tocbibind

tocbibind

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Mecânica

LEONARDO TADEU LIMA FRANCO

Título do seu Trabalho Acadêmico: dissertação de mestrado ou tese de doutorado

Campinas

2025

Leonardo Tadeu Lima Franco

Título do seu Trabalho Acadêmico: dissertação de mestrado ou tese de doutorado

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Mecânica.

Supervisor: Auteliano Antunes dos Santos

Este exemplar corresponde à versão final da Tese defendida pelo aluno Leonardo Tadeu Lima Franco e orientada pelo Prof. Dr. Auteliano Antunes dos Santos.

Campinas

2025

Acknowledgements

Inserir os agradecimentos, sem esquecer dos órgãos de fomento!

Resumo

Segundo a **??**, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Keywords: latex. abntex. text editoration.

List of Figures

List of Tables

Contents

1 Introduction

Este documento e seu código-fonte são exemplos de referência de uso da classe abntex2 e do pacote abntex2cite. O documento exemplifica a elaboração de trabalho acadêmico (teses e dissertações) produzido conforme a **Informação CCPG/001/2015** (que trata das *Normas para impressão de teses/dissertações* da UNICAMP). Encorajamos o leitor a consultar a Informação CCPG/001/2015 (**??**) antes de iniciar as alterações neste documento e seu código-fonte.

A elaboração deste modelo teve como base uma customização do "Modelo Canônico de Trabalho Acadêmico com abnTEX2" (**??**) para que as normas presentes na Informação CCPG/001/2015 fossem respeitadas. O modelo original produzido pela equipe abnTEX2 cumpre as seguintes normas ABNT:

1. **ABNT NBR 14724:2011**: Informação e documentação - Trabalhos acadêmicos - Apresentação;
2. **ABNT NBR 10520:2002**: Informação e documentação - Citações;
3. **ABNT NBR 6034:2004**: Informação e documentação - Índice - Apresentação;
4. **ABNT NBR 6028:2003**: Informação e documentação - Resumo - Apresentação;
5. **ABNT NBR 6027:2012**: Informação e documentação - Sumário - Apresentação;
6. **ABNT NBR 6024:2012**: Informação e documentação - Numeração progressiva das seções de um documento - Apresentação
7. **ABNT NBR 6023:2002**: Informação e documentação - Referência - Elaboração.

Este documento deve ser utilizado como complemento dos manuais do abnTEX2 (**??????**) e da classe memoir (**??**).

A leitura do teor desde documento (tanto o PDF quando os arquivos que compõem seu código-fonte), bem como do arquivo LEIAME.txt é altamente recomendada para melhor entendimento da dinâmica de funcionamento da classe abntex2 e do pacote abntex2cite. Seus principais comandos e usos estão exemplificados no decorrer do texto, bem como outras informações relevantes para a escrita de seu trabalho acadêmico.

USAR https://www.tandfonline.com/doi/full/10.1080/00423114.2025.2483972 para pegar algumas informações interessantes.

2 Literature Review

2.1 Time Series Anomaly Detection

In recent years, rapid technological advancements have led to a significant increase in the volume of data collected and generated from diverse sources such as sensors, databases, and files (ATTOH-OKINE, 2017). This large volume of data is now commonly referred to as Big Data. A substantial portion of Big Data consists of data points collected in sequential time steps, also known as time series. Analyzing time series data presents a complex set of challenges (TSAI et al., 2015).

One of the most studied problems in this area is anomaly detection. It consists of identifying outliers or abnormal patterns within data and has a variety of applications, from financial systems and medical diagnosis to urban management and fault detection (BLáZQUEZ-GARCíA et al., 2021; DARBAN et al., 2024; SAMARIYA; THAKKAR, 2021).

The following sections will dive deeper into what the concept of a time series is, what the different natures of anomalies in data are, and some of the models used to detect these anomalies.

2.1.1 Time Series

A time series (TS) can be defined as a set of points indexed sequentially over time and is often divided into univariate and multivariate. A univariate TS is a set of real values from a single variable that changes over time $\mathbf{X} = \{x_t\}$ for $t \in T$, where T is the total time. On the other hand, a multivariate TS is a set of real values from multiple variables that change over time, i.e., $\mathbf{X} = \{\mathbf{x}_t\}$ for $t \in T$, where $\mathbf{x}_t$ is a k-dimensional vector defined by $\mathbf{x}_t = (x_{1t}, ..., x_{kt})$ and each x_{it} is a univariate TS.

A subsequence of length $n \leqslant T$ can be defined as $\mathbf{S} = \{\mathbf{x}_{p:p+n}\}$ from the time series $\mathbf{X}$, where $p \in T$ and $p \leqslant T - n + 1$ (DARBAN et al., 2024; BLáZQUEZ-GARCíA et al., 2021).

Time series data exhibit dependencies that vary based on their type. In univariate time series, temporal dependency occurs when the value at time t, x_t, is influenced by its preceding values, $x_{p:t}$. In multivariate time series, in addition to temporal dependencies, there are spatial dependencies, representing the correlations between different variables within the same time step (DARBAN et al., 2024).

2.1.2 Time Series Anomalies

According to (GRUBBS, 1969) an anomaly can be defined as an outlier that deviates greatly from the general distribution of data. These outliers can be a single observation or a subsequence of the whole time series. Building on this, (BLáZQUEZ-GARCíA et al., 2021) divides anomalies into three main categories:

1. *Point Anomalies*: A point outlier can be defined as a single observation that deviates from the global behavior (global anomaly) or from the behavior of it's neighborhood (local anomaly). This type of anomaly can be identified using equation 2.1

$$|x_t - \hat{x}_t| > t \tag{2.1}$$

   where $\hat{x}_t$ is the expected result, x is the original data point and t is the threshold. If the difference is greater than the threshold t, then x_t is identified as an anomaly. Figure 1 shows an example of this anomaly, where O1 is a local outlier and O2 is a global anomaly.

2. *Subsequence Anomaly*: A subsequence outlier refers to a consecutive portion of points from the time series whose joint behavior is abnormal. It is important to note that each of these data points alone doesn't need to a point anomaly, but together they form an anomaly. Similarly to the point outliers, the subsequence outliers can be global or local. These anomalies can be identified by the equation (DARBAN et al., 2024):

$$diss\left(C, \hat{C}\right) > t \tag{2.2}$$

   where C is the the actual cycle or shape of the subsequence, $\hat{C}$ is the expected output, t is the threshold and $diss$ is a function that computes the dissimilarity between the two subsequences, such as the cosine similarity. Figure 2 shows an example of this, where O1 is a local anomaly and O2 is a global anomaly.

3. *Time Series Anomaly*: A time series anomaly is a entire timeseries that presents unusual behavior. Note that, this kind of anomaly can only be detected in multivariate time series. Figure 3 shows an example of this problem, where variable 4 behavior differs greatly from the other 3 variables.

(DARBAN et al., 2024) highlights another anomaly that can only occur in multivariate time series: the intermetric anomaly. An intermetric anomaly can be defined as an anomalous behavior in the correlation between two variables.

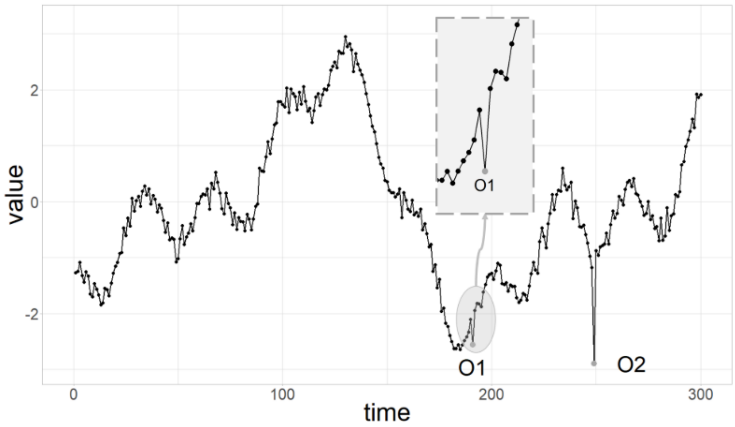


Figure 1 – Example of point anomalies in time series data, highlighting local (O1) and global outliers (O2). (BLáZQUEZ-GARCíA et al., 2021)

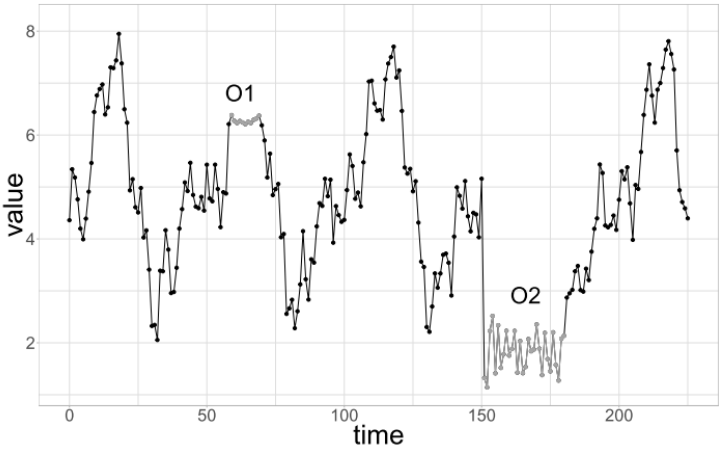


Figure 2 – Example of subsequence anomalies, showing local (O1) and global (O2) deviations in time series data. (BLáZQUEZ-GARCíA et al., 2021)

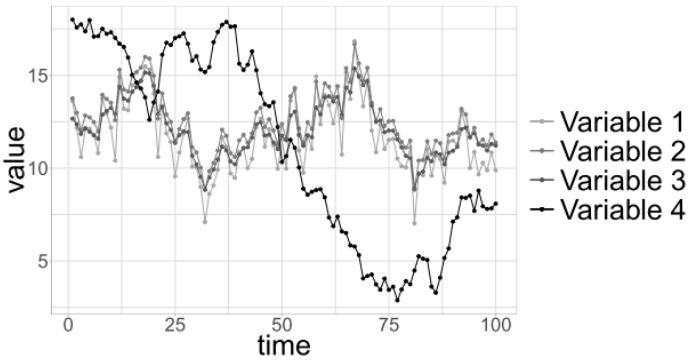


Figure 3 – Example of a time series anomaly in a multivariate dataset, where variable 4 significantly deviates from others. (BLáZQUEZ-GARCíA et al., 2021)

2.2 Time Series Anomaly Detection Models

Since the 1960s, many models for anomaly detection have been proposed, ranging from statistical models to recent deep learning models. The common main idea behind all of those is to identify low probability, i.e. low density, regions, classifying points in those areas as anomalies (SAMARIYA; THAKKAR, 2021).

Statistical models, for example, use statistical tests, like the χ^2 test, to identify abnormal points. Clustering-based models use cluster analysis to label anomalies, using, for example, the distance from the centroid of normal points or the number of points in a cluster. Distance-based models use the distance between the current time window and its neighborhood to classify anomalies. Density-based approaches use the density of the data points to find anomalies (DARBAN et al., 2024; SAMARIYA; THAKKAR, 2021). These methods described above are called traditional methods.

In recent years, with the increase of computing power, deep learning models have become increasingly prominent in anomaly detection. Unlike the traditional methods, the main advantage of deep learning models is that they are capable of identifying complex temporal and spatial correlations between variables, especially in larger multidimensional datasets, without the need for a labeled dataset (DARBAN et al., 2024; CHOI et al., 2021). (DARBAN et al., 2024) divides deep learning models into 3 main categories: forecasting-based models, reconstruction-based models, and representation-based models.

Forecasting-based models try to predict a future subsequence of the time series using historical data. These models learn the patterns and trends within data to anticipate what is expected to occur. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) are examples of this category. Anomalous values are detected if they deviate greatly from what the model predicts.

Reconstruction-based models attempt to recreate the input timeseries based on the values of past points or windows. These models use architectures like autoencoders (AE) and Generative Adversarial Networks (GANs). Anomalies are identified using the reconstruction error between the predicted series and the original series. The advantage of reconstruction-based models over forecasting ones is that they adapt better to rapid changes in a timeseries, which may cause it to become unpredictable. This happens because reconstruction models have access to the current timeseries data, which is not available in forecasting models (DARBAN et al., 2024). This better precision comes with a delay in the detection time of an anomaly, but, in general, these models are preferred over forecasting models.

Representation-based models focus on learning meaningful representations of the data that can be used in downstream tasks, like anomaly detection. These models use advanced architectures like transformers and self-supervised learning to extract high-level

features from the data. The learned representations are often used in conjunction with simpler models (e.g., clustering or density-based methods) to identify anomalies. The downside of these models is that they are computationally expensive to train, often needing a large amount of data.

An important point to highlight is that the majority of the models described above are unsupervised models. An unsupervised model is a model that can learn patterns from data without the need for a labeled dataset. This flexibility is desired because of the inherently unlabeled nature of historical data and the unpredictability of anomalies, which makes it harder to define an anomaly, especially if anomalies are rare. In this work, the focus will be on unsupervised reconstruction models.

Another important point to note is that, as will be detailed in Sections 2.10 and 3, the model proposed in this thesis is not designed specifically for anomaly detection. Instead, its primary objective is to learn how to reconstruct track irregularities from acceleration data. Despite that, the main idea that motivates the model's development is closely related to anomaly detection. For this reason, a review of anomaly detection methods was included.

2.3 Basics of Deep Learning Models

Before diving deeper into anomaly detection models, the fundamental concepts of deep learning will be presented first. This preliminary discussion will provide the necessary context and theoretical basis for the subsequent exploration of anomaly detection methodologies.

2.3.1 Neural Networks

Neural networks are the most basic structure in deep learning. It is inspired by the way the human brain works (AGATONOVIC-KUSTRIN; BERESFORD, 2000; ALZUBAIDI et al., 2021). These networks are composed of thousands, or even millions, of interconnected units called perceptrons, which mimic the behavior of biological neurons. In a neural network, these perceptrons are arranged into multiple layers, as shown in Figure 4(a). In this illustration, each circle represents a single perceptron.

The neural network is divided into three main components:

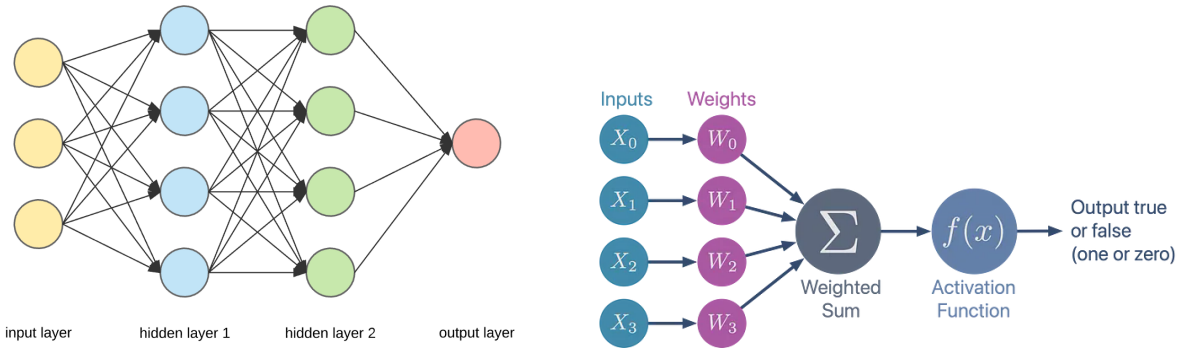
- The input layer: this layers receives raw data that will be precessed;
- The hidden layers: positioned between the input and output layers, these intermediate layers perform feature extraction;
- The output layer: this layer provides predictions or classifications

Each perceptron in a neural network processes input data linearly using the dot product. Given, them, an input array $\mathbf{x}$, the perceptron computes the output by multiplying $\mathbf{x}$ with an array of weights $\mathbf{w}$ and adding a bias b, producing a scalar output o (GURNEY, 1997):

$$o = \sum_i w_i x_i + b \tag{2.3}$$

The weights and biases are learned and iteratively updated during the training process such that they minimize the errors.

However, in its basic form, a perceptron can only solve linearly separable problems. To address this limitation, modern perceptrons include an additional element: the activation function. An activation function introduces non-linearity into the model, enabling it to solve complex, non-linear problems (ROTH, 2016).For reasons that will be explained later, it is common to choose differentiable activation functions. Common activation functions include the Rectified Linear Unit (ReLU) and the hyperbolic tangent (tanh). The general structure of a perceptron, including the activation function, is illustrated in Figure 4(b).



((a)) ((b))

Figure 4 – Structure of a neural network: (***a***) fully connected perceptrons arranged in layers, and (***b***) perceptron with activation function ((***a***) (P., 2025); (***b***) (WEAVER, 2024)).

As mentioned earlier, the weights and biases of the perceptrons are adjusted and optimized during the training phase of a neural network. This optimization process involves a method called backpropagation, which is essential for minimizing the error in predictions. Backpropagation, discussed in detail in Section 2.3.2, leverages the gradient descent algorithm to iteratively update the network's parameters, ensuring improved accuracy over successive training iterations.

2.3.2 Backpropagation

During the training phase of a neural network, its weights and biases are systematically optimized to minimize the error. This iterative process comprises two key

subprocesses: a forward pass followed by a backward pass. In a forward pass, the inputs are passed through the network and the outputs are calculated. Then, given an expected output, an error is computed using a function called the loss function. The backward pass, then, uses this error to update the weights and biases of the neural network, making it better at making right predictions. This process continues until the error is small enough.

The process of propagating the error back through the network during the backward pass is referred to as backpropagation. It can be broken down into four primary steps (CILIMKOVIC, 2015):

1. Feed foward the inputs;
2. Backpropagation to the output layer;
3. Backpropagation to the hidden layers;
4. Weights updates.

The first step is just feeding forward the inputs $\mathbf{x}$ through the network and obtaining the outputs $\mathbf{o}$. Then, a loss is computed using the loss function $\mathbf{L}$. There are a variety of loss functions, an example being the Mean Squared Error (MSE):

$$\mathbf{L} = \frac{1}{N}\sum_{i}(y_i - o_i)^2 \tag{2.4}$$

where y_i is the desired output and o_i is the network output. To backpropagate the error, the algorithm computes gradients of the loss function with respect to the parameters of the neural network. This is done using the chain rule twice (ROTH, 2016):

$$\frac{\partial \mathbf{L}}{\partial w_{ij}} = \frac{\partial \mathbf{L}}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \tag{2.5}$$

where w_{ij} is the i-th weight of the j-th perceptron, net_j is the result of the dot product before applying the activation function, i.e.,

$$net_j = \sum_{i} w_{ij}x_j \tag{2.6}$$

with x_j being the input, and o_j the perceptron output, i.e.,

$$o_j = \varphi(net_j) \tag{2.7}$$

where φ is the activation function of the perceptron. For this reason, it is necessary that the activation function is a differentiable function.

To update the weights, the algorithm computes, then, the incremental Δw_{ij} using:

$$\Delta w_{ij} = -\eta\delta_j x_{ij} \tag{2.8}$$

where η is a small positive number known as the learning rate, $\delta_j = \dfrac{\partial \mathbf{L}}{\partial net_j}$ and x_{ij} are the inputs of the j-th perceptron, and, thus, update the weights as:

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \tag{2.9}$$

For a full demonstration of these formulas, check Appendix A.

Despite its effectiveness, the classical backpropagation algorithm has some problems associated with its convergence. For example, if the learning rate is too small, the convergence is very slow and can converge to a local minimum; on the other hand, if it is too large, the algorithm can diverge (RUDER, 2016). Several modifications to the original algorithm have been proposed, like in (QIAN, 1999), (TIELEMAN, 2012) and (KINGMA; BA, 2017), but the contents of these works are outside the scope of this thesis.

2.3.3 Convolutional Neural Networks

Convolutional neural networks (CNN) achieved great popularity in modern machine learning, gaining large popularity with the (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) paper. In this work, the authors proposed a deep CNN that won the ImageNet LSVRC-2010 contest, outperforming second place by a large margin.

A CNN, similarly to the neural network, was inspired by the biological neurons, but, different from the conventional neural network, where all the neurons are connected and interact with all neurons in adjacent layers, the CNN employs a sparse architecture, i.e., the neurons are connected only to a local region of the input. This sparsity reduces the amount of training parameters, which reduces the computational cost and speeds up the training process (ALZUBAIDI et al., 2021; LI et al., 2022).

CNNs are widely used in computer vision to process and analyze 2D images and are used to solve a variety of image-related problems, like classification and object detection. While this section focuses on explaining CNN concepts with the assumption of a 2D input, it is important to note that the principles discussed are equally applicable to 1D inputs, like in a timeseries.

The basic block of a CNN is the convolution operation. A convolution can be defined as the dot product between an input x of shape (m, m, r), where m is the height, which is the same as the width, and r is the depth, also called the channel number, and k convolutional filters, called kernels. A kernel is a grid of weights, with shape (n, n, q), where $n < m$ and $q \leqslant r$, that convolves with the input, producing, each, an output of size $(m - n - 1)$, called the feature maps (ALZUBAIDI et al., 2021). These kernels compute the pass value similarly to the

$$h^k = f\left(W^k * x + b_k\right) \tag{2.10}$$

where W^k is the vector of weights from the kernel, f is an activation function, and b^k is the bias. Figure 5 shows an example of this operation between a 2D binary matrix and a 3×3 kernel.

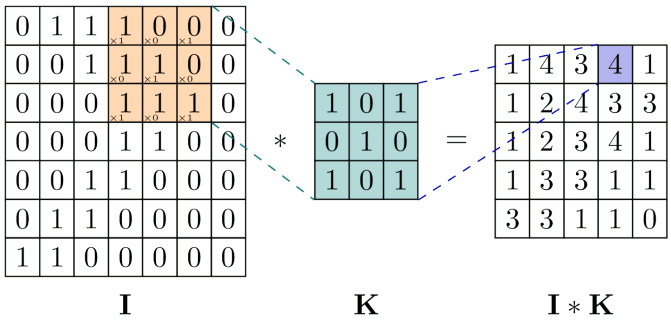


Figure 5 – Illustration of the convolution operation in a convolutional neural network. (RIEBESELL, 2022)

In addition to the convolution operation, there are two other important parameters in convolutional neural networks: the padding and the stride. Their main function is to control the output size of the convolution, helping to maintain the spatial characteristics of the input.

The padding is the process of adding additional rows and columns around the border of the input matrix, typically filled with zeros, a process known as zero padding. Without the padding, a convolution with a kernel of size $n \times n$ reduces the output size by $n-1$ in each spatial dimension. This would cause a rapid decrease in the size of the features, that would force the use of small kernels (GOODFELLOW; BENGIO; COURVILLE, 2018).

Stride, on the other hand, controls how the kernel moves across the input matrix, by skipping some of the elements from it. In a stride 1 convolution, the kernel shifts one unit at a time, covering all the positions. In a stride 2 convolution, the kernel would shift 2 elements each time, which causes the kernel to convolve with every other element. In general, a stride of size s means that the kernel will only convolve every s elements. This is done primarily to reduce computational cost, at the expense of extracting the features less finely (GOODFELLOW; BENGIO; COURVILLE, 2018).

Now with these two additional operations, the output size can be computed using the formula:

$$O = \left\lfloor \frac{I + 2 \cdot p - k}{s} \right\rfloor + 1 \tag{2.11}$$

where O is the output size, I is the input size, p is the padding, k is the kernel size, and s is the stride.

Another important feature that is widely used in CNNs is the pooling layer. Pooling layers are used to further reduce the spatial dimension of the feature map by

replacing the output value of the convolution operation with a local summary statistic, which helps the model to develop representations that are invariant to small translations (GOODFELLOW; BENGIO; COURVILLE, 2018).

One of the most widely used types is max pooling, which selects the maximum value within a local neighborhood, as shown in Figure 6. Other common pooling methods include average pooling, which computes the mean of the values within the region, and weighted average pooling, where each value contributes proportionally based on predefined weights.

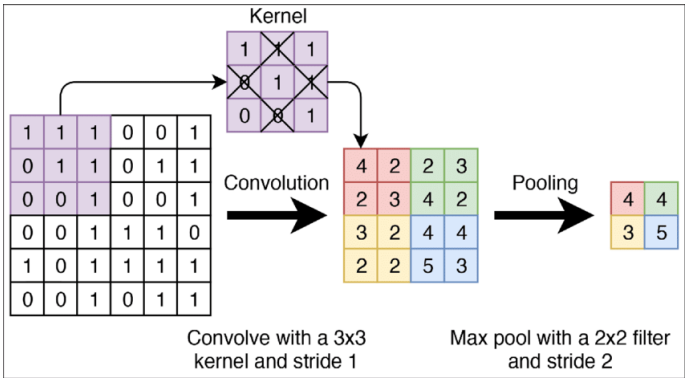


Figure 6 – Example of max pooling operation in a convolutional neural network. (VAART; LAMBERS, 2019)

2.3.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of artificial neural network specifically designed to work with sequential data, such as time series, speech recognition, text generation, and language modeling (SCHMIDT, 2019; FANG; CHEN; XUE, 2021; ALZUBAIDI et al., 2021). The main difference between an RNN and the networks discussed in Sections 2.3.1 and 2.3.3 is how data is processed inside the RNN. In the MLP and the CNN networks, it is assumed that each input is independent from each other, which can be a valid assumption when dealing with images, but might be a severe limitation when dealing with sequential data, like weather data, where, for example, the current temperature depends on the last few measurements.

RNNs deal with this kind of problem by introducing a cycle in their architecture allows information to persist across time steps (SCHMIDT, 2019), as shown in Figure 7. The key idea is to add a hidden-to-hidden weight matrix $\boldsymbol{W}_{hh}$ that carries information from the previous hidden state $\boldsymbol{H}_{t-1}$ in the computation of the current hidden state $\boldsymbol{H}_t$.

The current hidden state can be computed using Equation 2.12:

$$\boldsymbol{H}_t = \phi(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hh} + \boldsymbol{b}_h) \tag{2.12}$$

where $\boldsymbol{X}_t$ is the input vector, $\boldsymbol{W}_{xh}$ is the input-to-hidden weight matrix, $\boldsymbol{b}_h$ is the bias, and ϕ is an activation function.

An important observation is that it is not necessary to explicitly include all previous hidden states, from $\boldsymbol{H}_0$ up to $\boldsymbol{H}_{t-2}$, when computing $\boldsymbol{H}_t$. This is because, at each iteration, the RNN updates $\boldsymbol{H}_t$ by combining the input $\boldsymbol{X}_t$ with the previous hidden state $\boldsymbol{H}_{t-1}$, which is the result of the computation between $\boldsymbol{X}_{t-1}$ and $\boldsymbol{H}_{t-2}$, and so on recursively. This way, the network can keep information about the previous time steps when computing the next one.

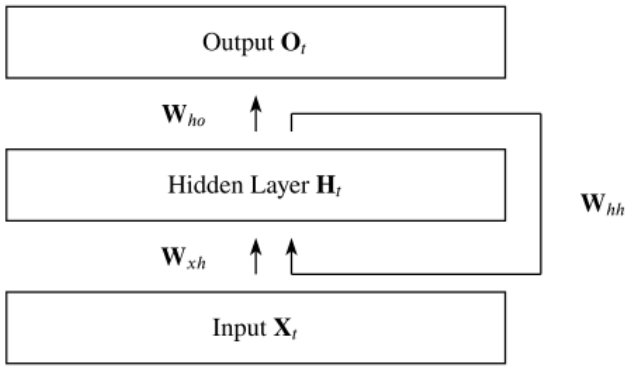


Figure 7 – Architecture of a recurrent neural network showing the recurrent connections between hidden states. Adapted from (SCHMIDT, 2019)

The main problem in the original RNN architecture arises during the training phase, specifically when computing gradients for weight updates. As gradients are propagated backward through time, a series of repeated multiplications occurs. If these values are greater than 1, the gradients can grow exponentially, causing the network weights to change abruptly fast. Conversely, if the values are less than 1, the gradients decay exponentially, which causes the gradient to vanish (KOLEN; KREMER, 2001; GLOROT; BENGIO, 2010). The vanishing gradient then causes the network to forget information from the initial time steps, which makes it difficult for RNNs to learn long-term dependencies in data.

This inherent problem motivated the introduction of more complex architectures, such as the LSTM networks. These networks will be explained in more detail in 2.3.5, but they mitigate the vanishing gradient problem, which caused the original RNNs to be barely used in today's research (FANG; CHEN; XUE, 2021; GAO et al., 2019).

2.3.5 LSTM

Long Short-Term Memory (LSTM) units were introduced by Hochreiter and Schmidhuber in 1997 (HOCHREITER; SCHMIDHUBER, 1997) and were specifically designed to deal with the vanishing gradient problem. Differently from vanilla RNNs, which struggle to retain information over long sequences, LSTM cells are specifically designed to capture long-term dependencies by preserving information across many time steps

(AL-SELWI et al., 2024; SCHMIDT, 2019).

The core idea behind the LSTM architecture lies in its gating mechanism, which regulates the flow of information through the cell. This mechanism is composed of three gates: the forget gate, the input gate, and the output gate, shown in Figure 8. Each gate uses a sigmoid activation function, producing values between 0 and 1 to determine how much information should pass through (SCHMIDT, 2019).

- The forget gate decides which parts of the previous cell state should be discarded, allowing the network to "forget" irrelevant or outdated information.
- The input gate determines which portions of the new input information should be added to the cell state.
- The output gate controls which parts of the internal cell state are exposed as the output hidden state at the current time step.

The forget and input gates are responsible for updating the cell state, which acts as an internal memory, carrying forward important information throughout the sequence. The output gate, meanwhile, regulates how much of the updated cell state should influence the hidden state output. Through this mechanism, LSTMs are able to maintain and regulate long-term information flow (AL-SELWI et al., 2024).

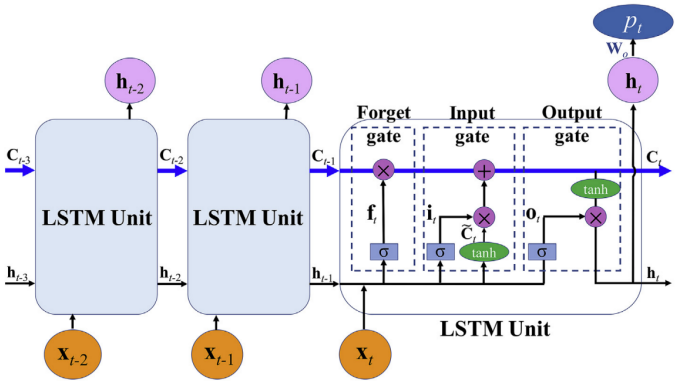


Figure 8 – Structure of an LSTM cell with input, forget, and output gates. Adapted from (WEI et al., 2021)

Using the same notation from Section 2.3.4 one can compute:

$$\boldsymbol{O}_t = \sigma\left(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o\right) \tag{2.13}$$

$$\boldsymbol{I}_t = \sigma\left(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i\right) \tag{2.14}$$

$$\boldsymbol{F}_t = \sigma\left(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f\right) \tag{2.15}$$

where $\boldsymbol{O}_t$, $\boldsymbol{I}_t$, and $\boldsymbol{F}_t$ are the inputs for the output gate, the input gate, and the forget gate, respectively. After this computation, the LSTM unit computes the candidate memory

cell $\tilde{\boldsymbol{C}}_t$, which represents the candidate information that could be added to the cell state, as:

$$\tilde{\boldsymbol{C}}_t = \tanh\left(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c\right) \tag{2.16}$$

After that, the LSTM introduces the previous memory content $\boldsymbol{C}_{t-1}$, which, together with the other gates, controls how much of the old memory information will be retained in the new memory state $\boldsymbol{C}_t$, which can be computed using:

$$\boldsymbol{C}_t = \boldsymbol{F}_t \otimes \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \otimes \tilde{\boldsymbol{C}}_t \tag{2.17}$$

where $\otimes$ denotes the element-wise multiplication. Finally, the LSTM unit computes the hidden state $\boldsymbol{H}_t$ as:

$$\boldsymbol{H}_t = \boldsymbol{O}_t \otimes \tanh(\boldsymbol{C}_t) \tag{2.18}$$

It is important to notice the hidden state $\boldsymbol{H}_t$ is used as the output of the LSTM cell and, together with the memory state $\boldsymbol{C}_t$, is used in the computation of the next time step.

2.3.6 Attention

Attention is an important concept in deep learning that was first introduced by (BAHDANAU; CHO; BENGIO, 2016) in a machine translation problem and gained widespread popularity with the famous paper "Attention Is All You Need" proposed by (VASWANI et al., 2023), where the authors proposed the transformer architecture. Inspired by the human attention of selectively focusing on relevant information, attention mechanisms enable neural networks to assign different weights to different parts of the input, allowing the model to concentrate on the most important elements of the data. This selective focus not only improves the model's interpretability but also makes it possible to handle large amounts of information more efficiently, reducing computational overhead (NIU; ZHONG; YU, 2021).

The attention mechanism works by taking a vector of features $\boldsymbol{F}$, which are obtained after the input data $\boldsymbol{X}$ is passed through a feature extraction model to extract the feature vectors $\boldsymbol{f}_1$, to $\boldsymbol{f}_n$. This feature extractor can vary depending on the task and, in the case of time series analysis, for example, it could be an encoder composed of CNNs, MLPs, or LSTMs, that produces the latent representations $\boldsymbol{f}_1$ to $\boldsymbol{f}_n$ (BRAUWERS; FRASINCAR, 2023).

From the feature vectors $\boldsymbol{F}$ the model then extracts the key and value vectors $\boldsymbol{K}$ and $\boldsymbol{V}$, respectively. These vectors are usually obtained through a linear transformation of $\boldsymbol{F}$ using the weight matrix $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$.

After that, the model introduces a query vector $\boldsymbol{Q}$, which is a task-specific vector that guides the attention mechanism to focus on the most important vectors

(BRAUWERS; FRASINCAR, 2023; NIU; ZHONG; YU, 2021). From the query and key vectors, the model then computes the energy score $\boldsymbol{e}$ using a score function f:

$$\boldsymbol{e} = f\left(\boldsymbol{Q}, \boldsymbol{K}\right) \tag{2.19}$$

where f can be a variety of functions, such as the scaled dot product (VASWANI et al., 2023):

$$f\left(\boldsymbol{Q}, \boldsymbol{K}\right) = \frac{\boldsymbol{Q}^T \boldsymbol{K}}{\sqrt{d_k}} \tag{2.20}$$

The energy scores $\boldsymbol{e}$ are then mapped to the attention weights $\boldsymbol{\alpha}$ using an attention distribution function g, usually the softmax function (NIU; ZHONG; YU, 2021):

$$\boldsymbol{\alpha} = \mathrm{g}(\boldsymbol{e}) \tag{2.21}$$

Using the attention weights $\boldsymbol{\alpha}$ and the value vector $\boldsymbol{V}$, the model computes the context vector $\boldsymbol{c}$ using:

$$\boldsymbol{c} = \sum_{l=1}^{n} \alpha_l \boldsymbol{V}_l \tag{2.22}$$

where $\alpha_l \in \mathrm{R}^1$ is the weight of the l-th vector. Figure 9 shows the architecture of the attention model.

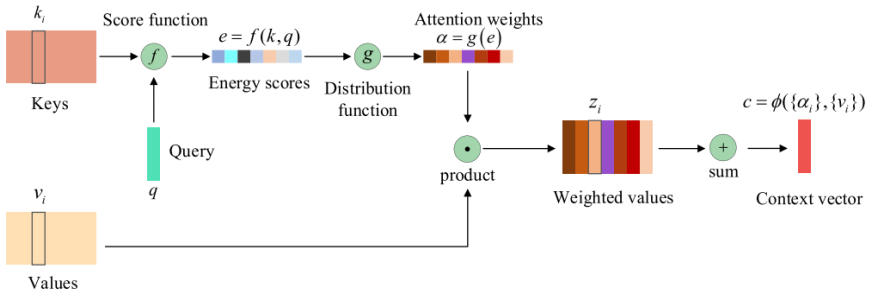


Figure 9 – Diagram of the attention mechanism showing query, key, value vectors, and the computation of the context vector. (NIU; ZHONG; YU, 2021)

2.3.7 Transformers

The transformer architecture was introduced by (VASWANI et al., 2023) and, since then, has gained widespread popularity in various fields, such as natural language processing (NLP), computer vision (CV) and time series analysis (LIN et al., 2021; WEN et al., 2023). It is built upon the attention mechanism, shown in Section 2.3.6, which makes it particularly effective in modeling long range dependencies in sequential data (WEN et al., 2023). Since the (VASWANI et al., 2023) paper, a variety of transformers architectures have been proposed to improve the original transformer, as shown in (LIN et al., 2021). However, this section will focus on the vanilla Transformer.

The vanilla transformer architecture is a sequence to sequence model composed of an encoder and a decoder. The encoder maps an input sequence $\boldsymbol{X} = (x_1, ..., x_n)$ to a latent representation $\boldsymbol{Z} = (z_1, ..., z_n)$. The decoder then utilizes this latent space to generate an output sequence $\boldsymbol{Y} = (y_1, ..., y_m)$ one step at a time. The process is auto-regressive, meaning it uses the previously generated output to generate the next output (VASWANI et al., 2023). Figure 10 shows the architecture of the transformer model.

The encoder is composed of a stack of L identical blocks. Each block is composed by two sub-layers: a multi-head attention and a feed-forward network. The multi-head attention sublayer is composed of H attention blocks, that use different projection matrices $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ and $\boldsymbol{W}_Q$ to compute the key, value and query vectors, respectively. The output of each attention block is then concatenated and multiplied by a matrix $\boldsymbol{W}^O$, as shown in Equation 2.23:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (\text{head}_1; ...; \text{head}_H) \cdot \boldsymbol{W}^O \tag{2.23}$$

where $\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$ is the i-th attention head, and Attention is the attention function defined in Section 2.3.6.

A residual connection and layer normalization follow the multi-head attention, as shown in Equation 2.24:

$$\boldsymbol{H}' = \text{LayerNorm}(\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) + \boldsymbol{X}) \tag{2.24}$$

where $\boldsymbol{X}$ is the input to the multi-head attention sublayer and LayerNorm is the layer normalization function. The output $\boldsymbol{H}'$ is then processed by a position-wise feed-forward network with two fully connected layers: 2.25:

$$\text{FFN}(\boldsymbol{H}') = \text{ReLU}(\boldsymbol{H}'\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2.25}$$

The decoder has a similar architecture to the encoder. It has L stacked blocks, but each block is composed of three sub-layers: a masked multi-head attention layer, a multi-head attention layer, and a feed-forward network. The masked multi-head attention layer works similarly to the multi-head attention, but the self-attention is masked to prevent the model from attending to future tokens in the sequence, ensuring that the prediction for a given token only depends on the previous tokens (LIN et al., 2021). Another difference is in the second multi-head attention layer, which uses the output of the encoder $\boldsymbol{Z}$ to project the key and value vectors of the attention.

Finally, one important point to highlight is the positional encoding layer. In vanilla transformers, the model does not have access to the order of the sequence. To deal with this, the authors propose adding a positional encoding to the input. They do so using the sine and cosine functions:

$$PE_{(pos,2i)} = \text{sen}\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.26}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.27}$$

where i is the dimension, pos is the token position and d_{model} is the model dimension.

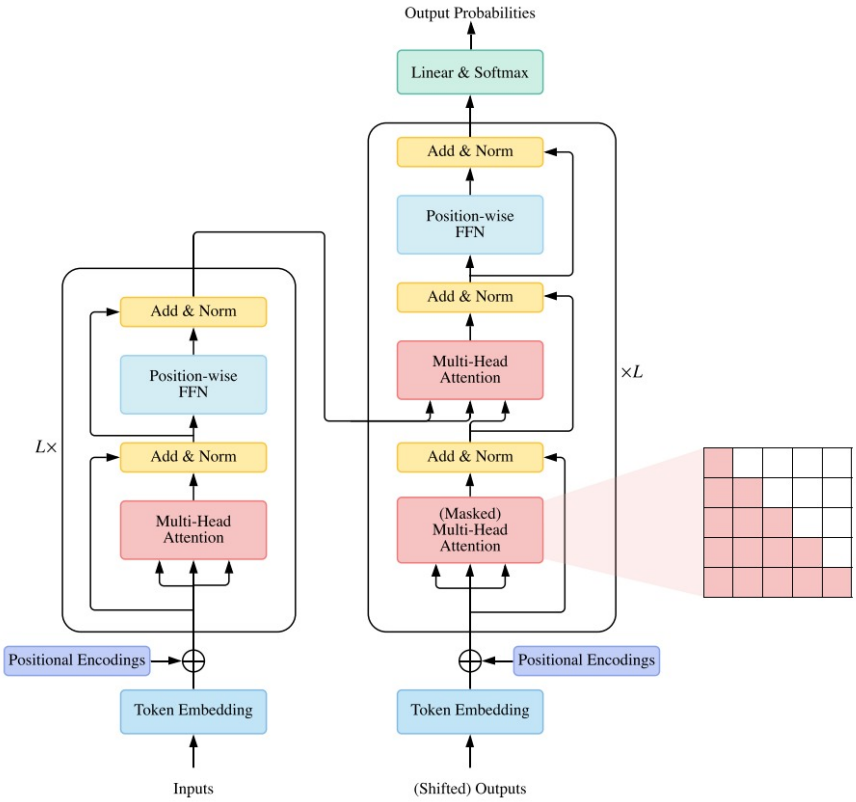


Figure 10 – Architecture of the vanilla Transformer model, with encoder and decoder components. (LIN et al., 2021)

2.4 Reconstruction-Based Models

Reconstruction-based models are a class of machine learning models that are designed to learn the intrinsic correlations in data by trying to reconstruct the original input signal. Differently from forecasting-based approaches, which attempt to predict future values, reconstruction-based models focus on accurately recreating the current input. This approach makes them great for anomaly detection, often outperforming forecasting-based models because they have access to the current time series data, which helps them to be more stable to rapidly changing time series. However, this advantage comes with a trade-off because they introduce a delay in the detection of anomalies as they rely on reconstruction error analysis. (DARBAN et al., 2024).

The main idea behind these networks is to train them to accurately reconstruct normal data, which is presumed to be the majority of the available data. This is usually done by minimizing a reconstruction loss, such as the Mean Squared Error (MSE), during the training phase.

During the test phase, the model encounters anomalous data, which it has not been trained to reconstruct, causing the reconstruction error to increase greatly compared to the error of normal data, as the model fails to reconstruct abnormal patterns. This way, an anomaly can be identified by thresholding the reconstruction error. Inputs with low reconstruction error will be classified as normal, and inputs with reconstruction error greater than the threshold will be classified as anomalous.

There are a variety of reconstruction-based models. In (LACHEKHAB et al., 2024), the authors proposed an autoencoder composed of multiple LSTM cells to identify anomalies in electrical motors vibrations. The model was trained using the MSE loss function to reconstruct normal vibration patterns, and anomalies were detected by applying a threshold to the reconstruction error.

In (XIE; XU; JIANG, 2023), the authors developed a model that combines convolutional layers, LSTM layers and a variational autoencoder to detect anomalies in multivariate time series. They computed correlation matrices between the variables to use as an input for the model and used a loss function that combines the MSE loss to the Kullback-Leibler divergence to train the model, achieving state-of-the-art results in three different datasets.

Another example is the model proposed in (TULI; CASALE; JENNINGS, 2022) where authors utilized the transformer architecture to detect anomalies. The model works on two phases: in the first phase the model tries to reconstruct the input sequence, obtaining a reconstruction error. From this, the model computes the focus score that highlight areas with bigger reconstruction errors. In the second phase, the focus score is used by the attention mechanism to focus on areas where the reconstruction error was bigger, allowing the model to concentrate on these high-error regions and attempt a refined reconstruction. This approach allowed the authors to outperform state-of-the-art models on six different datasets.

2.5 Railway Components

The railway track consists of different interdependent components that are divided into two categories: the superstructure and the substructure. The former comprises the rails, fastenings, and sleepers, while the latter consists of the ballast, sub-ballast, and subgrade (KAEWUNRUEN[1]; REMENNIKOV[1], 2008), as shown in Figure 11. These two structures are separated by the sleeper-ballast interface and are essential for ensuring a safe and cost-effective transportation system capable of guiding vehicles and transmitting loads to the subgrade (ATTOH-OKINE, 2017).

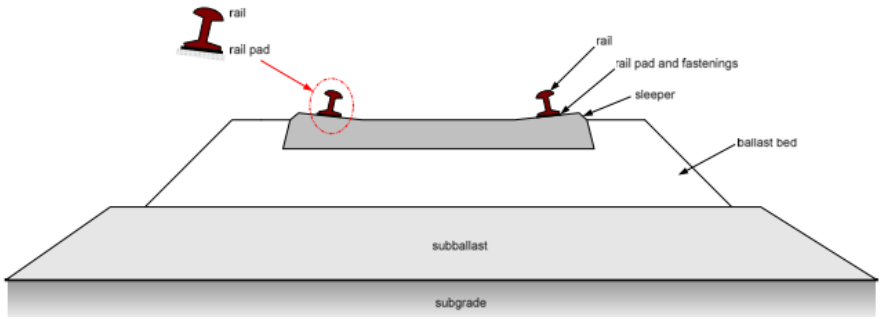


Figure 11 – Main railway track components, including superstructure and substructure. (KAEWUNRUEN[1]; REMENNIKOV[1], 2008)

2.6 Track Irregularities Standards

Railway track irregularities are deviations from the ideal track geometry that can affect the safety and comfort of train operations (SANSINENA; RODRíGUEZ-ARANA; ARRIZABALAGA, 2025). So, to ensure safe and efficient railway operations, a continuous assessment of track condition is necessary to support planning of maintenance actions. For this reason, different countries have established standards that quantify the acceptable limits for these irregularities. In subsections 2.6.1 and 2.6.2, two of these standards will be presented: the Federal Railroad Administration (FRA) standard, used in the United States, and the Brazilian standard, used in Brazil.

2.6.1 FRA Standard

Railway track irregularities are normally modeled as being random and, for this reason, are typically described by a Power Spectral Density (PSD) function. Countries like USA, China, Britain and Germany adopted this approach to describe track irregularities (PIRES et al., 2021; BERAWI, 2013). The PSD function parameters were estimated from measured data and, therefore, represent the average irregularity observed in each respective railway network.

FRA classifies tracks into nine categories based on their condition, assigning a PSD function to each category. Categories 1 to 6 are designed for ordinary tracks, while categories 7 to 9 are intended for high-speed tracks. Due to equipment limitations, the FRA standard can only describe irregularities with wavelengths between 1.524 m to 304.8 m (BERAWI, 2013).

The PSD formulas are given by Equations 2.28, 2.29 and 2.30:

$$S_{av}(\Omega) = \frac{k \cdot A_v \cdot \Omega_c^2}{\Omega^2 \cdot (\Omega^2 + \Omega_c^2)} \tag{2.28}$$

$$S_{al}(\lambda) = \frac{k \cdot A_a \cdot \Omega_c^2}{\Omega^2 \cdot (\Omega^2 + \Omega_c^2)} \tag{2.29}$$

$$S_{gauge/cl}(\lambda) = \frac{4 \cdot k \cdot A_v \cdot \Omega_c^2}{(\Omega^2 + \Omega_c^2) \cdot (\Omega^2 + \Omega_s^2)} \tag{2.30}$$

where $S_{av}(\Omega)$ is the vertical alignment PSD, $S_{al}(\Omega)$ is the lateral alignment PSD, and $S_{gauge/cl}(\Omega)$ is the gauge or the superelevation PSD, all expressed in $[cm^2/(rad/m)]$. The variable Ω is the spatial wavenumber, while Ω_c and Ω_s are the critical wavenumbers, all in $[rad/m]$. The variable k is a constant, and A_v and A_a are the roughness coefficients related to the line grade, expressed in $[cm^2 \cdot rad/m]$. Table 1 summarizes the parameters for each track class.

Class	Max velocity (km/h)		Parameters			
	Freight	Passengers	A_v (cm^2 rad/m)	A_a (cm^2 rad/m)	Ω_c^2 (rad/m)	Ω_s^2 (rad/m)
1	16	24	1.2107	3.3634	0.6046	0.8245
2	40	48	1.0181	1.2107	0.9308	0.8245
3	64	97	0.6816	0.4128	0.8520	0.8245
4	97	129	0.5376	0.3027	1.1312	0.8245
5	129	145	0.2095	0.0762	0.8209	0.8245
6	177	177	0.0339	0.0339	0.4380	0.8245

Table 1 – Summary of the parameters for each track class. (PIRES et al., 2021)

One important point to highlight is that track condition in USA have very similar conditions to those in Brazil. For this reason, the FRA standard can be used to describe track irregularities in Brazil (PIRES et al., 2021).

2.6.2 Brazil Standard

2.7 Assessing Railway Track Condition

Maintaining railway tracks in good condition is crucial to ensure safe and comfortable operations of trains (TSUNASHIMA, 2019; GHIASI et al., 2025). This maintenance is achieved through continuous monitoring of track conditions and the detection of irregularities. There are now two main methods to assess track quality: using track geometry cars (TGC) to directly measure the physical geometry of the track, and monitoring the dynamic response of the train using instrumented railway vehicles (IRV) equipped with onboard sensors capable of measuring the dynamics of the system (PIRES et al., 2024).

TGCs are equipped with sophisticated systems that directly measure the geometry of the track, that can be later compared to regulatory standards limits to identify parts of the track that need maintenance. However, this method has several drawbacks (PIRES et al., 2024; GHIASI et al., 2025; ONO et al., 2023):

- Operational disruption: the operation of the railway track needs to be disrupted during the TGC inspection, which can halt operations for several hours depending on the length of the track being measured;
- Operational high cost: the sophisticated equipment and logistics make TGC operation costly, which limits their frequent use;
- Low inspection frequency: normally, the TGC measures the condition of the track monthly, due to its high cost, which is a problem if a fault appears between inspections.

To overcome the limitations in TCGs, an alternative approach was developed using IRVs equipped with sensors that measure the vehicle's dynamic response due to track excitations. The underlying principle behind IRVs is that the vehicle's vibrations are deeply correlated to track excitations (PIRES et al., 2024; TSUNASHIMA, 2019), and an irregularity in the track will cause an anomalous dynamic response from the vehicle, which can be identified in the sensors' readings.

Using IRVs instead of the TCGs has several advantages, such as (PIRES et al., 2024; GHIASI et al., 2025; ONO et al., 2023):

- No disruptions: the operation is not halted during the measurements, since data can be collected during normal train services, which reduces the costs with logistics;
- Real time data collection: data is collected in real time, which speeds up possible defect detections;
- High frequency of inspections: Since the measurements are taken directly from in-service trains, IRVs enable near-continuous monitoring of track conditions, greatly increasing the inspection frequency.

Despite that, it is important to highlight that data collection using IRVs also comes with some drawbacks, such as (PIRES et al., 2024; GHIASI et al., 2025):

- High amount of data: the amount of data collected during IRV operation can be huge. This data needs to be carefully processed to obtain good quality data;
- Domain problem: different operations conditions, such as the wagon mass or velocity, can affect the dynamic response of the vehicle, so data collected from a one part of the track can be inherently different from another part;
- Correlation: since the measurements are done indirectly, they need to be correlated to the track condition, which can be a complex task that often needs the use of deep learning models;

- Data imbalance: since fault measurements are much more uncommon than normal measurements, the machine learning model needs to deal with class imbalance.

2.8 Velocity Effect on Train Dynamics

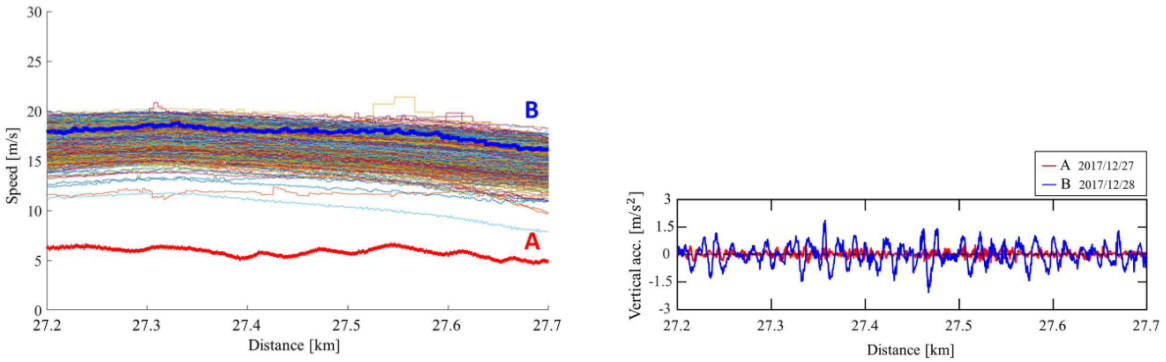
Train vibration measurements are strongly correlated to the velocity at which these measurements took place. As the wagon moves along the track, changes in velocity can alter the vibration responses measured by onboard sensors. At lower speeds, the dynamic response of the train to the track excitations tends to produce lower acceleration values than expected, while, for the same part of the track, higher velocities generate higher values of acceleration measured (ONO et al., 2023). Figure 12 shows this difference in measurement found by (ONO et al., 2023).

In Figure 12(a), the authors highlight two distinct velocity profiles labeled A and B, where the speed ratio between B and A is approximately 3. This difference in speed causes a significant change in the measured signal, as shown in Figure 12(b), where the velocity profile B generated bigger acceleration measures than A despite them traversing the same part of the track.

To address this velocity-induced distortion, the authors propose two different correction methods. The first method involves using the Mahalanobis distance to distinguish outliers from normal data. After that, the authors fitted a linear regression to normal data to predict the expected acceleration given the speed. The measurements are then normalized by dividing them by their predicted values.

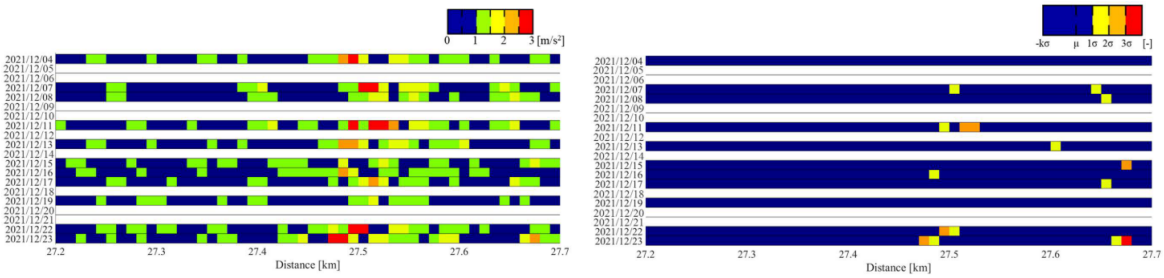
In the second method, the authors employed a Gaussian process regression to model the behavior of acceleration and speed, obtaining a regression that mapped the velocity to the expected acceleration. Similarly, they normalized the measurements by their respective predictions.

Both methods proved to be effective in mitigating the velocity effect, which reduced the number of false positives in anomaly detection, as shown in Figure 13.



((a)) ((b))

Figure 12 – Example of how velocity differences affect acceleration measurements on the same track segment. (***a***) shows the two velocity profiles, A and B. (***b***) illustrates the corresponding acceleration measurements for each profile. ((***a***) (ONO et al., 2023); (***b***) (ONO et al., 2023))



((a)) ((b))

Figure 13 – Reduction of false positives in anomaly detection after velocity correction. (***a***) shows the heatmap of the acceleration measurements before correction, while (***b***) shows the heatmap after correction. ((***a***) (ONO et al., 2023); (***b***) (ONO et al., 2023))

Another example demonstrating the impact of the velocity on measured acceleration is presented in (BALOUCHI; BEVAN; FORMSTON, 2021). In this work, the authors utilized the VAMPIRE vehicle dynamics simulation software to model a vehicle operating in a measured track geometry at a variety of speeds. The simulated results were then compared to the actual car body vibration measurements recorded over the same section of the track. The results are shown in Figure 14, where the continuous colored lines are the simulation results, blue circles represent the maximum acceleration found in the simulation, and the dashed lines are the car body measurements.

The authors fitted two models to the maximum simulated accelerations: a linear regression, the green line in Figure 14 and an exponential model, the purple curve in Figure 14, to the maximum simulation acceleration, selecting the exponential model as it has a bigger R^2 value. They then argue that at lower speeds, the acceleration response does not yield a good level of confidence to differentiate if the quality of the track is good

or bad, and so a normalization similar to (ONO et al., 2023) is necessary.

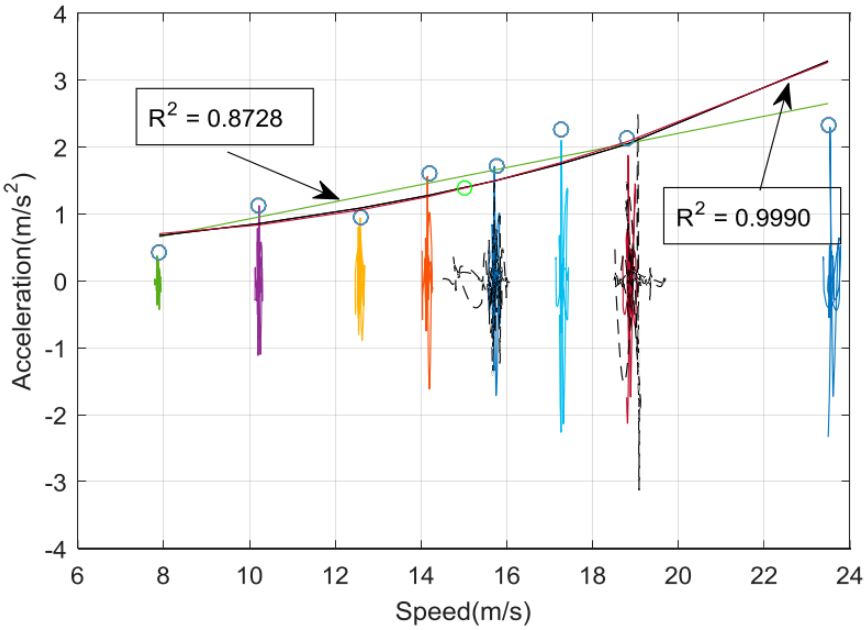


Figure 14 – Effect of train velocity on measured acceleration compared with simulation results and the linear and exponential model fitted to the maximum acceleration values with their corresponding R^2 values. (BALOUCHI; BEVAN; FORMSTON, 2021)

In the two papers presented above, the authors show the importance of correcting the acceleration measurements acquired at different velocities, as variations in speed alter the dynamic response of the train. However, their correction only considers the effect of the velocity and not the effect of the mass, as these studies were performed in a passenger car. As will be shown in Section 2.9, for freight wagon, the mass of the train might also affect the dynamics of the system.

2.9 Mass Effect on Train Dynamics

When analyzing the data collected by an IRV in a Brazilian railway, described in detail in (PIRES et al., 2024), it was noticed that the mass of the wagon influences the relationship between acceleration growth and velocity. This phenomenon is illustrated in Figure 15, which shows the 99th percentile of the absolute maximum vertical acceleration as a function of the velocity class. Each velocity class corresponds to data points that have a velocity between an 1 m/s interval. For example, the velocity class $7 < v < 8$ corresponds to all data points whose velocity is between 7 m/s and 8 m/s.

In the Figure, data points are colored according to the wagon mass class. The unloaded class corresponds to an empty wagon, i.e, that is not transporting any material, the loaded class corresponds to a wagon fully loaded with iron ore, and the 1/4 to 3/4

classes correspond to wagons that were either partially loaded or carrying lighter materials such as coal. One point to highlight, is that it was used the 99th percentile instead of the maximum value because the dataset consists of real-world measurements, which are prone to noise and outliers.

As shown in the Figure, different wagon masses produce distinct acceleration growth behavior: the unloaded wagon exhibits the lowest acceleration growth, the 1/4 loaded wagon shows a slightly higher growth, and the 2/4, 3/4, and fully loaded wagons display similar growth patterns.

These observations highlight the need to further investigate the correlation between wagon mass and acceleration behavior for freight operations.

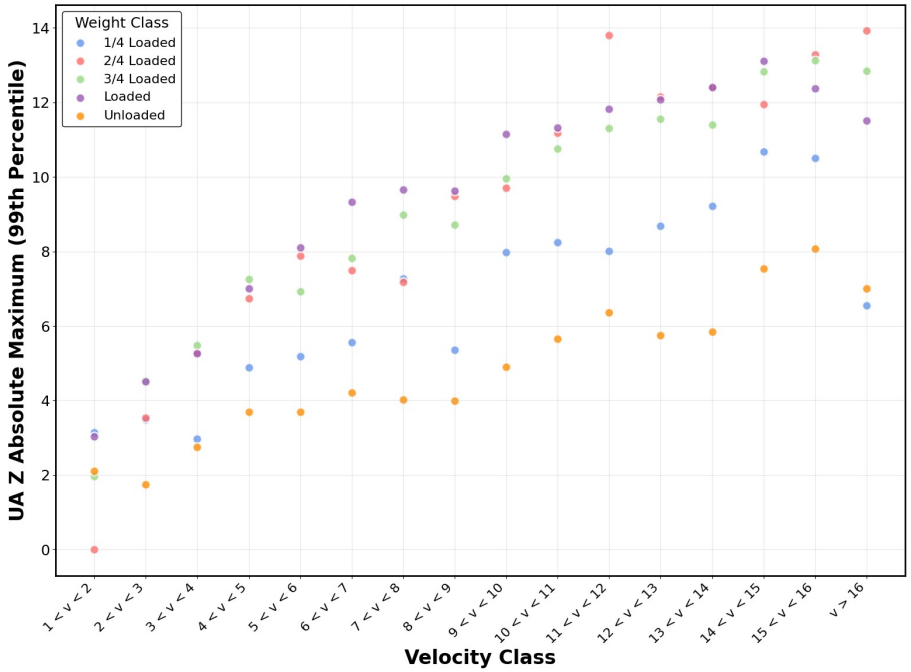


Figure 15 – Effect of wagon mass on the relationship between acceleration and velocity for freight trains. The data points represent the 99th percentile of the absolute maximum vertical acceleration for each velocity class, colored by wagon mass class.

2.10 Assessing Track Quality from Acceleration Data

As stated in Section 2.7, continuous measurements of track irregularities are an important factor in ensuring safety and comfort during railway operations. Traditionally, TGCs were used to assess track irregularities by directly measuring the track geometry. However, due to their high cost, constant inspection of the railway is not possible.

An alternative to this approach is to use an IRV equipped with sensors that can measure the dynamic response of the vehicle due to track excitations. Normally, in this setup, accelerometers are mounted on various one or more components of the train, such

as the axlebox, the bogie, or the car body, depending on the objective of the measurement and the maintenance costs of the sensors (TSUNASHIMA; YAGURA, 2024; SANSINENA; RODRíGUEZ-ARANA; ARRIZABALAGA, 2025).

In (SANSINENA; RODRíGUEZ-ARANA; ARRIZABALAGA, 2025) authors presented a review of research papers that proposed methods for estimating track irregularities using acceleration data. They divided the methods into three main categories:

- Model-driven methods;
- Data-driven methods;
- Hybrid methods.

In model-driven methods, a mathematical representation of the dynamic system is developed and validated on real data. Once the model accurately represents the system dynamics, the problem is inverted to estimate track irregularities from the measured acceleration data. The main advantage in model-driven methods is that they don't require a large amount of data to be designed, as they're based on expert knowledge.

For instance, in (De Rosa; ALFI; BRUNI, 2019) the authors used a 3D multibody model that was previously validated to measure the yaw and roll motions of the vehicle. Based on simulation data, they applied three reconstruction techniques, the Kalman filter, the Unknown Input Observer (UIO) and the Frequency Response Function (FRF), to reconstruct the lateral irregularities and the cross alignment. The performance of these methods was compared using two evaluation metrics, but the detailed results are out of the scope of this thesis. They then tested the FRF method on real measured data, which included lateral and vertical accelerations recorded at the axle-boxes, bogies, and car body, obtaining the reconstruction shown in Figure 16. The authors argue that although some similarity between the reconstructed and the original irregularities can be seen, the method did not produce satisfactory results. They justify that this deviation is caused by non-linear effects they didn't consider, but these deviations can occur because of the lack of robustness of the method to real-world noise in data, which is a common problem when dealing with simpler reconstruction methods.

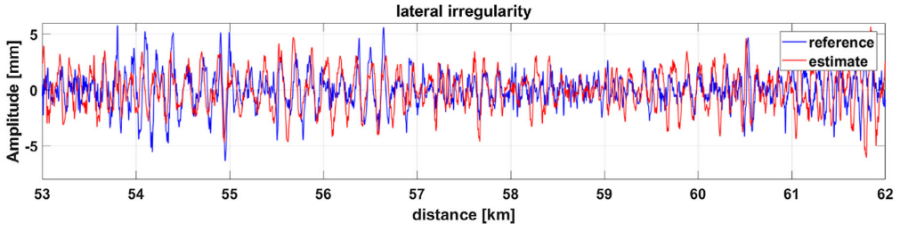


Figure 16 – Reconstruction of lateral irregularities using the Frequency Response Function method. (De Rosa; ALFI; BRUNI, 2019)

Data-driven methods, on the other hand, do not consider a mathematical model of the system, relying only on data processing to reconstruct or identify the irregularities. These methods typically utilize traditional machine learning algorithms or, more recently, deep learning architectures. Such approaches are generally more robust when dealing with real-world data, as they can learn from noisy and complex signals. This robustness, however, comes with a price, as they need a larger amount of data to be trained, especially in the case of deep learning (SANSINENA; RODRíGUEZ-ARANA; ARRIZABALAGA, 2025).

In (TSUNASHIMA; YAGURA, 2024) the authors propose a method of estimating railway track irregularities from car body vibration data. They first created a multibody simulation in SIMPACK. After that, they generated 12 different track irregularities, ranging from a good condition to a degraded condition, using FRA PSD formula and converted the generated profiles into 10 meters long track irregularity sections using the 10 m-chord versine method. This transformation is shown in Figure 17.

Using these irregularity profiles, they ran simulations over a 1000 meters track at speeds in the range of 40 to 80 km/h, varying the velocity in a 10 km/h interval. For each combination of track profile and speed, they collected the vertical and lateral acceleration as well as the roll rate of the carbody. Similarly to the irregularity profiles, a downsample was also applied to the data taking the maximum vibration measured in the 10 meters section, as shown in Figure 17. From this process, they obtained a dataset that correlates the maximum vibration of the carbody to the track irregularities for varying speeds and then applied a Gaussian Process Regressor (GPR) that was able to predict the irregularity from the vibration for a certain vehicle speed, as illustrated in Figure 18.

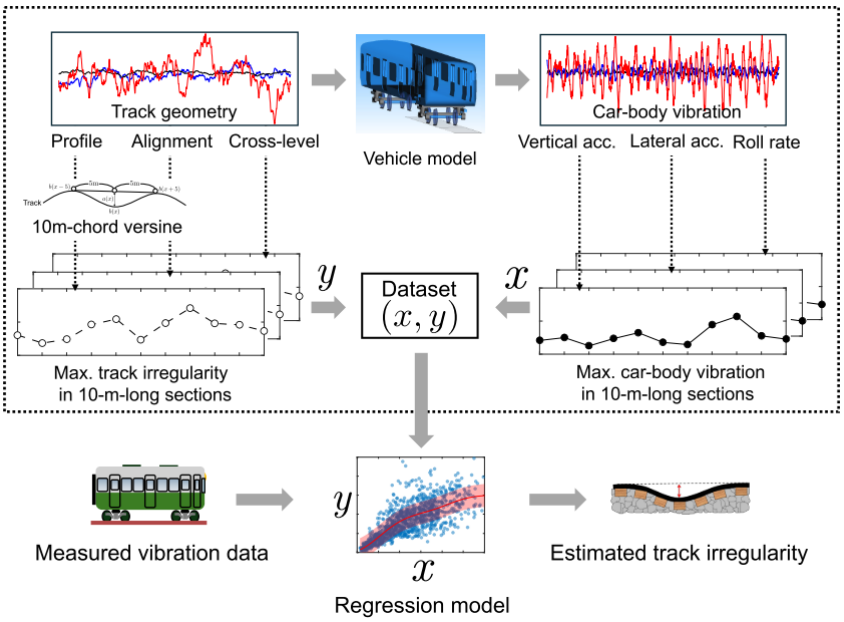


Figure 17 – Data processing pipeline for the creation of the simulated dataset. (TSUNASHIMA; YAGURA, 2024)

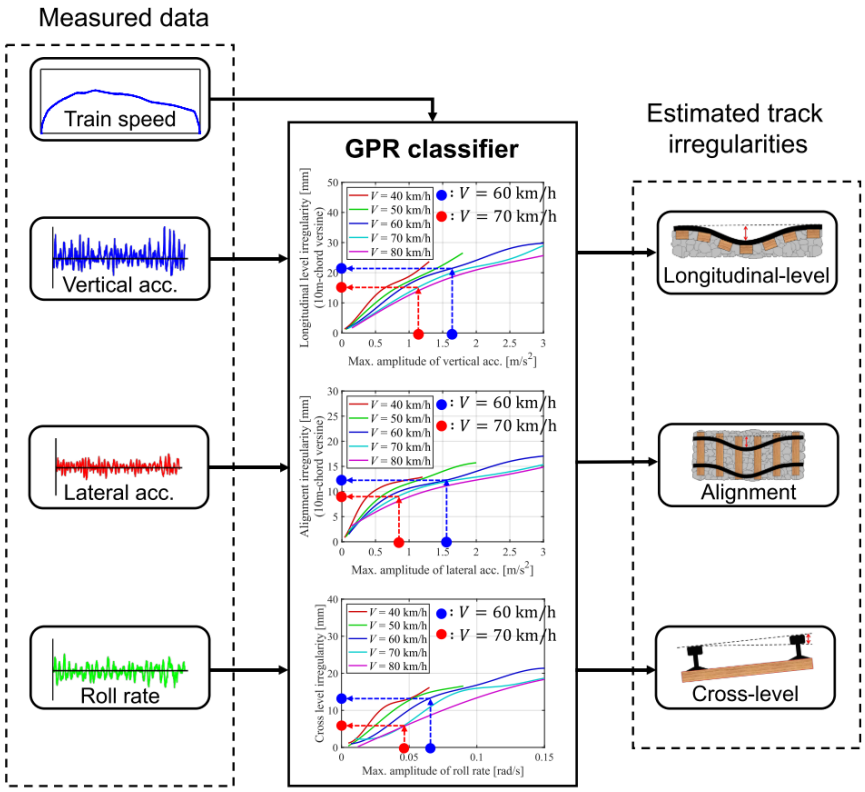


Figure 18 – Track irregularity prediction using Gaussian Process Regression for different vehicle speeds. (TSUNASHIMA; YAGURA, 2024)

The authors then tested their model utilizing real data measured using an onboard sensing device equipped with a triaxial accelerometer, a rate gyro, and a GNSS receiver. To ensure consistency, they applied the same downsample procedure to the real data and averaged the speed in the 10 m section to be used as an input for the regression model. The regression result for one part of the track is shown in Figure 19 and is presented with a 1σ confidence interval.

Tsunashima and Yagura argue that for the majority of the sections, the GPR predicts the correct value within the 1σ confidence interval. However, for some of the sections, the predicted value significantly deviated from the actual measurement. They argue that there is a factor other than the track irregularity that influences the dynamics of the system and superimposes on the carbody's vibration. This difference might also occur because of the low complexity model used that might not have been able to learn the correlations between the variables very well.

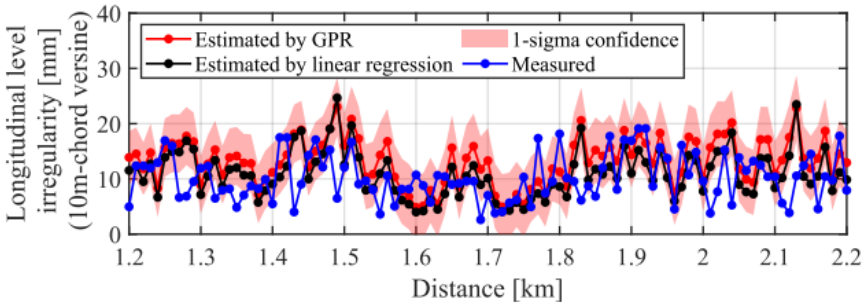


Figure 19 – Regression results comparing predicted and actual track irregularities from real measurement data. (TSUNASHIMA; YAGURA, 2024)

In (ROSA et al., 2021), the authors try to develop a threshold to detect not safe railway track irregularities. In the paper, the authors propose a machine learning-based classification model that divides the data into two classes based on the standard deviation of the lateral and roll bogie frame accelerations. Class 1 corresponds to normal conditions, i.e., track irregularities below a defined threshold that do not require maintenance, while Class 2 corresponds to sections with irregularities exceeding that threshold, thus requiring maintenance intervention.

The authors use a dataset composed of real measurements collected onboard the high-speed TGC, operating at 300 km/h, that was also equipped with accelerometers located on the carbody, the bogie frames and the wheelsets, and a set of simulated data generated using a validated multibody dynamics model. From real data, the athor applies a pass-band filter in the range of 3 to 27 Hz and then computed the standard deviation of the signals over 100-meter-long track length. The simulation data was obtained considering a straight track, with a vehicle with speed of 300 km/h and for three different cases of track irregularities: considering only the lateral irregularity, considering only the cross level irregularity, and considering both at the same time. The authors didn't consider vertical irregularities in their simulation. For each case, they ran the simulation 10 times. The standard deviation was computed similarly to the real data. Figure 20 shows the whole dataset, where the red lines shown the standard defined limits for the standard deviation.

Using this dataset, the authors then fitted the data three classifiers: a decision tree, a linear Support vector Machine (SVM) and a Gaussian SVM. Results are shown in Table 2. The authors state that

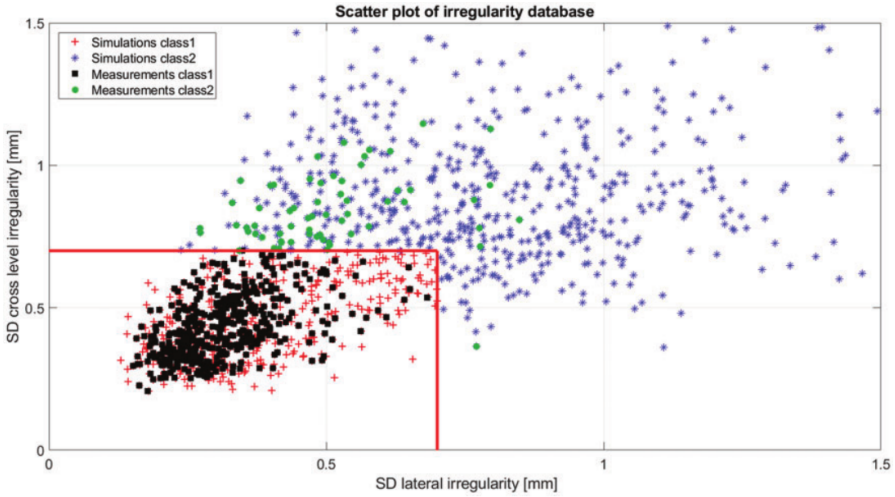


Figure 20 – Dataset distribution of standard deviation of the irregularities with defined safety thresholds. (ROSA et al., 2021)

	Accuracy (%)	Precision (%)	Recall (%)	F_1 score (%)	Kappa
Decision tree	87.6	49.6	92.1	64.4	0.58
Linear SVM	92.9	70.3	71.4	70.9	0.67
Gaussian SVM	92.9	68.1	77.8	72.6	0.69

SVM: support vector machine.

Table 2 – Summary of classifier performances in the testing phase.

3 Methodology

This chapter describes the methodology used to develop a machine learning model that is capable of reconstructing the track irregularities from acceleration data. It is organized as follows: Section 3.1 defines the problem to be solved, Section 3.2 describes the simulation parameters for the velocity and mass correction, Section 3.3 describes the machine learning model that will be used to reconstruct the track irregularities, and Section 3.4 presents a flowchart of the step by step methodology.

3.1 Defining the Problem

In this thesis, the main objective is to develop a machine learning model that can reconstruct track irregularities from acceleration data. The reconstructed profiles can be then compared to the standard limits to support maintenance planning, or, in the case of an off limits irregularity, a defect can be identified and repaired before it causes a serious issue.

To acquire data, a multibody simulation will be created in the software SIMPACK. First, the simulation model will be validated using real data from a section of track geometry measured by the TCG and the corresponding acceleration measurements collected by the IRV. The simulated wagon will replicate the IRV sensor configuration, described in detail in (PIRES et al., 2024), measuring the vertical acceleration on the side frames, the displacement of the secondary suspension and the triaxial carbody acceleration. Once the simulation is validated, a dataset will be generated containing the simulated sensor data for different track irregularities, velocities and for different wagon masses. This search space will be explained in more details in Section 2.8.

With this dataset, a study will be conducted to analyze the effect of the wagon mass and velocity on the acceleration measurements. The goal is to understand how these parameters affect the data and develop a correction method, using similar approaches described in Sections 2.8 and 2.9, to correct the acceleration data.

The corrected acceleration data will then be used to train a machine learning model for reconstructing track irregularities. Model performance will be evaluated using real-world data. Finally, a threshold will be defined on the reconstructed profiles to determine when maintenance is required, enabling condition-based planning based solely on acceleration measurements.

3.2 Velocity and Mass Correction

As stated in Section 3.1 a SIMPACK simulation will run across a search space composed of different velocities, wagon masses and track irregularities. Table 3 summarizes the parameter ranges. Each simulation will test one combination until all possibilities have been explored.

Parameter	**Values**
Velocity (km/h)	5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65
Wagon mass (tons)	0, 51, 102
Track irregularity	No irregularity, FRA 6, FRA 5, FRA 4, Real Irregularity

Table 3 – Search space for the SIMPACK simulation.

The velocity range was defined to cover the typical train velocity operation in the railway. Wagon masses corresponds to the unloaded wagon class, the half loaded wagon and the fully loaded wagon class. The track irregularities were defined to cover a good track condition (FRA 6), a regular condition track (FRA 5) and a bad condition track (FRA 4). The real irregularity corresponds to a measured track geometry that contains some defects, such as a switch and some misalignments. The no irregularity serves as a baseline case to the study.

From this generated data, an approach similar to (BALOUCHI; BEVAN; FORMSTON, 2021) will be used to correct the measurements. In this method, a linear or quadratic regression model will be fitted to the absolute maximum acceleration data across different velocities and different wagon masses. The goal is to develop a correction method that is independent from the track irregularity. The final regression model will be evaluated on real world data.

3.3 Machine Learning Model

The machine learning model to be used in this thesis needs to satisfy a set of requirements. These requirements are:

- The model can not depend on labels, i.e., it needs to be an unsupervised model;
- The model should consider input data as being sequential, i.e., it needs to take into consideration temporal correlations;
- The model needs to be able to handle noisy data;
- The model should be able to be trained on simulated data and perform well on real world data without the need of retraining.

Based on these requirements, the chosen approach is to use a deep learning model that utilizes the methods described in Section 2.2, as it is the most robust approach for handling complex data. This outputs of this model will be then compared to the geometric limits in the Brazilian standard to develop a threshold for maintenance planning based on acceleration data.

3.4 Flowchart

4 Results

5 Conclusion

Bibliography

AGATONOVIC-KUSTRIN, S.; BERESFORD, R. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, v. 22, n. 5, p. 717–727, 2000. ISSN 0731-7085. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0731708599002721>. Cited in page 16.

AL-SELWI, S. M.; HASSAN, M. F.; ABDULKADIR, S. J.; MUNEER, A.; SUMIEA, E. H.; ALQUSHAIBI, A.; RAGAB, M. G. Rnn-lstm: From applications to modeling techniques and beyond—systematic review. *Journal of King Saud University - Computer and Information Sciences*, v. 36, n. 5, p. 102068, 2024. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157824001575>. Cited in page 23.

ALZUBAIDI, L.; ZHANG, J.; HUMAIDI, A. J.; AL-DUJAILI, A.; DUAN, Y.; AL-SHAMMA, O.; SANTAMARíA, J.; FADHEL, M. A.; AL-AMIDIE, M.; FARHAN, L. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, v. 8, n. 1, Mar 2021. Cited 3 times in pages 16, 19, and 21.

ATTOH-OKINE, N. O. *Big data and differential privacy: analysis strategies for railway track engineering*. [S.l.]: John Wiley & Sons, 2017. Cited 2 times in pages 12 and 28.

BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. Disponível em: <https://arxiv.org/abs/1409.0473>. Cited in page 24.

BALOUCHI, F.; BEVAN, A.; FORMSTON, R. Development of railway track condition monitoring from multi-train in-service vehicles. *Vehicle System Dynamics*, Taylor & Francis, v. 59, n. 9, p. 1397–1417, 2021. Disponível em: <https://doi.org/10.1080/00423114.2020.1755045>. Cited 4 times in pages 6, 33, 34, and 42.

BERAWI, A. R. B. Improving railway track maintenance using power spectral density (psd). In: . [s.n.], 2013. Disponível em: <https://api.semanticscholar.org/CorpusID:111822997>. Cited in page 29.

BLáZQUEZ-GARCíA, A.; CONDE, A.; MORI, U.; LOZANO, J. A. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, v. 54, n. 3, p. 1–33, Apr 2021. Cited 4 times in pages 6, 12, 13, and 14.

BRAUWERS, G.; FRASINCAR, F. A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 35, n. 4, p. 3279–3298, abr. 2023. ISSN 2326-3865. Disponível em: <http://dx.doi.org/10.1109/TKDE.2021.3126456>. Cited 2 times in pages 24 and 25.

CHOI, K.; YI, J.; PARK, C.; YOON, S. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, v. 9, p. 120043–120065, 2021. Cited in page 15.

CILIMKOVIC, M. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, v. 15, n. 1, p. 18, 2015. Cited in page 18.

DARBAN, Z. Z.; WEBB, G. I.; PAN, S.; AGGARWAL, C.; SALEHI, M. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, v. 57, n. 1, p. 1–42, Oct 2024. Cited 4 times in pages 12, 13, 15, and 27.

De Rosa, A.; ALFI, S.; BRUNI, S. Estimation of lateral and cross alignment in a railway track based on vehicle dynamics measurements. *Mechanical Systems and Signal Processing*, v. 116, p. 606–623, 2019. ISSN 0888-3270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327018303790>. Cited 2 times in pages 7 and 36.

FANG, W.; CHEN, Y.; XUE, Q. Survey on research of rnn-based spatio-temporal sequence prediction algorithms. *Journal on Big Data*, v. 3, p. 97–110, 01 2021. Cited 2 times in pages 21 and 22.

GAO, C.; YAN, J.; ZHOU, S.; VARSHNEY, P. K.; LIU, H. Long short-term memory-based deep recurrent neural networks for target tracking. *Information Sciences*, v. 502, p. 279–296, 2019. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025519305778>. Cited in page 22.

GHIASI, R.; LESTOILLE, N.; DIAINE, C.; MALEKJAFARIAN, A. Unsupervised domain adaptation for drive-by condition monitoring of multiple railway tracks. *Engineering Applications of Artificial Intelligence*, v. 139, p. 109516, 2025. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197624016749>. Cited 2 times in pages 30 and 31.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256. Disponível em: <https://proceedings.mlr.press/v9/glorot10a.html>. Cited in page 22.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MITP, 2018. Cited 2 times in pages 20 and 21.

GRUBBS, F. E. Procedures for detecting outlying observations in samples. *Technometrics*, v. 11, n. 1, p. 1, Feb 1969. Cited in page 13.

GURNEY, K. *Introduction to neural networks Kevin Gurney*. [S.l.]: Taylor Francis, 1997. Cited in page 17.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, p. 1735–1780, 11 1997. Cited in page 22.

KAEWUNRUEN[1], S.; REMENNIKOV[1], A. M. Dynamic properties of railway track and its components: a state-of-the-art review. *New research on acoustics*, Nova Publishers, p. 197, 2008. Cited 3 times in pages 6, 28, and 29.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível

em: <https://arxiv.org/abs/1412.6980>. Cited in page 19.

KOLEN, J. F.; KREMER, S. C. Gradient flow in recurrent nets: The difficulty of learning longterm dependencies. In: _____. *A Field Guide to Dynamical Recurrent Networks*. [S.l.: s.n.], 2001. p. 237–243. Cited in page 22.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C.; BOTTOU, L.; WEINBERGER, K. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Cited in page 19.

LACHEKHAB, F.; BENZAOUI, M.; TADJER, S. A.; BENSMAINE, A.; HAMMA, H. Lstm-autoencoder deep learning model for anomaly detection in electric motor. *Energies*, v. 17, n. 10, 2024. ISSN 1996-1073. Disponível em: <https://www.mdpi.com/1996-1073/17/10/2340>. Cited in page 28.

LI, Z.; LIU, F.; YANG, W.; PENG, S.; ZHOU, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, v. 33, n. 12, p. 6999–7019, 2022. Cited in page 19.

LIN, T.; WANG, Y.; LIU, X.; QIU, X. *A Survey of Transformers*. 2021. Disponível em: <https://arxiv.org/abs/2106.04554>. Cited 4 times in pages 6, 25, 26, and 27.

NIU, Z.; ZHONG, G.; YU, H. A review on the attention mechanism of deep learning. *Neurocomputing*, v. 452, p. 48–62, 2021. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S092523122100477X>. Cited 3 times in pages 6, 24, and 25.

ONO, H.; TSUNASHIMA, H.; TAKATA, T.; OGATA, S. Development and operation of a system for diagnosing the condition of regional railways tracks. *Mechanical Engineering Journal*, v. 10, n. 3, p. 22–00239–22–00239, 2023. Cited 6 times in pages 6, 30, 31, 32, 33, and 34.

P., P. *What is a neural network how does it work? ai guide*. Roboflow Blog, 2025. Disponível em: <https://blog.roboflow.com/what-is-a-neural-network/>. Cited 2 times in pages 6 and 17.

PIRES, A.; MENDES, G.; SANTOS, G.; DIAS, A.; SANTOS, A. Indirect identification of wheel rail contact forces of an instrumented heavy haul railway vehicle using machine learning. *Mechanical Systems and Signal Processing*, v. 160, p. 107806, 2021. ISSN 0888-3270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327021002016>. Cited 3 times in pages 8, 29, and 30.

PIRES, A.; VIANA, M.; SCARAMUSSA, L.; SANTOS, G.; RAMOS, P.; SANTOS, A. Measuring vertical track irregularities from instrumented heavy haul railway vehicle data using machine learning. *Engineering Applications of Artificial Intelligence*, v. 127, p. 107191, 2024. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197623013751>. Cited 4 times in pages 30, 31, 34, and 41.

QIAN, N. On the momentum term in gradient descent learning algorithms.

Neural Networks, v. 12, n. 1, p. 145–151, 1999. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608098001166>. Cited in page 19.

RIEBESELL, J. *Janosh Riebesell*. 2022. Disponível em: <https://tikz.net/conv2d/>. Cited 2 times in pages 6 and 20.

ROSA, A. D.; KULKARNI, R.; QAZIZADEH, A.; BERG, M.; GIALLEONARDO, E. D.; FACCHINETTI, A.; BRUNI, S. Monitoring of lateral and cross level track geometry irregularities through onboard vehicle dynamics measurements using machine learning classification algorithms. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, v. 235, n. 1, p. 107–120, 2021. Disponível em: <https://doi.org/10.1177/0954409720906649>. Cited 3 times in pages 7, 39, and 40.

ROTH, D. *Upenn*. 2016. Disponível em: <https://www.cis.upenn.edu/~danroth/Teaching/CS446-17/LectureNotesNew/neuralnet1/main.pdf>. Cited 2 times in pages 17 and 18.

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Cited in page 19.

SAMARIYA, D.; THAKKAR, A. A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, Nov 2021. Cited 2 times in pages 12 and 15.

SANSINENA, A.; RODRíGUEZ-ARANA, B.; ARRIZABALAGA, S. A systematic review of acceleration-based estimation of railway track quality. *Vehicle System Dynamics*, Taylor & Francis, v. 0, n. 0, p. 1–28, 2025. Disponível em: <https://doi.org/10.1080/00423114.2025.2483972>. Cited 3 times in pages 29, 36, and 37.

SCHMIDT, R. M. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. Disponível em: <https://arxiv.org/abs/1912.05911>. Cited 4 times in pages 6, 21, 22, and 23.

TIELEMAN, T. *Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude*. 2012. 26 p. Disponível em: <https://cir.nii.ac.jp/crid/1370017282431050757>. Cited in page 19.

TSAI, C.-W.; LAI, C.-F.; CHAO, H.-C.; VASILAKOS, A. V. Big data analytics: A survey. *Journal of Big Data*, v. 2, n. 1, Oct 2015. Cited in page 12.

TSUNASHIMA, H. Condition monitoring of railway tracks from car-body vibration using a machine learning technique. *Applied Sciences*, v. 9, n. 13, 2019. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/9/13/2734>. Cited 2 times in pages 30 and 31.

TSUNASHIMA, H.; YAGURA, N. Railway track irregularity estimation using car body vibration: A data-driven approach for regional railway. *Vibration*, v. 7, n. 4, p. 928–948, 2024. ISSN 2571-631X. Disponível em: <https://www.mdpi.com/2571-631X/7/4/49>. Cited 5 times in pages 7, 36, 37, 38, and 39.

TULI, S.; CASALE, G.; JENNINGS, N. R. *TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data*. 2022. Disponível em: <https://arxiv.org/abs/2201.07284>. Cited in page 28.

VAART, W. Verschoof-van der; LAMBERS, K. Learning to look at lidar: The use of r-cnn in the automated detection of archaeological objects in lidar data from the netherlands. *Journal of Computer Applications in Archaeology*, v. 2, p. 31–40, 03 2019. Cited 2 times in pages 6 and 21.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. *Attention Is All You Need*. 2023. Disponível em: <https://arxiv.org/abs/1706.03762>. Cited 3 times in pages 24, 25, and 26.

WEAVER, M. *Perceptron 101: The Building Blocks of a neural network*. AI Mind, 2024. Disponível em: <https://pub.aimind.so/perceptron-101-the-building-blocks-of-a-neural-network-496f6b9b3826>. Cited 2 times in pages 6 and 17.

WEI, X.; ZHANG, L.; YANG, H.-Q.; ZHANG, L.; YAO, Y.-P. Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks. *Geoscience Frontiers*, v. 12, n. 1, p. 453–467, 2021. ISSN 1674-9871. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1674987120301134>. Cited 2 times in pages 6 and 23.

WEN, Q.; ZHOU, T.; ZHANG, C.; CHEN, W.; MA, Z.; YAN, J.; SUN, L. *Transformers in Time Series: A Survey*. 2023. Disponível em: <https://arxiv.org/abs/2202.07125>. Cited in page 25.

XIE, T.; XU, Q.; JIANG, C. Anomaly detection for multivariate times series through the multi-scale convolutional recurrent variational autoencoder. *Expert Systems with Applications*, v. 231, p. 120725, 2023. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417423012277>. Cited in page 28.

Appendix

APPENDIX A – Deriving the Backpropagationg Rule