Estimation of car body vibration from axle box acceleration by combining machine learning and model-based methods

Studienabschlussarbeit

Tobias Weiland
453 148

08.07.2024

Supervisor: Prof. Dr.-Ing. C. Gühmann
Prof. Dr.-Ing. D. Kolossa

Technische Universität Berlin
School of Electrical Engineering and Computer Science
Department of Energy and Automation Technology
Chair of Electronic Measurement and Diagnostic Technology

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 30th September 2024

Kurzfassung

Die vorliegende Studie hatte zum Ziel, ein Machine-Learning-Modell zur Vorhersage der vertikalen Beschleunigungsübertragung zwischen dem Achslager und der Karosserie eines Zuges zu entwickeln. Dafür wurde ein Datensatz aus einem Projekt zur vorausschauenden Wartung moderner Eisenbahninfrastruktursysteme verwendet, welcher Aufzeichnungen eines Multisensorsystems mit Achslagerbeschleunigungssensoren (ABA) und einer Inertialmesseinheit (IMU) beinhält. Der Datensatz wurde neu abgetastet, gefiltert, skaliert, synchronisiert und in Trainings-, Validierungs- und Testdatensätze aufgeteilt, um die Daten für das Modelltraining vorzubereiten. Im Rahmen der Arbeit wurden drei neuronale Netzwerktypen berücksichtigt: konventionelle neuronale Netze (NNs), Faltungsneuronale Netze (CNNs) und Rekurrente neuronale Netze (RNNs). Die Modelle wurden in Python implementiert und es wurde eine Optimierung für ausgewählte Parameter (z.b. für die Modellbreite und die Modelltiefe) durchgeführt. Die verschiedenen Ansätze wurden miteinander verglichen und bewertet. Die Studie zeigte, dass die Entwicklung eines zuverlässigen Vorhersagemodells mit erheblichen Herausforderungen verbunden ist. Alle Modelltypen wiesen während des Trainings eine starke Unteranpassung auf und erreichten schließlich ein lokales Minimum, was zu einer suboptimalen Parametrisierung und Vorhersagen nahe Null führte. Durch die Optimierung konnte keine Parameterkombination gefunden werden, welche eine bessere Vorhersagegüte aufwies, stattdessen unterschieden sich die Konfigurationen in der Geschwindigkeit, mit der das Plateau erreicht wurde. Die unzureichende Modellqualität lässt sich auf die Verwendung von verrauschten und uneinheitlich korrelierten Signalen für das Training zurückführen, was dazu führte, dass die zugrunde liegende Dynamik nicht exakt erfasst werden konnte. Als potenzielle Verbesserungen können die Einbeziehung zusätzlicher Daten, die Untersuchung von Mehrkanal-Lösungen mit mehreren Sensoren sowie die Untersuchung größerer Modelle mit mehr Rechenleistung genannt werden. Des Weiteren könnten Modellierungsansätze, die partielle Differentialgleichungen einbeziehen, potenzielle Forschungsfelder für die Zukunft darstellen.

Abstract

The objective of this study was to develop a machine learning model to predict the vertical acceleration transfer between the axle box and the body of a train. The modelling was based on a data set from a project on predictive maintenance of modern railway infrastructure systems and included recordings from a on-board multi-sensor system with axle box acceleration (ABA) sensors and an inertial measurement unit (IMU). The data set was resampled, filtered, scaled, synchronised and split into training, validation and test sets in order to prepare the data for model training. Three neural network types were considered: conventional neural networks (NNs), convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The models were implemented in Python and parameter optimisation was carried out, for example for the model width and model depth. The different approaches were evaluated and compared with each other. The study demonstrated that the development of a reliable prediction model is associated with considerable challenges. All model types exhibited severe underfitting during training and eventually reached a local minimum, resulting in suboptimal parameterisation and predictions that were close to zero. No parameter combination could be identified through optimisation that demonstrated superior prediction quality; instead, the configurations differed in the speed at which the plateau was reached. The suboptimal model quality can be attributed to the use of noisy and inconsistently correlated signals for training, which prevented the underlying dynamics from being accurately captured. Potential improvements include the inclusion of additional data, the investigation of multi-channel solutions with multiple sensors and the investigation of larger models with more computing power. Furthermore, modelling approaches that incorporate partial differential equations could represent promising avenues for future research.

Contents

List of Figures

List of Tables

List of Abbreviations

CBM	Condition-based Maintenance
DLR	Deutsches Zentrum für Luft- und Raumfahrt e. V.
IMU	Inertial Measurement Unit
ABA	Axle Box Acceleration
GNSS	Global Navigation Satellite System
NN	Neural Network
NNs	Neural Networks
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
RNN	Recurrent Neural Network
RNNs	Recurrent Neural Networks
MSD	Multibody System Dynamics
MLP	Multilayer Perceptron
LSTM	Long Short-Term Memory
RelU	Rectified Linear Unit
ELU	Exponential Linear Unit
GRU	Gated Recurrent Unit
RMSE	Root Mean Squared Error
PINN	Physics Informed Neural Network
PINNs	Physics Informed Neural Networks
PDEs	Partial Differential Equations

1 Introduction

Railway transportation of goods and passengers is considered an environmentally friendly and energy efficient alternative to cars. Therefore, the development and expansion of railway infrastructure is becoming increasingly important in many countries [1]. However, railway operators face additional costs due to the high maintenance requirements of vehicles and the rail network. Modern maintenance solutions have significant potential for cost reduction through optimized resource utilization. Furthermore, the intensity of effortful maintenance activities, such as inspections or patrolling, can be reduced [1].
Condition Monitoring as part of Condition-based Maintenance (CBM) is becoming recognised as the most effective strategy for accomplishing maintenance tasks in various industries [2] including the area of rail transport infrastructure systems.
To implement a CBM strategy, it is necessary to have a quasi-continuous and comprehensive collection of track condition data. Sensor-based technologies are crucial for measuring the track condition. Within several projects, scientists at the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR) are working on an approach based on the use of embedded sensor systems on regular rail vehicles instead of dedicated measurement trains. The main focus of this approach is on utilizing inertial sensors to capture the dynamic response of the vehicle to track irregularities. Specifically, acceleration sensors located on the axle boxes are used in combination with a positioning system in the driver's cabin [3]. The positioning system comprises an Ineratial Measurement Unit (IMU), a Global Navigation Satellite System (GNSS) antenna and receiver. By combining these systems with the Axle Box Acceleration (ABA), georeferenced data can be obtained, which can be used to determine the track condition at a given position.
A high level of measuring accuracy and a reliable sensor system are necessary to monitor the current state of the tracks and implement a CBM strategy. Using a sensor system that is prone to errors to monitor irregularities in the track will result in unreliable anomaly detection. This is particularly important in autonomous systems that collect data offline most of the time. A robust sensor that can detect abnormal behavior before sending it to the back-end is desirable. In light of the aforementioned considerations, the necessity for an advanced sensor validation method is apparent, as it is of paramount importance to ascertain whether anomalies in the measured data are attributable to substandard track quality or sensor failure.
One potential methodology for validation is the autonomous, data-driven validation of the sensor system, utilising the data from the individual sensors within the system and their interrelationships. The key to this approach is that both the acceleration sensors in the axle boxes and the IMU in the positioning system measure accelerations in the vertical direction, creating redundancy. As the position of the sensors is different (acceleration sensor at the axle boxes and IMU in the cabin), the acceleration they measure is also different because the vibration is damped by the vehicle's suspension system. The underlying principle of this model-based method is to forecast the vibration in the body where the IMU is situated, utilising the ABA data as input information. The comparison between the model prediction and the actual vibration measured in the cabin can then be used to evaluate the actual behaviour of the sensor

system and to detect sensor errors. In order to implement this approach, it is essential to develop a model with high prediction accuracy.
In the field of railway dynamics, a conventional and established approach for on-board state and vibration monitoring is Multibody System Dynamics (MSD) modelling. However, these models require high computational power due to the non-linear behaviour associated with wheel-rail contact and hysteretic damping characteristics. Especially for real-time systems, it is essential to reduce the computation time by increasing the efficiency of the models [4]. In addition, physical models for determining dynamic vehicle responses are generally difficult to parameterise with sufficient accuracy, as the necessary parameters are often unknown and may change over time [3].
Recent publications have focused on a purely data-driven modelling approaches, which has become increasingly popular due to recent advances in machine learning, e.g. [5]–[9]. These approaches are particularly suitable for dynamic system modelling [5]. In addition to classical machine learning methods such as classification and linear regression, deep learning techniques are also being developed. Neural networks are particularly advantageous for these tasks due to their high nonlinear modelling capability [6].
This work presents several neural network models for estimating the body vibrations of a railway vehicle based on the axle box acceleration. The background to this research is the autonomous evaluation of the functionality of multi-sensor systems with redundant signals. The models, which include a conventional Neural Network (NN), a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN), are compared and evaluated with regard to the specific task. The data sets needed to train and test the models originate from the HavenZug [10] project and are provided by the DLR.

2 Theoretical Background

This chapter elaborates on the theoretical background for this work. It includes an introduction to the necessary aspects of railway vehicle vertical dynamics, a description of the machine learning methods used to derive car body vibration from ABA data, and the necessary techniques applied during the course of data preprocessing.

2.1 Railway Vehicle Vertical Dynamics

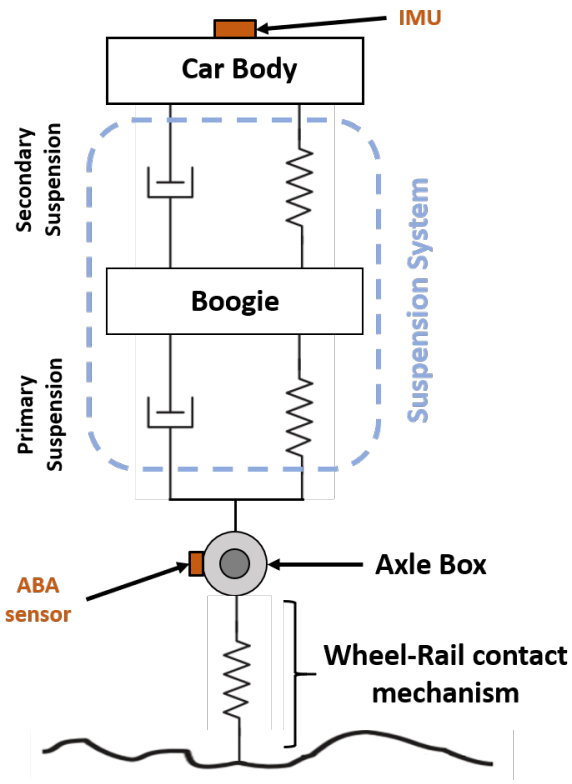

Figure 1: Sensor Positioning in Two-Stage Suspension System. Modified from Rao 2021 [11]

The topic of the vertical dynamics of rail vehicles encompasses a number of different aspects, including mechanical, system-theoretical, and design considerations. As a result, this chapter has been divided into a number of sub-chapters, each of which deals with a specific aspect in greater detail. The first subchapter provides an in-depth analysis of the specific characteristics of rail vehicles, followed by a detailed examination of the modelling of damping systems. Finally, a mechanical quarter-vehicle model is derived.

It is important to note that the entire vehicle is not considered, and the focus is on the transmission of vibration between the axle bearing and the cab. Therefore, this section does not discuss the non-linear properties at the wheel-rail contact point as they are not relevant. Instead, the focus is on the non-linear properties in the suspension system located between the axlebox and cabin.
To illustrate the arrangement of the sensors within the measurement set-up used in the underlying database, Figure 1 shows a simple quarter-car model representing the rail vehicle. The positioning of the ABA sensor, the IMU and the two-stage suspension system is shown schematically. The quarter-car model itself is discussed in more detail in section 2.1.3.
The two-stage suspension system, illustrated as a combination of a damper and a spring connected in parallel, isolates the vehicle body from road shocks and unwanted vibrations [12]. Its characteristics determine the comfort and handling qualities of the vehicle [13]. The next chapter outlines further characteristics of the suspension and elucidates the mechanism of energy dissipation in such systems.

2.1.1 Suspension Characteristics

Trains are often fitted with a two-stage suspension system to manage vibrations caused by track irregularities and ensure passenger and cargo safety and comfort. [14]. The primary suspension is predominantly responsible for reducing the amplitude and frequency content of vibrations to maintain the vehicles dynamic stability. In contrast, the secondary suspension is primarily designed to reduce vibrations for increased comfort [15], [16].
In addition to helical coil springs of various designs, rubber springs are frequently used in the railway sector. The dataset that will be used for model training in subsequent chapters, is recorded on a train equipped with rubber springs as a primary suspension. In comparison to helical springs, rubber springs offer greater stiffness and damping properties as excitation frequency increases and amplitude decreases. Frequencies between 0-20 Hz are typically significant in primary and secondary suspension frequency analysis [16], [17]. Designed to have certain force-displacement or, in case of dampers, force-velocity characteristics, suspension elements can have linear or non-linear properties. Given non-linear characteristics, the deflection rate changes for increasing load. In railway applications deflection rates usually increase with growing loads [16].
Although nonlinear stiffness can complicate the dynamic response of a system, damping is often the dominant source of non-linearity in many applications [13]. Damping characteristics can also include a hysteresis, which means that the force-displacement curve may not follow the same path for the loading and unloading part of the cycle [16]. The area inside the curve can be interpreted as the amount of damping in the suspension element. Figure 2 displays three force-displacement curves from an experimental characterization of a hydraulic damper, showing the damper stroke along the x-axis and the corresponding force along the y-axis. The study compared a standard damper to a modified one, that was designed to reduce the amount of force transmitted by the damper at higher frequencies [15].
The damping behaviour is significantly affected by the excitation frequency. At 1 Hz, the standard damper (blue) exhibits typical purely dissipative suspension behaviour, with a regular and symmetrical hysteresis cycle. In contrast, the modified damper shows a stronger hysteresis at the same excitation frequency. Higher frequencies can cause distorted hysteresis cycles, resulting in a significant reduction of the total damping amount, especially for the modified damper. This reduction occurs as the area described inside the curve decreases significantly.

The experiments were conducted to identify spring and damper parameters for a complex mechanical secondary suspension model [15]. It is crucial to know the model's parameters and dependencies for an accurate simulation. However, suspension components are often inaccessible, and parameters may be unknown, which reduces the model's quality. The modeling of suspension systems is complicated by parameter uncertainties, non-linear damping, and complex structures. The following section presents approaches to address these issues.

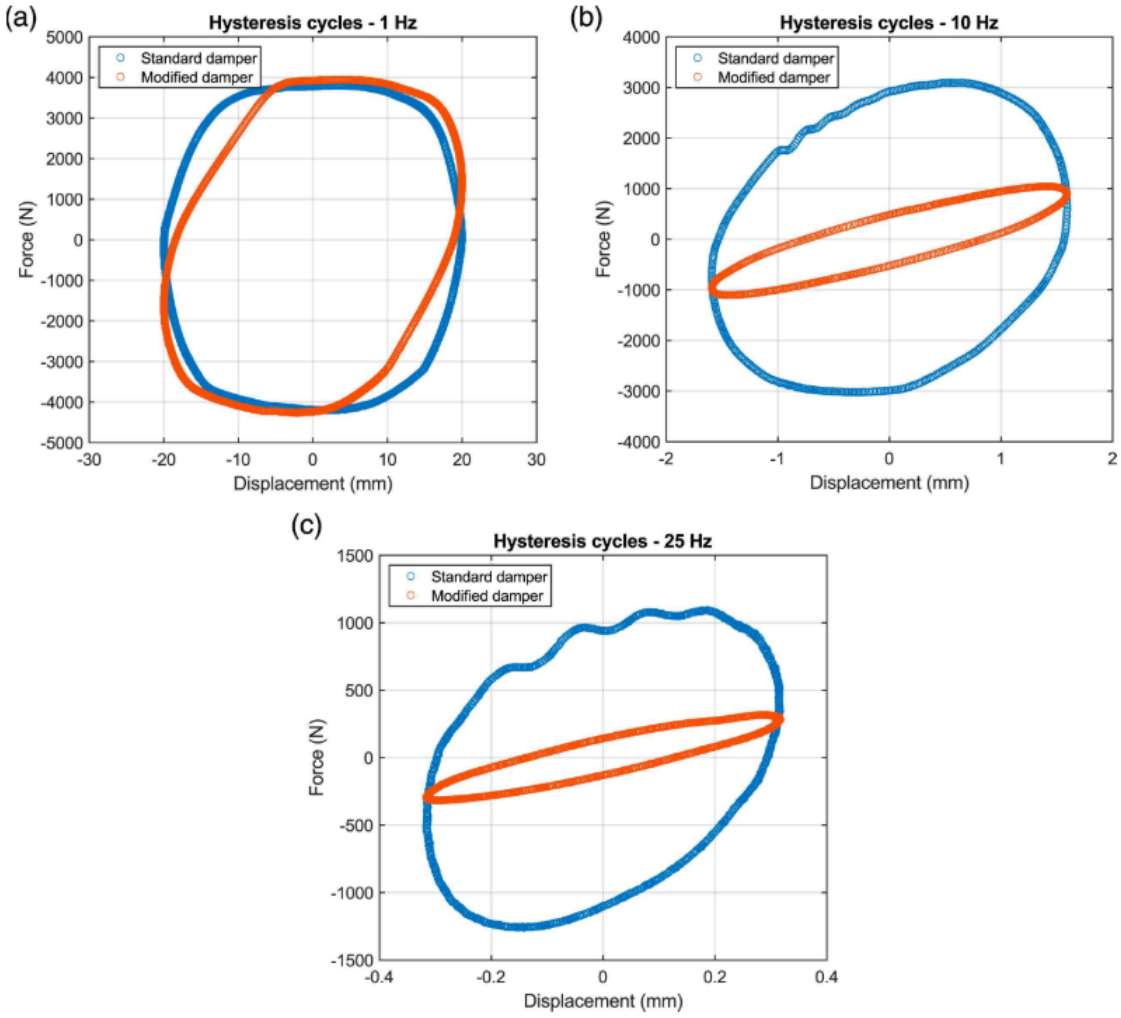

Figure 2: Hysteresis cycles in the force-displacement diagram of a hydraulic damper measured at a) 1 Hz, b) 10 Hz, c) 25 Hz [15]

2.1.2 Suspension Modeling

The objective of this study is to develop a machine learning model that accurately forecasts the transmitted vibrations in the suspension system of a rail vehicle. The model should emulate the mechanical behaviour of the suspension system, which dissipates energy as the vehicle moves along a track.

In order to model and simulate the suspension dynamics mathematically, several first-order differential equations are employed, which result from the nonlinear behaviour associated with the various suspension components [17].
Coil springs, leaf springs, rubber springs, air springs, and hydraulic dampers are the most frequently utilized suspension elements. All of these components are categorized as passive, denoting that the forces and moments they exert rely on the displacement and velocities at their interfaces, given their specific properties and potential pre-load. This is in contrast to active suspension components, inclusive of semi-active ones, wherein additional factors also impact the exerted forces [17]. No active components are used in the vehicle in this study, hence modeling these elements will not be discussed further. More information on the locomotives suspension system and the used sensors is indicated in chapter 3.1.
The origin of the mechanical characteristics of suspension components can be traced back to the disciplines of solid mechanics, fluid mechanics, and tribology. Modeling suspension components in detail, regarding the geometry, the material as well as pre-load and friction interfaces, can be computationally intensive and time-consuming. Instead simplified models of springs, dashpots, and friction elements are used frequently in vehicle-track simulations [17]. By combining simple sub models, complex suspension structures can be realized.
Detailed mechanical 2D and 3D models for vertical vibration studies in railway applications are discussed in a number of recent publications. Muñoz *et al.* [4] presented a comparative study with a full 3D coupled dynamic multibody model for different vehicles. Multibody railway models are commonly used for on-board state observation, parameter identification as well as track geometry measurement.
Another study by Dumitriu *et al.* [18] discussed various sub models for secondary suspensions, for integrating them into a mechanical half vehicle model. Illustrating the combination of simple elements as dampers and springs forming a complex suspension system, the half vehicle model is displayed below in figure 3. By combining multiple sub models, which itself are a mechanical models representing various suspension elements, the necessary model complexity can be reached.

However, mechanical models are easily affected by variations in reality such as increasing preloads or worn components and depend heavily on the reliability of the model's parameters. Considering a changing environment where parameters are difficult to obtain, this fact is even more crucial. Furthermore, the numerical iteration method used to solve mechanical model equations during simulation is time-consuming [9].
An alternative modeling approach is purely data driven modeling and the use of machine learning methods. Unlike mechanical models, machine learning models are computationally efficient, can handle non-linear mapping relationships, and are applicable to varying speeds and vehicles. Machine learning models can be categorised into shallow learning (or traditional machine learning) and deep learning techniques [9]. Both are being applied in the railway vehicle vertical dynamics.
A study [5] showed regression models can be used for predicting the dynamic response amplitude for a railway vehicle, given the input from a number of on-board inertial sensors . Classification models on the other hand are suitable for classifying the track sections by their condition based on the dynamic response distribution. However, the use of shallow learning methods (traditional machine learning) is not the main focus of interest nowadays.
Instead, the focus is on the use of deep learning methods, especially Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and combinations of those as they hold great potential in suspension modeling. Developments in recent decades have led to great suc-

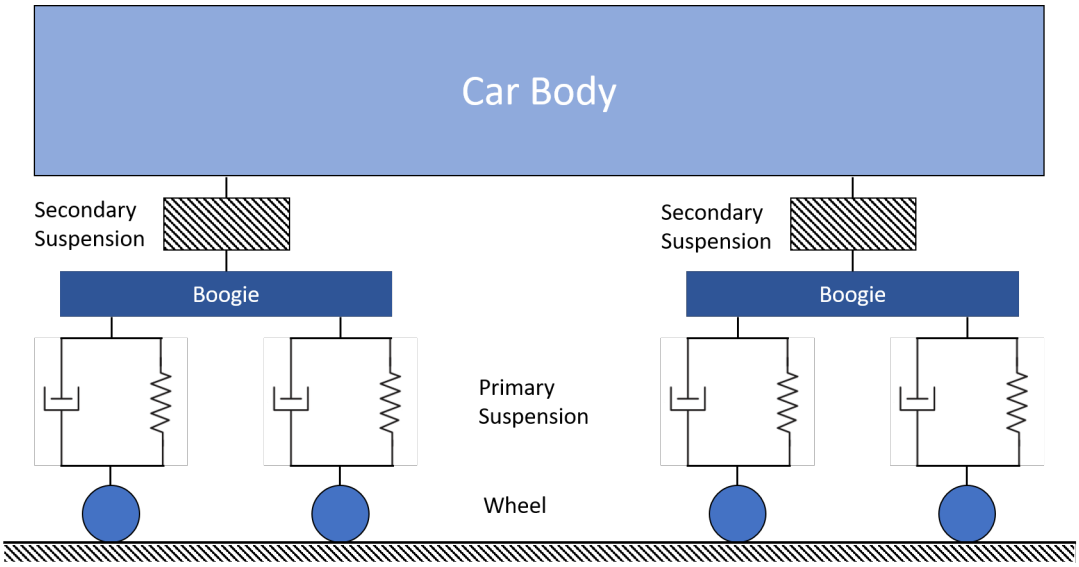

Figure 3: Mechanical half vehicle suspension model for vertical vibration analysis. Redrawn from [18]

.

cess in sequence modeling with these deep learning techniques [9] and a number of studies have proven their efficiency. Li *et al.* for instance used an RNN to estimate car body vertical and lateral acceleration [8]. Ma *et al.* proposed a combined CNN-RNN model to predict car body vibrations from track irregularities [7]. Details about these types of networks will be outlined in chapter 2.3.3 and chapter 2.3.2.
However, prior to the transition to the subject of data processing and machine learning concepts, an approach to mechanical modelling will be examined. This is done to present a comprehensive and complete examination of the problem. Furthermore, it is intended to illustrate where the strengths of purely data-driven methods often lie: mechanical models, even when relatively simple quarter car models, require, unlike machine learning models, sound domain knowledge and information about model parameters, which can be difficult to obtain.

2.1.3 Quarter-Car Model

The following subchapter will briefly convey the basis for the quarter-vehicle model that correlates with the sensor system. For analyzing the dynamic behaviour of a vehicles suspension, the quarter-car is a useful and widespread model for passive, active and semi-active configurations [19]. Numerous publications deal with modeling quarter-car models for dynamics simulation [12], [19]–[21]. Predominantly quarter-car models for the railway vehicle include the wheel-rail contact the nonlinear behaviour associated to it. In regard of the application, using ABA data to estimate the car body vibration, the model must exclude the contact point. Furthermore a two-stage suspension system needs to be represented. A modified quarter-car model is illustrated in figure 4.

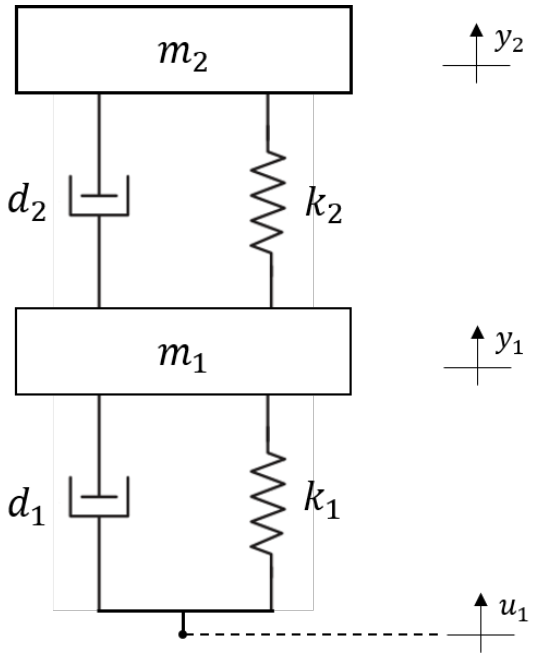

Figure 4: Modified Quarter-Car Model for ABA Data with excluded Wheel-Rail Contact. Redrawn from [20]

The simplified model excludes the wheel-rail connection as u_1 is representing the axle box position. The indices are related to whether the parameter refers to the primary or secondary suspension. Same applies for the coordinates y_1 and y_2. The model of the passive suspension system can be also found in various publications [12], [20]
The mathematical model is represented by two differential motion equations, each describing one of the masses m_1 and m_2:

$$m_2\ddot{y}_2(t) - d_2(\dot{y}_1(t) - \dot{y}_2(t)) - k_2(y_2(t) - y_1(t)) = 0 \tag{1}$$

$$m_1\ddot{y}_1(t) + d_2(\dot{y}_1(t) - \dot{y}_2(t)) + k_2(y_2(t) - y_1(t)) - d_1(\dot{u}_1(t) - \dot{y}_1(t)) - k_1(u_1(t) - y_1(t)) = 0 \tag{2}$$

The two second-order differential equations of the masses positions y_1 and y_2 have to be transferred into a equivalent first order differential equation system $\dot{\vec{x}}(t) = f(x(t), u(t))$, where $x(t)$ is a vector consisting the masses positions and their derivatives and $u(t)$ is the input on the axle box. Considering a dynamic application, $u(t)$ and $x(t)$ are variables depending the time t. For simplifying the equations while transferring them in the course of this section, the time dependency will be ignored in the formula. Following substitutions are made to transfer the system to a first-order:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dot{y}_1 \\ \dot{y}_2 \end{bmatrix}, \vec{u} = \begin{bmatrix} u_1 \\ \dot{u}_1 \end{bmatrix}$$

Using the substitutions, 1 and 2 can be summarised and converted to matrix notation:

$$\dot{\vec{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-d_1-d_2}{m_1} & \frac{d_2}{m_1} & \frac{-c_1-c_2}{m_1} & \frac{c_2}{m_1} \\ \frac{d_2}{m_2} & \frac{-d_2}{m_2} & \frac{c_2}{m_2} & \frac{-c_2}{m_2} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{c_1}{m_1} & \frac{d_1}{m_1} \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} u_1 \\ \dot{u}_1 \end{bmatrix} \tag{3}$$

$$\dot{\vec{x}} = A \times \vec{x} + B \times \vec{u} \tag{4}$$

Equation 4 is the state-space representation of the system, with the state vector $\vec{x}$, the input or control vector $\vec{u}$, system matrix A and the input matrix B. The representation describes the suspension system as a system of first-order differential equations, $\dot{\vec{x}}(t) = f(x(t), u(t))$.
The selection of parameters is a crucial aspect of any modelling. Numerous examples of the parameterisation of quarter car models for railway applications can be found in the literature. Goodall *et. al* [22] presented a simplified perturbed railway vehicle model based on a quarter-car approach. Following physical parameters for the vehicle are used in their research and can be regarded as a guideline for locomotives for freight transport:

Car body mass	m_1	9500 [kg]
Bogie mass	m_2	2500 [kg]
Stiffness primary suspension	c_1	0.5×10^6 [N/m]
Stiffness secondary suspension	c_2	2.5×10^6 [N/m]
Damping coefficient primary suspension	d_1	100 [N - s/m]
Damping coefficient secondary suspension	d_2	1.79×10^3 [N - s/m]

Table 1: Parameters for passive two-stage railway vehicle suspension system model [22]

Unfortunately, no information was available regarding the locomotive that was utilised in the HavenZug project. No specific parameters, such as weight, stiffness, or damping coefficients, were provided. Although the values listed in Table 1 can be used as an approximation, it is technically possible to estimate parameters. However, uncertain parameters may lead to poor model quality and complicate the process of analysis and model debugging.
In contrast, machine learning methods are based exclusively on data and their interrelationships. Consequently, the parameters of the physical model are no longer relevant. One of the most crucial aspects of utilising machine learning methodologies is the preparation and processing of the data.

2.2 Signal Processing

Preprocessing is the most time-consuming phase in data analysis and data science projects. It involves cleaning, reformatting, and integrating raw data to improve its quality and usability. Data scientists spend a significant amount of time, ranging from 60 to 80%, on data preprocessing [23]. Efficient methods for preprocessing data are required when dealing with sequence

models, as datasets for their training often exceed multiple million data points, causing significant computational effort to process the data. The used methods in this work will be outlined in the following subsections.

2.2.1 Data Synchronisation

The model is based on measurement data acquired from the ABA sensor and the IMU. Given that the sensors are positioned differently and have different sampling rates, temporal discrepancies may arise between measurement signals. This desynchronization may present a significant challenge, particularly in the context of data-driven modelling, where a model is trained exclusively on data. Consequently, it is of the utmost importance to detect desynchronization and synchronize the two time series prior to model training. A standard method for estimating the degree to which two time series are correlated is cross-correlation.[24].
The cross-correlation operation is equivalent to complex conjugate convolution. In the field of machine learning, cross-correlation is commonly used in convolutional neural networks due to this identity, but it is referred to as convolution for ease of implementation [25], [26].
For two continuous functions $f(t)$ and $g(t)$, the cross-correlation $(f \star g)(\tau)$ is defined as:

$$(f \star g)(\tau) = \int_{-\infty}^{\infty} f(t)^* g(t+\tau) dt \tag{5}$$

where τ denotes the displacement and $f(t)^*$ the complex conjugate of input signal $f(t)$. The displacement τ refers to the time difference by which the two functions are shifted against each other.
When working with discrete time series data, it is not possible to integrate as described in the equation 5. Instead, when given two time series $f[k]$ and $g[k]$, the cross correlation $z[k]$ can be calculated as indicated below. Here, the displacement k refers to the number of elements by which the two arrays are shifted against each other.

$$z[k] = (x * y)(k - N + 1) = \sum_{l=0}^{||f||-1} f_l g^*_{l-k+N-1} \tag{6}$$

$$\text{for } k = 0, 1, ..., ||f|| + ||g|| - 2$$

where ||x|| is the length of the array x and $N = max(||f||, ||g||)$. Cross-correlation can determine the necessary shift of $g[k]$ along the x-axis to align it with $f[k]$. The formula shifts g along the x-axis and calculates the integral of its product at each position. For discrete functions, the sum of the element-wise multiplication is calculated. The cross-correlation value is maximised when the functions match, because aligned peaks (positive areas) make a significant contribution to the correlation value. In the same way, negative areas also make a positive contribution. When the functions align, the displacement is referred to as k_{opt}.
Because $N = max(||f||, ||g||)$, this method for synchronizing works for arrays with different sizes. It is also necessary for the application which is dealt with in this work, as different sampling frequencies of the sensors lead to different sized data. However, when sliding $g(t)$ by k_{opt} to synchronize the sequences, padding is required. Commonly padding is done with zeros or the last valid value of the array. Figure 5 illustrates an example for a padding after

synchronization. In this case $g[k]$ is slided by 3 elements, causing the last three elements to be deleted (hatched). The green marked element refers to the last valid element of the array and yellow indicates a padded value.

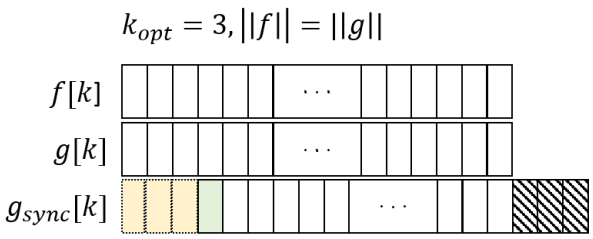

Figure 5: Schematic of the padding process after data synchronisation with an optimum shift of 3 samples

Another method used during signal preprocessing is applying digital filters to emphasize specific frequency ranges of the signal. The following subsection will elaborate on this topic.

2.2.2 Data Filtering

The purpose of signal filtering is to eliminate unwanted components and emphasise important signal elements. This is typically achieved by amplifying or attenuating specific frequency ranges. There are multiple types of filters for instance high-pass, low-pass, and band-pass. Bandpass filters can attenuate both excessively high and low frequencies simultaneously, unlike high-pass and low-pass filters which only limit one end of the frequency range. The transfer function mathematically describes the filter and can be represented by its amplitude and phase response. As an example, the amplitude response of a low-pass filter is shown in Figure 6.

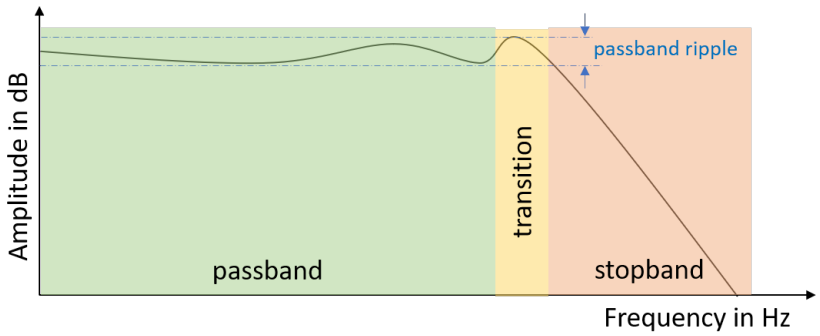

Figure 6: Schematic example of the amplitude response of a low pass filter with significant ripple in the passband

Various approaches exist for designing digital filters. Butterworth filters have a maximally flat amplitude response in the passband, which minimally affects the useful signal in this range. Additionally, the amplitude response is monotonic, meaning it has no ripple. This filter type is commonly used due to its desirable characteristics [27].

The Butterworth filter's transfer function is defined by only two parameters: the cut-off frequency and the filter order. For a butterworth bandpass filter, a high-cut and a low-cut frequency need to be defined. The filter order determines the steepness of the amplitude response's decrease in the transition range. The higher the filter order, the steeper the amplitude response decreases, and the smaller the transition range.
The filter's cut-off frequency is defined as the frequency at which the normalised amplitude response reaches the value of $\frac{1}{\sqrt{2}} \approx -3dB$, according to its transfer function. This applies to all filter orders. Therefore, when designing the filter, it is important to ensure that spectral components of a signal are already attenuated by 3 dB at the cut-off frequency.
Applying digital butterworth filters to a data series may cause time distortion. Polyphase or zero-phase filtering can prevent time offsets. This is achieved by processing the input data both forwards and backwards. The filtered sequence is reversed and filtered again after being filtered in the forward direction. When designing a filter for polyphase filtering, it is important to maintain the desired filter order and overall gain, which is determined by the order and gain of the individual filter passes forwards and backwards.
As previously described in section 2.2.1, it is required for the signals to be synchronous. To achieve precise synchronization of the data sets, the IMU and the ABA Sensor data must be sampled at the same frequency. As the sampling rates of the two sensors in the setup used differ greatly, it is necessary to equalize the signals for further processing. To achieve this, one method is to use polyphase filtering. The next subsection outlines resampling by applying digital filters.

2.2.3 Data Resampling

Data resampling refers to the procedure of modifying the sampling rate of a signal. The change in the sample rate is categorized as interpolation when it increases and as decimation when it decreases. Interpolation may be defined as a process whereby the duration of existing samples is reduced and samples with zero values are included to fill in the gaps. This process is referred to as upsampling. Conversely, decimation entails the elimination of a specific number of samples, thereby extending the duration of the remaining samples. This is referred to as downsampling.
However, neither of these operations is without its consequences. The process of upsampling results in the replication of the signal throughout the expanded spectrum. Conversely, downsampling causes high frequency elements that cannot be accurately represented at the new sampling rate to be shifted to the lower part of the spectrum. This phenomenon is commonly known as aliasing. Figure 7 illustrates the effects of upsampling and downsampling a signal.
In order to eliminate the unwanted effects, it is essential to filter the signal. When performing interpolation, it is necessary to filter the signal after upsampling to ensure the elimination of spectral images. During decimation, the filter is applied before downsampling to remove any components that may pass into the new passband. Consequently, both interpolation and decimation consist of two distinct stages making it computationally intensive .
Polyphase filtering is a method for efficient sample conversion. During the interpolation process, the filter is required to perform numerous operations on input samples that have a value of zero. These operations have the potential to be omitted. To avoid performing operations on zeros, the filter is divided into sub-filters, also known as phases, hence the name of the filter [28].
The decimation process encounters a similar problem, where the majority of the filtered samples are discarded during the downsampling phase. Consequently, the same approach is used for

the decimation process, but in reverse order. As the downsampling process regularly discards samples, it becomes feasible to activate the filter only when there is a need to produce output [28].

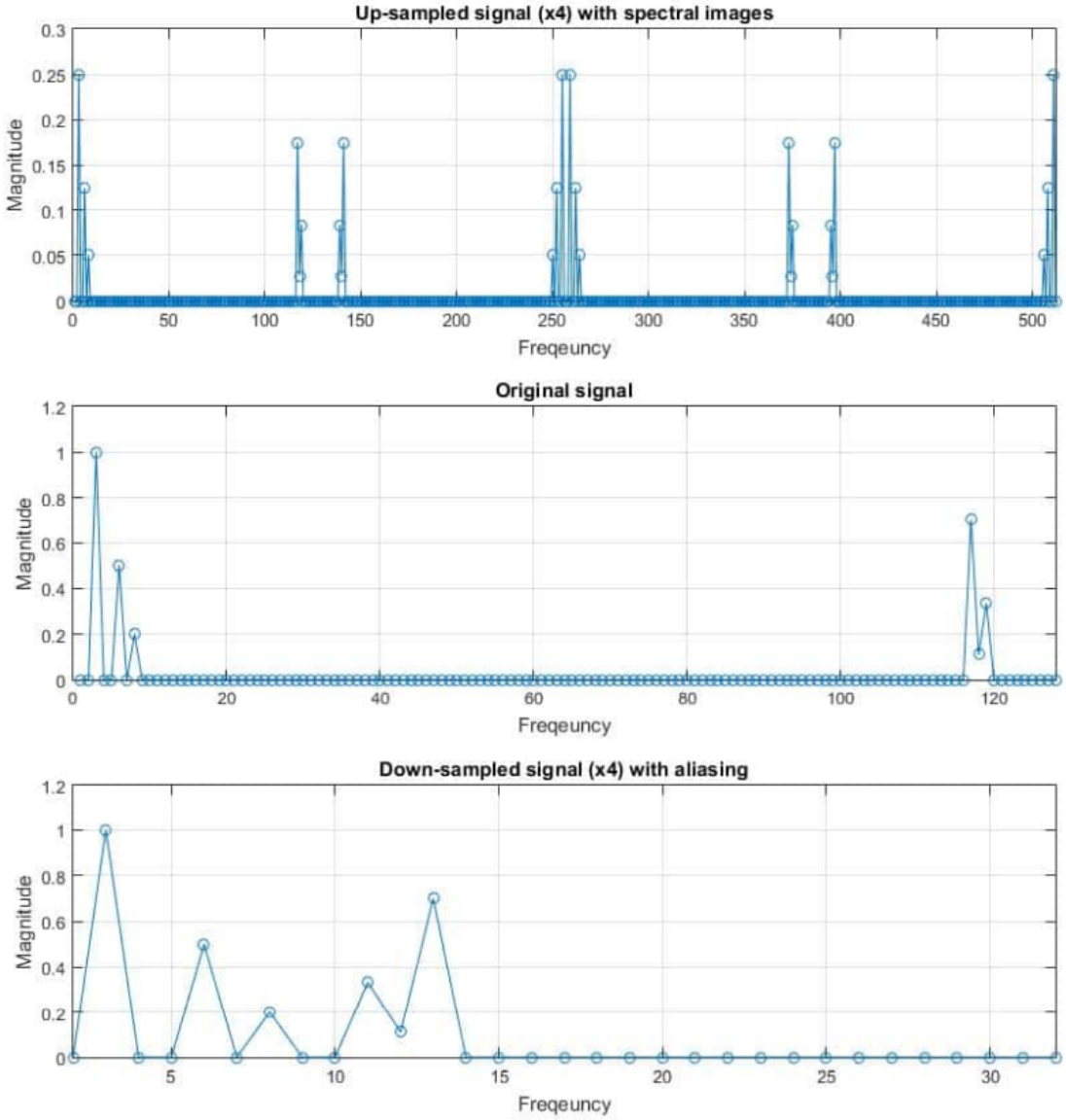

Figure 7: Example of spectral images and aliasing caused by sampling rate conversion using Polyphase Filters [28]

2.2.4 Data Scaling

Data scaling represents an indispensable preprocessing step in machine learning, offering numerous advantages for model training. It enhances the accuracy and performance of the resulting model by reducing numerical issues in the calculation. In particular, when gradient methods are employed, which are utilized in the models developed in course of this work, scaling expedites convergence and facilitates more effective optimization of the model parameters.

In the event of multiple input features, scaling serves to prevent distortion and to ensure that all features exert the same influence on the training process. This results in an improvement in the convergence speed and stability of the learning algorithm.
There are a number of different methods for scaling a data set. One common method is standardisation. In this instance, the features are transformed in such a way that their mean value is 0 and their standard deviation is 1. This approach is beneficial for algorithms that assume a normal distribution. Standardisation is achieved through the application of formula 7:

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{7}$$

where μ is the mean value of the signal and σ its standard deviation.
An alternative approach is Min-Max-Scaling, which is a data transformation technique that normalises the data to a fixed range, typically between 0 and 1. This is particularly useful when the data does not follow a Gaussian distribution. This method also preserves the original distribution, although it is susceptible to outliers. In contrast, standardisation is more robust against outliers, but it can alter the original data distribution.
Having presented the selected preprocessing methods, this section will now proceed to explain the machine learning modelling methods used in the work.

2.3 Machine Learning

The past ten years have seen significant advancements in the field of artificial intelligence. On a regular basis, novel machine learning models are emerging, surpassing state-of-the-art methods in computer vision and natural language processing [7].
In addition to image and speech processing, machine learning methods designed to process sequential data have been applied to a wide range of areas, including safety surveillance [29], disease prediction [30], health condition monitoring [31] and vehicle fault diagnosis [32].
Various techniques can be used to model the vertical dynamics of a vehicle, all of which come from the field of machine learning. An important distinction is made between classical machine learning methods and deep learning methods. Classical machine learning techniques for predicting a vehicles dynamic response include support vector machines, decision trees, MLP and other regression algorithms [9].
Alternatively, common deep learning methods for sequence modelling are CNNs, RNNs and Long Short-Term Memory (LSTM) networks. Combinations of these are also promising approaches and the subject of recent research [7], [9]. The foundation of all deep learning sequence models are classical neural networks, which are discussed at the beginning of this chapter. Depending on the depth of the network, Neural Networks (NNs) can be classified as either classic machine learning or deep learning. The following subsection explains NNs in more detail. It will provide a knowledge base for the subsequent elaborations on more complex types of networks.

2.3.1 Classical Neural Network

Non-linear behaviour can be modelled using NNs. These models are inspired by the way the brain is thought to process external stimuli. By using activation functions, NNs derive their ability to infer information from features that have a non-linear relationship to the target. The present subsection explains the functioning of neural networks and is derived from [33].

Model of a single neuron

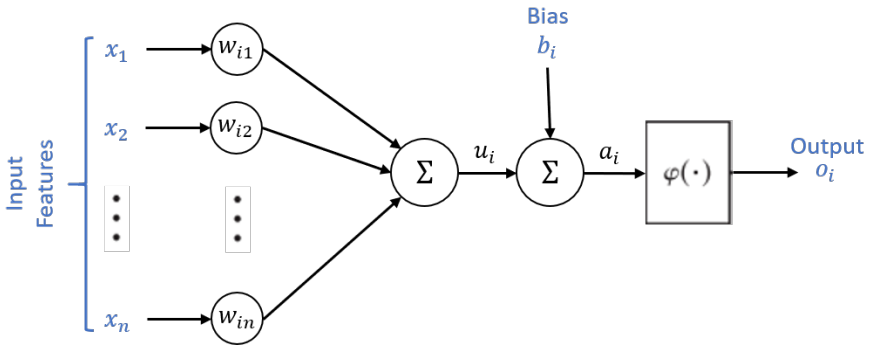

Figure 8: Nonlinear model of a single neuron, labeled i. Redrawn from [33]

A single neuron is the fundamental information-processing unit in a neural network. Figure 8 displays a single neuron schematically. It serves as the foundation for the creation of an extensive range of neural networks discussed in subsequent chapters. Here we identify three basic components:

- A set of connecting links, which are each characterized by a synaptic weight w_{ij}. The subscripts of the weight w_{ij} refers to the connection of input x_j with neuron i. Synaptic weights may be positive and negative values.
- A summation block for summing the input signals weighted by the respective synaptic weight of the neuron. It serves as a linear combiner for the weighted input signals.
- An activation function $\phi(\cdot)$ that determines whether the neuron should be activated or not based on the weighted sum of its inputs. It introduces non-linearity into the network, allowing it to learn and model complex patterns and relationships in the data. Common activation function will be outlined in the course of this chapter.

An externally applied bias, denoted by b_i, is included in the neural model shown in Figure 8. This bias has the effect of increasing or decreasing the net input of the activation function. Like weights, biases can be positive or negative values. Following equations describe the neuron's model mathematically:

$$u_i = \sum_{i=1}^{n} w_{in} x_i \tag{8}$$

$$o_i = \phi(a_i) = \phi(u_i + b_i) \tag{9}$$

where $x_1, x_2, \ldots, x_n$ and $w_{i1}, w_{i2}, \ldots, w_{in}$ are the neuron's input signals and the respective weights, $\phi(\cdot)$ is the activation function and b_i the bias of neuron i. Neurons essentially combine multiple

inputs to generate one (o_i) or multiple ($\vec{o}_i$) outputs by summing the weighted input and bias and applying the activation function to the sum.

Network architectures

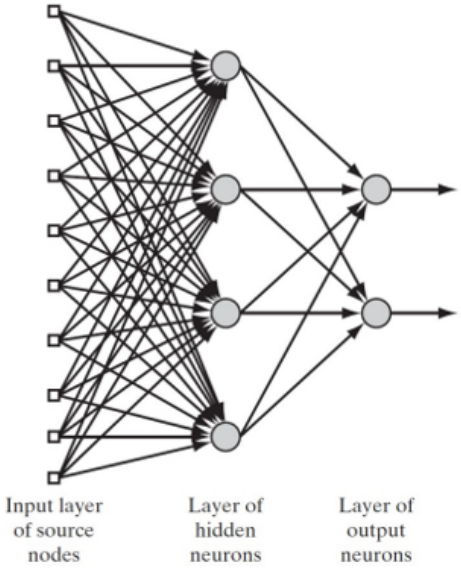

Figure 9: Fully connected feedforward network with one hidden layer and one output layer [33]

It is possible to realise different network architectures by combining multiple neurons. One approach to neuron arrangement is a fully connected feed-forward network. This involves each neuron in one layer being connected with each neuron in the next layer and with all neurons in the previous layer. Figure 9 illustrates this model structure, which shows that neurons in the hidden layer are fully connected to all neurons in neighbouring layers.
Feedforward neural networks are composed of an input layer, an indeterminate number of hidden layers, and a final output layer. The input layer of the network provides the activation pattern (input vector) to the computation nodes in the second layer (i.e. the first hidden layer). The output signals of the second layer are then used as inputs to the third layer and so on for the rest of the network. Neurons in each layer typically only have the output signals of the preceding layer as their inputs. The output signals of the neurons in the final layer of the network make up the network's overall response to the activation pattern provided by the source nodes in the input layer. The network that has been elucidated is a fully-connected feedforward neural network, which means the neurons of neighbouring layers are all connected to each other. Alternatively in partially-connected networks, only some of the neurons of the present layer are connected to the ones in the next layer.
Two crucial parameters when modelling neural networks are depth and width. While depth refers to the number of hidden layers, width refers to the number of neurons per layer. The network depicted in Figure 9, for instance, has a depth of one and a width of four.
A feedforward NN will be used to examine the training algorithm in the course of this section. It is necessary to recall the general formulation for these networks, as the nomenclature used in the neuron model is extended to include the various layers:

Definitions:

w_{ij}^k:	weight for node j in layer l_k for incoming node i
b_i^k:	bias for node i in layer l_k
a_i^k:	product sum plus bias for node i in layer l_k
o_i^k:	output for node i in layer l_k
r_k:	number of nodes in layer l_k
m:	number of layers
n:	number of input signals
N:	number of data pairs in the data set
ϕ:	activation function for hidden layer nodes
ϕ_o:	activation function for output layer nodes

Back-propagation

Back-propagation is a method to train a NN, which means optimizing the weights and biases of the connecting links in order to achieve the desired mapping between input and output. Neural networks can undergo training through two methods: supervised learning and unsupervised learning. In supervised learning, the network is trained using correlating input and output pairs, while in unsupervised learning, the network identifies patterns or structures in the data on its own, without specified data pairs.

Back-propagation is mainly used for supervised learning, where there are known desired outputs for each input. Although the method can technically be used in both supervised and unsupervised learning, it is typically considered a supervised learning algorithm. As the thesis focuses on supervised learning, unsupervised learning methods are not elaborated upon.

There are three prerequisites for supervised learning using back-propagation:

- A dataset containing input-output pairs $(\vec{x_d}, \vec{y_d})$, where $\vec{x_d}$ is the input, that creates the output $\vec{y_d}$. The whole set of data pairs of size N is denoted as $X = \{(\vec{x_1}, \vec{y_1}), (\vec{x_2}, \vec{y_2}), ..., (\vec{x_N}, \vec{y_N})\}$.
- A feedforward neural network as defined above with k number of layers. The parameters of the network (weights w_{ij}^k and biases b_i^k) are collectively denoted as θ. Weight w_{ij}^k refers to the weight between node j in layer l_k and node i in layer l_{k-1}. Analogue, is the bias for node i in layer l_k denoted as b_i^k.
  For simplifying mathematics, the bias b_i^k will be integrated into the weights as w_{0i}^k with a fixed output $o_0^{k-1} = 1$ for node 0 in layer l_{k-1}.
  $$b_i^k = w_{0i}^k$$
- a loss function $E(X, \theta)$, that calculates the error between the predicted output of the model $\hat{\vec{y}_i}$ and the desired output $\vec{y_i}$. A classic loss function used in regression analysis is the mean squared error:
  $$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{10}$$

Should the aforementioned prerequisites be satisfied, the optimisation of the parameters θ may be conducted via the back propagation method. The first step of the back propagation algorithm is calculating the forward pass for each input-output pair $(\vec{x_d}, \vec{y_d})$ in X and storing the results $\hat{y_d}$, a_i^k and o_i^k for each node i in layer k, starting from layer 0 to layer n, the output layer.

Subsequently the backward pass is calculated for each input-output pair $(\vec{x_d}, \vec{y_d})$ in X. and the results for for all weights is stored. This requires to calculate $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ and store the gradients of the loss function $E(X, \theta)$ with respect to each weight w_{ij}^k and bias b_i^k. As the error function can

be broken down into a sum of individual error terms for each input-output pair, the derivative can be calculated for each pair separately and then combined at the end.

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial}{\partial w_{ij}^k}(\frac{1}{2}(\hat{y_d}-y_d)^2 = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial E_d}{\partial w_{ij}^k} \tag{11}$$

By applying the chain rule, the partial derivatives are calculated for each weight w_{ij}^k connecting node i in layer $k-1$ to node j in layer k. The partial derivative of the error function for one input-output pair E_d with respect to a weight w_{ij}^k is generally defined as:

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \tag{12}$$

with the error δ_j^k denoted as:

$$\delta_j^k = \frac{\partial E_d}{\partial a_l^k} \tag{13}$$

By substituting the error, the general form can be adapted to obtain the derivative with respect to weights in a certain layer. For instance, the derivative of the error function with respect to a weight w_{i1}^m in the output layer is denoted:

$$\frac{\partial E_d}{\partial w_{i1}^m} = \delta_1^m o_i^{m-1} = (\hat{y_d}-y_d)\phi_0'(a_1^m)o_i^{m-1} \tag{14}$$

The equation for the derivative of the error function with respect to a weight w_{ij}^k a hidden layer is defined for $1 <= k <= m$ as:

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = \phi'(a_j^k)o_i^{k-1}\sum_{l=1}^{r^{k+1}} w_{jl}^{k+1}\delta_l^{k+1} \tag{15}$$

The process starts with calculating the derivatives with respect to the weights in the output layer working its way to the input layer. For a full formal description of the algorithm, please refer to [33]. Once all gradients are calculated and combined (equation 11), the model parameters θ are updated according to the learning rate α and the calculated total gradient $\frac{\partial E}{\partial w_{ij}^k}$:

$$\Delta w_{ij}^k = -\alpha\frac{\partial E(X,\theta)}{\partial w_{ij}^k} \tag{16}$$

The derivation in the back-propagation process is straightforward and follows the chain rule and the product rule. However, the application of these rules is dependent on the differentiation of the activation functions $\phi(\cdot)$ and $\phi_0(\cdot)$ used in various layers of the network.

Activation functions

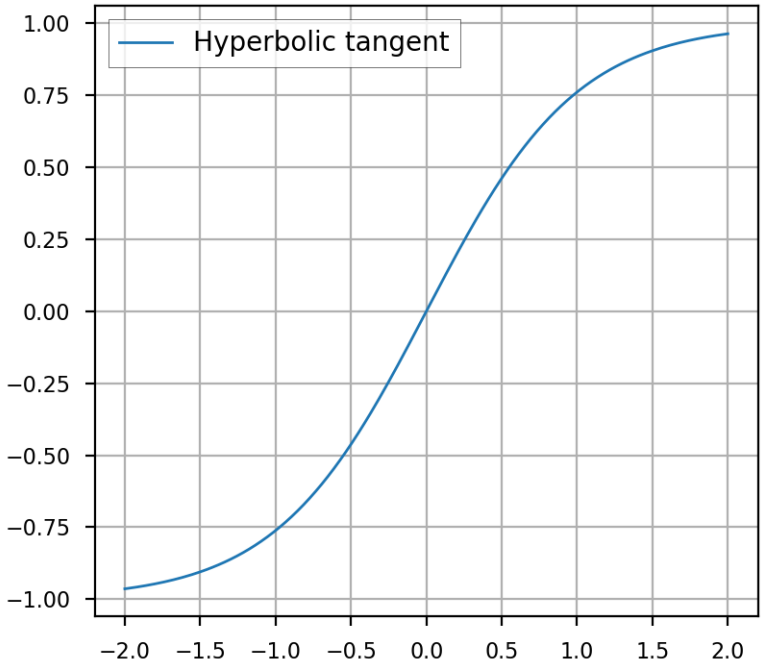

Figure 10: Hyperbolic tangent activation function

Activation functions can be linear or non-linear and should be chosen to suit the application. The linear activation function, known as "no activation" or "identity function" $f(x) = x$, basically just passes the input to the output without changing it. Using linear activation functions creates a linear regression model, that has limited modelling capacities of non-linear behaviour. Modeling nonlinear systems, like the two-stage suspension, requires nonlinear activation functions.
The sigmoid function $\sigma(z)$, a special variant of the logistic function, is a commonly used nonlinear activation. The S-shaped function takes any real value as input and outputs values in the range of 0 to 1. The output is calculated according to formula 17. Since the output is limited between 0 and 1, it is commonly used for models where we need to predict probability as an output.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{17}$$

Another nonlinear commonly used activation function is the hyperbolic tangent function $tanh(x)$. Its output is constrained to a range of -1 to 1, and like the sigmoid function, it has an S-shaped curve. Figure 10 displays the $tanh(x)$ function. The function is mathematically represented as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{18}$$

In addition to the mentioned functions, other commonly used nonlinear activation functions are the Rectified Linear Unit (RelU), the softmax function, and the Exponential Linear Unit (ELU).

Model and Training Parameters
When designing and training models based on neural networks, it is essential to select and adjust a number of parameters in order to achieve optimal performance and generalisation capability of the models. A selection of parameters is presented, with an explanation of their role in achieving optimal performance in the models.

- **Learning Rate:** The learning rate is a crucial parameter in neural network training. It determines the speed with which the network learns from the training data and adjusts its weights. If the learning rate is set too high, the network may overshoot the optimal weights and fail to converge. Conversely, a learning rate that is too low can result in slow convergence.
- **Batch Size:** The batch size is the number of training examples that are processed simultaneously before the weights are updated. A larger batch size may result in more stable updates, although this may necessitate a greater allocation of memory. A smaller batch size can result in faster convergence, although this may be accompanied by less stability.
- **Number of Epochs:** An epoch is defined as a complete iteration of the entire training data set. The number of epochs determines how often the neural network is presented with the entirety of the data set. Insufficient epochs may result in underfitting, whereas an excess of epochs may lead to overfitting.
- **Regularization:** Regularization methods serve to prevent overfitting by constraining the complexity of the model. A common regularization technique is dropout, whereby individual neurons per layer are specifically deactivated. Another technique is early stopping, which interrupts training when a plateau is reached.
- **Weight Initialization:** The initialisation of the weights at the commencement of training can have a significant impact on the convergence and performance of the model. Consequently, numerous weight initialization methods exist, for instance, the Xavier Weight Initialization.

Having established the fundamental principles of neural networks and the methodology of supervised model training, the subsequent sections will examine the various types of neural networks.

2.3.2 Convolutional Neural Networks

CNNs are a special type of neural network designed to process data with a structured grid-like topology, especially image data, which represent a two-dimensional grid of pixels. It is named after the mathematical linear operation between matrices known as convolution. The operation involves sliding a filter function (or kernel) over the input data, multiplying the kernels values with the values in the input at each position, and then summing up these products to produce a single output. Mathematically, the discrete convolution of two series $f(t)$ and $g(t)$ is similar to the cross-correlation examined in chapter 2.2.1. The convolution $(f * g)$ of the discrete function $f(t)$ with kernel function $g(t)$ is defined as:

$$s(t) = (f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a) \tag{19}$$

where a refers the shift of $g(t)$ in the course of the operation. The first argument (here, the function $f(t)$) to the convolution is often called the input, and the second argument (here, the

function $g(t)$) the kernel. The output of the function may referred to as the feature map [25]. Furthermore, convolution is also possible in multiple dimensions. A multidimensional filter is employed for the processing of image data, where 2D image I stored as a matrix of pixel values is convoluted by a two-dimensional kernel function K according to:

$$S(i,j) = (I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)K(i-m, j-n) \tag{20}$$

where i, j represent the pixel coordinates.
Originally developed for computer vision tasks in the 1960s [34], CNNs have become one of the most widely used deep learning algorithms and are increasingly applied to image classification, object detection, video processing, natural language processing and speech recognition [25], [34], [35].
The utilisation of the convolution operation also confers advantages when processing time series data, as exemplified by the case of ABA and IMU sensor signals. The suitability of CNNs for time series data lies in their ability to take into account the local structure and dependencies within the data. When considering a time series $X = x_1, x_2, x_3, ..., x_n$, a fully connected neural network would ignore the inherent sequential nature of this data. This may result in the loss of information, for instance, the fact that (x_1, x_2) and (x_2, x_3) are both adjacent pairs. Consequently, a difference in x_1 and x_2 should be interpreted in a similar manner to an equivalent difference in x_4 and x_5. Conversely, a similar difference between x_4 and x_9 may be interpreted in a completely different manner. CNNs exploit the fact that neighbouring data points have a stronger relation than more distant points and the convolution operation enables the extraction of local features along the time axis. This allows temporal patterns and dependencies to be captured on different time scales. By applying a convolution to the series of data instead of processing every timestep independently, CNNs can efficiently extract features of interest and relations in the data set [34]. Recent publications examine CNN-based models on a wide range of time-series applications from stock market and gold-price predictions to forecasting time series data of solar radiance and photovoltaic power [36].
In conjunction with the convolution operation, so-called pooling layers are frequently employed to reduce the dimensionality of the convolved data. This method is employed with particular frequency in conjunction with CNNs for the processing of images, although it can also be applied to one-dimensional time series data. Pooling modifies the data by generally replacing the output of the network at a given location with a summary statistic of nearby outputs. Commonly used statistic features are peak values and averages. Figure 11 illustrates a max-pooling operation of a time-series with 20 integer data points and a filter size 4x1.

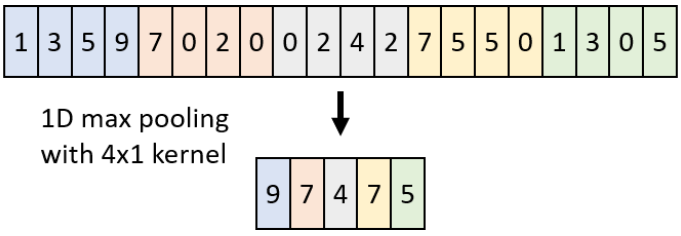

Figure 11: One-dimensional max pooling operation with filter size 4x1. Redrawn from [37]

Further aspects of modelling CNNs for processing time series are presented in Chapter 4.2.

In the following section, a further type of neural network is introduced, which is particularly suitable for sequential data such as time series.

2.3.3 Recurrent Neural Network

An RNN is a type of neural network that processes sequential data by utilizing feedback loops that create a bidirection flow of information. The network's output at one time step is used as an input at the next time step, along with the current inputs. This allows information learned from one time step to be passed on to subsequent time steps, effectively capturing temporal relationships within data sequences. Due to this ability RNNs offer significant advantages in time series forecasting in comparison to conventional NNs. They excel at modelling complex sequences and are particularly useful for predicting non-stationary time series data, such as those found in dynamic responses of vehicles to external influences [8]. Figure 12 illustrates a RNN schmatically. The individual cells in this example contain a hyperbolic tangent function, which is applied on the sum of the current input x_t and the output h_{t-1} at the last time step. The Output h_t at timestep t is passed on to the cell at timestep $t+1$ and so on.

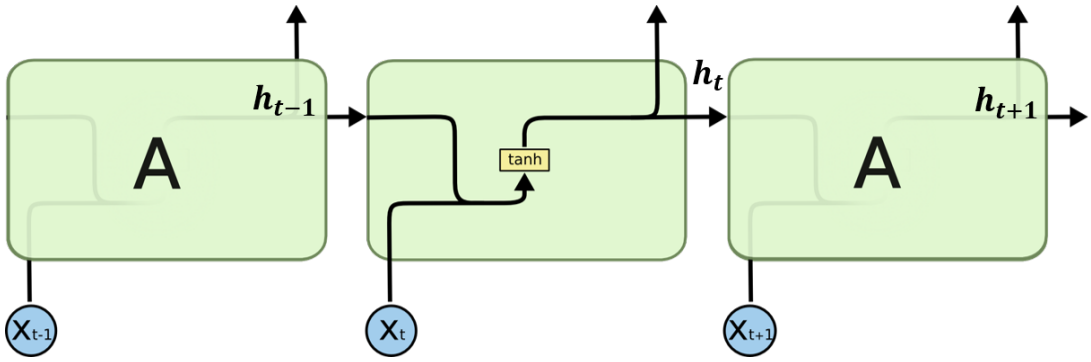

Figure 12: Schematic RNN with tanh activation function. Modified from https://dagshub.com/blog/rnn-lstm-bidirectional-lstm/

RNNs are available in different architectures such as LSTM and Gated Recurrent Unit (GRU) networks, each designed to address challenges such as vanishing gradients and effective processing of long sequences [38]. Especially the LSTM modeling approach is capable of handling long-term dependencies and is therefore frequently applied in recent studies [7], [39], [40]. The structure of a LSTM cell is much more complex than the RNN cells shown above in Figure 12. Figure 13 in comparison shows the schematic structure of a LSTM network.

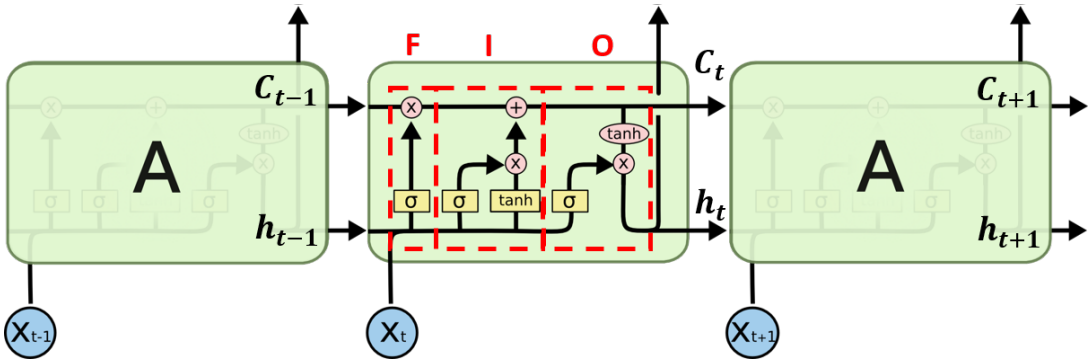

Figure 13: Recurrent Neural Network: Schematic LSTM Network. Modified from https://dagshub.com/blog/rnn-lstm-bidirectional-lstm/

It can be observed that, in contrast to the RNN network illustrated (Figure 12), the LSTM cell transmits two signals to the subsequent cell, the cell state c_t and the output h_t. The cell state retains information over long sequences, allowing the network to remember important information for extended periods.
Furthermore, three areas are identified in the structure of the LSTM cell: the forget gate (F), the input gate (I) and the output gate (O). Each gate fulfils a distinct function:

- **Forget-Gate**: This gate determines which information from the previous cell state should be forgotten or retained. It assists the LSTM in determining which parts of the sequence are relevant and delete information from the context that is no longer needed [38].
- **Input-Gate**: The more complex input gate controls how much the input x_t and the last ouput h_{t-1} contributes to the current cell state c_t.
- **Output-Gate**: The output gate regulates the flow of information that is output from the cell. It determines which components of the cell state c_t are employed in the computation of the final output h_t of the cell.

By utilising these three gates to regulate the cell state, the LSTM network employs distinct pathways for long-term and short-term memory. This enables the model to address the long-term dependency issues commonly encountered in RNNs by filtering out irrelevant information. However, due to the more complex structure and the increased number of trainable parameters, LSTM networks require more memory and are more computationally intensive when fitting the model. Additionally, the large number of parameters and complex architecture, causes the risk of overfitting, especially when the available training data is limited.

3 Description of the Database

This chapter introduces the underlying database for this work and its origin. Important aspects of the dataset are outlined to assess the quality of the data regarding the context of modeling with deep learning methods. It then describes the preprocessing pipeline used before providing the data for model training.
The datasets originate from the HavenZug project [10], which aims to optimise port operations through predictive embedded condition monitoring of the track infrastructure. The project was carried out from September 2018 to November 2021, and was initiated by the German Federal Ministry of Transport and Digital Affairs.
The measurement system used in the project consists of multi-sensor systems on the rail vehicles in the port of Braunschweig and a background system for automatic data analysis and information processing. The measurement data collected by this system enables the development of data driven analysis methods for the detection and diagnosis of rail infrastructure irregularities [10]. To further benefit from the data generated, it is used in this work for developing machine learning models to predict vibrations in the car body. The following subsection elaborates on the specific acceleration sensors in the set-up and the IMU.

3.1 Description of the Sensors used

As part of the project, a locomotive was equipped with a multi-sensor system containing two acceleration sensors and a positioning system. The acceleration sensors in both sides of the axle boxes are industrial multiple-axis vibration sensors type PCB-629A61 from PCB electronics. The Positioning System consists of a GNSS module and a 3D IMU that gives 3D acceleration, 3D rate of turn and 3D earth-magnetic field data. The IMU is type MTi-10 IMU and produced by xSense. Both, the acceleration sensor and the IMU from the measurement set-up are displayed in the Figure 14.

Figure 14: Illustration of the sensors used (PCB-629A61 Acceleration sensor on the left and MTi-10 3D Inertial Measurement Unit (IMU) on the right

Table 2 presents a comparative analysis of the most significant characteristics of the sensors. It can be observed that the ABA sensors have a much broader measurement and frequency range. Furthermore, the anticipated measurement signal noise is lower throughout the entire frequency range despite the considerably higher sample rate. The noise of ABA sensors is minimised, especially at high frequencies. Nevertheless, the IMU outperforms the ABA sensors in terms of non-linearity.

	Unit	629A61 Accelerometer	MTi-10 IMU Accelerometer
Measurement Range	m/s^2	± 490	± 200
Frequency Range	Hz	0.8 - 8000	0 - 375
Sampling Rate	Hz	20625	100
Spectral Noise	$\mu g/\sqrt{Hz}$	7.0 (10Hz) 2.8 (100Hz) 1.0(1000 Hz)	60
Non-Linearity	%	± 1	± 0.1

Table 2: Comparison of the sensor characteristics of the PCB-629A61 Acceleration sensor and the MTi-10 3D Inertial Measurement Unit [41], [42]

The IMU operated with a sample rate of 100 Hz with decreased the effective frequency range even further to 0 - 50 Hz.
During the execution of the project, measurement data was obtained from both normal operations and dedicated measurement runs on specific parts of the track network. The data provided for this work is mainly obtained from these dedicated measurements. The velocity of the locomotive during these recordings was limited to 22 km/h. The locomotive in question is of type MaK G 321 B and equipped with a two-stage suspension system. The primary suspension consists of a combination of rubber springs and dampers. The secondary suspension isolates the car body from the axes with two suspension elements per side. The total weight of the vehicle is approximately 40 tonnes. The locomotive used for the project is shown in Figure 15.

Figure 15: Locomotive equipped with Multi-Sensor-System [43]

3.2 Data Exploration

The initial step in the data analysis process is data exploration, which aims to uncover important patterns and relationships in the datasets and gain valuable insights. It is beneficial to develop a deeper understanding of the data before conducting more complex processing steps like model training.
This section will present the structure of the database and evaluate the quality of the individual measurements in order to select the quantity of samples used for modelling. In addition to the conventional evaluation metrics for time series data, such as the mean value, standard deviation, duration and extreme values, the IMU and ABA data must also be evaluated in terms of their signal correspondence.

3.2.1 Data Structure

The data was transferred in HDF5 format and is divided into so-called sessions and journeys. Each session comprises a series of journeys, recorded in sequence and defined from vehicle stillstand to stillstand. The journeys in each session are of different lengths, with the average length of journeys per session and the number of journeys also differing. For every journey, the measurement is grouped into the various signals from the IMU and the two ABA sensors. The structure of the ABA and IMU data records differs slightly from each other, as the ABA data records contain not only time stamps and 3-axis acceleration, but also vehicle speed, vehicle orientation and information on the track section on which the vehicle was located.

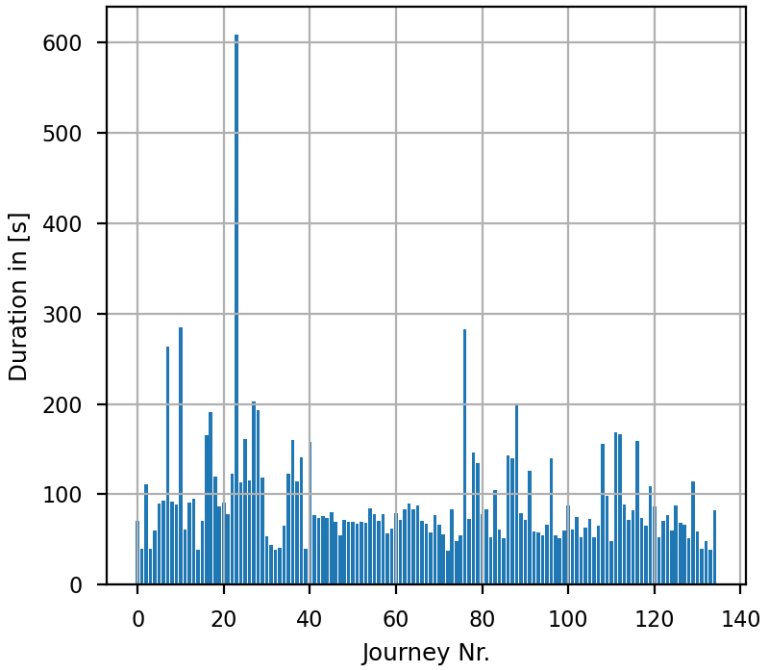

Figure 16: Distribution of journey length across all sessions

The additional information to the ABA signal actually origin from the positioning system, but

were integrated and upsampled to the ABA record before handing over the data.
A total of five sessions were conducted to develop the model, comprising a total of 136 journeys. The longest trip contained 60894 valid IMU samples, resulting in a duration of 608.94 seconds at an IMU sampling rate of 100 Hz. Conversely, the shortest journey consisted of only 3808 samples. The average length of a recording is approximately 93 seconds. The average length of all recorded trips is 93,1 seconds. A graphical representation of the journeys durations across all sessions is shown in Figure 16. It can be seen that only a few recordings significantly exceed the average length.

3.2.2 Data Handling

The disparate journey lengths and strongly differing sampling rates of the ABA sensors and the IMU present distinct challenges when handling and preprocessing the data for the actual modelling. The volume of ABA data is considerable due to the high sampling rate, which necessitates significant computing time for processing, visualisation and storage. In order to ensure efficient processing, the data was also sampled down to the IMU's 100 Hz sampling rate for analysis and exploration.
Conversely, the storage of data in tabular format is constrained. For example, Session one comprises 15 recordings with a total duration of 263.04 seconds. At a recording frequency of 100 Hz, this results in an array of dimensions 15x26304. The shortest recording is 38.89 seconds in duration. It is therefore necessary to populate the array with values in order to avoid the failure of subsequent arithmetic operations. The process of padding with values, typically zeros or boundary values, enables the arithmetic operations to be performed, but also results in the use of an unnecessarily large amount of memory usage. The impact is particularly evident when considering the magnitude of the discrepancy in the maximum case. When the shortest recording is padded to match the length of the longest in the case of session 1, it is observed that not even 15 percent of the memory is utilized for actual values, while the remainder contains padding values.
The issue can be addressed by employing sparse arrays. This specific form of array is particularly advantageous when the majority of elements within an array possess a default value that is typically either one or zero. In contrast to conventional arrays, sparse arrays do not require the storage of all elements. The indices of the elements do not have to start from 0 throughout. Consequently, only the indices of the elements that deviate from the aforementioned default value are saved. This property renders sparse arrays highly memory-efficient, as they require a significantly reduced amount of memory than a complete array with the same dimensions. Furthermore, the time required for computations on sparse arrays is often reduced, as only the pertinent elements must be considered.
In the freely available Python libraries, such as NumPy and SciPy, there are functions for padding and then concatenating padded multiple journeys of multiple lengths into a sparse array. This format is conducive to efficient processing, as it is straightforward to save and reload. This method of handling the data was found to be the most efficient during the course of the work.

3.2.3 Sensor Selection

Although two ABA signals are available for each journey, the focus was on working with the data from a single sensor. The utilisation of a single signal confers a number of advantages. In addition to the decrease in data volume and therefore less computational effort, the three signals

must be synchronised to benefit from the effect of the multi-channel usage. Conversely, the potential for information gain from utilising the second signal is constrained if the two signals contain largely analogous information. In order to fulfil a more general objective, the work is limited to the prediction of one output signal based on one input signal.
In order to ascertain which of the two sensors is more suitable for model training, it was necessary to examine a number of exemplary journeys in greater detail. It is of particular importance to ensure synchronisation with the IMU signal, which can be mathematically expressed by the signals cross-correlation. High synchronisation results in a high maximum value in the vector, which arises from the cross-correlation operation between the IMU and an ABA signal. If this extreme value is also situated at the centre of the vector, this indicates that the signals are optimally aligned with each other.
The cross-correlation indicates that one of the two signals exhibits a slight advantage over the other in terms of synchronicity. Nevertheless, the hypothesis is tested randomly. Approximately 20 percent of all journeys were randomly visualised to test the above hypothesis in order to ultimately select the sensor for model training. Figure 17 presents an example of a recording from session one. The four plots illustrate the vertical acceleration signal of ABA Sensor 1, situated below the analogue signal from ABA Sensor 2, followed by the IMU vertical acceleration signal and the vehicle velocity. The plotted signal is recorded by ABA sensor 1.
Upon examination, a clear correlation is evident between both ABA signals and the IMU signal. A correlation can also be observed with the speed signal, as the acceleration amplitudes decrease with a reduction in vehicle speed. Nonetheless, there is a more pronounced correlation between the ABA2 signal and the IMU signal, particularly in the latter part of the signal. This trend is further substantiated by the visual inspection, which is consistent with the cross-correlation results.
Furthermore, it is apparent that the IMU signal is considerably less pronounced in amplitude than that of the ABA signals. Additionally, it is observed that the IMU signal exhibits a relatively higher degree of noise and is slightly displaced in negative vertical direction. The noise is most noticable during the initial and final stages of the recording, which correspond to periods of the vehicle stopping.
A comparison of the x-axis of the diagrams reveals that the IMU signal comprises a significantly smaller number of samples than the other signals, which were all recorded by one of the two ABA sensors. The discrepancy in sample rate is the primary factor contributing to this observation, although a more detailed examination reveals a notable difference in the recording length between the IMU and ABA signals. The average difference between the signals is 0.99 seconds across all journeys and sessions. The deviation is relatively constant, with a range of 0.85 to 1.15 seconds across all recordings, with the exception of one instance. In all other cases, the IMU signals are consistently longer than the ABA signals.

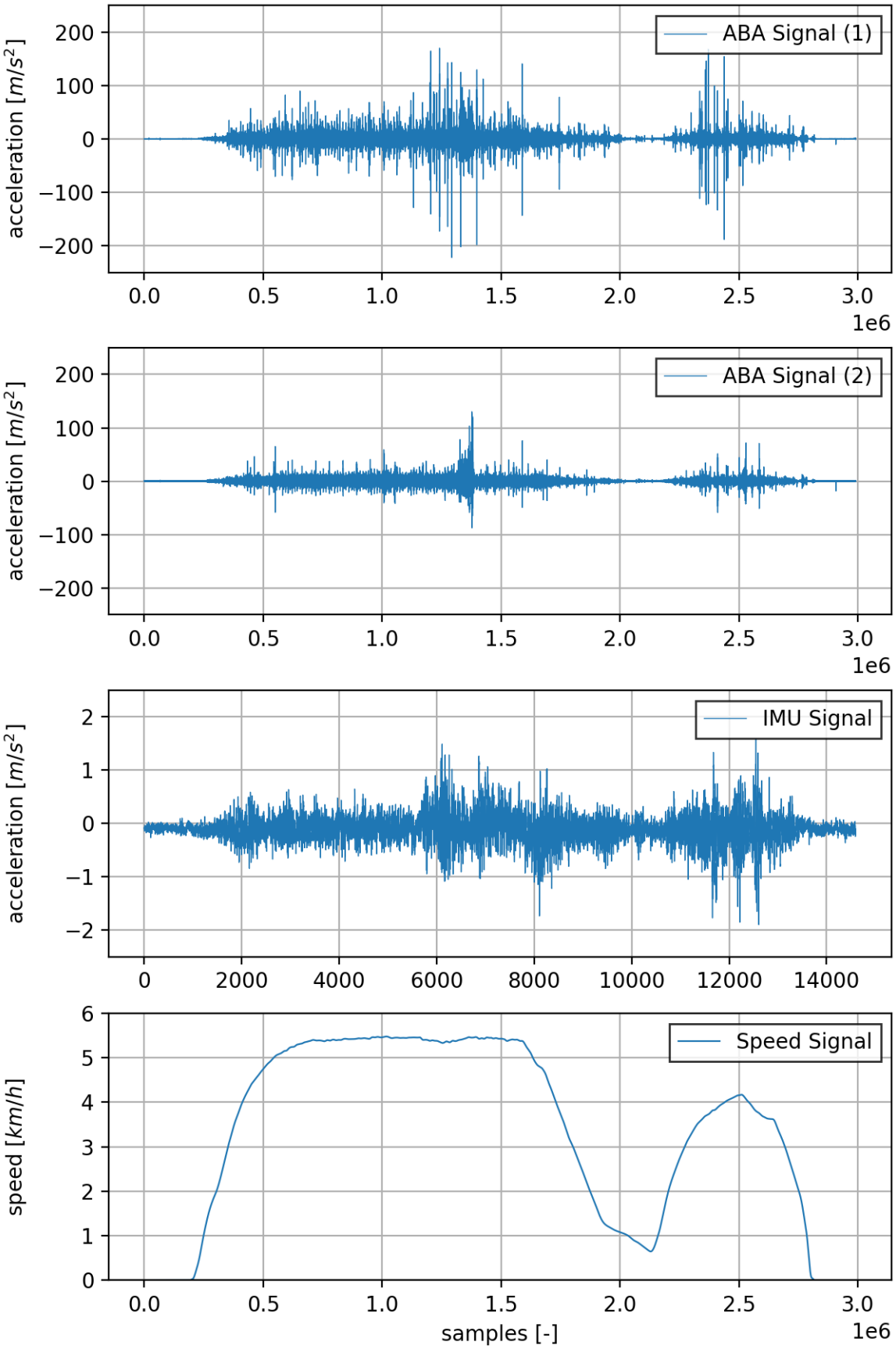

Figure 17: Plot of the acceleration signals from ABA1, ABA2, IMU and speed signal from an exemplary journey

3.2.4 Data Rating

Viewing the data set and the subsequent evaluation and selection of the training data is done to gain further insights to the data set. All recordings of the IMU and the selected ABA were assessed visually based on the synchronisation of the two signals. A rating system was implemented to categorise the recordings into three distinct categories. Ratings A, B and C were assigned to the journeys, based on the signals correspondance between ABA and IMU.
A clear correlation was observed between the ABA and the IMU signal throughout the entirety of the signal in journeys with an A rating. Class B journeys exhibit a clear correlation, although there are a few signal parts that do not appear to correlate strongly. Journeys with a C rating exhibited a considerable number of signal parts that exhibited a weak correlation with the IMU signal. This phenomenon is frequently compounded by the presence of substantial noise and generally low signal strength.
A total of 26 journeys were assigned an A rating. A high correlation was only partially observed for 59 journeys, which corresponded to a B rating. The remaining 51 journeys lack in consistent ABA and IMU signal correlation, and thus were assigned a C rating. Figure 18 illustrates the ABA and IMU signals of a C-rated journey. It is evident that the peaks at 2000 samples and shortly afterwards in the IMU signal are barely discernible in the ABA signal.

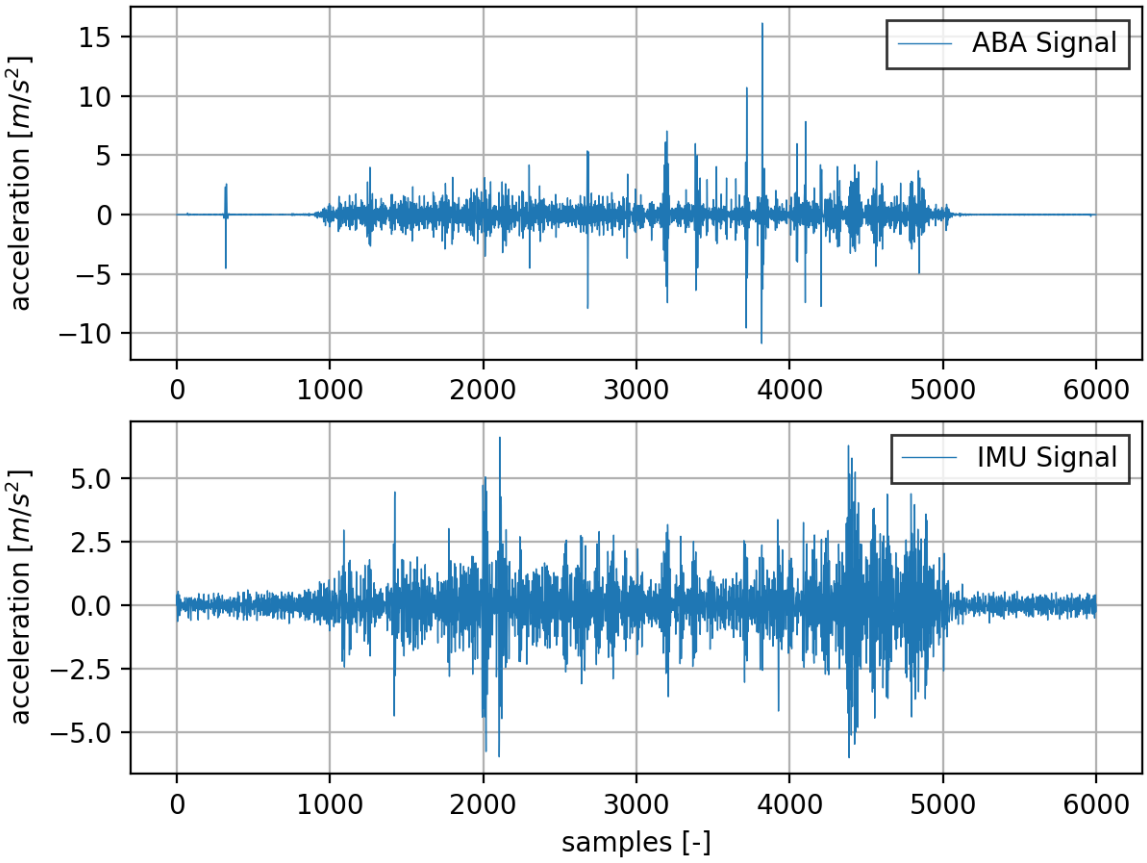

Figure 18: Journey with weak correlation between ABA and IMU signal (C-rating)

In order to facilitate the automation of the evaluation process and to obtain further information about the data set and the individual journeys, they were evaluated using a variety of

metrics. The objective of this study is to identify a metric that is consistent with the visual evaluation. This can be expressed by the correlation of the two metrics. In addition to the classic stochastic metrics, such as the mean value, standard deviation and maximum values, similarity-related metrics were also determined for all journeys. The following metrics were employed: the Dynamic Time Warp Score (DTW), the Cosine Similarity (Cosine), and an autocorrelation-based correlation metric, which does not appear in literature. The autocorrelation is the cross-correlation of the signal with itself and provides information about the noise component of the signal. The correlation metric is defined in formula 21 as P_C:

$$P_C = \frac{2 * \max |z_{X,y}|}{\max |z_{X,X}| + \max |z_{y,y}|} \tag{21}$$

where $z_{X,y}$ denotes the cross correlation of the ABA and IMU signal and $z_{X,X}$, $z_{y,y}$ the autocorrelations of the individual signals.
A heat map was then created by computing the correlation matrix for the evaluation metrics and the rating. The coefficients for the heat map were calculated based on the Pearson correlation coefficient. Figure 19 visualises the results and provides further information on the signal and the correlations between the metrics.

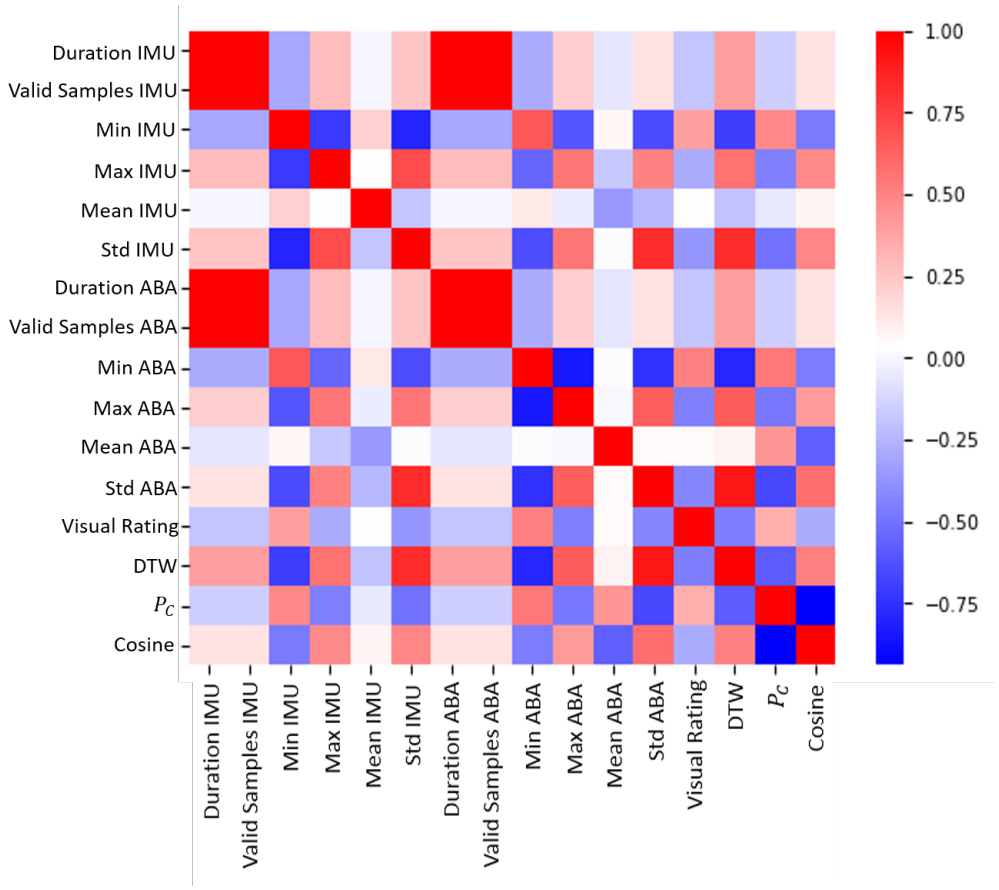

Figure 19: Heatmap showing the relationships between the evacuation metrics and the visual assessment of the data

It should be noted that the Pearson correlation is appropriate for continuous, quantitative data types. However, this is not the case with the transformed visual rating, which is ordinal data. The application of this method to this type of data contravenes the underlying assumption of the Pearson correlation that the data is continuous and normally distributed. This discrepancy may result in erroneous conclusions, which is why the results relating to the visual rating may be invalid.
However, the strongest correlation to the visual evaluation is evident with the Dynamic Time Warp Score, but this is not particularly meaningful as it correlates as strong as the standard deviation of the ABA signal for instance. The score also appears to correlate with low extreme values and large standard deviations for both IMU signal and ABA signal.
Low scatter, expressed by a low standard deviation, correlates inversely with the minimum values of both signals. This can be explained by the fact that the noise component is greater with a higher average amplitude. Furthermore, the standard deviation of one sensor correlates with that of the other sensor, indicating plausible sensor behaviour. The strong negative correlation between the minimum and maximum values of both signals is equally positive and plausible.
The evaluation of the journeys could be used in the further course of this work to select the best journeys for model training. In addition to selecting the most appropriate journeys, the correct preprocessing of data is also of crucial importance for the successful implementation of data-based modelling. The subsequent section presents the data preprocessing pipeline, which was employed in this work.

3.3 Data Preprocessing

Prior to the creation of a model, the raw data must be subjected to preprocessing. In this context, it is crucial to inspect and analyse the data in relation to the specific task at hand in order to select and apply the most appropriate preprocessing steps.
In cases where input and output data exhibit marked differences in their characteristics, it is imperative to pay special attention to the implications of this discrepancy. Especially considering the information and dimensions contained within the data. The information encoded in a time series data set is dependent upon the shape of the signal, which can be described by the frequency range of the signal and the associated amplitudes.
It is crucial to highlight that at the outset of the preprocessing pipeline, the ABA data was downsampled to the IMU sample rate of 100 Hz. In accordance with the Nyquist theorem, this implies that information above 50 Hz is irrecoverable. However, as previously stated, this is not a significant issue for the IMU signal, as it is not susceptible to detection. It is also important to note that the downsampling process resulted in a significant reduction in the amount of data. The exceptionally high sampling rate of the ABA data resulted in considerable computing times during their exploration and processing, particularly in view of the number of journeys.
The applied preprocessing pipeline comprises the filtering of the data in order to equalise the frequency spectra of the signals. Subsequently, the signals are aligned based on their cross-correlation, and any necessary adjustments are made to ensure that the dimensions of the input and output data are identical. Finally, the data is subjected to a scaling process with the objective of removing any trends that may be present. The individual steps are described in detail below.

3.3.1 Bandpass Filter

In view of the distinctive characteristics of the sensors as set out in table 2, it is imperative to implement a filtering process, given that the frequency ranges covered by the sensors exhibit a considerable disparity. The ABA sensors operate at a much higher sample frequency, which means that they can detect information that is not visible in the measurement of the IMU. This is because the frequency range cannot be detected by the IMU. Furthermore, the vibration detected by the ABA is transmitted by the two-stage damping system and damped by its low-pass character, which attenuates the amplitudes in the high frequency ranges.
By applying the same bandpass filter to both the ABA and IMU signals, it is possible to equalise the frequency ranges of the signals. Furthermore, the filter is designed to attenuate noise components simultaneously and to smooth out abnormal peaks.
In the course of the work, a digital filter was applied to the data forward-backward. The filter was designed as a second-order Butterworth bandpass filter, which results in a total filter order of 4. The advantage of forward-backward filtering is that it equalises the phase offset that can occur through simple filtering.
The bandwidth visible for the IMU is 0 to 50 Hz. However, the vibrations are strongly damped by the damping system, which results in the emergence of less dominant frequencies above 18 Hz in the IMU data. In order to filter out high-frequency components from the ABA data, the high cut frequency of the digital filter used was set to 18 Hz.
Upon analysing the IMU signals, low-frequency noise was identified. In order to eliminate these noise components, a comparatively high low-cut frequency of 6 Hz was selected and applied.

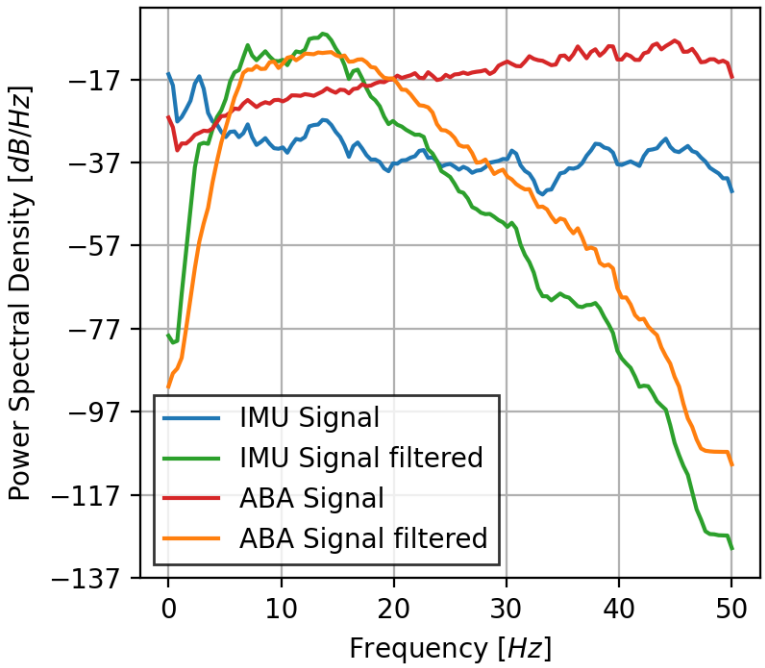

Figure 20: Frequency spectra of the raw and filtered IMU and ABA signals of an exemplary journey

Figure 20 illustrates the frequency spectra of the IMU and the ABA signal of an exemplary

journey. It also shows the frequency spectra of the filtered signals. It is evident that the filtered signals exhibit greater similarity in the frequency range than the raw data, which greatly differ in the frequency range. Furthermore, the filter demonstrates a high degree of efficacy in limiting the frequency range. In the raw data, it is apparent that the ABA signal comprises a significantly larger proportion of high frequencies, while the IMU signal is predominantly comprised of low frequencies. This pronounced tendency persists even after the application of the filter, indicating its inability to fully cancel out the effect. Although the two spectra are more similar than before the filter, the spectrum of the IMU signal appears to be shifted slightly towards lower frequencies in contrast to that of the ABA signal.
Figure 21 illustrates the outcome of data processing utilising the designed filter. The upper plot depicts the unfiltered and filtered ABA signal, while the two IMU signals are shown below.
It is evident that the filter has considerably reduced the overall amplitude of the ABA data. This effect is less pronounced in the IMU data. This can be interpreted as indicating that the high-frequency components of the ABA signals are more strongly involved in the overall amplitude than is the case with the IMU signal. The peaks were effectively suppressed for both signals. Furthermore, the IMU signal exhibited a shift of the mean towards zero, and the noise in the standstill phases was significantly reduced.
The application of a digital filter to the signals in question allows for the smoothing and limiting of the frequency range, thus facilitating the recognition of the individual vibration events present in the sensor data. This data can then be synchronised and subjected to further processing.

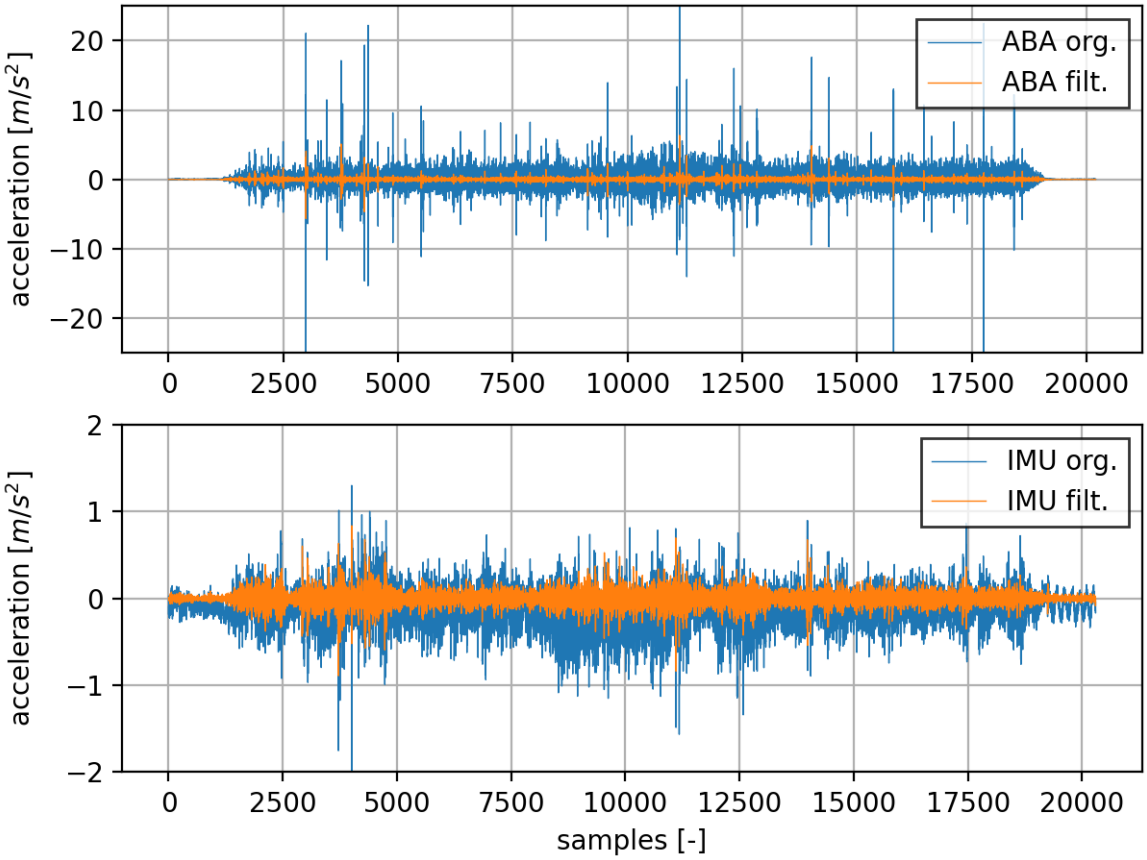

Figure 21: Result of the bandpass filter with a low cut-off frequency of 6 Hz and a high cut-off frequency of 18 Hz

3.3.2 Data Synchronisation

The process of data synchronisation involves the shifting of signals in relation to each other in order to optimise the alignment of the IMU and ABA signals. As described in chapter 2.2.1, the cross-correlation was employed for this purpose. The mathematical operation enabled the determination of the shift at which the two signals exhibited the closest degree of correlation.
Following the completion of the shift, it is essential to equalise the resulting discrepancy and pad the vector. To illustrate, if the ABA signal is shifted by 20 samples in a positive direction, the initial 20 values of the newly formed array must be padded and the final 20 values trimmed in order to achieve the same dimensionality of the signal following synchronisation.
A function has been developed which aligns the ABA signal with the IMU signal and equalises the length of the vectors. As chapter 3.2.3 demonstrates, the signals in question do not possess the same number of samples, necessitating harmonisation. In this instance, the function shortens the IMU signal by the difference and returns the aligned and equalised two signals.
Furthermore, the output includes the value by which the signal has been shifted. The value in question was subjected to analysis for all journeys. The objective was to ascertain whether journeys should be excluded from the model training process due to their extreme shifts.
Table 3 provides an overview of the sessions and the shifts in their journeys due to synchronisation. It is important to note that the number of journeys considered for training was reduced from 135 to 130 due to the identification of poor data quality during the data exploration process. In this context, the term Delta is used to refer to the shift that occurs as a result of synchronisation.

	Session 1	Session 2	Session 3	Session 4	Session 5
Nr. Journeys	14	17	42	38	19
Mean Delta	−1.36	−4.05	−3.78	65.88	−4.50
Max Delta	6	2	49	2826	3
Min Delta	−19	−25	−48	−189	−15

Table 3: Overview of the resulting sample shifts by signal synchronisation. Maximal, minimal and mean value for each session

It can be observed that there are two recordings in session 4 that exhibit highly pronounced shifts. Both extremes, one in a positive and one in a negative direction, distort the mean value and reduce the comparability of the value to the other sessions. After adjusting for the outliers and discarding the faulty recordings, the mean value is calculated as -3.68. The new mean value lies within the value range of the other sessions and indicates plausible synchronisation. It may be reasonably assumed that the observed shift, as identified through cross-correlation, is relatively consistent in nature.
Given the frequently considerable length of the recordings, up to 600 seconds in duration, a shift of only four samples on average indicates a highly synchronised measuring system. Signal offsets may be observed due to differences in the signal processing power electronics of the respective sensors, as well as by cable length and other parameters.
In the final preprocessing step, the synchronised data is scaled.

3.3.3 Data Scaling

Standardisation was employed to scale the data, with the objective of containing mean-free data with a standard deviation of 1. However, the mean value was already adjusted incidentally through filtering. As illustrated in Figure 21, the signals were not only constrained in their frequency ranges by the forward and backward bandpass filtering, but a shift of the signals in the y direction was also generated. This is particularly evident in the IMU signal. This shift corresponds to the adjustment of the mean value. Consequently, the scaling of the data is not essential for adjusting the signals' means, yet it must be applied due to the unbalanced standard deviation.
Through standardisation, the standard deviation of all signals was successfully set to 1, meaning that the data is now fully preprocessed. It is important to note that by standardising the acceleration data, specifically by dividing it by the standard deviation, the acceleration is rendered unitless. Consequently, no unit is indicated on the Y-axis in the subsequent figures that display measured or predicted accelerations.

3.3.4 Preprocessing Evaluation

The preprocessing of the data attempted to process the data in such a way as to emphasise the relevant signal components present in both signals, as well as to identify outliers and thus cleanse the data set. The data pipeline comprises three principal stages: filtering, synchronisation and scaling.
During the application of the pipeline, the data was saved between the individual operations. This was done so that the data could be analysed at different points in the processing.
This section presents a comparison between the original, raw signals and those that have undergone the three processing steps. The relationship between the signals and the manner in which signal changes are transmitted is of significant importance for the purposes of model training. Consequently, the data is always evaluated in terms of its degree of similarity or synchronicity. The synchronicity of signals is of significant importance not only in the advance stage but also during the later development process, when evaluating model predictions. It is therefore essential to further familiarise with the metrics employed for the evaluation of time series-like signals.
Besides classic plots of the data, metrics for analysing the synchronicity were collected at different preprocessing steps and compared. Such an analysis enables the evaluation of whether the desired outcome has been achieved. Moreover, the efficacy of individual preprocessing steps can be assessed and modified as necessary.
In addition to the applied metrics for time-series like signals mentioned in chapter 3.2.4, there are alternative approaches for processing signals in this manner and evaluating two signals synchronicity. One such method is the the Pearson correlation coefficient.
This coefficient, as well as the dynamic time warp score, the cosine similarity, and the value described in formula 21 resulting from correlation and autocorrelation, were calculated and analysed. For all four metrics, the higher the value, the greater the similarity of the functions or signals. In the case of the Pearson correlation and the correlation-based metric, the value range is standardised between -1 and 1 and can also be negative. Although no negative correlations between the signals are to be expected, errors in averaging can be avoided by using the absolute value. Therefore, we proceeded with the absolute values of both correlation based metrics. For the Dynamic Time Wrap score and the cosine similarity, the value range is not standardised but

always positive.
Table 4 presents the key figures for all five sessions. The average value across all journeys is displayed in each case. Each metric is applied four times: once on the raw data (index raw), after filtering (index filtered), after synchronisation (index synced) and at the end of the pipeline (index scaled).

	Session 1	Session 2	Session 3	Session 4	Session 5
DTW raw	82.462	95.626	113.698	77.992	42.105
DTW filtered	13.989	20.818	16.483	12.915	10.898
DTW synced	13.989	20.818	16.483	12.913	10.898
DTW scaled	66.362	72.747	54.966	60.265	53.134
Pearson raw	0.026	0.028	0.023	0.023	0.049
Pearson filtered	0.145	0.146	0.160	0.125	0.225
Pearson synced	0.236	0.218	0.246	0.210	0.237
Pearson scaled	0.236	0.218	0.246	0.210	0.237
Cosine raw	0.045	0.045	0.061	0.228	0.061
Cosine filtered	0.145	0.146	0.160	0.125	0.225
Cosine synced	0.236	0.218	0.246	0.210	0.237
Cosine scaled	0.236	0.218	0.246	0.210	0.237
P_C raw	0.042	0.032	0.038	0.164	0.082
P_C filtered	0.171	0.149	0.150	0.135	0.179
P_C synced	0.171	0.149	0.150	0.136	0.179
P_C scaled	0.248	0.233	0.271	0.224	0.280

Table 4: Dynamic time warp score, pearson correlation coefficient, cosine similarity and cross-correlation based factor 21 during the course of the prepocessing pipeline (mean value for each session)

The Dynamic Time Warp Score was unable to detect any adjustment of the signals in 80 percent of journeys on average. Consequently, it is not a meaningful metric and should not be considered further in this study. Nevertheless, it is evident that standardisation greatly increases the metric, while filtering greatly reduces it. This correlation is evident in the average amplitude, which can be greatly reduced by filtering and amplified by scaling. The rationale behind this phenomenon when scaling is the adjustment of the standard deviation. In the case of a narrow value range, this can result in an increase in the aforementioned range, thereby raising the average amplitude.
All other metrics experience an increase due to the preprocessing pipeline and indicate that the ABA and IMU signals are more similar due to the processing steps. It should be noted, however, that the remaining three metrics are situated within a range where the correlation is weak.
It is also notable that the average Pearson correlation factor and the cosine similarity are in perfect alignment once the data have been filtered. With the exception of a few instances, this observation can be made to five decimal places.
The scaling of the data does not affect the cosine similarity or the Pearson correlation, whereas the cross-correlation-based factor is significantly influenced by it. Conversely, this metric is unable to detect a shift of the signals against each other in the x direction, as the metric appears to be unaffected by the synchronisation.
In general, it can be stated that the metrics across all sessions exhibit a high degree of consist-

ency in their response to the respective preprocessing methods. Given that each session comprises a disparate number of journeys, which in turn vary in length, the narrow range of values observed in the metrics can be interpreted as indicative of consistent measurement quality.

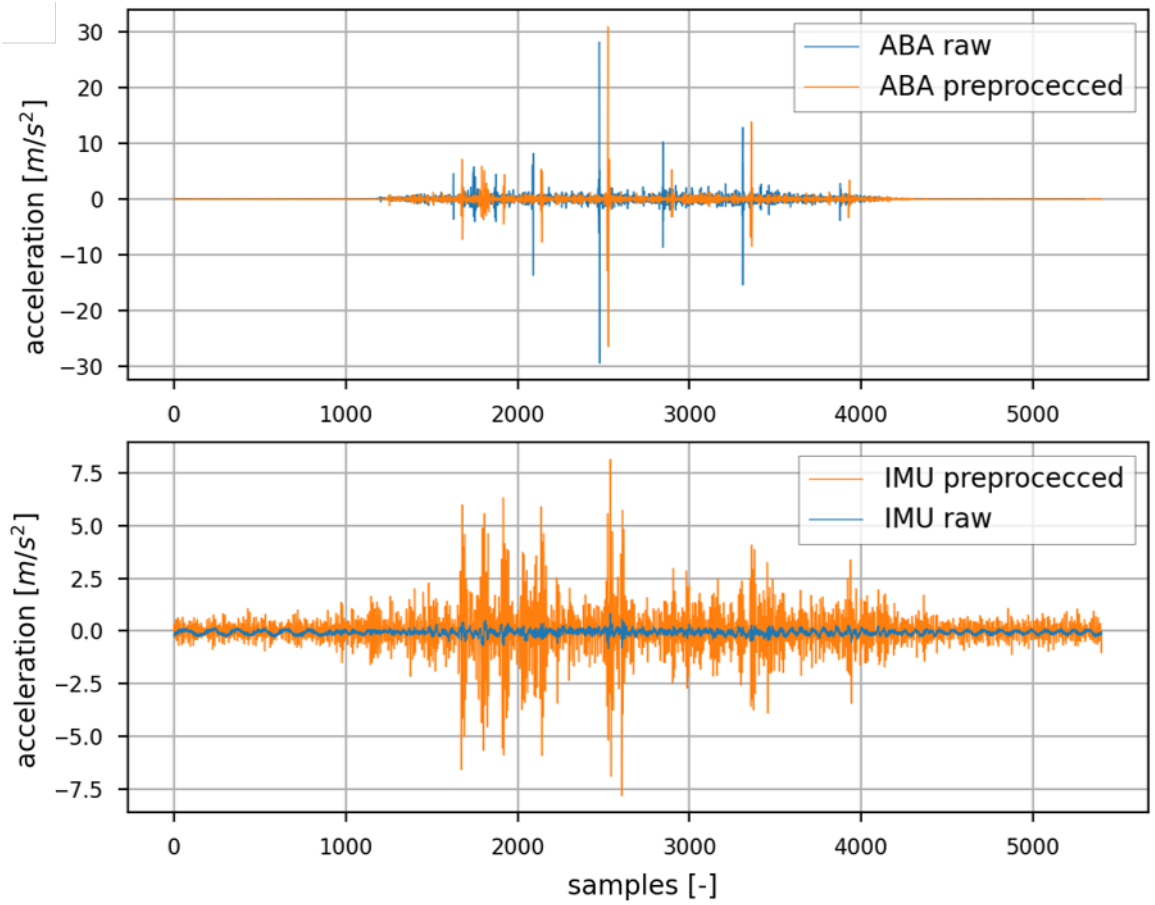

Figure 22: Plotted raw and processed signals for IMU and ABA as a result of preprocessing

Figure 22 illustrates the signals before and after preprocessing. It is evident that the ABA signal has been shifted in the positive X direction to align with the IMU signal. Additionally, both signals exhibit greater centering in the y-direction, which corresponds to the zeroing of the mean value due to the scaling. It is apparent that the IMU signal has experienced a notable increase in amplitude, yet certain signal components appear to have been accentuated. This phenomenon can be attributed to the applied filter.
The preparation of the data for subsequent modelling represents a significant and time-consuming step that serves to establish the foundation for model quality. Following the extraction and processing of the data, 135 journeys are available for model development.

4 Model Development and Evaluation

This chapter describes the modelling approaches for the prediction of the IMU signal based on the ABA signal, which employs different types of neural networks. It presents a detailed analysis of three distinct network types: classical neural networks (i.e., feedforward MLP), CNNs and RNNs.
The extent to which classical neural networks, as described in chapter 2.3.1, are capable of solving a task such as the prediction of acceleration data is analysed in the first section. This fundamental form of neural network also serves as a foundation upon which subsequent chapters are built, guiding the reader through the intricacies of modelling using NNs.
The following sections examine the potential of more complex network forms, which are similarly promising at an early stage. Recent studies [7], [9] were able to demonstrate the superiority of complex networks, such as combinations of RNNs and CNNs, in particular for time series applications.
A detailed general procedure is presented below, which outlines the necessary measures in a step-by-step manner and is applicable to all network types.

- **Data preparation and pre-processing:** The process of preprocessing was previously described in detail in the preceding chapters and consists of the same procedures for all employed network types. Subsequently, the quantity of processed data is divided into three distinct data sets: a training data set, a validation data set and a test data set. The training and validation data sets are employed for the purpose of evaluating the models performance in early development stages. The test data set is later utilized exclusively for the purpose of final testing.
- **Architecture selection and initialisation:** In this phase, the fundamental framework of the model is established. It is essential to define the dimensions of both the input and output layers in a manner that aligns with the specific requirements of the application and the data. Variable data dimensions, such as those involving journeys of varying lengths, present a particular challenge.
  In addition to the choice of dimension, further parameters must be defined. These include the choice of optimiser, the activation function of the individual layers or the initial parameterisation of the networks weights.
- **Hyperparameter optimisation:** The optimal performance of neural networks is contingent upon the optimization of a multitude of hyperparameters. As part of the optimisation process, a number of parameters are selected as hyperparameters and a comprehensive test run is initiated, which trains and evaluates a multitude of models with different configurations. The models are trained on the test data set and validated with the validation data set. The objective is to identify a parameter configuration that yields promising results, thereby enabling the subsequent final training process to be conducted with greater capacity.
- **Final training, testing and evaluation:** In the final phase, the model is again trained with the training data but with way more epochs. It is of the utmost importance to monitor the

training process meticulously and to track metrics such as loss, accuracy and validation performance. Stopping the training process prematurely or adjusting the learning rate can help to avoid overfitting.
Following the training, the model is subjected to an evaluation on the test data in order to ascertain its prediction performance on a data set that hasn't been used for training. The results of the evaluation allow for the implementation of fine-tuning techniques, which may include adjustments to hyperparameters, modifications to the architecture, or the incorporation of additional data.

4.1 Classical Neural Network

The development of an MLP for time series forecasting is described in the following section. The Keras machine learning API was employed for the project and implemented in the Python programming language.

4.1.1 Architecture Selection and Initialisation

When modeling in Keras, the first question that arises is the dimensions of the model, which are directly related to the dimensions of the training and test data. The time series data for training comprises sequences of varying lengths. This presents a challenge for the standard architectures of neural networks, which typically assume fixed input dimensions. In contrast, RNNs and CNNs are able to handle variable sizes due to their inherent properties. Nevertheless, a number of solutions have been devised to address this issue for classical neural networks. A common approach is to extend the shorter inputs with special padding tokens to a maximum length. Concurrently, masking is employed to differentiate the authentic input data from the padding tokens. This method enables the processing of variable-length inputs by bringing them to a fixed size. Nevertheless, this approach may result in inefficiencies and an increased computational effort, due to the number of padding values. Similarly, as previously outlined in Chapter 3.2.2, sparse arrays were employed for data storage during preprocessing with the objective of enhancing computational efficiency.
An alternative approach is to restructure the data in such a way that the signals retain the same form, despite their differing lengths. This approach employs window functions, which divide the signal into sub-signals of uniform length and summarize them in a matrix. The window size is of critical importance in this context. If the window is excessively large, it is possible that crucial data may be overlooked. To illustrate, consider a relatively brief recording comprising 5895 samples, processed with a window size of 1000. This yields a 5x1000 matrix, with 895 data points being discarded. One potential solution is the implementation of a padding mechanism. Conversely, if the window is too small, the system response of the system to be modelled may be too lengthy for it to be represented in a partial signal. Consequently, the model would attempt to make predictions based on input signals that do not correspond to the actual output of the system.
The window function method was employed in the development of the model. This decision was made on the grounds of computational efficiency. Furthermore, the window size can be optimised as a hyperparameter, thereby affording further possibilities and insights. The subsequent decision to be made when selecting the network architecture is that of the output dimension. Consequently, the training data, in this case the IMU signals, must also be structured

in accordance with the aforementioned principles. Once more, there are a number of potential approaches to be taken when utilising the window function. While it is theoretically possible to select the output dimension at will, not all dimensions are logically coherent. A window size of 100 permits the prediction of a single value as well as to predict the next 100. However, it appears that maintaining the dimensions of the input and output variables in a consistent manner is the most logical approach. For this reason, the IMU signals were structured in such a way that they corresponded to the input, the ABA signals. The result is a network that processes a subsignal of the ABA sensor as a variable-size window and outputs the corresponding IMU subsignal.

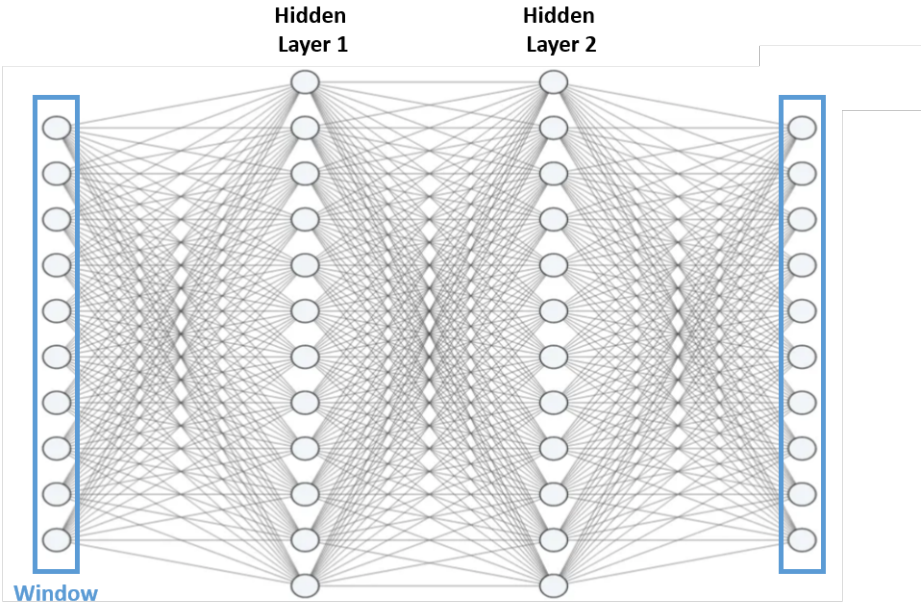

Figure 23: Schematic MLP with two hidden layers and a width of units 12 per layer

For the concrete implementation in Python, the dimensions of the input layer depend on the size of the window. The input layer is followed by one or more hidden layers. The final hidden layer is succeeded by the output layer, which serves to output the values in the form of an interface. The dimensions of the output layer are identical to those of the input layer. Consequently, the window size determines the input and output size of the network and represents a significant hyperparameter. Models with different window sizes can be tested and evaluated using an automated data processing pipeline.
The number of neurons in the hidden layers is interpreted as the width of the network, while the number of layers is generally referred to as its depth. The width and depth of the network represent important hyperparameters of the model, which are examined in more detail during optimisation.
Figure 23 provides an illustrative example of an MLP, which is used to visualise the dimensions and the relationship between the input and output data. The illustration depicts an MLP with two Hidden layers, each comprising 12 neurons. The window size employed is 10, which is evident in the dimensions of the input and output layers.
Another crucial decision is the selection of the activation function to be applied to the respective neurons. A number of common activation functions were subjected to testing. However, due to

the value range of the acceleration signals, it is necessary, at least in the output layer, to select an activation function that can also cover a negative value range. It was found that the hyperbolic tangent function was well suited to this purpose.
The selection of a loss function is of significant consequence for the calculations that occur in the background during the process of model fitting. In the context of time series predictions, there are a number of potential options. However, the Root Mean Squared Error (RMSE) is considered to be the most reliable. This is particularly the case when the target variables have a value range that includes both positive and negative numbers, as the Absolute Mean Error (AME) can lead to erroneous results. The RMSE was employed as a loss function for the models constructed as part of the study. The Keras API provides the requisite functions, which are then transferred to the corresponding interfaces when the model is compiled.
In addition to the aforementioned parameters, more must be taken into account when initialising the network. These include the learning rate, the batch size and the choice of optimisation algorithm. The majority of the parameters employed in this study were derived from standard literature sources or other online information sources.
Once the model has been initialised and no calculation problems have occurred, the first models can be trained and evaluated. The objective here is not to achieve the greatest accuracy; rather, it is to ascertain whether the design of the network and its dimensions are correct. If this is the case, the most crucial parameters will be identified in the subsequent step and then tested in conjunction with one another as part of the hyperparameter optimisation process.

4.1.2 Hyperparameter Optimisation

The optimisation of hyperparameters is of paramount importance when utilising multilayer perceptrons (MLPs) for time series prediction, as it the decisive factor in achieving optimal performance. Two critical hyperparameters are model depth and model width. The depth of the model, defined as the number of hidden layers, enables the network to capture more complex patterns and relationships in the data. Nevertheless, an excessive depth can give rise to issues such as overfitting or difficulties in training. Consequently, the optimal depth must be meticulously selected in order to guarantee balanced performance.
The model width, defined as the number of neurons in the hidden layers, enhances the network's capacity to approximate complex functions. However, an excessive width can also result in overfitting and longer training times. Consequently, an appropriate width must be identified that enables satisfactory performance with an acceptable computation time.
Another crucial hyperparameter in time series prediction with multilayer perceptrons (MLPs) is the window size. As outlined in previous subsection, a window size that is too small may result in the loss of information, while a window size that is too large may unnecessarily increase complexity and make training more difficult.
The hyperparameter optimisation for the work is implemented through the use of a training loop in Python. In this approach, the selected hyperparameters are systematically varied over multiple iterations within a previously defined range. This enables the comprehensive testing of the full configuration space. The number of epochs is kept to a minimum in order to achieve a manageable computing time despite the large number of variants. The outcomes of this optimisation process are then evaluated and analysed in order to identify the most promising configuration and to conduct further training on it. Furthermore, the results can be employed to elucidate the interrelationships between hyperparameters and their influence on the model behaviour.
It is prudent to define the ranges within which the hyperparameters are varied in advance, as this

will reduce the amount of unnecessary computing and programming effort required. The limits were initially determined through empirical experimentation for the aforementioned purpose.
While the optimal parameters for the optimisation were being sought, it was discovered that a classical neural network in the form of an MLP does not yield promising results for the task of time series prediction. It appears that the network is unable to recognise the connection between the signals. As a result, the parameterisation selected by the model training is such that the output is mapped directly to the input, rather than the desired IMU signal corresponding to the input.

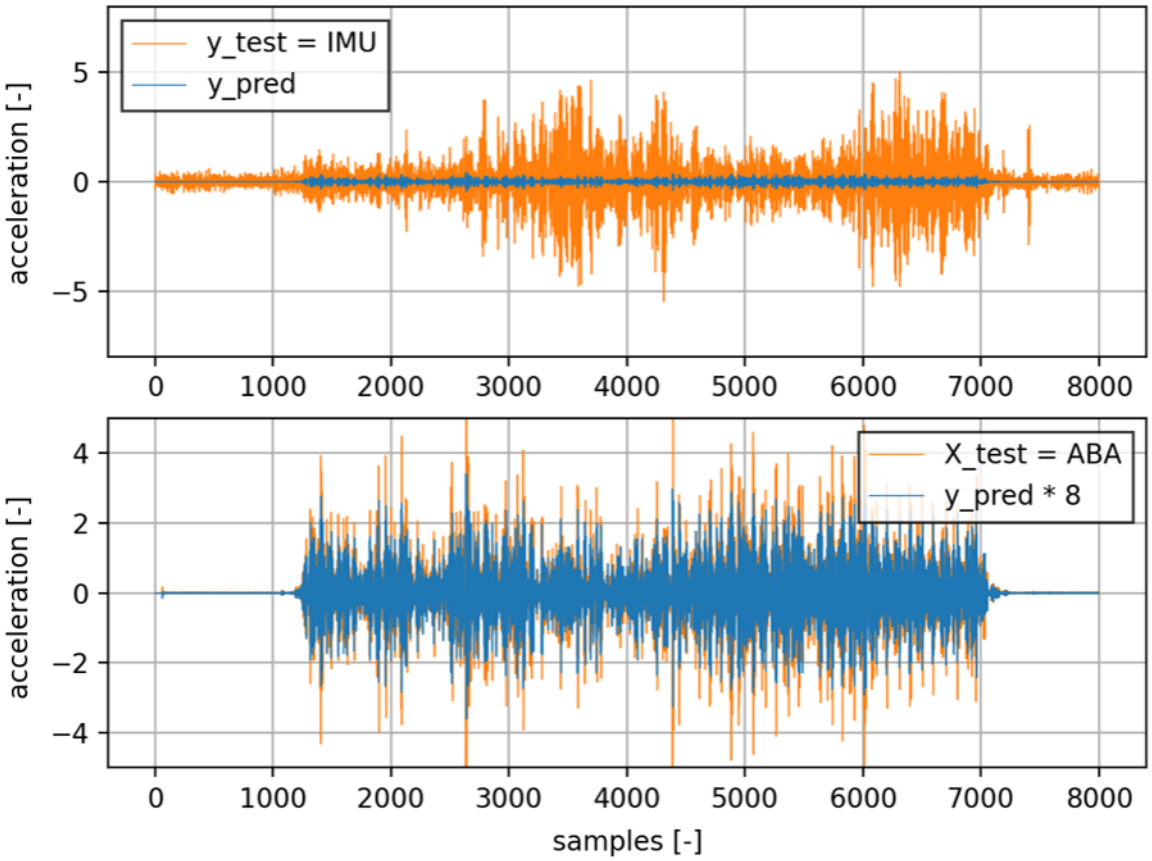

Figure 24: Plotted prediction of classical neural network in comparison to the scaled input and output signals

Figure 24 illustrates this effect and depicts the input and output signals, as well as the predicted signal by the model. This is a journey from the test data set, which was not used in the model training. The prediction was generated by a model comprising two hidden layers with 100 neurons each. The data is processed with a fixed window size of 100.
The prediction is shown in both plots, but is scaled up by a factor of 8 in the lower plot for better visualisation. The value range of the network output appears to be limited and does not exceed that of the input data. Furthermore, as described above, the prediction appears to be a damped version of the input and is not transformed by the model to follow the IMU signal.
Varying the hyperparameters did not change this behaviour. However, with more complex structures, deeper and wider meshes, there was a better fit to the input in terms of amplitude. An enhancement in this regard could also be attained by increasing the initial weights of the dense

layers. However, no discernible improvement could be achieved with regard to the prediction quality.

4.1.3 Training and Evaluation

Given the lack of predictive power exhibited by the model, no further attempts were made to optimise the MLP. Instead of a computationally intensive test run and analysis, more complex network types were employed directly.
MLPs treat each input as an independent instance, and do not take into account any sequence information or dependencies on previous points in time. Consequently, they are unable to identify and represent the inherent patterns and trends in temporal sequences of data. The prediction in MLPs is based solely on the current input values, without consideration of the sequence and correlations between successive points in time.
This assumption of independence is at odds with the nature of time series, where values often depend on previous values and have a sequential structure. Consequently, multilayer perceptron (MLP) networks are unable to adequately capture the temporal dynamics and dependencies in time series data, which significantly limits their predictive power for this application.
The absence of literature on the use of MLPs for time series forecasting supports the assumption that this type of model is not suitable for this purpose.

4.2 Convolutional Neural Network

This section outlines the methodology employed in the development of a CNN for time series prediction. The implementation is carried out using special one dimensional convolutional layers in the Keras machine learning API.

4.2.1 Architecture Selection and Initialisation

CNNs are capable of handling variable input sizes due to the fact that the convolutional operation with filters can be applied to inputs of any size by moving the filter over the entire input. It is therefore unnecessary to restructure the data with a window function. However, the data cannot be processed directly on the sparse array. Instead, the data is transformed into a NumPy array, whereby shorter time series are padded with zeros so that all signals have the same number of values. In the interface to the model, only those time steps that were not padded can then be filtered using a masking layer.
The input layer with masking is followed by a specific number of convolutional layers. The number of layers corresponds to the model depth and is a hyperparameter that is optimised in the subsequent section. The model depth exerts a profound influence on the number of parameters to be trained, which in turn affects the computational intensity of the model training. Consequently, networks that are excessively deep are excluded due to a lack of computing capacity. In a convolutional layer, several kernels can be applied to the signal simultaneously, in a manner analogous to the number of neurons in a dense layer. The number of kernel units is another hyperparameter, corresponding to the model width. As with the depth, this hyperparameter is set in a variety of parameter configurations. Given that the width of the model has a relatively minor impact on the computational intensity of the training process, the parameter in question can be varied over a wider range than the model depth. The final hyperparameter to

be analysed in detail is the length of the one-dimensional convolution kernel. A larger kernel is capable of capturing a greater degree of context, but this can also result in an increased number of parameters and the potential for overfitting.
In addition to the hyperparameters, further parameters are defined during design of the convolutional layers. These parameters are then not varied during optimisation, ensuring that they are the same in all configurations. One parameter to be considered here is the padding method. The padding method employed in the Conv1D layer of Keras determines the manner in which the input data is padded prior to the convolution operation. A number of options are available within the API, with the 'same' option being selected. The input is padded in such a way that the spatial output size is equal to the input size.
As with the development of the MLP, the predictions of the initial model tests were found to be considerably smaller than required in terms of amplitude. The same behavioural pattern was observed in the initial drafts of the CNN model. To address this issue, the initial weights of the convolution kernel were selected to have a standard deviation of 1. This resulted in relatively large initial weights and corresponding outputs.
The selection of an activation function applied on the convolutional layer has a significant impact on the model's predictive capabilities. A number of different activation functions were trialled during the design of the convolutional layers. Figure 25 illustrates the model prediction of the same network with distinct activation functions in the convolutional layers. The network in question is a convolutional neural network (CNN) with a width of 10 units. The kernel size was set to 100 for the purposes of this comparison. A rectified linear unit (ReLU) is applied at the top, while a hyperbolic tangent activation function is applied at the bottom. It is evident that the output amplitude of the prediction is considerably higher than the actual IMU signal when the ReLU is applied. In contrast, the lower graph depicts a considerably smaller prediction than the target value. Nevertheless, the hyperbolic tangent function was identified as a more promising approach for further modelling.
The output of a convolution operation can be one dimension larger than the input depending on the layers units, with the dimensions of the output varying according to the width of the model. Following convolution with, for instance, 10 kernels, 10 distinct signals are generated, each calculated from the respective kernel weights and the signal. The kernel size plays no role in this effect. In order to obtain an output of the network that corresponds to the dimensions of the generated output, the signals must be recombined into one signal. This can be achieved by adding another convolutional layer if the kernel size is set to 1 and the unity number to 1. This special parameterisation generates a linear combiner. This calculates a signal from the 10 resulting signals by weighting the individual data points and calculating a sum for each point in time. Consequently, the output dimension is equivalent to the input dimension.

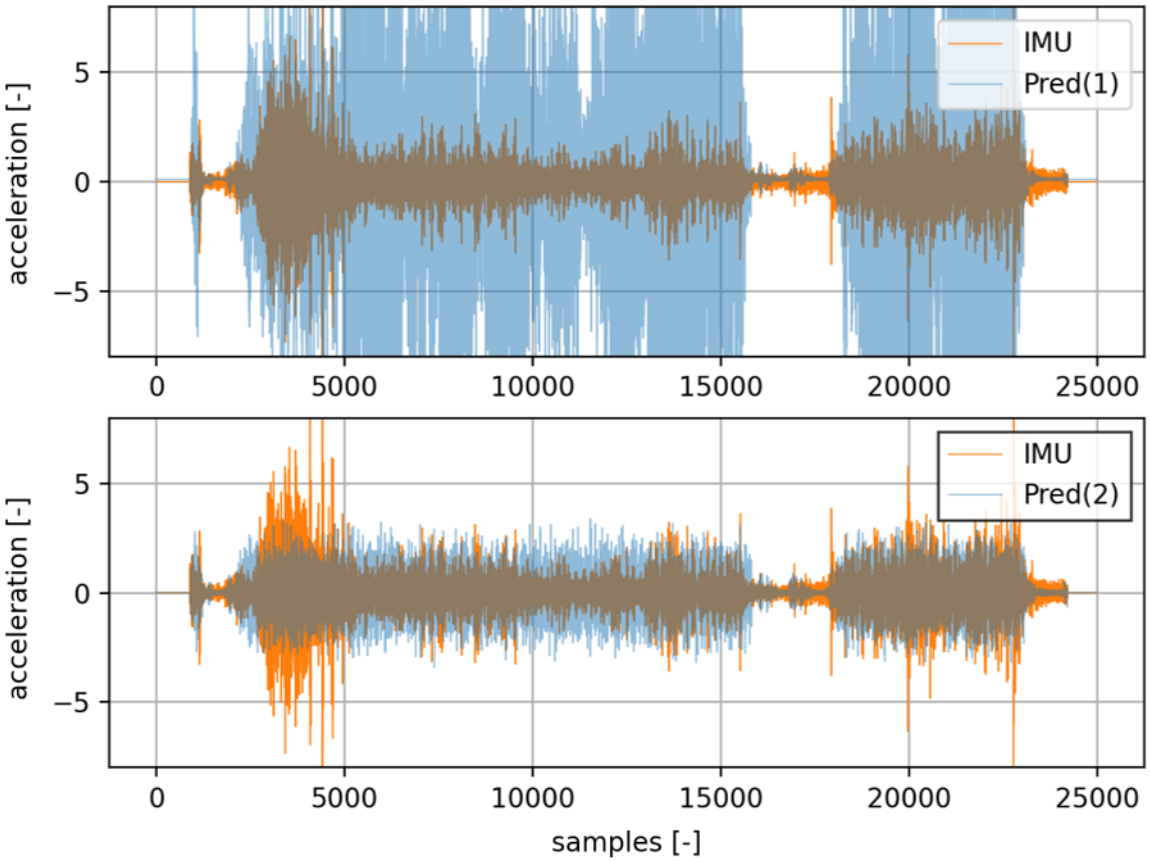

Figure 25: Plotted predictions of two convolutional neural networks in comparison to the scaled IMU Signal. Prediction 1: Convolutional Layer with RelU activation in hidden layer, Prediction 2: Convolutional Layer with hyperbolic tangent activation in hidden layer

As in the case of the MLP, the loss function employed is the RMSE. The learning rate represents a distinguishing factor between the MLP and this approach. This was selected as a variable parameter, which stabilises the training process and prevents over-adaptation in the event of learning stagnation. The learning rate may be adjusted during the training process via a callback mechanism, which is a specific method that is called at fixed times during training. The ReduceLROnPlateau callback in Keras is designed to automatically reduce the learning rate during training if a certain metric is no longer improved over a number of epochs. This callback was employed in the development of the CNN. The metric that is monitored is the RMSE of the validation dataset. The initial learning rate is set to 0.001 and is reduced by a factor of 0.1 when the loss function stagnates until a minimum learning rate of $1 * 10^{-9}$ is reached.
Once the fundamental model structure has been established and initialisation has been completed, the hyperparameters can be optimised in the subsequent step.

4.2.2 Hyperparameter Optimisation

In order to identify a promising parameter configuration efficiently, a comprehensive test run was initiated, during which the parameters kernel size, model depth (number of layers) and model width (number of units) were systematically varied. A total of 18 distinct models were

subjected to training and subsequent comparison. The ranges in which the parameters were varied are as follows:

- Kernelsize: 10, 100, 250
- Number of units: 1, 5, 10
- Number of layers: 1, 2

In comparison to the other two parameters, the number of layers was only tested on a smaller scale. This decision was made because the networks depth has a significant impact on the number of trainable parameters of the respective model. In particular, very deep networks result in a very high number of parameters, which in turn leads to lengthy training times. Should it be demonstrated that models with two layers outperform shallower networks, regardless of their kernelsize and number of units, further combinations can be tested in the course of the work, with a particular focus on deeper networks. The use of fewer units allows for the training of deeper nets to be completed in a relatively short period of time. Figure 26 depicts the RMSE over the entire training period for all 9 models with just one hidden layer. The RMSE is calculated for the training data only. Each model was trained for a total of 750 epochs.

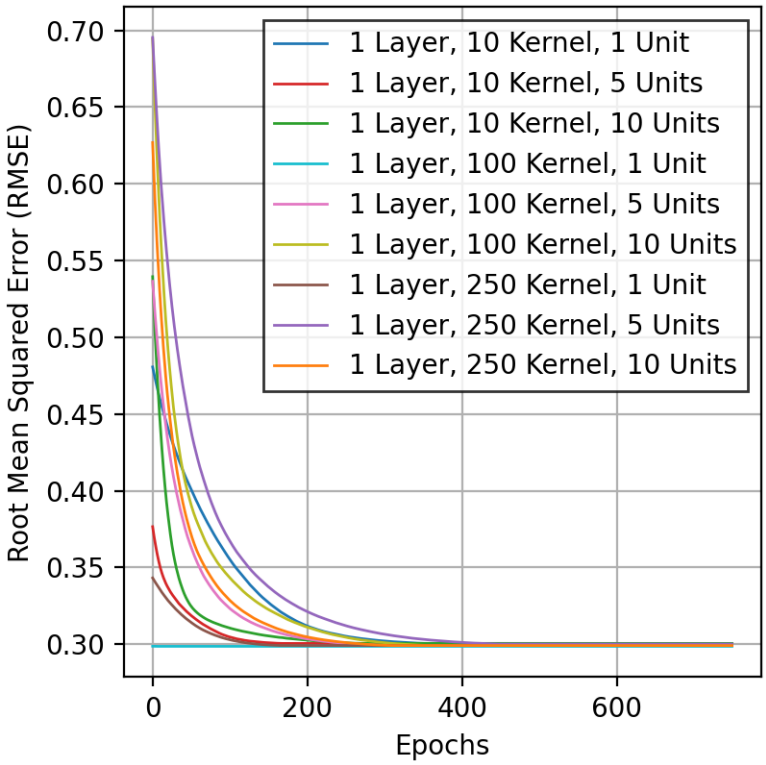

Figure 26: Root Mean Squared Error of model prediction and scaled training data labels for single hidden layer CNNs

It is evident that the RSME reaches a relatively stable value for all models with minimal training required. The threshold value appears to be 0.300 in the case of single layers of CNNs. It appears that the training process reaches a local minimum, which cannot be overcome even by adjusting the learning rate. It is also noteworthy that this phenomenon occurs in all models.

Nevertheless, it is evident that the rate at which the models reach the plateau differs. In the case of a kernel size of 100 and a single unit, the model reaches the minimum within a few individual epochs. In contrast, the model with a kernel size of 250 and five units only reaches the steady-state value after approximately 400 epochs. The models with a single unit and a large kernel (250 and 100) reach the value the fastest. It can be observed that more complex networks tend to take a longer period of time to reach a steady state.
For purposes of comparison, Figure 27 depicts the same diagram for the CNN models with two hidden layers. It is clearly visible that the same behaviour occurs as for models with only a single layer in Figure 26. Once again, all curves approach a limit value, which is approximately 0.3. It is notable that the model with a model size of 10 and only one unit exhibits a markedly different behaviour as it approaches the limit value. A local minimum is reached with the greatest speed when the kernel size is set to 10 and the number of units is 5. Nevertheless, there does not appear to be a correlation between speed of reaching the threshold and model complexity.

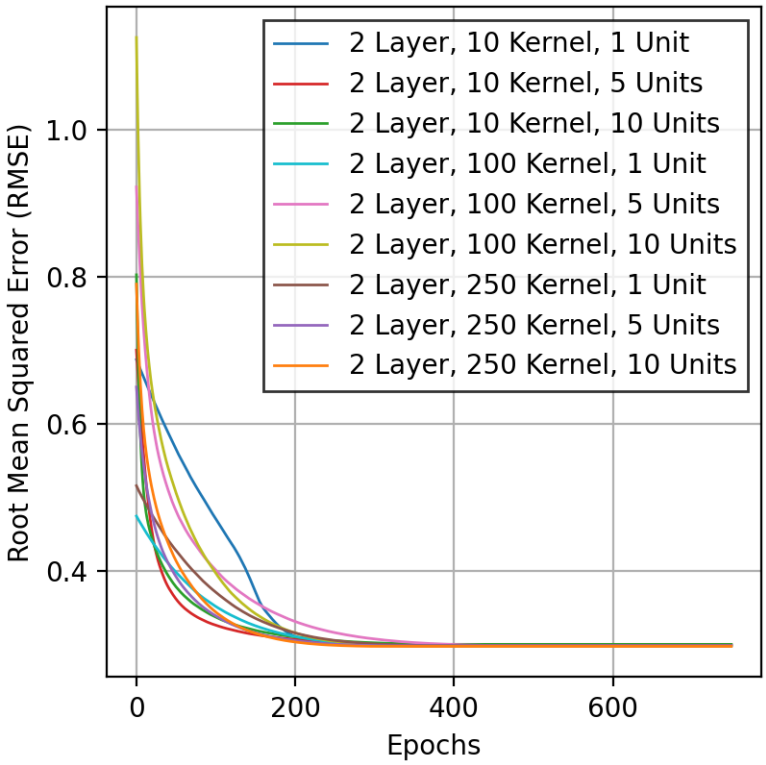

Figure 27: Root Mean Squared Error of model prediction and scaled training data labels for double hidden layer CNNs

A common feature of all models is that the limit value can be observed with sufficient clarity well before the maximum number of training epochs have been reached, and the model parameters are no longer adjusted after a maximum of 400 epochs. In order to minimise the number of unnecessary calculation steps in the final epochs, a specific Keras callback is employed in the subsequent process. This enables the training to be terminated at an early stage when an insurmountable plateau is reached.
Figures 26 and 27 only refer to the application of the model to the training data. Validation using

the validation data set is important for every model training. Through the relationship between the prediction accuracy on the training data and that on the validation data, conclusions can be drawn about the generalisation of the model. It is therefore immensely important to also evaluate the predictions for the validation data set. Figure 28 estimates the RMSE for the validation data set. CNN models with a single hidden layer are shown.

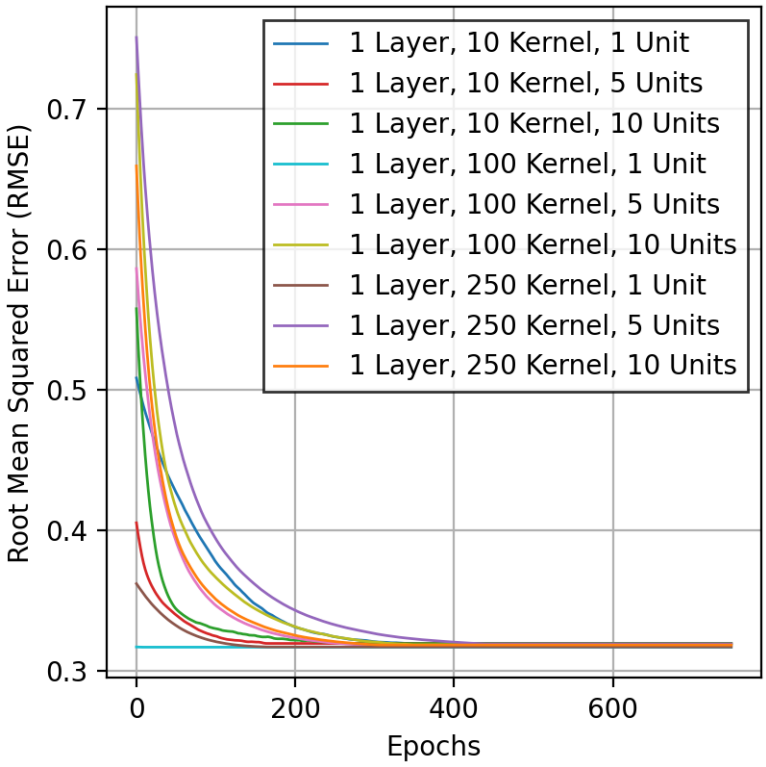

Figure 28: Root Mean Squared Error of model prediction and scaled validation data labels for single hidden layer CNNs

The behaviour of the models on the validation data is identical to that observed on the test data. The sole distinction between the two sets of data is the threshold, which has undergone a slight increase from 0.3 to 0.318. In all other respects, the curves exhibit identical behaviour. The model with a kernel size of 100 and a single unit again demonstrates the most rapid adaptation to the new threshold.
The application of double hidden layer models to the validation data yielded the same results, which is why the RMSE of these models is not illustrated again. As with the single-layer models, the RMSE has also changed, with a value of 0.318.
It can be observed that all models appear to converge towards a similar local minimum, as they all approach a similar threshold. Having demonstrated that the model with a single hidden layer, a kernel size of 100 and a single convolutional unit is the fastest to reach the threshold, the specific model's prediction is analysed in greater detail below. The models prediction to the validation data set is compared to a prediction generated by a model with the same kernel size and number of layers, but with five units in order to discover any differences.

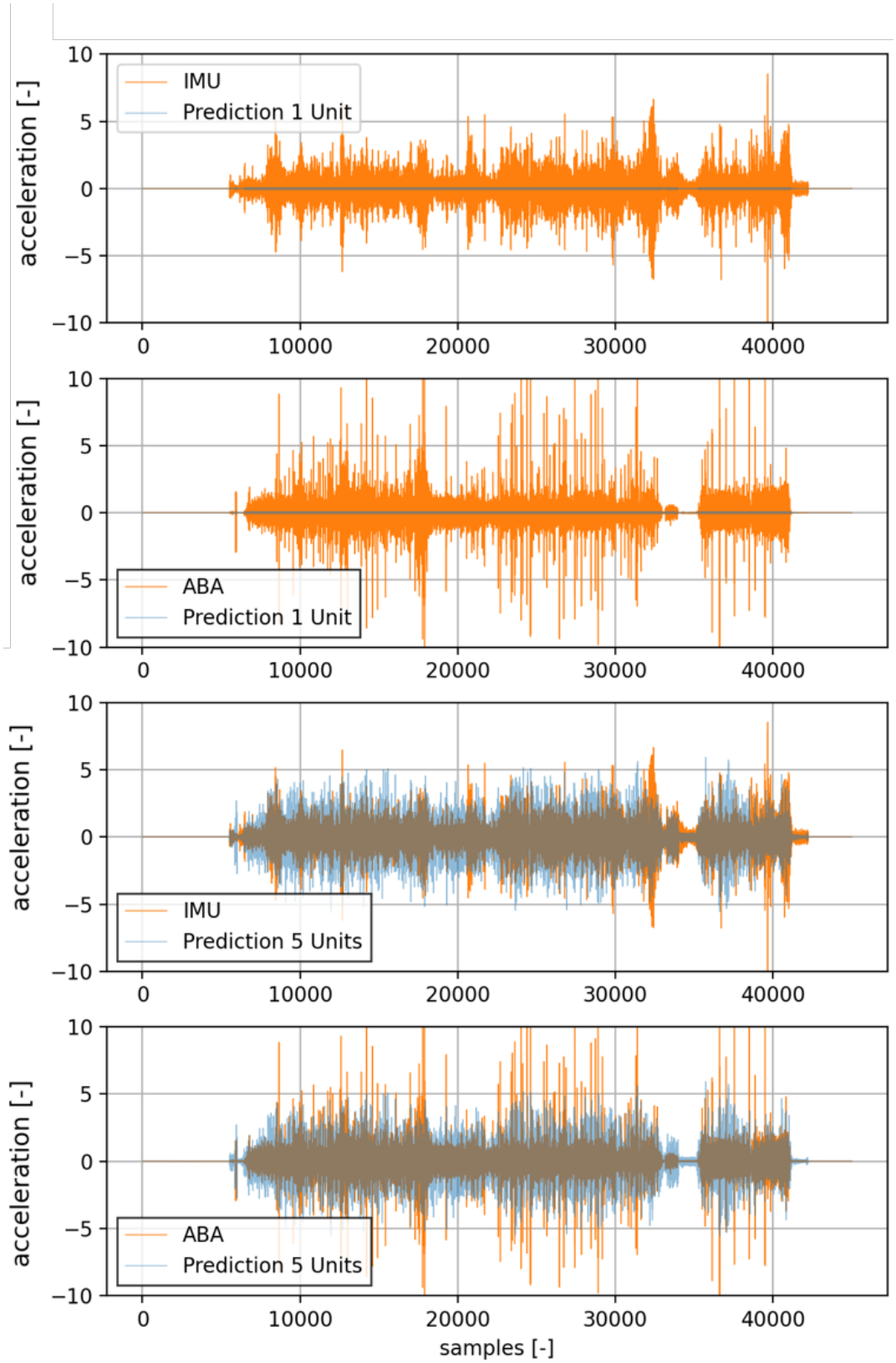

Figure 29: Plotted predictions of two convolutional neural networks in comparison to the scaled input and output signals

Figure 29 displays the predictions and corresponding IMU and ABA Signals. The prediction of the model with a single hidden layer, a kernel size of 100 and one convolutional unit exhibits undesirable behaviour. The model prediction is notably small and exhibits high-frequency fluctuations around the zero point, suggesting that the model may be attempting to adjust the output to the value zero independently of the input. This appears to be a local minimum that was identified during the training of the model. The model with multiple units appears to offer a more promising prediction in comparison. Consequently, it appears that the initially promising parameter combination converges rapidly, yet ultimately results in an inconsequential parameterisation.
In comparison to the predictions of the classic NN as shown in Figure 24 the displayed figure reveals that the CNN model with five units attempts to approximate the IMU signal. A clear illustration of this phenomenon can be observed in the accompanying figure. Subsequently, a phase with a relatively weak ABA input signal is observed at approximately 32,000 samples. Nevertheless, the model generates an output that attempts to emulate the IMU signal.Nevertheless, it is important to note that there is no clear correlation between the input signal and the prediction. The larger peaks in the ABA data, such as those observed at approximately 18,000 samples, which are clearly visible in the IMU signal, are lost in a prediction that appears to be a noise-like signal and are therefore not recognisable. Furthermore, the average amplitude of the prediction appears to exceed that of the ground truth.
The subsequent section will analyse the training of model with 5 units in depth and apply it to the training data.

4.2.3 Training and Evaluation

In order to better understand the model training process, a model was partially trained with the previously defined parameter configuration and repeatedly applied to the test data for the prediction. The result of this process is shown in Figure 30. It can be seen that as the training progresses (number of epochs), the prediction becomes smaller in terms of amplitude. After 1000 epochs of parameterisation with the training data, the result is a very weak signal in terms of amplitude, which bears little or no relation to the IMU signal. This behaviour corresponds to the prediction when using a CNN with only one unit as shown in Figure 29.
This finding corresponds to the picture that has already emerged during the testing of the different parameter configurations. The model training leads to a stagnation of the prediction quality for all tested parameter combinations. However, the speed at which this plateau is achieved is different.
In comparison to the outcomes achieved with conventional neural networks, the forecasts of convolutional neural networks exhibit a significant disparity. As training progresses, the predictions of the conventional neural networks converge to the input signals. However, the model parameters of the CNNs converge in such a way that a time series is predicted which approaches a local minimum, which is unable to reflect the characteristics of the IMU signal. The resulting prediction is of insufficient amplitude and fails to respond adequately to impulses in the input data.

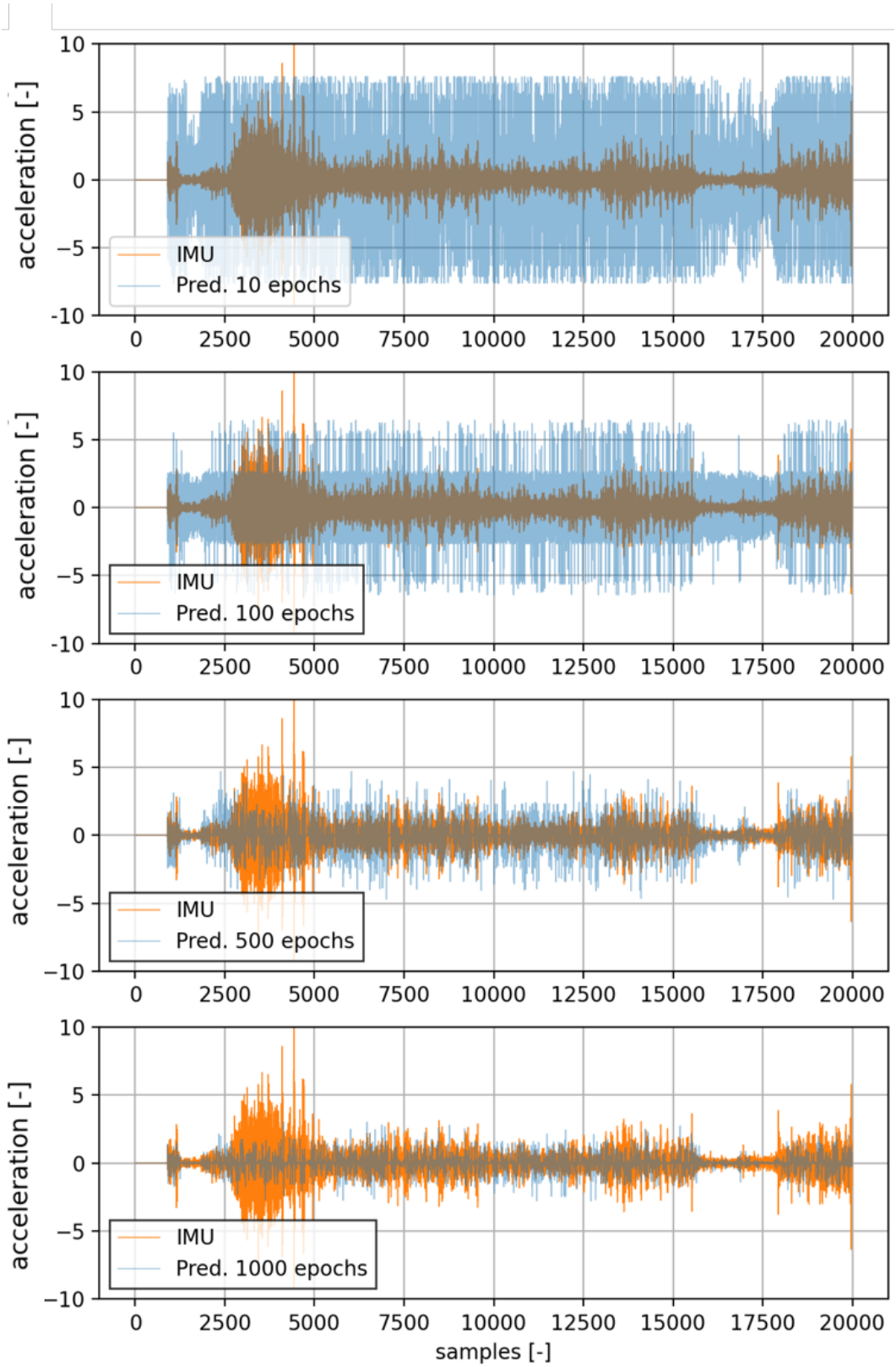

Figure 30: Plotted predictions of a convolutional neural network at different steps of the training process in comparison to the scaled IMU Signal

There are a number of potential explanations for the model's inability to converge against this parameterisation, as well as the poor quality of the predictions on the test data. One might be inclined to associate such model behaviour with overfitting. However, the fact that predictions based on training data also exhibit a lack of quality speaks against this hypothesis. In this instance, it is more probable that underfitting is present, as the model has not been adequately adapted to both the test and training data.
The development of neural networks based on the convolutional operation will be terminated at this point. The focus will then shift to RNNs, with the aim of determining whether similar issues arise or whether the sequential processing of signals in the case of RNNs can lead to an improvement in this application. In the subsequent discussion to the model development section, the potential for further improvement is identified through the use of CNNs. Additionally, the capabilities of CNNs in pattern identification are considered.

4.3 Recurrent Neural Network

The following chapter presents a detailed account of the modelling techniques employed in RNNs. In particular, the LSTM layers of the Keras machine learning API are used. The process commences with a description of the underlying model structure, in accordance with the methodology described in Chapter 4. This is followed by a process of optimising selected hyperparameters. Finally, the selected model is trained and applied to the test data set in the concluding section of the chapter.

4.3.1 Architecture Selection and Initialisation

Prior to commencing the modelling process, it is essential to define the input and output dimensions of the model. The sequential processing and output of information in an LSTM allows the realization of a multitude of model architectures with this element. The length of the input vector is variable, as is the length of the resulting output. It is also possible to predict only one value, resulting in a scalar output. In practice, this would mean that the model processes the first 1000 samples of the ABA data set to predict the 1001 sample of the IMU data set. Nevertheless, the initial 1000 samples of the IMU data set can also be predicted. This is contingent upon the selection of the model structure and the corresponding adaptation of the data sets for training, testing, and validation.
In the event that the input signal is only partially supplied as input to the network, the data records must be restructured once more using a window function, as is the case in Chapter 4.1.1 when utilising conventional neural networks. In contrast, as with the CNN, the entire time series can be fed into the network simultaneously. This obviates the necessity for the processing step with the window function. Furthermore, the model is designed to receive the entirety of the input signal for prediction, rather than a subset, which appears to be advantageous and sensible. Consequently, the model architecture was chosen to accommodate the processing of entire ABA signals in order to output the full IMU Signal. This type of RNN is referred to as a many-to-many model.
As with the CNN, the data records must first be transferred from the sparse array format to a padded array in order to compensate for the different journey lengths. In the input layer of the RNN, the padded zeros must be ignored using a masking layer.
The input layer is succeeded by the LSTM layers. The number of subsequent layers is inter-

preted here as the model depth, while the unit number of individual layers represents the model width. The subsequent chapter will provide a detailed analysis of both parameters. The hyperbolic tangent function was selected as the activation function for the LSTM layers. The initial weights are generated from a normal distribution with a standard deviation of 1, ensuring an optimal starting position for learning.
The 'return-sequences' argument is of paramount importance for the selected model architecture when defining the LSTM layer. This setting determines whether the LSTM layer outputs the full sequence or only the output after the last time step. In order to implement the many-to-many approach, the LSTM layers were defined in such a way that each layer outputs the full sequence. In the event that a network comprising a multitude of units is employed, for instance five, this will result in the generation of five sequences. The length of each sequence will correspond to that of the input.
A further layer is incorporated into the model in order to integrate the sequences and generate an output with the same dimensionality as the input. This dense layer is embedded in a wrapper, which allows each element of the sequences to be processed individually. In the case of an LSTM layer with five units, the initial values of the five sequences are inserted into a dense layer, which generates a single final value for the initial time step. A linear activation function was selected for this layer. This implies that the output value is a linear combination of the values of the individual sequences. This is carried out independently for each time step. The 'TimeDistributed' wrapper is particularly well-suited to the processing of sequential information and can be employed in conjunction with other layers. Furthermore, the loss function and learning rate were selected for the modelling of the RNN in the same manner as for the CNN.
Once the fundamental structure of the model had been established, the objective was to identify a suitable parameter combination as part of the hyperparameter optimisation process. Prior to commencing the test run, which involved varying the two hyperparameters, namely model depth and model width, a comparatively simple model was trained and its prediction analysed. This was done to ascertain whether a local minimum occurs during model training, as was observed with the other two model approaches previously, which severely limits the prediction capability. A model comprising a single layer of LSTM cells with a depth of one and a width of 10 units was trained for a total of 10 epochs. Figure 31 illustrates the model's prediction on the validation dataset for a representative journey.
It can be observed that the amplitude of the prediction is once again relatively insignificant in comparison to the IMU and ABA signal. Despite the limited number of epochs, the parameters are adjusted by the model training to achieve a very small output value with great rapidity. Furthermore, the prediction appears to align with the signal of the ABA sensor, exhibiting a similar response to larger deflections of the IMU, despite low activation, as evidenced by the lack of prediction at approximately 32,000 samples.
These observations indicate the presence of a local minimum, which is reminiscent of the results presented in previous chapters. However, they may also be indicative of a lack of model complexity. Consequently, more complex models are trained and evaluated in the following section.

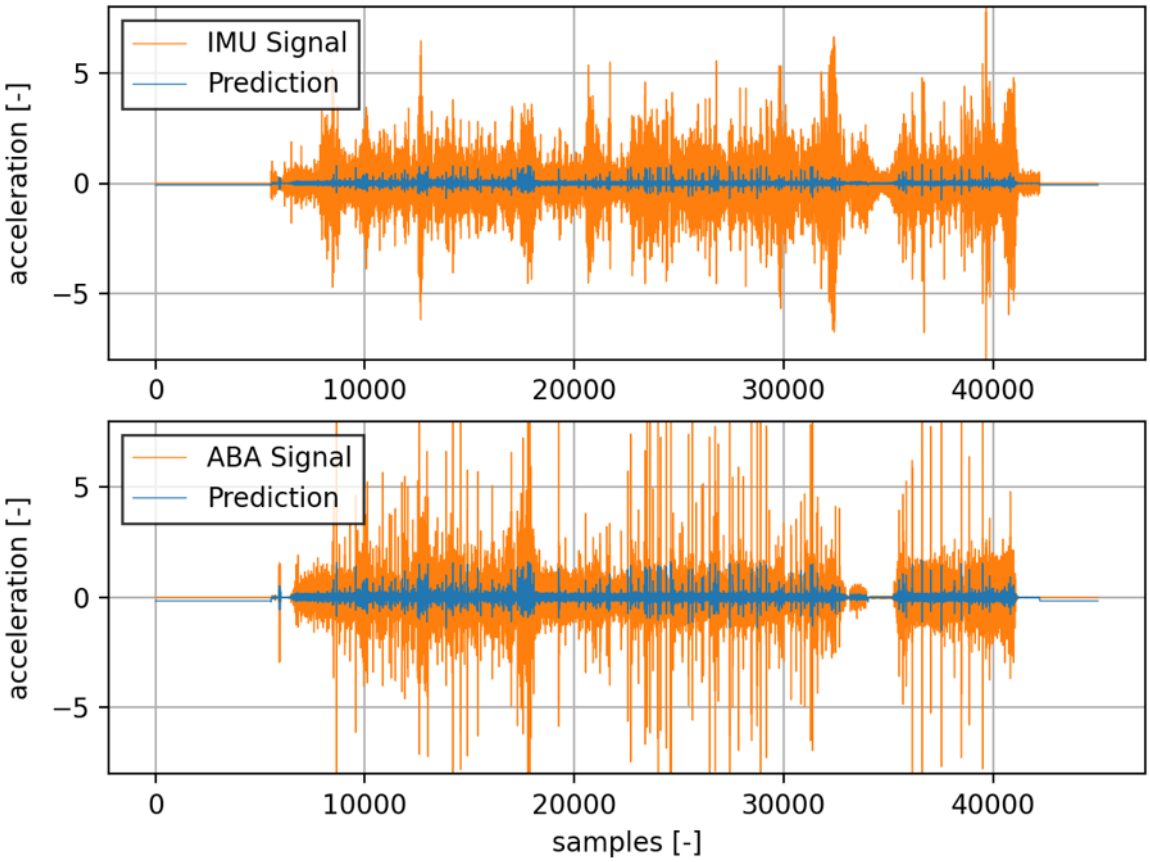

Figure 31: Plotted predictions of a recurrent neural network in comparison to the scaled input and output signals

4.3.2 Hyperparameter Optimisation

In comparison to the CNNs, fewer models were trained and subjected to comparative analysis. The depth of the model was varied between one and two LSTM layers, while the width of the model was tested in the range of one to 10. The number of layers in the dense layer following the LSTM was not varied. The models were each trained for 300 epochs. Figure 32 illustrates the root mean square error over the course of training. The losses in comparison to the training data are presented in each case.
It can be observed that all models, with the exception of one, achieve a plateau within a relatively short period, typically less than 50 epochs. At this point, the model parameters cease to be adjusted. The model with one layer and ten units is an outlier, requiring almost 100 epochs to reach a stable value. Nevertheless, no correlation can be drawn with a pronounced model width, as the model with two layers and the same width stabilises very quickly. Given that all models exhibit a similar value, it is reasonable to conclude that they also exhibit a similar parameterisation.
Upon examination of the RMSE of the models in comparison to the validation data (not shown), a similar pattern emerges. Following less than 50 epochs, all models exhibited a loss of approximately 0.318, which was slightly above the level of the test data. In this case, the model with one layer and 10 units is also an outlier. The model that attains the stationary value in the

shortest time is the one with two layers and one unit. It is noteworthy that the most complex model, comprising two layers and 10 units, attains the final steady-state value with remarkable swiftness in comparison to the model with the same width, while also exhibiting the lowest root mean square error.

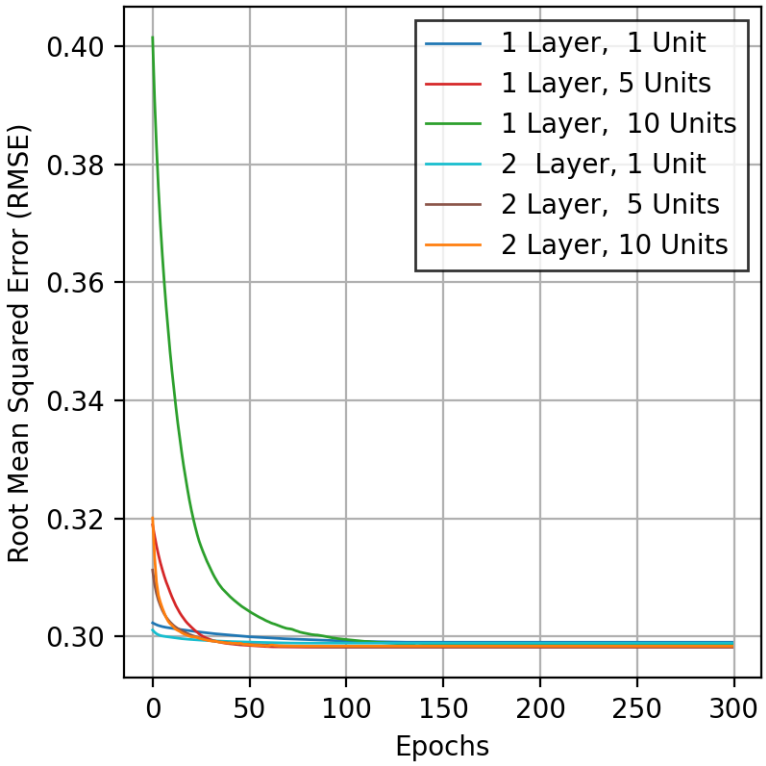

Figure 32: Root Mean Squared Error of model prediction and scaled training data labels from RNN models

4.3.3 Training and Evaluation

Based on the results from the parameter optimisation, extensive model training beyond 300 epochs was not carried out. When applied to the test data, the picture was similar to that of the validation data and the desired level of prediction accuracy could not be achieved. Subsequently, the model comprising two layers with one unit per layer which reached the stationary value first was subjected to examination, with a particular focus on the manner in which the prediction undergoes modification during the training process. This was undertaken with the objective of attaining a more comprehensive understanding of the parameterisation process.
Figure 33 shows the prediction of the RNN model on the test data set during different phases of training. It can be seen that the prediction becomes smaller in amplitude as training progresses. However, from epoch 100 onwards there are hardly any parameter adjustments. This can be concluded from the fact that the two predictions are identical after 100 and 300 epochs. Finally, the prediction of the RNN is also more correlated with the input signal of the ABA sensor than with the signal of the IMU, and a satisfactory prediction accuracy could not be achieved.

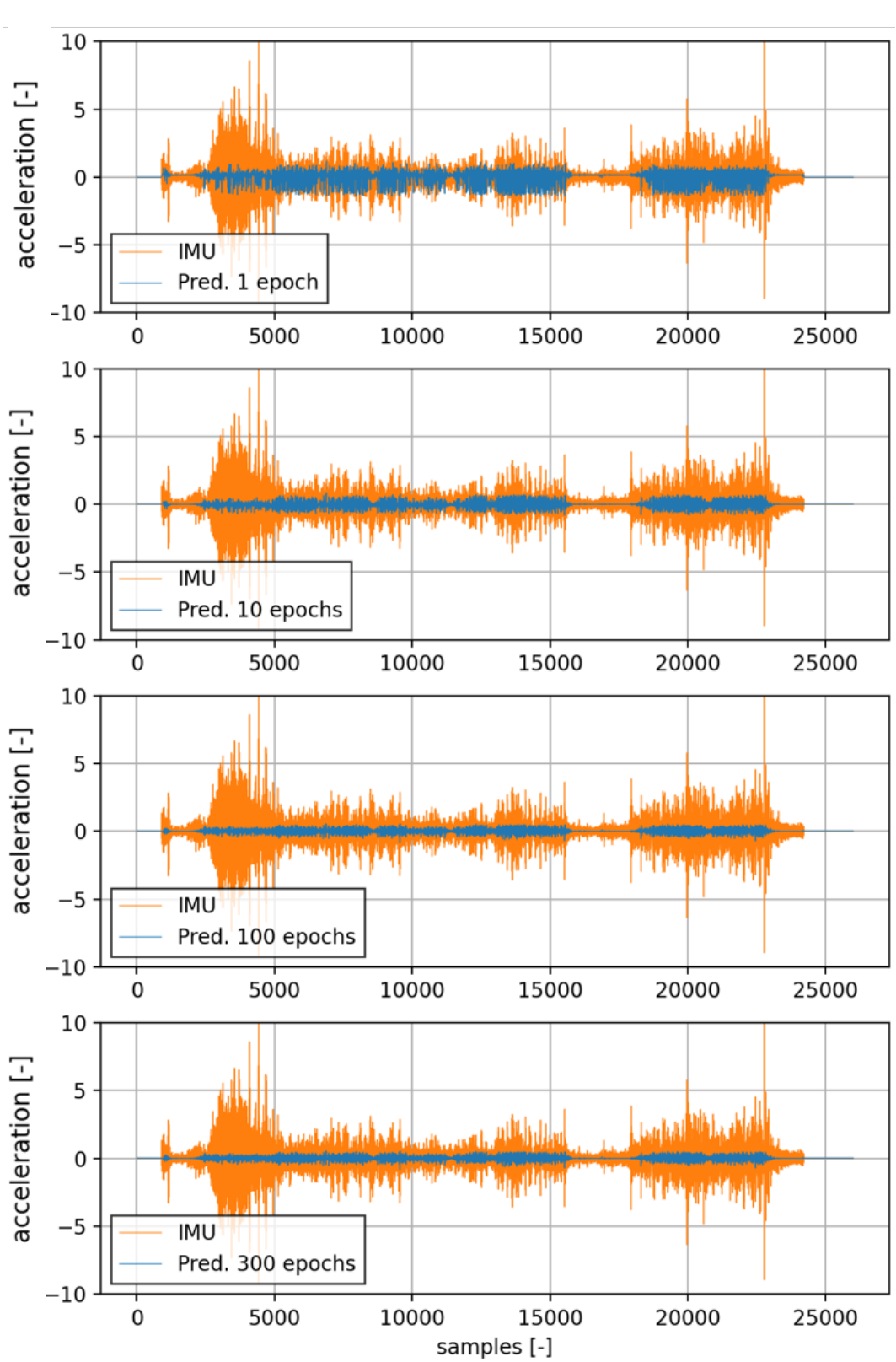

Figure 33: Plotted predictions of a recurrent neural network at different steps of the training process in comparison to the scaled IMU Signal

Investigations of the predictions of the models with varying parameter configurations demonstrated comparable behaviour to that of the model used for the prediction in Figure 31. As the number of training epochs increases, the output of the model tends towards a time series that approaches zero and does not reflect the behaviour of the IMU signal. Instead, the prediction correlates more strongly with the input signal, but is significantly smaller in the amplitude range than the original ABA Signal.
Further adjustments were made to the model in order to prevent this behaviour, including variations of the batch size, the learning rate and the initialisation method of the weights for the LSTM layers. None of the selected measures were found to result in a significant improvement.
Following the modelling, a discussion is initiated in the subsequent chapter, which presents a summary and comparison of the results of this chapter. Subsequently, an attempt is made to discuss the reasons for the lack of model quality and to identify any potential for improvement.

5 Further Results and Discussion

The preceding chapter outlined three modelling approaches and their concrete implementation via the machine learning API Keras. Conventional neural networks (multilayer perceptrons), convolutional neural networks and recurrent neural networks of various configurations were created and trained on the pre-processed data to find a model that could predict the IMU signal to a corresponding ABA signal. Regrettably, none of the methodologies employed yielded a predictive model with an acceptable degree of accuracy. Despite a comprehensive analysis of the model parameters and an effort to adapt them, no parameter configurations could be identified that effectively addressed this issue. The subsequent section presents a further comparison of the results of the modelling.

5.1 Comparison of the Modelling Results

	NN	CNN	RNN
RMSE Train Data [-]	1.0207	0.2988	0.2894
RMSE Validation Data [-]	1.0167	0.3165	0.3163
RMSE Test Data [-]	1.0178	0.46928	0.4700
Nr. of Epochs to stop [-]	14	10	10

Table 5: Overview of the resulting RMSE of predictions and scaled IMU Signals from different models

Subsequently, an effort is made to comprehend the underlying reasons for the models' lack of predictive accuracy and to identify any additional measures that may be necessary.
Table 5 presents the root mean square error of three models when applied to test, validation, and training data. The models are one model per treated model type. The following parameterisations were used for the predictions in Table 5:

- NN: Number of Layers: 2, Number of Units: 100, Window size: 100
- CNN: Number of Layers: 1, Number of Units: 1, Kernel size: 100
- RNN: Number of Layers: 2, Number of Units: 1

All models were trained using a regularisation technique that monitors a certain metric during training and, in the event of stagnation, terminates training prematurely before the set epochs have elapsed. The aforementioned callback is employed to circumvent overfitting, whereby the loss observed with respect to the validation data is typically considered rather than that pertaining to the training data. For all three model types, a parameterisation was reached at which the training process reached a plateau, indicating that no further progress could be made in terms of prediction accuracy. In addition to the RMSE for the different data sets, Table 5

also shows the number of epochs after which the training processes were terminated by the aforementioned callback.
It can be observed that the RNN has the lowest RMSE for the training and validation data set. In relation to the test data set, the RMSE of the CNN model is minimally lower. Nevertheless, it is evident that the results of the CNN and the RNN are highly comparable across all data sets. Furthermore, the models exhibit a similar pattern of increasing RMSE from the test data set to the validation data set and then to the test data set, indicating a deterioration in prediction accuracy. The conventional NN exhibits RMSEs that are approximately three times as large as those of the other two models. Additionally, the loss for the NN differs only insignificantly between the data sets. It is uncommon for the accuracy on the test data set to be higher than on the training data set. This eliminates the possibility of overfitting.
Upon examination of the number of epochs passed through, it becomes evident that both the CNN and the RNN have the same number of epochs. This is due to the threshold value stored in the callback, which has the effect of causing the system to wait a few epochs before cancelling the training.
By analysing the progression of the RMSE over the epochs, as illustrated in Figure 34, it can be observed that the loss is stabilising prematurely. The figure depicts the RMSE in relation to the training and validation data for all network types.

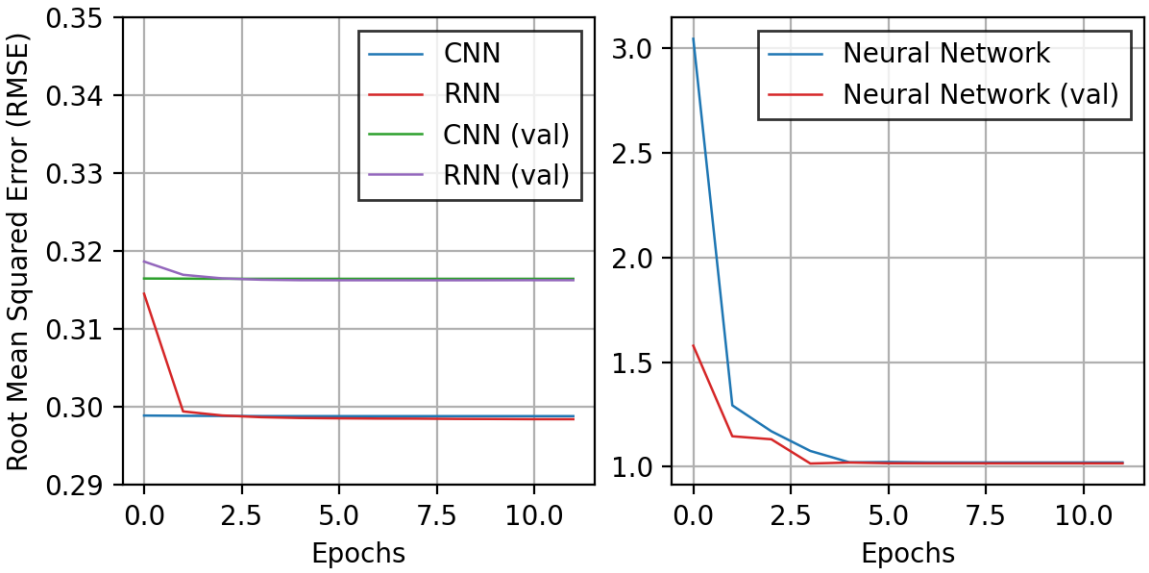

Figure 34: Root Mean Squared Error of model prediction and scaled validation data labels for single hidden layer CNNs

It can be observed that the CNN exhibits a loss of performance after just one epoch, with minimal adaptation during the remainder of the training period. In contrast, the RNN demonstrates a plateau after one epoch. These findings apply to both the training and validation data sets. The conventional neural network stabilises after four epochs. Notably, the neural network is the only model in which the initial value of the loss is lower for the validation data set than for the training data set.
At this early stage of training, achieving optimal parameterisation is unusual. This indicates that the models have reached a local minimum during parameterisation, which cannot be over-

come. It can be theorised that reducing the learning rate may prove beneficial in this instance. Nevertheless, this approach was implemented, yet the observed outcome did not demonstrate a discernible impact on the model behaviour.
Given that the RMSE of the CNN and RNN models is comparable across all data sets, it can be inferred that the predictions generated by the two models will also exhibit a similar correlation. To gain a more profound understanding of the limitations of the models, a prediction for an exemplary journey from the validation data set was created for each model and displayed graphically. Figure 35 illustrates the prediction of each model in a stacked plot, with the actual IMU signal to be predicted superimposed. In the case of the CNN and the RNN, the predictions are so small in terms of amplitude that no deflections can be discerned with the naked eye.
In order to facilitate visualisation, the signals were increased by a factor. The signal of the CNN was close to zero, therefore the factor, which is 10 for the RNN, must be 100 for the CNN in order to recognise shapes in the signal. For purposes of comparison, the bottom plot depicts the input signal, which served as the correlating ABA signal and was employed as input for all three models.
The commonality between the CNN and RNN models becomes apparent when examining the tendency of the models to end in a parameterisation that generates predictions close to zero in order to minimise the loss related to the IMU signal. However, when zooming in, a different shape between the CNN prediction and the RNN prediction can be observed. While the output of the CNN is almost identical to the input signal, with only a slight reduction in magnitude, the RNN produces a prediction that is strongly positive in the y-direction. It appears that the RNN model is unable to predict negative accelerations.
In terms of optical comparison, the prediction of the conventional NN appears to be the most closely aligned with the desired prediction, despite exhibiting a relatively larger RMSE. Nevertheless, it cannot be stated with certainty that specific deflections are identified in the actual IMU signal. Furthermore, the output appears to be constrained to a range of values between -4.4 and 4.4. Nevertheless, the prediction of the NN is the only one that is able to almost predict the deflection of the IMU in the range of approximately 26,000 samples.
It is regrettable that none of the employed methodologies were able to generate models that could accurately predict the IMU signals. The parameterisation of the models appears to result in the occurrence of wrong local minima after a few training epochs, which precludes any improvement in the prediction. The parameterisation of the CNN and RNN produces predictions that are close to zero and strongly influenced by the input signal. The parameters of the neural network generate a prediction that approaches an upper and lower limit value, significantly inflate the input signal with a low-pass character. The occurrence of false local minima during the training of neural networks, as observed in the training of the suspension model, is a phenomenon that has been well documented in the literature.

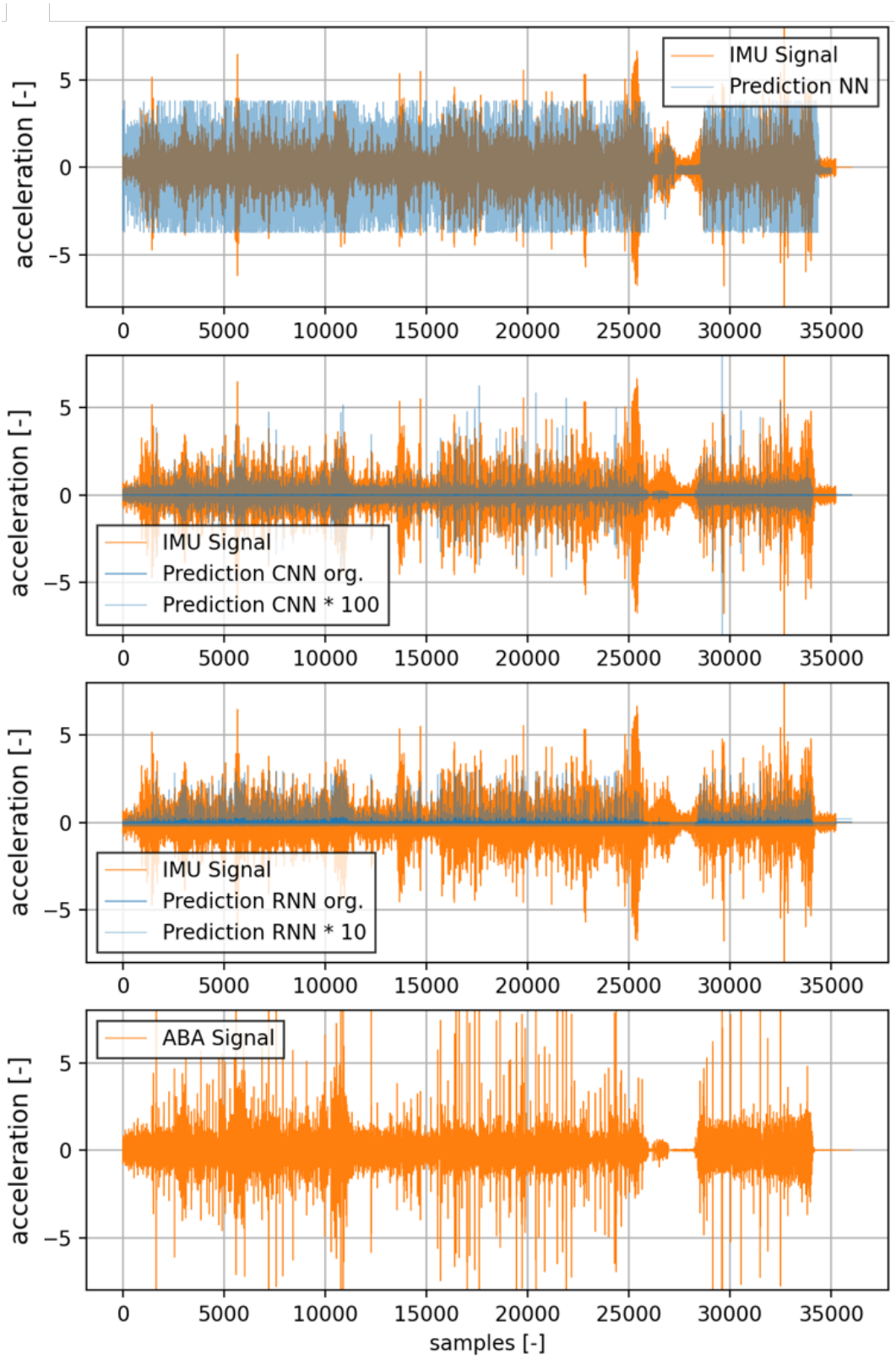

Figure 35: Plotted predictions of different neural networks in comparison to the scaled input and output signals

5.2 Spurious Local Minima

During the training of neural networks, suboptimal minima may arise, which indicates that the algorithm utilised in backpropagation is no longer capable of adapting the parameters. Consequently, the adaptation of the model is said to have reached a plateau. This is in contrast to the fact that neural networks serve as universal function approximators, capable of learning complex patterns in data and approximating any continuous function. The universal approximation theorem asserts that a neural network comprising a sufficient number of neurons and hidden layers is capable of approximating arbitrary continuous functions with arbitrary accuracy. Nevertheless, the occurrence of such spurious local minima can be demonstrated to follow directly from the theorem and the underlying mechanisms. This is particularly evident when non-affine activation functions are employed, such as in the case of the ReLU or tangent hyperbolic function, as was the case in this study [44].
Modelling based on neural networks is predicated on the assumption that non-convex functions can be optimised using local gradient descent methods during the model training phase, which is known as the back propagation. However, there is no guarantee that the optimisation algorithm will converge to a meaningful minimum, or indeed that it will converge at all [45]. In the case of the models analysed, the models converged to two different local minima, which did not result in a meaningful prediction of the model. The quality of the minimum achieved is often much worse than the global maximum and depends, among other things, on the initialisation of the model parameters and the utilized data set[46].
The data employed for the training were acceleration signals recorded by real sensors. In addition to the informative signal components, the raw signals also exhibited noise components. This was particularly evident in the IMU signals, such that the noise component was not entirely removed even by the filtering operation applied during the preprocessing stage. This phenomenon can be observed in Figure 35. Upon examination of the IMU signals, it becomes evident that residual noise persists in the signal at the conclusion of the ABA excitation period. The input of an accelerometer, particularly the noise component, can be expressed by a Gaussian distribution. This factus may have affected the model training. A study published in 2019 examines the occurrence of special local minima when Gaussian inputs are employed. These inputs adhere to the principle of least loss of symmetry with respect to the target weights, and thus may be far away from the global minimum [47].
The conclusion of training in a spurious local minimum results in an inadequate level of prediction quality and a lack of adaptation to the task. This phenomenon is commonly referred to as underfitting. The term is closely related to a fundamental problem in machine learning, namely the bias-variance trade-off.

5.3 Bias Variance Trade-Off

The bias-variance problem represents a significant challenge when training machine learning models. The issue can be described as a conflict between two opposing sources of error. Bias and Variance.
Bias may be defined as the inability of a model to capture important patterns in the data, due to its simplicity. A model exhibiting high bias demonstrates suboptimal performance on both the training data and new data. This phenomenon is referred to as underfitting. Conversely, the variance represents the error caused by overfitting to the training data. A model with high variance

exhibits a high degree of learning of the training data, including noise and outliers. Although the model performs well on the training data, it generalises poorly to new data. This phenomenon is known as overfitting. This can be attributed to an excess of functions or parameters in relation to the quantity of data, an extended training period, or a deficiency in regularisation.
The objective is to identify a model that strikes a balance between simplicity and complexity. A model that is overly simplistic or overly complex is to be avoided. Such a model is able to identify the fundamental patterns in the data without being overly influenced by random fluctuations.
The fact that the RMSE in relation to the test and validation data is similar for all models strongly suggests that the models were underfitting. The potential causes of this phenomenon include an insufficient number of parameters in the model, inadequate training time, or excessive regularisation. In this work, a regularisation technique was employed to interrupt training when model adaptation was no longer occurring. However, other regularisation techniques, such as dropout layers, were not utilised.
Upon examination of the number of parameters that can be trained, it becomes evident that this is a negligible fraction in comparison to the number of training data points, which comprises all the data points of the individual journey and exceeds 2 million.The models that were subjected to comparison in chapter 5.1 were found to possess the following number of trainable parameters an results from the model architecture:

- Conventional NN: 10401
- CNN: 103
- RNN: 26

It is evident that both models, whose predictions exhibit minimal amplitude, possess a limited number of trainable parameters. Chapters 4.2.2 and 4.3.2, in which the hyperparameters were optimised for the respective model architecture, demonstrated that simple models are more effective than complex models in terms of achieving a stable deviation quickly. Furthermore, more complex networks did not demonstrate any improvement in terms of prediction quality, even with extensive training runs. It can be observed that the number of trainable parameters is not directly proportional to the complexity of the network. In fact, more complex networks, such as those with wider and deeper layers, can only achieve a greater number of trainable parameters. This is in contrast to the assumption that a greater number of parameters would easily prevent underfitting.
Nevertheless, it is important to note that even the most complex models tested in the optimisation runs were not in the same order of magnitude as the number of data points in the training data set. The largest trained models reached a number of approximately 100,000 parameters, representing only approximately 5 % of the training data.
A pervasive conviction within the machine learning community is that smaller networks exhibit superior generalisation capabilities compared to complex ones. Additionally, it is widely believed that the number of parameters in a model should be smaller than the number of training data points [48]. Nevertheless, it appears that the lack of prediction quality is not directly attributable to the number of parameters, as more complex networks did not yield better results.
Since more complex models did not perform better, i.e. the models were not too simple, no extreme regularisation was applied and the training time was not too short, the cause of the underfitting could also be the poor quality of the database. The quality of the data set used to train the model is of equal importance to the number of parameters in determining the quality of the model. The following chapter will examine the extent to which modifications to the data

set could be beneficial for model quality.

5.4 Modification of the Database

All modelling approaches failed to accurately predict the IMU signals based on the ABA signals. The fact that the models have the same problem despite their different structure, in the case of the CNN and RNN models, could indicate that there is a common cause. And since all models were trained, validated and tested on the same datasets, it stands to reason that the modelling difficulty is related to the relationship between the ABA and IMU signals in the underlying datasets.
Chapter 3.2.4 discussed the quality of the data. Particular attention was paid to the synchronisation of the signals and the degree of correlation between the IMU and ABA signals on each trip. The individual runs were analysed in detail and compared using various metrics. They were also subjected to an visual analysis.
This showed that about 50 % of the trips had a correlation between the signals that was not continuous across the signal. Portions of the IMU signals do not appear to originate from the ABA signal that correlates with it in the data set. Figure 36 illustrates this with an example. It shows a trip where there is very little or no correlation between the IMU and ABA signals in comparison to a journey with strong correlation.

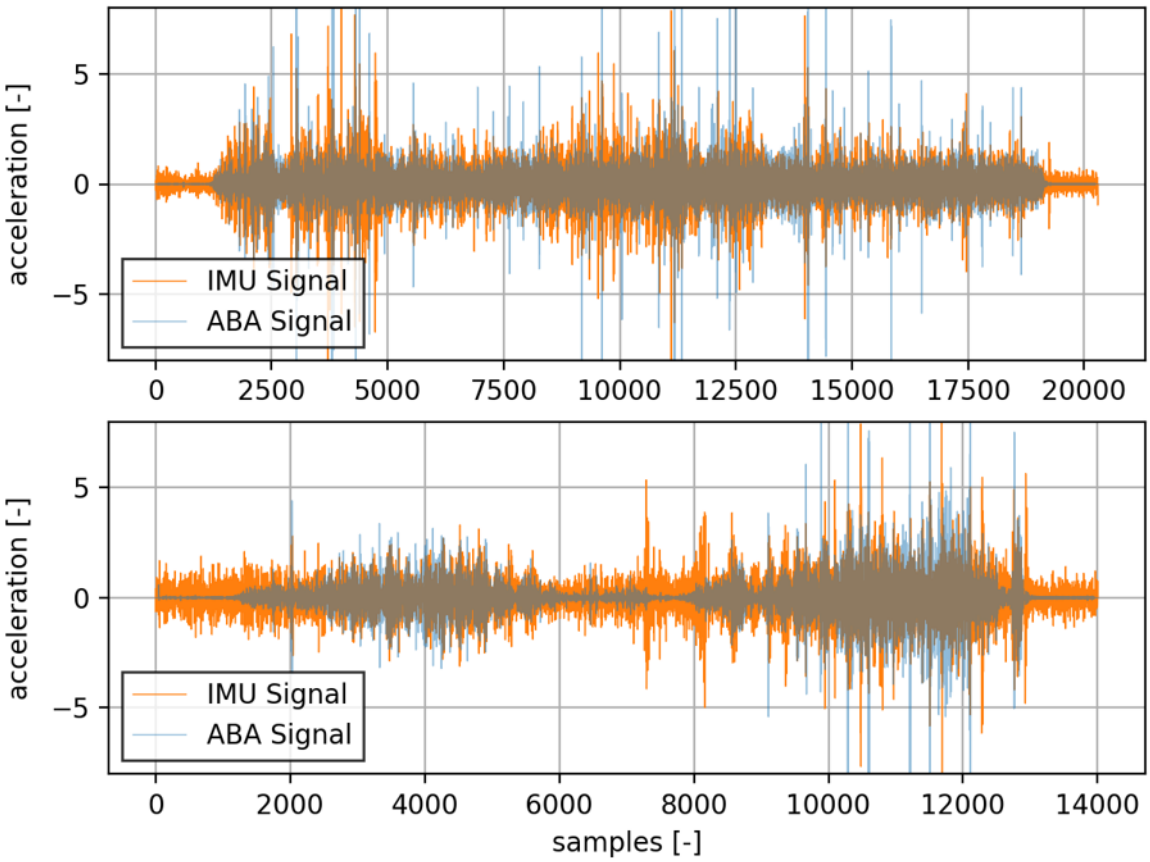

Figure 36: Scaled ABA and IMU signals of exemplary journeys with high correlation (upper plot) and low correlation (lower plot)

The training process from such a journey presents a number of challenges for neural networks, with back-propagation parameterisation being particularly affected. Such models are capable of recognising and learning complex behaviours and patterns in the data. However, this is problematic when there is no clear or discontinuous behaviour in the training data, in this case, no clear correlation between the IMU and ABA signal.
Figure 37 depicts two scatter plots that illustrate the correlation between ABA and IMU signals for two exemplary journeys. As previously mentioned, the correlations between IMU and ABA signals are not constant across the data set. The regression lines and the additionally depicted Pearson correlation factor r demonstrate that there is a positive correlation in the left journey and a negative correlation in the right journey. The discrepancy within the training data has a severely detrimental impact on the model training process, preventing the model from generalising and mapping the relationship between input and output data. Consequently, the backpropagation algorithm converges to a parameterisation that produces predictions that are close to zero.

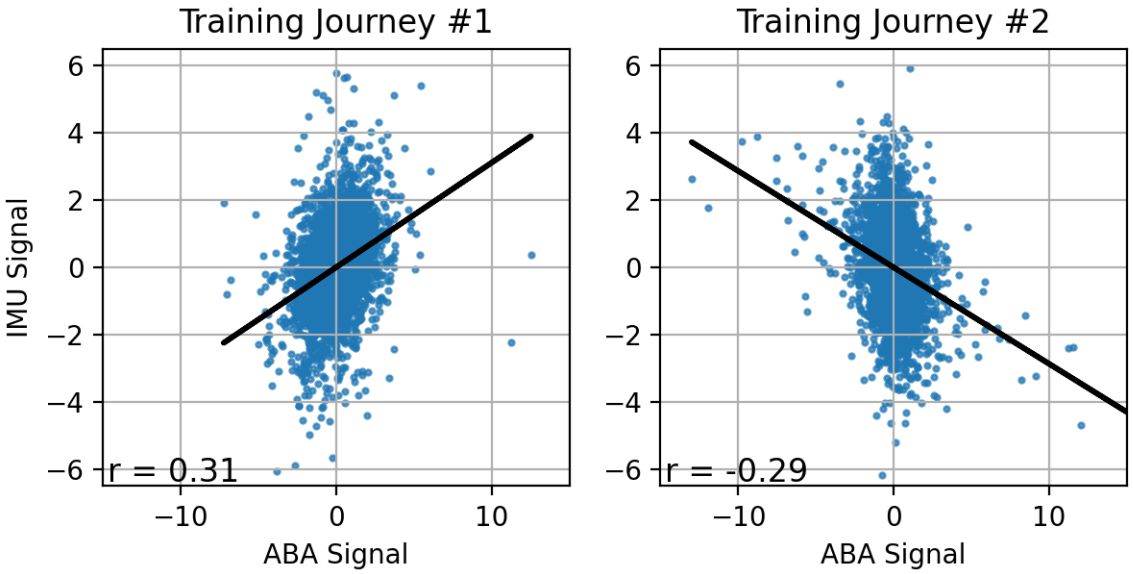

Figure 37: Scatter plot of the ABA and IMU Signal from two exemplary journeys of the training data

It is possible that modifying the training data set by deleting the data that does not demonstrate a strong correlation between the signals could enhance the prediction quality. However, if the model is only trained on data exhibiting this characteristic, the total amount of training data is significantly reduced. This would result in the utilisation of only approximately 25 % of the available recordings, thereby further reducing the already limited quantity of training data. Nevertheless, the approach is promising and has been subjected to further analysis.
The models presented at the outset of this chapter, one for each modelling approach, were subjected to further training, this time utilising only the data presented. A subset of the highly correlated data was employed to validate the model training. Nevertheless, the application and ultimate utilisation of the newly trained models is conducted once more on the same dataset, in order to guarantee comparability with the resulting RMSEs from the preceding work.
Table 6 presents the RMSEs resulting from the predictions of the three retrained models in rela-

tion to four distinct data sets: the modified training data set and the three known test, validation and training data sets.

	NN	CNN	RNN
RMSE Modified Train Data [-]	0.9917	0.5053	0.5051
RMSE Train Data [-]	1.0074	0.2979	0.3020
RMSE Validation Data [-]	1.0085	0.3161	0.3218
RMSE Test Data [-]	1.0198	0.4703	0.4859
Nr. of Epochs to stop [-]	18	89	15

Table 6: Overview of the resulting RMSE of predictions and scaled IMU Signals from different models when trained with modified train data set

The data indicates that training with the modified data set did not result in any significant improvements. The RMSE related to the individual data sets has not demonstrably improved in comparison to the previous state, when the models were trained with all journeys. The RMSEs remains comparable between the RNN model and the CNN model, while the conventional NN exhibits values in a higher value range across all data sets.

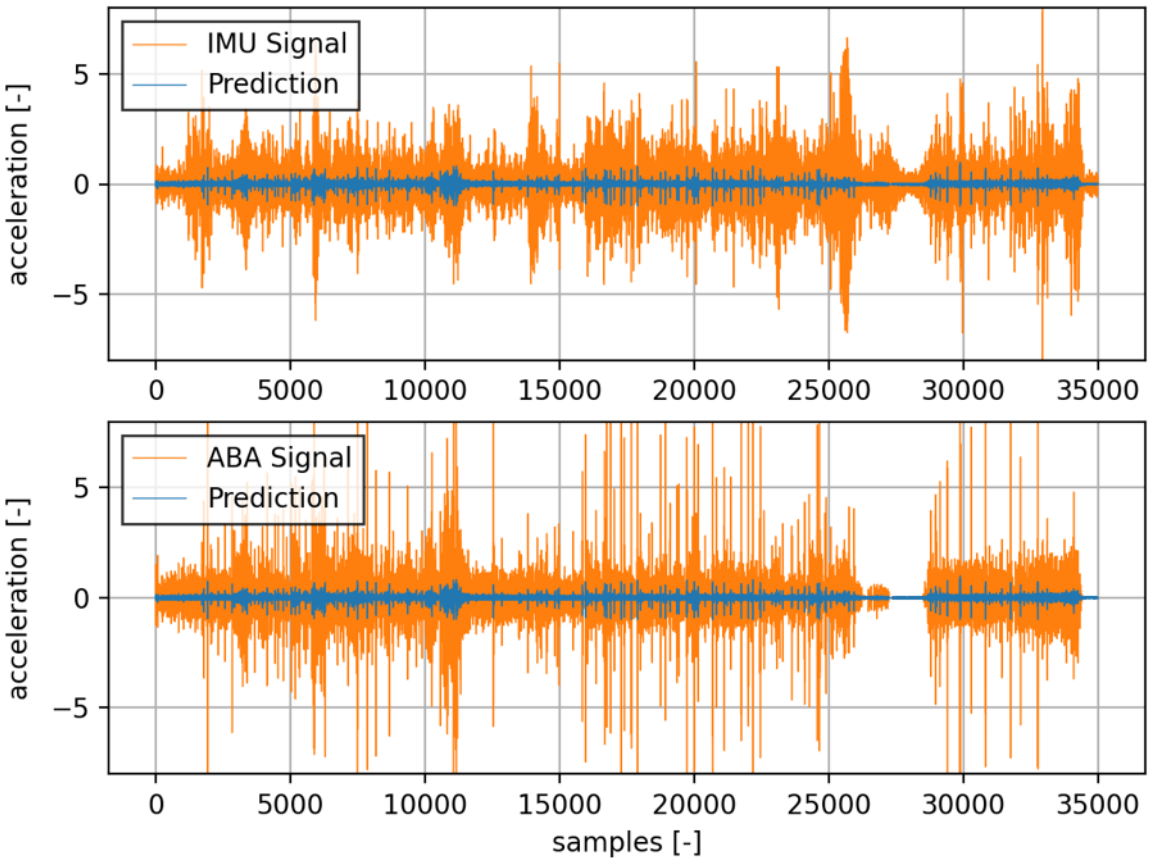

Figure 38: Prediction of conventional NN in comparison to scaled input and output signals. Model trained with only highly correlating IMU and ABA Signals

Two of the three models also demonstrate a relatively rapid stabilisation of the RMSE after less than 20 epochs. Although the CNN model requires a greater number of training epochs, 89 epochs are still a relatively small number of training runs.
It was hypothesised that the modified data set would facilitate the learning of the model's behaviour, resulting in improved prediction and a reduced RMSE. This hypothesis was tested on a NN and found to be true, as the RMSE is lower when applied to the modified training data than to the regular training data. Nevertheless, the enhancement for the NN is relatively modest, and the other two networks did not exhibit any improvement whatsoever, as they demonstrated a higher loss in the modified data set.
To gain further insight into the results of the model training with only highly correlated IMU and ABA signals, the prediction of the conventional neural network is presented graphically in Figure 38. This is an example journey from the validation data set for better comparability. The figure also shows the corresponding IMU and ABA signals.
It can be observed that the amplitude of the predicted signal is considerably smaller than that of the two real signals. This behaviour is reminiscent of the CNN and RNN models. Furthermore, the prediction appears to align with the input signal, and the characteristics of the IMU signal are not discernible.
In general, it can be stated that the measure was not a success. The observation that the value ranges in which the RMSEs occur remained unchanged by the data set modification indicates that the action taken did not produce the desired result.
Other potential modifications to the database that could enhance the modelling process include the utilisation of a single journey for model training. This offence could eliminate a potential source of error that could result from a variable transmission of the signal between the IMU and ABA over several sessions. It is important to note that the size of the training data would be significantly reduced, which could have a negative impact on the model due to its tendency towards poor generalisation.
An alternative approach would be to utilise both ABA sensors in the left and right axle bearings for prediction. This adaptation of the models would theoretically be straightforward to implement and would also result in a natural increase in the parameters. However, as the models are already highly simplified, this is not a significant issue. Nevertheless, it would be necessary to conduct a detailed analysis to ascertain whether the amount of information is significantly increased by using the second sensor and the model in order to better predict the transmission. The two proposed approaches are not addressed in this paper, but offer a potential avenue for further investigation.
In addition to further modifications to the data set, alternative modelling approaches can also be discussed as part of further investigations. This topic is further explored in the subsequent chapter.

5.5 Alternative Model Approach

Besides the employed sequence modeling methods, another technique, which combines data driven and physical modeling, archived great results when modeling nonlinear dynamical systems. A Physics Informed Neural Network (PINN) represent a novel approach to integrating neural networks with physical laws. The utilisation of partial knowledge of the dynamic system to be modelled serves to enhance the predictive precision of the model [49].
The fundamental concept underlying PINNs is the incorporation of the governing Partial Differential Equations (PDEs) of the physical system into the loss function during the training of

the neural network. This approach enables the network to not only fit the available data, but also to respect the underlying physical constraints described by the PDEs. The computation of the PDE residuals is typically performed using the automatic differentiation capabilities of deep learning frameworks such as Keras.
In order to implement a PINN for a specific problem, it is first necessary to define the desired network architecture. Subsequently, the PDE must be derived. The corresponding PDE to the system in question can be derived directly from a mechanical model. A commonly used and simple approach is the quarter-car model [19]. Section 2.1.3 presented this conventional approach and derived the representing quarter-car model equations for the specific use case of this work.
The PDEs governing the system dynamics are then coded into a custom loss function along with traditional data-fitting terms. During training, the network weights are optimised in order to minimise violations of the PDEs in addition to minimising errors on the available data.
By leveraging knowledge of the governing physics, PINNs can achieve high prediction accuracy even with limited data, while avoiding unphysical predictions that purely data-driven models may produce when extrapolating [50]. This innovative combination of machine learning and physical modelling has the potential to revolutionise the way complex nonlinear systems are modelled across a wide range of scientific and engineering domains.
Nevertheless, the construction of complex physical systems with PINNs is challenging when the governing partial differential equations or their parameterisation are unknown or inaccurate. This is a significant issue that can severely impact the performance and reliability of the PINN model. The key idea behind PINNs is to incorporate the known PDEs that describe the physical system into the neural network training process. However, if these partial differential equations (PDEs) are not fully understood or incorrectly parameterised, the PINN will essentially be trained on flawed physical constraints. By enforcing incorrect physical laws during training, the PINN may produce predictions that violate fundamental principles or conservation laws, rendering the model outputs unphysical or unrealistic.
In the case of the investigated dynamic system for the thesis, a two-stage damping system of a locomotive, the model parameters required for the full initialisation of the quarter vehicle model, such as vehicle mass and spring and damper data, were unknown. Consequently, no mechanical model could serve as the basis for a PINN that models the relationship between the measured accelerations in the IMU and the ABA with sufficient accuracy. The necessity for a profound comprehension of the physical system and the availability of known parameters in order to successfully implement a PINN represents a disadvantage of this type of modelling in comparison to the other approaches presented.

6 Summary

The objective of this study was to develop a machine learning model that predicts the transmission of vertical accelerations between the axle box of a train and its body. The theoretical framework underlying the modelling approach is to use the redundancy of the measurement data of individual sensors in a multi-sensor system to validate the measurement autonomously. For this purpose, a data set from a research project in the field of predictive maintenance of modern railway infrastructure systems was used.
The data set was divided into sessions and journeys and subjected to analysis and pre-processing for modelling purposes. A series of process steps were carried out, including resampling, filtering, scaling, synchronisation and the splitting of the data into training, validation and test data. The individual preprocessing steps and their results were presented and critically discussed in the thesis. Subsequently, three different modelling approaches were investigated: conventional NNs, CNNs and RNNs. A model architecture was proposed for each model type and implemented in Python using methods from the machine learning API Keras. This was followed by a parameter optimisation to enable the comparison of different parameter configurations. Subsequently, the accuracy of the models was evaluated using the test data.
It was not possible to develop a model that could generate reliable predictions based on this dataset. Regardless of the chosen model type, signs of severe underfitting occurred during the training of the neural networks. Suboptimal minima, due to which the back-propagation method was no longer able to further adapt the parameterisation, impaired the model training and led, among other things, to predictions that were constantly close to zero. This effect occurred with any parameter configuration and all three modell approaches, with the only differences being marginal differences in the RMSE of the prediction and the number of training epochs required to reach the suboptimal plateau. More complex models took longer than simpler ones. Moreover, the RMSEs between the convolutional neural network models and the recurrent neural network models were interestingly very similar, while the neural network models exhibited deviations of three orders of magnitude.
The reasons for the insufficient model quality may be attributed to the use of noisy and not continuously strongly correlated ABA and IMU signals, which were employed for model training. The incorporation of additional data for model training or the integration of multiple sensors in a multi-channel solution could prove advantageous. The models tested were also of moderate size, which raises the question of whether a superior result could be achieved with higher computing power and a larger amount of data. The investigation of a modelling approach involving partial differential equations also appears promising and could provide a foundation for further work.

Bibliography

[1] M. Binder, V. Mezhuyev and M. Tschandl, 'Predictive maintenance for railway domain: A systematic literature review,' *IEEE Engineering Management Review*, vol. 51, no. 2, pp. 120–140, 2023. DOI: 10.1109/EMR.2023.3262282.

[2] R. B. Randall, 'Introduction and background,' in *Vibration-based Condition Monitoring*. John Wiley Sons, Ltd, 2011, ch. 1, pp. 1–23, ISBN: 9780470977668. DOI: https://doi.org/10.1002/9780470977668.ch1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470977668.ch1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470977668.ch1.

[3] B. Baasch, J. C. Groos and M. H. Roth, 'Sensorgestützte anomaliedetektion zur zustandsbewertung der schiene mit regelzügen,' *ETR - Eisenbahntechnische Rundschau*, no. 12, Dezember 2018. [Online]. Available: https://elib.dlr.de/121905/.

[4] S. Muñoz, J. F. Aceituno, P. Urda and J. L. Escalona, 'Multibody model of railway vehicles with weakly coupled vertical and lateral dynamics,' *Mechanical Systems and Signal Processing*, vol. 115, pp. 570–592, 2019, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2018.06.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327018303534.

[5] N. Nadarajah, A. Shamdani, G. Hardie, W. K. Chiu and H. Widyastuti, 'Prediction of railway vehicles' dynamic behavior with machine learning algorithms,' *Electronic Journal of Structural Engineering*, vol. 18, no. 1, pp. 38–46, 2018, ISSN: 1443-9255. DOI: 10.56748/ejse.182271.

[6] X. Hao, J. Yang, F. Yang, X. Sun, Y. Hou and J. Wang, 'Track geometry estimation from vehicle–body acceleration for high-speed railway using deep learning technique,' *Vehicle System Dynamics*, vol. 61, no. 1, pp. 239–259, 2023, ISSN: 0042-3114. DOI: 10.1080/00423114.2022.2037669.

[7] S. Ma, L. Gao, X. Liu and J. Lin, 'Deep learning for track quality evaluation of high-speed railway based on vehicle-body vibration prediction,' *IEEE Access*, vol. 7, pp. 185 099–185 107, 2019. DOI: 10.1109/ACCESS.2019.2960537.

[8] H. Li, T. Wang and G. Wu, 'Dynamic response prediction of vehicle-bridge interaction system using feedforward neural network and deep long short-term memory network,' *Structures*, vol. 34, pp. 2415–2431, 2021, ISSN: 23520124. DOI: 10.1016/j.istruc.2021.09.008.

[9] F. Yang, X. Hao, H. Zhang, J. Jiang, Z. Fan and Z. Wei, 'Estimation of vehicle dynamic response from track irregularity using deep learning techniques,' *Shock and Vibration*, vol. 2022, pp. 1–9, 2022, ISSN: 1070-9622. DOI: 10.1155/2022/2136464.

[10] DLR. 'Havenzug – schäden am gleis effizient vorbeugen.' (), [Online]. Available: https://verkehrsforschung.dlr.de/de/projekte/havenzug-schaeden-am-gleis-effizient-vorbeugen (visited on 14/03/2024).

[11] P. S. Rao, A. K. Desai and C. H. Solanki, 'Application of quarter car model for assessment of attenuation characteristics of soil at low strain,' *Transportation Infrastructure Geotechnology*, vol. 8, no. 3, pp. 329–348, 2021, ISSN: 2196-7202. DOI: 10.1007/s40515-020-00139-2.

[12] I. M. Ahmed, M. Y. Hazlina and M. M. Rashid, 'Modeling a small-scale test rig of quarter car railway vehicle suspension system,' *Journal of robotics and mechatronics*, vol. 2, no. 4, pp. 149–153, 2015. DOI: 10.21535/IJRM.V2I4.888.

[13] S. J. Elliott, M. Ghandchi Tehrani and R. S. Langley, 'Nonlinear damping and quasi-linear modelling,' *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 373, no. 2051, 2015. DOI: 10.1098/rsta.2014.0402.

[14] I. Paglia, L. Rapino, F. Ripamonti and R. Corradi, 'A methodology for including suspension dynamics in a simple context of rail vehicle simulations,' in *15th World Congress on Computational Mechanics (WCCM-XV) and 8th Asian Pacific Congress on Computational Mechanics (APCOM-VIII)*, CIMNE, July 31 to August 5, 2022. DOI: 10.23967/wccm-apcom.2022.005.

[15] I. La Paglia, L. Rapino, F. Ripamonti and R. Corradi, 'Modelling and experimental characterization of secondary suspension elements for rail vehicle ride comfort simulation,' *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 238, no. 1, pp. 38–47, 2024, ISSN: 0954-4097. DOI: 10.1177/09544097231178858.

[16] S. Iwnicki, M. Spiryagin, C. Cole and T. McSweeney, *Handbook of Railway Vehicle Dynamics, Second Edition*. Milton, UNITED KINGDOM: Taylor & Francis Group, 2019, ISBN: 9780429890635. [Online]. Available: http://ebookcentral.proquest.com/lib/dlr-ebooks/detail.action?docID=5981899.

[17] S. Bruni, J. Vinolas, M. Berg, O. Polach and S. Stichel, 'Modelling of suspension components in a rail vehicle dynamics context,' *Vehicle System Dynamics*, vol. 49, no. 7, pp. 1021–1072, 2011, ISSN: 0042-3114. DOI: 10.1080/00423114.2011.586430.

[18] M. Dumitriu, I. I. Apostol and D. I. Stănică, 'Influence of the suspension model in the simulation of the vertical vibration behavior of the railway vehicle car body,' *Vibration*, vol. 6, no. 3, pp. 512–535, 2023. DOI: 10.3390/vibration6030032.

[19] L. Rónai, 'Investigation of the vibrational behavior of a quarter-car model,' in *Vehicle and Automotive Engineering 4*, ser. Lecture Notes in Mechanical Engineering, K. Jármai and Á. Cservenák, Eds., Cham: Springer International Publishing, 2023, pp. 824–834, ISBN: 978-3-031-15210-8. DOI: 10.1007/978-3-031-15211-5{\textunderscore}68.

[20] M. Boldis and K. Zakova, 'Web presentation of quarter car model,' in *2019 5th Experiment International Conference (exp.at'19)*, IEEE, 2019, pp. 167–171, ISBN: 978-1-7281-3637-0. DOI: 10.1109/EXPAT.2019.8876578.

[21] J. Yang, J. Wang and Y. Zhao, 'Simulation of nonlinear characteristics of vertical vibration of railway freight wagon varying with train speed,' *Electronic Research Archive*, vol. 30, no. 12, pp. 4382–4400, 2022, ISSN: 2688-1594. DOI: 10.3934/era.2022222.

[22] R. M. Goodall, H. Sira-Ramírez and A. Matamoros-Sánchez, 'Flatness based control of a suspension system: A gpi observer approach,' *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 103–11 108, 2011, ISSN: 14746670. DOI: 10.3182/20110828-6-IT-1002.03102.

[23] C. El Morr, M. Jammal, H. Ali-Hassan and W. El-Hallak, 'Data preprocessing,' in *Machine Learning for Practical Decision Making: A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics*, Cham: Springer International Publishing, 2022, pp. 117–163, ISBN: 978-3-031-16990-8. DOI: 10.1007/978-3-031-16990-8{\textunderscore}4.

[24] P. Bourke, 'Cross correlation,' *Cross Correlation", Auto Correlation—2D Pattern Identification*, vol. 596, 1996.

[25] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[26] L. Stanković and D. Mandic, 'Convolutional neural networks demystified: A matched filtering perspective-based tutorial,' *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 6, pp. 3614–3628, 2023, ISSN: 2168-2216. DOI: 10.1109/tsmc.2022.3228597.

[27] H. Zumbahlen, 'Analog filters,' in *Linear Circuit Design Handbook*, Elsevier, 2008, pp. 581–679, ISBN: 9780750687034. DOI: 10.1016/B978-0-7506-8703-4.00008-0.

[28] D. Marinov. 'Part 5: Polyphase fir filters.' (2023), [Online]. Available: https://vhdlwhiz.com/part-5-polyphase-fir-filters/ (visited on 14/03/2024).

[29] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, 'Convolutional neural networks based fire detection in surveillance videos,' *IEEE Access*, vol. 6, pp. 18 174–18 183, 2018. DOI: 10.1109/ACCESS.2018.2812835.

[30] M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, 'Disease prediction by machine learning over big data from healthcare communities,' *IEEE Access*, vol. 5, pp. 8869–8879, 2017. DOI: 10.1109/ACCESS.2017.2694446.

[31] S. Zhang, Z. Sun, J. Long, C. Li and Y. Bai, 'Dynamic condition monitoring for 3d printers by using error fusion of multiple sparse auto-encoders,' *Computers in Industry*, vol. 105, pp. 164–176, Feb. 2019. DOI: 10.1016/j.compind.2018.12.004.

[32] H. Shao, J. Hongkai, L. Xingqiu and W. Shuaipeng, 'Intelligent fault diagnosis of rolling bearing using deep wavelet auto-encoder with extreme learning machine,' *Knowledge-Based Systems*, vol. 140, Oct. 2017. DOI: 10.1016/j.knosys.2017.10.024.

[33] S. S. Haykin, *Neural networks and learning machines*, 3. ed. New York: Pearson, 2009, ISBN: 0-13-147139-2.

[34] P. Purwono, A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky and Q. M. u. Haq, 'Understanding of convolutional neural network (cnn): A review,' *International Journal of Robotics and Control Systems*, vol. 2, no. 4, pp. 739–748, 2022. DOI: 10.31763/ijrcs.v2i4.888.

[35] S. P and R. R, 'A review of convolutional neural networks, its variants and applications,' in *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*, IEEE, 2023, pp. 31–36, ISBN: 979-8-3503-3583-5. DOI: 10.1109/ICISCoIS56541.2023.10100412.

[36] A. P. Wibawa, A. B. P. Utama, H. Elmunsyah, U. Pujianto, F. A. Dwiyanto and L. Hernandez, 'Time-series analysis with smoothed convolutional neural network,' *Journal of big data*, vol. 9, no. 1, p. 44, 2022, ISSN: 2196-1115. DOI: 10.1186/s40537-022-00599-y.

[37] K. Aurangzeb, M. Alhussein, K. Javaid and S. I. Haider, 'A pyramid-cnn based deep learning model for power load forecasting of similar-profile energy customers based on clustering,' *IEEE Access*, vol. 9, pp. 14 992–15 003, 2021. DOI: 10.1109/ACCESS.2021.3053069.

[38] D. Jurafsky and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. Chapter 9: Sequence Processing with Recurrent Networks*. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009, ch. 9, ISBN: 9780131873216 0131873210. [Online]. Available: http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y.

[39] Q. El Maazouzi, A. Retbi and S. Bennani, 'Automatisation hyperparameters tuning process for times series forecasting: Application to passenger's flow prediction on a railway network,' *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-4/W3-2022, pp. 53–60, 2022. DOI: 10.5194/isprs-archives-XLVIII-4-W3-2022-53-2022.

[40] Huile Li, Tianyu Wang and Gang Wu, 'Dynamic response prediction of vehicle-bridge interaction system using feedforward neural network and deep long short-term memory network,' *Structures*, vol. 34, pp. 2415–2431, 2021, ISSN: 23520124. DOI: 10.1016/j.istruc.2021.09.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352012421008559.

[41] *Precision triaxial industrial icp ® accelerometer*, 629A61, Rev. B, PCB electronics, Mar. 1999.

[42] *Mti user manual*, MT0605P, Rev. 2020A, Xsens Technologies, Feb. 2020.

[43] J. Oprel, 'Railway track monitoring from vehicles in daily operations,' Projektarbeit, 2023. [Online]. Available: https://elib.dlr.de/201910/.

[44] C. Christof and J. Kowalczyk, *On the omnipresence of spurious local minima in certain neural network training problems*. DOI: 10.1007/s00365-023-09658-w. [Online]. Available: http://arxiv.org/pdf/2202.12262v2.

[45] G. Swirszcz, W. M. Czarnecki and R. Pascanu, *Local minima in training of neural networks*. [Online]. Available: http://arxiv.org/pdf/1611.06310v2.

[46] G. Elidan, M. Ninio, N. Friedman and D. Schuurmans, 'Data perturbation for escaping local maxima in learning,' *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2003.

[47] Y. Arjevani and M. Field, *On the principle of least symmetry breaking in shallow relu models*. [Online]. Available: http://arxiv.org/pdf/1912.11939v3.

[48] S. Lawrence, A. C. Tsoi and C. L. Giles, 'Local minima and generalization,' in *Proceedings of International Conference on Neural Networks (ICNN'96)*, IEEE, 1996, pp. 371–376, ISBN: 0-7803-3210-5. DOI: 10.1109/ICNN.1996.548920.

[49] H. Robinson, S. Pawar, A. Rasheed and O. San, *Physics guided neural networks for modelling of non-linear dynamics*, 2022. DOI: 10.48550/arXiv.2205.06858.

[50] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli, 'Scientific machine learning through physics–informed neural networks: Where we are and what's next,' *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022, ISSN: 1573-7691. DOI: 10.1007/s10915-022-01939-z.