}{\text{sampling frequency}} \tag{3.1}$$

$$\text{number of samples} = \frac{\text{track length} - 50\ \text{m}}{\text{sampling interval}} \tag{3.2}$$

However, train simulation stops can occur when there is a difference between the wheel and the rail that is greater or equal to 17 mm. This is interpreted as a derailment. As a matter of fact, 23 out of the 56 simulations did not complete, amounting to 59 % of successful simulations. In comparison, the total simulation work sums up to 8,419,203 data samples which results in 78.4 % of the scheduled work. This does not imply that 21.6 % of the scheduled simulation data is useless, instead, it was never generated.

Chapter 4

Machine learning

4.1 Overview

This chapter explains the experimental procedure around the machine learning models used. First of all, machine learning is categorised into three different areas, each having its own purposes and use cases. These areas are called supervised learning, unsupervised learning and reinforcement learning. The underlying problem of predicting rail irregularities lies in the supervised learning setting. This generally means that a model is trained on labeled data which is then used to predict the labels of new, previously unseen data.

Convolutional neural networks are introduced and applied to the dynamics data generated in Chapter 3. More precisely, there are two differently sized models which are trained on the dynamics of some train and tested on some other train's dynamics. The model outputs are compared to assess if the size and complexity of the model influence the prediction precision and to what extent a model is even capable of predicting the irregularities based on another train's dynamics due to a parameter change.

The software in this part is completely implemented in Python, version 3.11. The code skeleton was made available. The primary libraries include NumPy, Pandas, Matplotlib, Scikit-learn and PyTorch [11]. Model training was conducted using CUDA [12] on GPU nodes within the computer cluster of the Distributed Computing Group.

4.2 Motivating example

Before introducing the more complex machine learning models, the linear regression setting is presented first. To recall, the simulation data provides the 30-dimensional train dynamics data serving as input for the linear regression model. The model output is the set of the lateral and vertical displacements of the left and right front and rear wheels.

The model implements the ordinary least squares linear regression. It is fitted on the dynamics with the corresponding eight-dimensional labels from train_0 on track seed_1. The same model is then used to predict the labels of its training data. This means the model predicts the labels of the data that is used to estimate its coefficients. As an illustration, a 500-sample-long part of the model's prediction, starting at position 90,000 for no particular reason, is plotted against the true labels in Figure 4.1. The four plots show the lateral and vertical displacements of the front wheels only. The rear wheel plots are omitted because they display similar characteristics. It can be seen that the linear model works acceptably well in predicting the vertical irregularities of the rails. However, it is not capable of predicting the lateral values at all. The error statistics show that the mean absolute error is around 1.8 mm for the lateral labels and around 0.8 mm for the vertical labels.

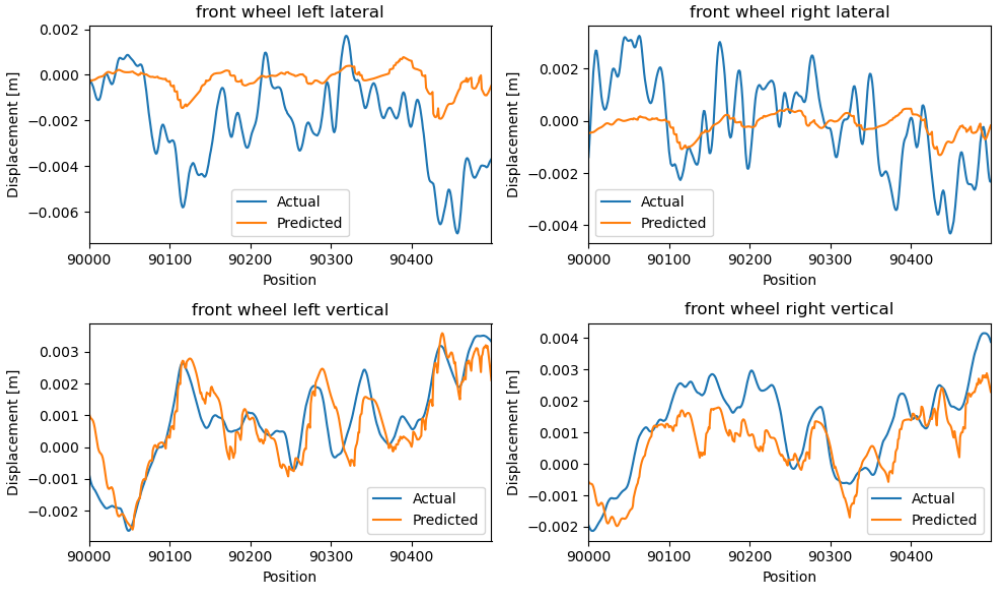

Figure 4.1: Linear regression prediction showing the displacements along a 500-sample-long track segment

Moreover, for the prediction to be useful, the precision needs to be increased up to a mean error of the order of 0.1 mm. This benchmark was already used in [3]. Considering the weakness of learning the lateral irregularities, another machine model architecture, capable of extracting the lateral behaviour, is needed. This is where CNNs demonstrate their utility. As mentioned in Chapter 1.4, CNNs have been proven to be able to predict the vertical as well as the lateral irregularities of the rails.

4.3 Data batches

Before delving into CNNs, the data pre-processing steps are outlined first. In general, the available data is divided into batches of equal length. This optimises the memory efficiency as well as the gradient calculation when using for example the stochastic gradient descent technique. This is also useful for batch processing and enables parallelism. The structure of the collection of training data files generated in this project leads to two noteworthy design decisions. The first decision details the partitioning of a single data file into batches while the second decision explains the allocation of the resulting batches for training, validation and testing of the machine learning models.

4.3.1 Batch generation optimisation

First and foremost, the opening 2 km of the train dynamics are affected by a transient. They are therefore ignored when splitting the data into segments [5].

Next, the data can be split in two distinct ways. Consider therefore a data set of 100 samples numbered x_1, x_2, ..., x_{100} with labels y_1, y_2, ..., y_{100} and a segment length of 10. The naive way is to split the data x_{1-10}, x_{11-20}, ..., x_{91-100} with labels y_{1-10}, y_{11-20}, ..., y_{91-100}. This batch generation method can however be optimised for CNNs to obtain more batches from the same data. Due to the nature of a convolution, a reduction of the input size, there are fewer labels than samples in a batch. For a reduction of 4, this means x_{1-10} would only map to a subset of y_{1-10}, namely y_{3-8}. Instead of the next segment mapping x_{11-20} to y_{13-18}, the segment interval can be shifted to the left by 4 to seamlessly connect the y values. The segments would then include x_{7-17} and y_{9-14}. In other words, the data segment split is done by sliding the reduced selection window for y over the labels and selecting the corresponding x values to form a batch. The two approaches are summarised in Table 4.1 where the CNN has a dimension reduction of 4 and the basic neural network (NN) represents the counterpart without convolution.

In conclusion, the generation of batches from a given dataset depends on the type of neural network and convolution used. The number of segments is given by equation 4.1 in the non-convolutional case and by equation 4.2 in the convolutional case.

$$\text{number of segments} = \left\lfloor \frac{\text{number of samples}}{\text{segment length}} \right\rfloor \tag{4.1}$$

$$\text{number of segments} = \left\lfloor \frac{\text{number of samples} - \text{reduction}}{\text{segment length} - \text{reduction}} \right\rfloor \tag{4.2}$$

	NN	CNN
Data segments	(x_{1-10}, y_{1-10}) (x_{11-21}, y_{11-21}) ... (x_{91-100}, y_{91-100})	(x_{1-10}, y_{3-8}) (x_{7-17}, y_{9-14}) ... (x_{91-100}, y_{93-98})
Number of segments	10	16

Table 4.1: The data split of 100 samples, using a segment length of 10, is shown for a non-convolutional neural network (NN) and a convolutional neural network (CNN) with a reduction of 4.

4.3.2 Seed split

As described in Chapter 3, for every train variation there is data available from eight different seeds. These eight seeds are split conceptually into a training, a validation and a test data pool to guarantee a model is not validated nor tested on data it has used for training. Since the required volume of training data is larger than the volume for validation and testing, the seeds are split according to Table 4.2. In summary, this means all batches originating from the dynamics data files that have been simulated using track segments defined by seed_{1-5} are used for training.

	Training pool	Validation pool	Testing pool
Seeds	1, 2, 3, 4, 5	6, 7	8

Table 4.2: Seed split

All files that contain fewer than 50,000 samples are not considered for providing data batches [10]. For this reason, 7 out of 56 files are ignored.

4.4 Convolutional neural networks

CNNs represent a subclass of deep neural networks and are often used for image processing. They are the main tool in this part of the project. They have multiple hidden layers, each applying a convolution and a non-linear activation function.

There are two different neural networks used throughout this part of the project. They are called Model 1 and Model 2 and only differ in size and kernel lengths. Therefore, they have a different number of learnable parameters resulting in distinct learning behaviours. The design specifics are explained below.

First of all, the neural networks' input layer has 30 channels that correspond to the 30 train dynamics' metrics. The output layer has eight channels, corresponding to the displacements of the wheels. Next, the two models have three

and four hidden layers, respectively. The hidden layers apply a one-dimensional convolution, followed by the ReLu activation function. The kernel size of the first hidden layer was chosen such as to detect wavelengths of around 100 m. With a spacing of $\frac{1}{6}$ m between every data point, this gives a kernel size of 601 samples. The last network layer's convolutions of both Model 1 and Model 2 have a kernel length of 3. Note that kernel size and kernel length are used interchangeably because the convolution is applied in one dimension only, the longitudinal direction of the rails to be specific. The exact differences between the two architectures are described in Table 4.3 where the list notation describes the features of the hidden layers in order. A CNN has additional stride and padding parameters. Since the neural network should give predictions for every sample, the stride is 1 and the padding equals 0.

Since the largest kernel length equals 801, the size of a training batch needs to be significantly greater. The segment size is set to 12,000 samples which represents a track segment length of 2 km. Moreover, training a neural network involves an optimiser and a loss function. The Adam optimiser and the mean squared error (MSE) loss function are selected as they are widely recognised and well-suited to the problem context of this project. The model is trained for exactly 3,000 epochs. Additionally, a learning rate scheduler is used to decrease the optimiser's learning rate with increasing number of epochs. The scheduler kicks off with a learning rate of 10^{-4} and reduces it by a factor of 0.33, meaning multiplying it by 0.33, every 750 epochs. The models use a batch size of 1 which means that stochastic gradient descent (SGD) is employed for updating the model parameters. Finally, the model is validated every 10 epochs. All mentioned parameters are inspired by [5, 10] and hand-tuned by trial and error.

	Model 1	**Model 2**
Number of hidden layers	3	4
Number of channels in hidden layers	[16, 48, 32]	[256, 256, 64, 32]
Convolution kernel lengths	[601, 7, 7]	[801, 9, 7, 7]

Table 4.3: Model differences for Model 1 and Model 2. The list notation describes the hidden layers in order. This means Model 1's first of three hidden layers has 16 channels and uses a convolution of kernel length 601.

4.5 Training

From now on, in the phrase 'the model is trained on the dynamics of train$_i$ and tested on the dynamics of train$_j$', the subphrase 'the dynamics of' is omitted to enhance readability and improve the flow of the text.

The model training is conducted as described in Chapter 4.4. Model 1 and Model 2 are both trained and validated on the same trains, so as to compare

their respective training performances.

As previously mentioned, Model training is conducted on GPU nodes of the computer cluster. Using Model 1, training takes about 30 minutes and using Model 2, the duration ranges from 1 to 4 hours, depending on the node's processing power.

Both models are trained with the dynamics data of every single one of the seven trains. The focus lies on the models that are trained with the reference train. The training losses of all 14 cases are listed in Table 4.4. A row in the table represents the final training loss when the model is trained with the according train. Additionally, the training and validation loss curves of Model 1 and Model 2, trained on train_0 for 3,000 epochs, are plotted in Figures 4.2 and 4.3. Both plots serve as a proxy for all models trained with a single train because the respective convergence patterns resemble each other.

On the one side, the training loss of Model 1 reduces to roughly $2{\cdot}10^{-6}$ m^2 and has a validation loss of approximately $3 \cdot 10^{-6}$ m^2. The model's learning process plateaus very quickly from epoch 250 onwards. The learning rate adjustment points at epochs 750 and 1,500 can be identified because the variance of the loss peaks decreases. On the other side, Model 2 training performs better and attains a training loss of roughly $4 \cdot 10^{-7}$ m^2 while having a validation loss of approximately $5{\cdot}10^{-6}$ m^2. The training loss gradually decreases until it starts to plateau at epoch 1,500. It is important to notice that in this case, the validation loss is significantly larger than the attained training loss. Again, the locations of the learning rate updates can be identified. Contrary to what happens during the training of Model 1, the validation loss decreases in the beginning to a minimum of around $4{\cdot}10^{-6}$ m^2 at epoch 250 when it starts to increase again up to a value of approximately $5{\cdot}10^{-6}$ m^2. To conclude, there is no noticeable difference between the usage of the dynamics of train_{0-6} for model training.

Model 1	**Training loss [m^2]**
train_0	$1.90 \cdot 10^{-6}$
train_1	$1.81 \cdot 10^{-6}$
train_2	$1.94 \cdot 10^{-6}$
train_3	$1.94 \cdot 10^{-6}$
train_4	$2.00 \cdot 10^{-6}$
train_5	$1.94 \cdot 10^{-6}$
train_6	$1.87 \cdot 10^{-6}$

Model 2	**Training loss [m^2]**
train_0	$3.89 \cdot 10^{-7}$
train_1	$4.37 \cdot 10^{-7}$
train_2	$4.08 \cdot 10^{-7}$
train_3	$3.74 \cdot 10^{-7}$
train_4	$4.31 \cdot 10^{-7}$
train_5	$3.92 \cdot 10^{-7}$
train_6	$3.58 \cdot 10^{-7}$

Table 4.4: Training losses are shown for Model 1 and Model 2, each trained with the dynamics of a single train. The row indicates which train's dynamics are used for model training and the resulting training loss.

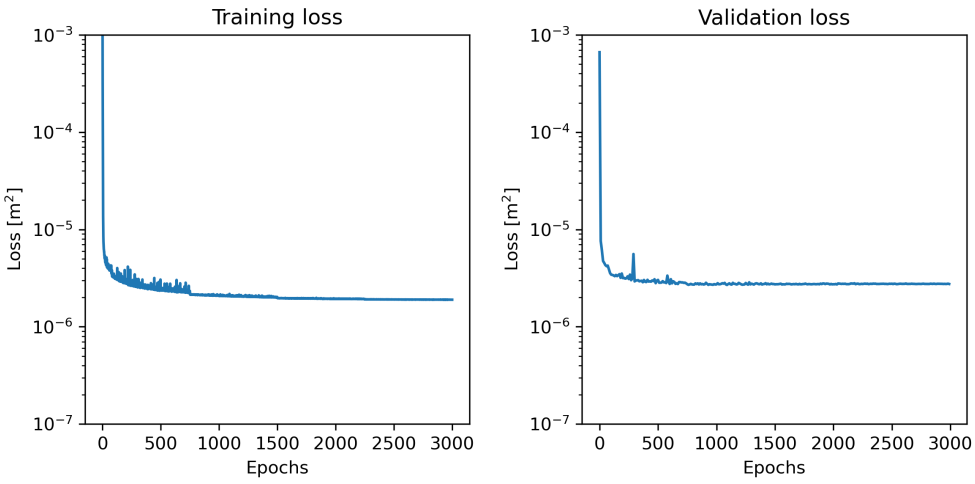

Figure 4.2: Training and validation losses are shown for Model 1, trained on train_0, in relation to the epochs.

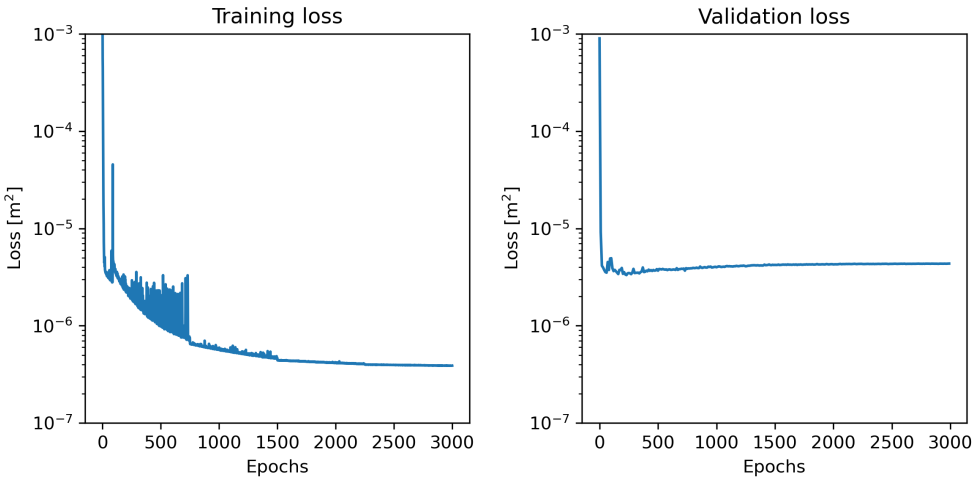

Figure 4.3: Training and validation losses are shown for Model 2, trained on train_0, in relation to the epochs.

Unfortunately, the levels of the training and validation losses of the neural networks do not meet the precision of the models attained in prior work [3, 5]. The training and validation loss mismatch of the Model 2 models signifies that the model is overfitting the training data which means it cannot make valuable predictions on new data. This is a serious problem because the model should

be able to predict the rail irregularities precisely. Therefore, the future decisions and conclusions are based on the exact numbers instead of the precision of the prediction that can be visualised in plots, while always referring to this problem.

4.6 Testing

After training 14 different models, they are evaluated. Model 1 and Model 2 are again tested on the same trains to be able to compare the prediction performances.

The evaluation is done by testing the model trained with the reference train, with every other train individually. This is the main result of this project, as it shows if and how well a model can predict the irregularities of the rails given the dynamics of another train. Then, the constellation is reversed, meaning all models trained with train_{1-6} individually are tested on train_0. This step verifies that the predictive capability is of the same level as the other way round. Finally, all models trained with train_i are evaluated on train_i to verify that the models have effectively learned to map their own dynamics to irregularities. All test outcomes are depicted in Table 4.5 where the row indicates the model from Chapter 4.5 that is tested on a train's dynamics, which is in turn given by the column. For example, the entry in row train_4 and column train_0 provides the mean errors of the prediction of the displacements from train_0 of the model trained with train_4.

Based on the results, it is clear that the model trained with train_0 is able to predict the irregularities based on train_{1-4}. The precision is of the exact same quality as the prediction of the displacements from train_0. However, the model does not predict the displacements based on train_{5-6} as precisely. Looking at the reversed constellation, the outcomes is mirrored. This means all models trained with train_{1-4} predict the displacements from train_0 as precisely as their own. Again, the models trained with train_{5-6} experience an error increase when predicting the displacements from train_0, although the mean vertical error is not as elevated. The diagonal values prove that the models learn the mapping from train dynamics to irregularities best for the train dynamics they are trained with, putting into perspective the fact that all error outcomes are nearly identical for all models trained and tested with train_{0-4} and only models using train_{5-6} have cross-train prediction difficulties. All observations made count for Model 1 and Model 2. To be precise, for Model 1, the mean lateral error is roughly 1.72 mm and the mean vertical error is approximately 0.58 mm. For Model 2, the mean lateral error is roughly 2.12 mm and the mean vertical error is approximately 0.61 mm. This means Model 2 performs worse, as there is a relatively large drop in the lateral precision and a tiny increase in the mean vertical error. Furthermore, it is important to notice that for Model 1, the error increase in the cross-train prediction of the displacements from train_{5-6} is much larger in the vertical than in the lateral direction. The prediction results average 2.35 mm in the lateral and

2.20 mm in the vertical direction. For Model 2, the prediction results average 3.49 mm in the lateral and 1.62 mm in the vertical direction. This does not hold true for the prediction of the displacements from train_0 using a model trained with train_{5-6} nor for the same tests using Model 2.

As an illustration of the observations made above, the predictions from Model 1 and Model 2 with training data being the dynamics of train_0 using test data derived from train_0 and train_6 are plotted in Figures 4.8, 4.9, 4.10 and 4.11, respectively. The figures are moved to Chapter 4.8 to enhance the comparative analysis. Setting the predictions for train_0 and train_6 using either model side by side, it can be observed that the latter predictions fluctuate more.

The conclusions of the test results need to be approached with care because, as already explained, the models overfit during training and attain a worse training error than the models used in prior work [3, 5]. However, it seems evident that the models using train_0's dynamics as training data can predict the irregularities from train_{1-4} as precisely as from their own training data, but are not capable of predicting the irregularities from train_{5-6} because of their large parameter changes.

Model 1	train_0	train_1	train_2	train_3	train_4	train_5	train_6
train_0	1.69	1.68	1.67	1.70	1.68	2.31	2.39
	0.58	0.57	0.57	0.59	0.56	2.37	2.03
train_1	1.76	1.76	—	—	—	—	—
	0.59	0.58	—	—	—	—	—
train_2	1.70	—	1.69	—	—	—	—
	0.56	—	0.56	—	—	—	—
train_3	1.75	—	—	1.75	—	—	—
	0.60	—	—	0.60	—	—	—
train_4	1.77	—	—	—	1.73	—	—
	0.58	—	—	—	0.56	—	—
train_5	2.41	—	—	—	—	1.73	—
	0.94	—	—	—	—	0.62	—
train_6	2.11	—	—	—	—	—	1.75
	0.81	—	—	—	—	—	0.59

Model 2	train_0	train_1	train_2	train_3	train_4	train_5	train_6
train_0	2.10	2.09	2.11	2.14	2.14	2.95	4.03
	0.61	0.60	0.61	0.62	0.59	1.64	1.59
train_1	2.15	2.11	—	—	—	—	—
	0.61	0.61	—	—	—	—	—
train_2	2.12	—	2.12	—	—	—	—
	0.60	—	0.60	—	—	—	—
train_3	2.16	—	—	2.17	—	—	—
	0.65	—	—	0.64	—	—	—
train_4	2.08	—	—	—	2.11	—	—
	0.62	—	—	—	0.62	—	—
train_5	2.43	—	—	—	—	2.19	—
	1.38	—	—	—	—	0.64	—
train_6	2.78	—	—	—	—	—	2.13
	1.64	—	—	—	—	—	0.64

Table 4.5: Evaluations are shown for Model 1 and Model 2, each trained with the dynamics of a single train. The row indicates which train's dynamics are used for model training while the column indicates which train's dynamics are used for testing. The upper (white) and lower (gray) values correspond to the mean lateral and vertical errors, respectively. Units are in millimetres. Note that '—' marks the absence of a value.

4.7 Improvements

There are two methods presented that could improve the prediction of the rail irregularities. The first method uses the idea that the model learns the mapping from train dynamics to irregularities in a broader way by using training data consisting of the dynamics of multiple trains. In this way, the model could learn a mapping that generalises more trains with diverse parameter changes. Another idea is to fine-tune the models that are pre-trained in Chapter 4.5 to adapt to the influence of the parameter change in the train specification.

4.7.1 Generalised training

The settings for model training are exactly the same as before.

The two models Model 1 and Model 2 are used again. This time, they are trained with a larger training set, consisting of all of the train dynamics data from multiple trains. It was chosen to train the models twice, once using the combined data from train_{0-2} which is called $\text{train}_{0,1,2}$ and once using the data from train_{0-2} and train_{5-6} which is in turn called $\text{train}_{0,1,2,5,6}$. The training losses of the resulting four models are listed in Table 4.6. Again, a row in the table represents the final training loss when the model is trained with the accumulated dynamics from the according trains. Exemplary plots of the training and validation loss curves of Model 1 and Model 2 trained on $\text{train}_{0,1,2,5,6}$ can be seen in Figures 4.4 and 4.5. The learning progress portrayed in the plot resembles the learning behaviour of the other two models trained using the dynamics of $\text{train}_{0,1,2}$.

Both Model 1 instances attain training loss minima of $1.92 \cdot 10^{-6}$ m^2 and $1.95 \cdot 10^{-6}$ m^2, respectively. Their validation losses are of the same level, converging to $2 \cdot 10^{-6}$ m^2. This means both Model 1 instances are comparable to the Model 1 instances explained in Chapter 4.5, only having a slight increase in training loss of $3.5 \cdot 10^{-8}$ m^2 on average and a small, but noticeable decrease in validation loss. Model 2 instances however perform not as well as their predecessors, only attaining a training loss of $6.32 \cdot 10^{-7}$ m^2 and $7.72 \cdot 10^{-7}$ m^2, respectively. Their validation losses are again around $4 \cdot 10^{-6}$ m^2 and hence significantly larger than the training loss. It is worth noticing that the training loss variance peaks are not as large as those in Chapter 4.5 as there is 3 to 5 times more training data available, leading to a better gradient estimation. The learning rate update points can again be identified at epoch 750.

Model 1	Training loss [m^2]
Recap $train_0$	$1.90 \cdot 10^{-6}$
$train_{0,1,2}$	$1.92 \cdot 10^{-6}$
$train_{0,1,2,5,6}$	$1.95 \cdot 10^{-6}$

Model 2	Training loss [m^2]
Recap $train_0$	$3.89 \cdot 10^{-7}$
$train_{0,1,2}$	$6.32 \cdot 10^{-7}$
$train_{0,1,2,5,6}$	$7.72 \cdot 10^{-7}$

Table 4.6: Training losses are shown for Model 1 and Model 2, each trained with the dynamics of multiple trains. The row indicates which trains' dynamics are used for model training and the resulting training loss. 'Recap' means restatement of the results from Table 4.4 in Chapter 4.5.

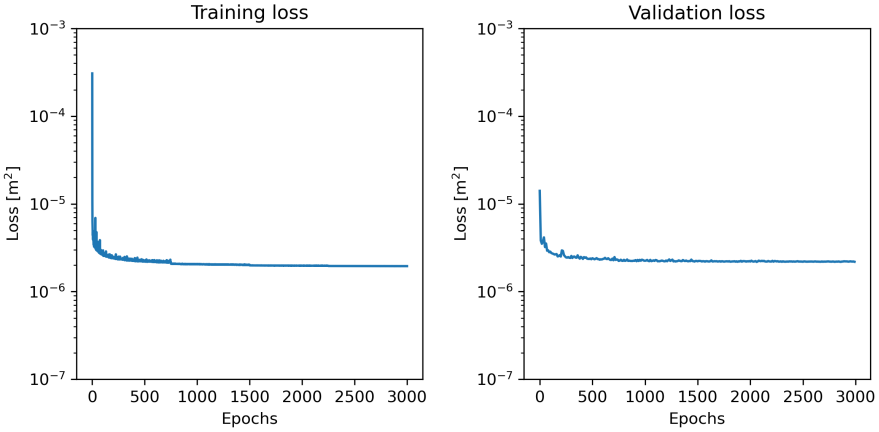

Figure 4.4: Training and validation losses are shown for Model 1, trained on $train_{0,1,2,5,6}$, in relation to the epochs.

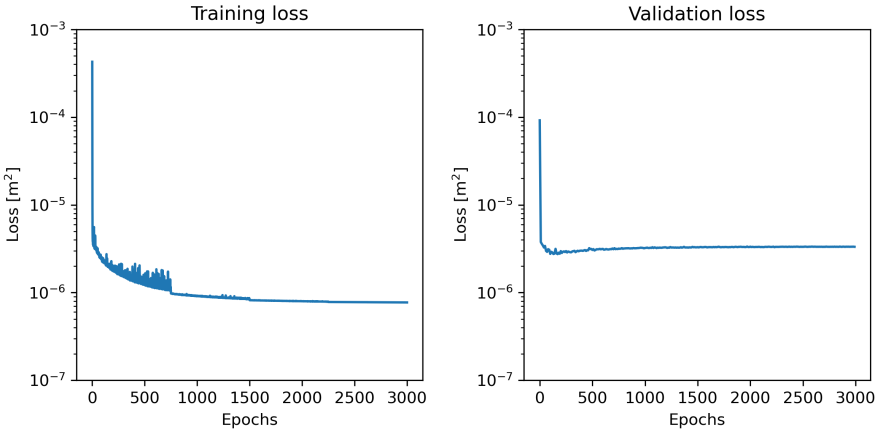

Figure 4.5: Training and validation losses are shown for Model 2, trained on $train_{0,1,2,5,6}$, in relation to the epochs.

To analyse if the prediction quality changes in any way, the four models are tested on all seven trains. This leads to a total of 28 tests. The final mean lateral and vertical errors can be seen in Table 4.7 where the row defines the train whose dynamics are used as training data and the column defines the train the model is tested on.

In general, both Model 1 and Model 2 instances perform better than the models trained with a single train's dynamics. Moreover, the models trained with $\text{train}_{0,1,2,5,6}$ outperform the model trained with $\text{train}_{0,1,2}$. The former not only decrease the mean errors for train_{5-6} compared to the latter considerably because those train's dynamics are new part of the training data, but it reduces the mean errors of all predictions by 0.07 mm in lateral and 0.01 mm in vertical direction on average in the Model 1 case. The Model 2 instances confirm this observation. However, there is no difference in the prediction precision from train_{3-4} compared to those from train_{0-2}. It is also clear that Model 1 instances are superior to Model 2 instances because their mean lateral errors are around 0.35 mm smaller, although their mean vertical errors are identical, which was not the case with single-train dynamics training.

The visualisation of the prediction of Model 1 and Model 2, trained with $\text{train}_{0,1,2,5,6}$ using train_3 can be seen in Figures 4.12 and 4.13. train_3 plays the role of the unknown train and is used to test the models on a new parameter combination. The figures are again moved to Chapter 4.8 to enhance the comparative analysis.

Model 1	train_0	train_1	train_2	train_3	train_4	train_5	train_6
Recap train_0	1.69	1.68	1.67	1.70	1.68	2.31	2.39
	0.58	0.57	0.57	0.59	0.56	2.37	2.03
$\text{train}_{0,1,2}$	1.56	1.55	1.56	1.58	1.56	2.06	2.29
	0.54	0.53	0.53	0.54	0.52	0.89	1.39
$\text{train}_{0,1,2,5,6}$	1.49	1.49	1.48	1.50	1.49	1.50	1.49
	0.53	0.52	0.52	0.53	0.51	0.54	0.53

Model 2	train_0	train_1	train_2	train_3	train_4	train_5	train_6
Recap train_0	2.10	2.09	2.11	2.13	2.14	2.95	4.03
	0.61	0.60	0.61	0.62	0.59	1.64	1.59
$\text{train}_{0,1,2}$	1.92	1.89	1.90	1.93	1.96	2.31	2.33
	0.54	0.53	0.54	0.55	0.52	1.51	1.49
$\text{train}_{0,1,2,5,6}$	1.82	1.81	1.83	1.85	1.82	1.86	1.83
	0.52	0.52	0.53	0.53	0.51	0.54	0.54

Table 4.7: Evaluations are shown for Model 1 and Model 2, each trained with the dynamics of multiple trains. The row indicates which train's dynamics are used for model training while the column indicates which train's dynamics are used for testing. The upper (white) and lower (gray) values correspond to the mean lateral and vertical errors, respectively. Units are in millimetres. 'Recap' means restatement of the results from Table 4.5 in Chapter 4.6.

4.7.2 Fine-tuning

The setting for model fine-tuning is similar to the setting for normal model training but includes some minor changes. The differences are that model training is continued with different data, the number of epochs is fixed at 100 instead of 3,000 and the learning rate scheduler updates the learning rate after 50 epochs instead of 750. This means there is only one such update.

To evaluate the fine-tuning strategy, the trade-off between the amount of training data used for fine-tuning the model and the resulting prediction precision is analysed. Therefore, Model 1 and Model 2, both trained with train_0, are considered the pre-trained models. These are then fine-tuned to train_{1-6} according to the training parameter adjustments already mentioned, three times, each with a different volume of training data. Each model is trained once with only 1 random data batch, once with exactly 10 data batches and another time with all available data batches. The number of available dynamics data batches ranges between 50 and 70 which describes an increase by a factor of 5 to 7 compared to the 10-batch scenario. The training losses of the resulting 36 model fine-tuning constellations are listed in Table 4.8. To recall, the pre-trained model used train_0 for training, the row only indicates the train chosen for fine-tuning. The training and validation loss curves for Model 1 and Model 2, both pre-trained with train_0

and fine-tuned with train_1 using 10 batches, are plotted in Figures 4.6 and 4.7, respectively.

First, it is evident that the training loss strictly relates to the amount of training data used. For Model 1, the average training loss of the fine-tuning step is $1.19 \cdot 10^{-6}$ m^2 using 1 batch, $1.89 \cdot 10^{-6}$ m^2 using 10 batches and $2.17 \cdot 10^{-6}$ m^2 using all available batches. Model 2 fine-tuning experiences the same conceptual loss increase and confirms the observation. The validation loss for both model architectures remains at the same level at approximately $3 \cdot 10^{-6}$ m^2. Taking a closer look, the Model 1 models using 1 batch for fine-tuning beat the pre-training loss by a huge margin while Model 2 models using 1 batch for fine-tuning only match their corresponding pre-training loss. All this is a clear indication of overfitting and poor generalisation, as one batch does not contain enough information to make the model learn a good general mapping from train dynamics to wheel displacements.

Model 1	**Recap pre-training**	**1 batch**	**10 batches**	**all batches**
train_1	$1.81 \cdot 10^{-6}$	$1.15 \cdot 10^{-6}$	$1.69 \cdot 10^{-6}$	$2.10 \cdot 10^{-6}$
train_2	$1.94 \cdot 10^{-6}$	$1.17 \cdot 10^{-6}$	$2.00 \cdot 10^{-6}$	$2.17 \cdot 10^{-6}$
train_3	$1.94 \cdot 10^{-6}$	$1.24 \cdot 10^{-6}$	$1.97 \cdot 10^{-6}$	$2.18 \cdot 10^{-6}$
train_4	$2.00 \cdot 10^{-6}$	$1.26 \cdot 10^{-6}$	$1.94 \cdot 10^{-6}$	$2.31 \cdot 10^{-6}$
train_5	$1.94 \cdot 10^{-6}$	$1.22 \cdot 10^{-6}$	$1.86 \cdot 10^{-6}$	$2.18 \cdot 10^{-6}$
train_6	$1.87 \cdot 10^{-6}$	$1.09 \cdot 10^{-6}$	$1.85 \cdot 10^{-6}$	$2.14 \cdot 10^{-6}$

Model 2	**Recap pre-training**	**1 batch**	**10 batches**	**all batches**
train_1	$4.37 \cdot 10^{-7}$	$3.89 \cdot 10^{-7}$	$7.93 \cdot 10^{-7}$	$1.19 \cdot 10^{-6}$
train_2	$4.08 \cdot 10^{-7}$	$3.78 \cdot 10^{-7}$	$8.97 \cdot 10^{-7}$	$1.34 \cdot 10^{-6}$
train_3	$3.74 \cdot 10^{-7}$	$4.20 \cdot 10^{-7}$	$9.18 \cdot 10^{-7}$	$1.49 \cdot 10^{-6}$
train_4	$4.31 \cdot 10^{-7}$	$4.03 \cdot 10^{-7}$	$9.59 \cdot 10^{-7}$	$1.39 \cdot 10^{-6}$
train_5	$3.92 \cdot 10^{-7}$	$4.00 \cdot 10^{-7}$	$9.24 \cdot 10^{-7}$	$1.38 \cdot 10^{-6}$
train_6	$3.58 \cdot 10^{-7}$	$4.12 \cdot 10^{-7}$	$8.28 \cdot 10^{-7}$	$1.27 \cdot 10^{-6}$

Table 4.8: Training losses are shown for Model 1 and Model 2, each pre-trained with the dynamics of train_0 and fine-tuned with the dynamics of a single train using various amounts of data. The row indicates which train's dynamics are used for model fine-tuning and the resulting training losses. Units are in square meter. 'Recap' means restatement of the results from Table 4.4 in Chapter 4.5.

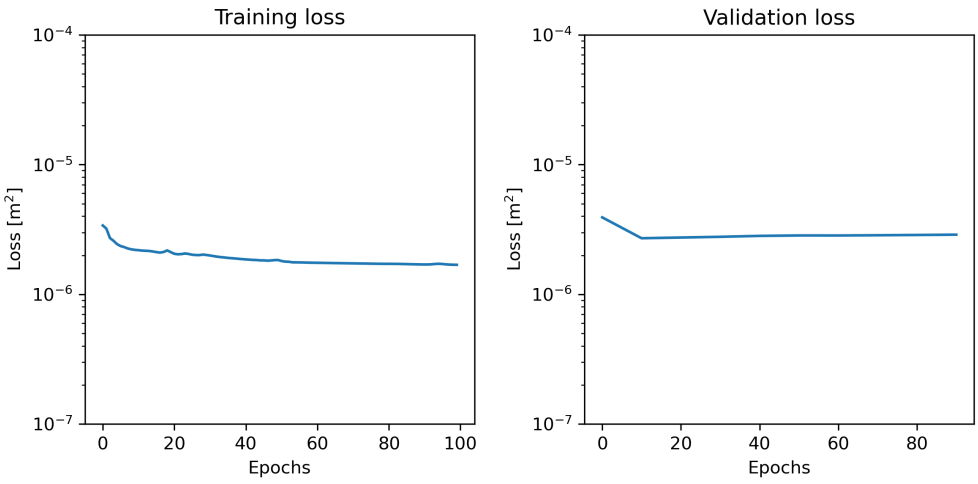

Figure 4.6: Training and validation losses are shown for Model 1, trained on train_0, fine-tuned on train_1 using 10 data batches, in relation to the epochs.

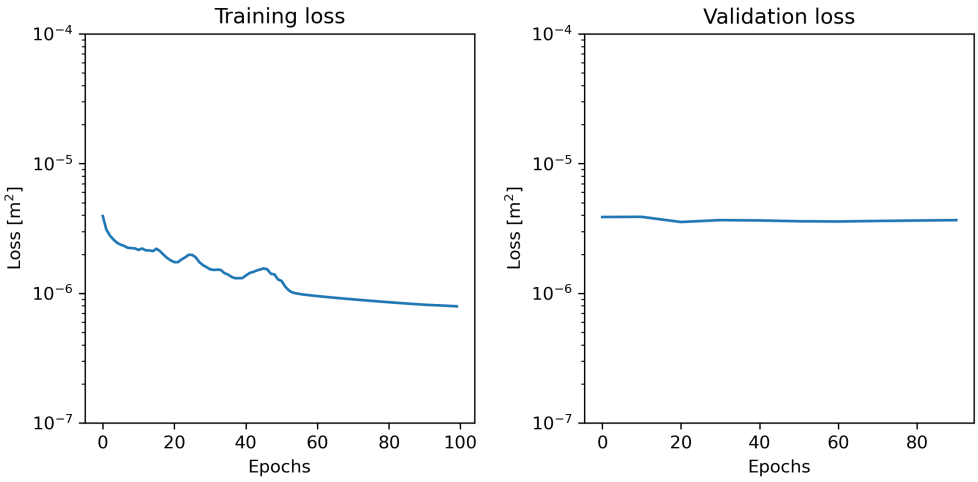

Figure 4.7: Training and validation losses are shown for Model 2, trained on train_0, fine-tuned on train_1 using 10 data batches, in relation to the epochs.

Finally, the fine-tuned models are evaluated. All of them are tested on the same train's dynamics they have used for fine-tuning. This means a model fine-tuned with train_i is also tested on train_i. With the pre-trained models being Model 1 and Model 2 trained with train_0, they are both fine-tuned on the other 6 trains using 1, 10 and all available data batches. This leads to 36 different

testing scenarios. The outcome and the errors are all listed in Table 4.9.

There are two universal trends in the analysis of the error statistics. First, there is an error decrease when increasing the amount of training data for fine-tuning. This shows that more data helps the model learn a better mapping from train dynamics to wheel displacements and keeps the model from overfitting a fraction of the data. Overall, if all available batches are used for fine-tuning, the final prediction errors beat the errors after the pre-training stage. This observation is noticed for both Model 1 and Model 2, while Model 2 instances show a larger mean error drop compared to the Model 1 instances. Second, the fine-tuned models having the Model 1 architecture perform better in predicting the irregularities. Their mean lateral errors are roughly 0.2 mm lower than those of their Model 2 equivalents and the mean vertical errors are on the same level. Finally, it is important to mention that, after fine-tuning to train_{5-6}, the respective mean lateral and vertical errors drop to the same level as those after fine-tuning to the other trains. This counts for both Model 1 and Model 2. This leads to the conclusion that the fine-tuning method helps to overcome the cross-train prediction weakness for trains having large parameter changes and generally improves the prediction precision.

The visualisation of the predictions of Model 1 and Model 2, pre-trained using train_0 and fine-tuned using 10 batches of train_1, can be seen in Figures 4.14 and 4.15. The figures are again moved to Chapter 4.8 to enhance the comparative analysis.

Model 1	Recap pre-training	1 batch	10 batches	all batches
$train_1$	1.68	1.86	1.75	1.65
	0.57	0.66	0.60	0.58
$train_2$	1.67	1.82	1.72	1.63
	0.57	0.63	0.60	0.58
$train_3$	1.70	1.86	1.77	1.65
	0.59	0.61	0.65	0.61
$train_4$	1.68	1.84	1.75	1.66
	0.56	0.68	0.62	0.59
$train_5$	2.31	2.02	1.76	1.68
	2.37	0.67	0.63	0.60
$train_6$	2.39	2.05	1.83	1.66
	2.03	0.91	0.62	0.60

Model 2	Recap pre-training	1 batch	10 batches	all batches
$train_1$	2.09	2.20	1.96	1.85
	0.60	1.00	0.62	0.56
$train_2$	2.11	2.12	1.98	1.87
	0.61	0.64	0.59	0.55
$train_3$	2.13	2.10	2.02	1.86
	0.62	0.65	0.62	0.56
$train_4$	2.14	2.10	1.95	1.88
	0.59	0.64	0.57	0.54
$train_5$	2.95	2.53	2.05	1.97
	1.64	1.39	0.64	0.60
$train_6$	4.03	3.72	2.03	1.83
	1.59	1.05	0.67	0.59

Table 4.9: Evaluations are shown for Model 1 and Model 2, each pre-trained with the dynamics of $train_0$ and fine-tuned with the dynamics of a single train. The row indicates which train's dynamics are used for model fine-tuning and testing while the column indicates the amount of training data used. The upper (white) and lower (gray) values correspond to the mean lateral and vertical errors, respectively. Units are in millimetres. 'Recap' means restatement of the results from Table 4.5 in Chapter 4.6.

4.8 Prediction plots

All prediction figures mentioned in Chapters 4.6 and 4.7 are listed below. Every figure includes four plots, namely the displacements of the front wheels of a bogie. The rear wheels are omitted because they show the same characteristics. The plots visualise a 500-sample-long interval of the track starting at position 40,000 of track segment $seed_8$. The choice regarding $seed_8$ is detailed in Chapter 4.3.2.

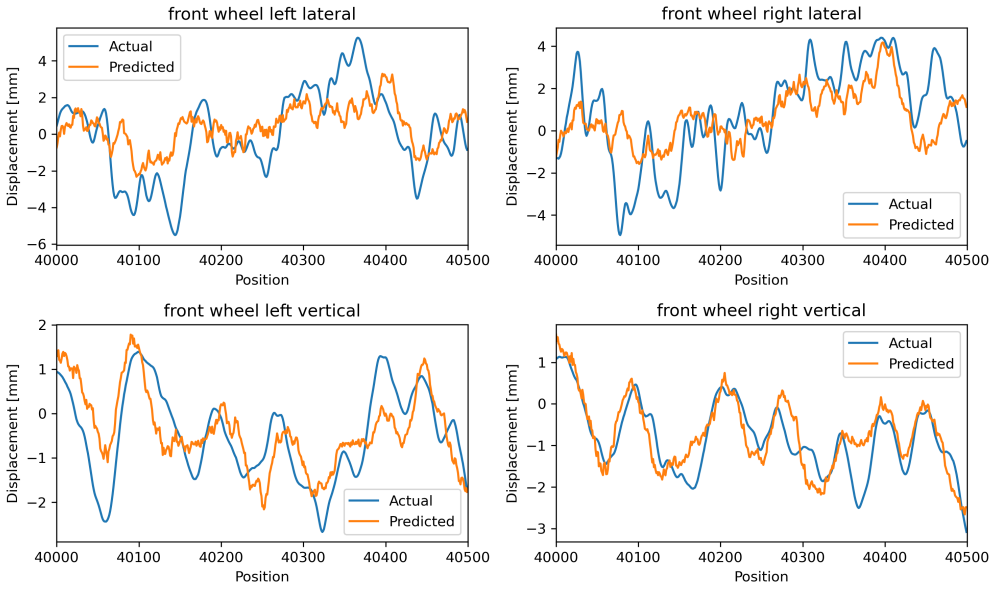

Figure 4.8: Model 1 trained on train_0 and tested on train_0

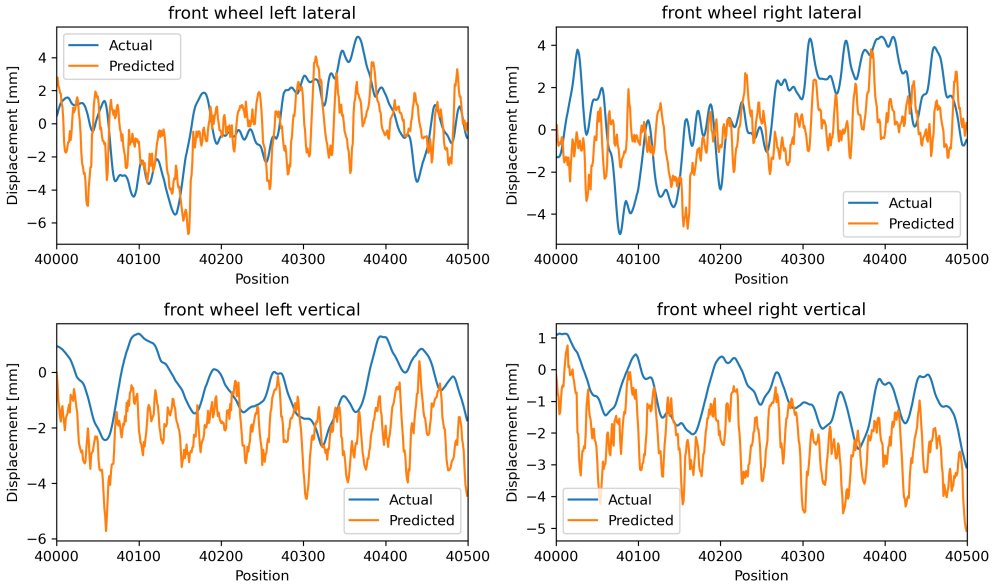

Figure 4.9: Model 1 trained on train_0 and tested on train_6

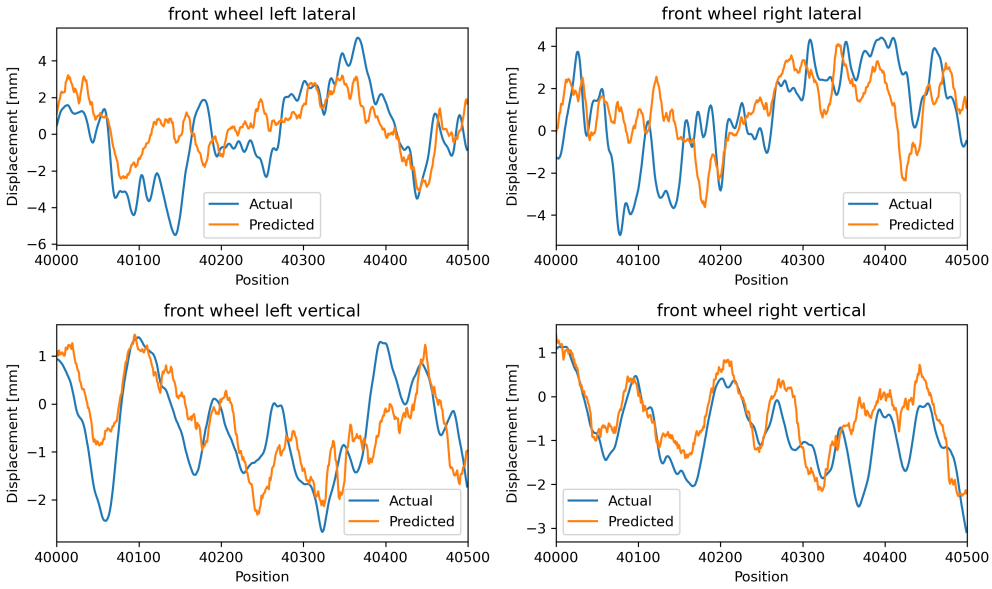

Figure 4.10: Model 2 trained on train_0 and tested on train_0

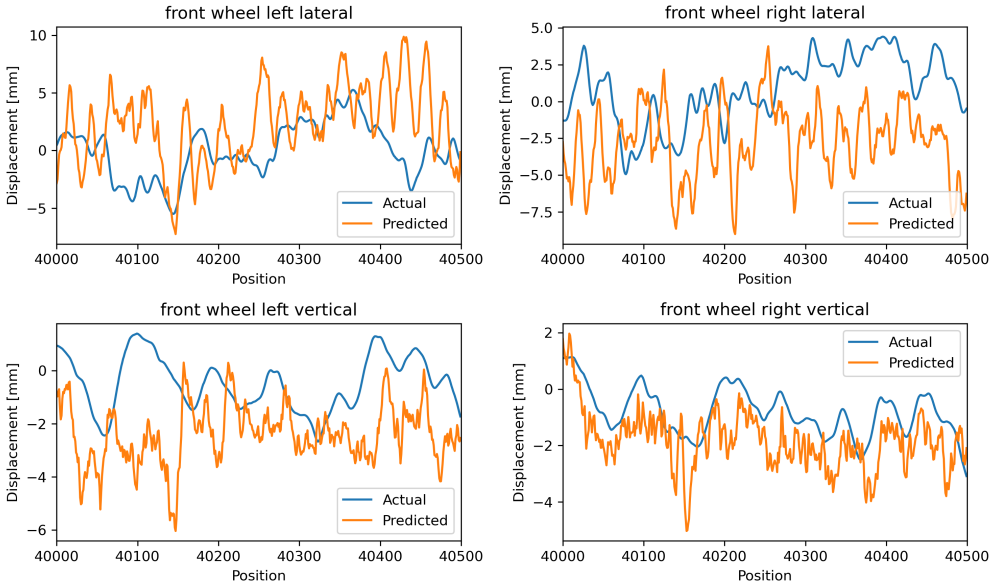

Figure 4.11: Model 2 trained on train_0 and tested on train_6

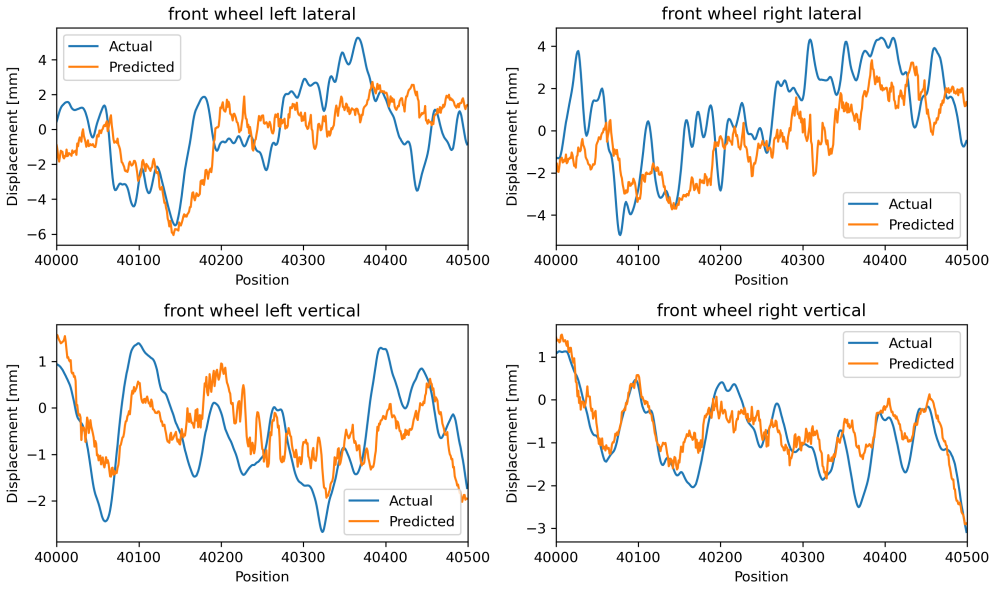

Figure 4.12: Model 1 trained on $\text{train}_{0,1,2,5,6}$ and tested on train_3

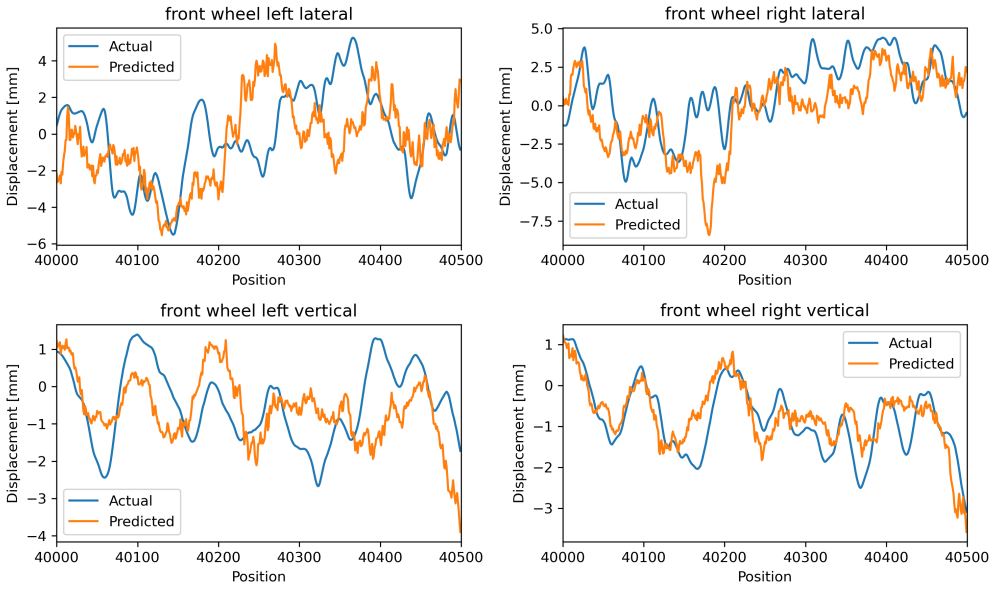

Figure 4.13: Model 2 trained on $\text{train}_{0,1,2,5,6}$ and tested on train_3

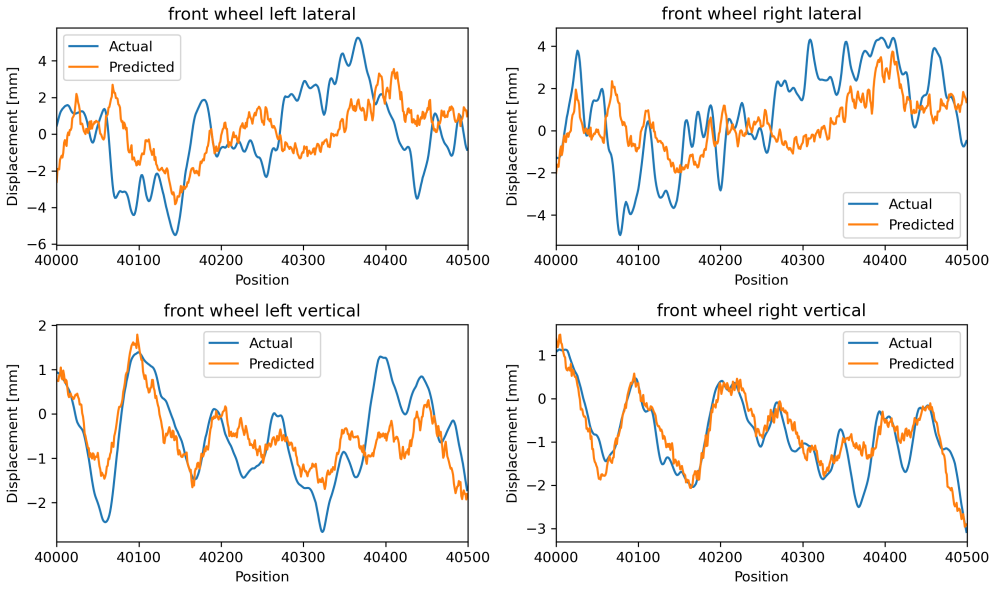

Figure 4.14: Model 1 trained on train_0, fine-tuned on train_1 and tested on train_1

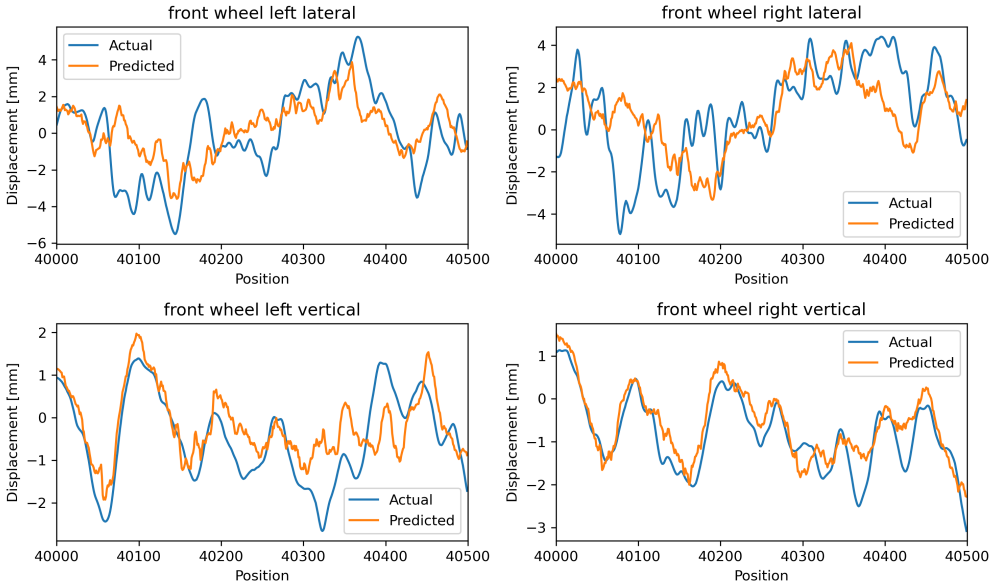

Figure 4.15: Model 2 trained on train_0, fine-tuned on train_1 and tested on train_1

Chapter 5

Conclusion and future work

5.1 Conclusion

To sum up, the evaluation of the models trained with the dynamics of a single train shows that they are capable of predicting the rail irregularities based on the dynamics of another train if the parameter change is small. This means the models trained with the dynamics of the reference train are able to predict the displacements from the dynamics of train_{1-4} as precisely as from their own, but not as exact as from the dynamics of train_{5-6}. To recall, train_{1-4} have a single parameter modified by 5 % while train_{5-6} have eight parameters varied by 5 % each.

The two optimisation strategies presented are generalised training and fine-tuning. On the one side, generalised training uses more training data and helps to increase the prediction precision, thus reducing the mean errors by a small amount overall. The model trained with the dynamics of $\text{train}_{0,1,2}$ manages to predict the irregularities from the dynamics of train_{3-4} precisely, but still struggles to do so from the dynamics of train_{5-6}, though there is an improvement compared to models trained on individual train dynamics. On the other side, in the fine-tuning scenario, the overfitting property of the models can again be observed when analysing the trade-off between the amount of training data and training duration in the fine-tuning setting. Fine-tuning using one data batch attains the lowest training loss and the highest prediction error while fine-tuning using all available data batches reaches the highest training loss and leads to the best prediction errors. Overall, fine-tuning demonstrates an enhancement, effectively reducing both the training loss and prediction errors to the level attained by pre-training and even below, when using the dynamics of train_{5-6}.

Last but not least, it needs to be mentioned that model training does not attain the same level that was reached in previous work. Model overfitting is observed as the more complex Model 2 has a much smaller training loss compared to its validation loss. This is not the case for Model 1. Therefore, the resulting predictive capability is weak and does not provide a strong basis to draw accurate conclusions.

5.2 Future work

In a further step, the same set-up of generalised training and fine-tuning can be used to analyse more thoroughly the differences in the cross-train predictions. It is worth exploring whether there is a better starting point for model fine-tuning than the local minimum achieved after the classical model training, typically at the end of 3,000 epochs. Identifying an optimal starting point could enhance fine-tuning performance, potentially at the expense of some pre-training performance.

Bibliography

[1] World Economic Forum, "The travel & tourism competitiveness report 2019," Accessed: 2024-07-24. [Online]. Available: https://www.weforum.org/publications/the-travel-tourism-competitiveness-report-2019

[2] SBB, "Infrastructures report," Accessed: 2024-07-24. [Online]. Available: https://reporting.sbb.ch/en/infrastructures

[3] A. Plesner, "Using data-driven state-of-the-art machine learning and conformal prediction for track irregularities from observed dynamics of in-service railway vehicles," Master's thesis, Technical University of Denmark, 2022.

[4] L. E. Christiansen, "The dynamics of a railway vehicle on a disturbed track," Master's thesis, Technical University of Denmark, 2001. [Online]. Available: https://backend.orbit.dtu.dk/ws/portalfiles/portal/126118469/thesis.pdf

[5] A. Plesner, "Estimating track irregularities from railway vehicle dynamics using deep learning," 2021.

[6] L. Xiao, "Using data-driven state-of-the-art machine learning and conformal prediction for track irregularities from observed dynamics of in-service railway vehicles," 2024.

[7] MathWorks, "Matlab," Accessed: 2024-07-24. [Online]. Available: https://www.mathworks.com/products/matlab.html

[8] Z. Eric and W. Jiahui, *Vector Autoregressive Models for Multivariate Time Series*. Springer New York, 2003. [Online]. Available: https://doi.org/10.1007/978-0-387-21763-5_11

[9] MathWorks, "Information criteria for model selection," Accessed: 2024-07-24. [Online]. Available: https://ch.mathworks.com/help/econ/information-criteria.html

[10] A. Plesner, "Personal communication," 2024.

[11] PyTorch Developers, "PyTorch," Accessed: 2024-07-24. [Online]. Available: https://pytorch.org/

[12] NVIDIA Corporation, "CUDA," Accessed: 2024-07-24. [Online]. Available: https://developer.nvidia.com/cuda-zone