Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)
Faculty of Computer Science and Business Information Systems

Bachelor Thesis

Comparison and Evaluation of Concept Drift Handling in Anomaly Detection of Streaming Data in Formula Student Racing Cars

submitted to the Technical University of Applied Sciences Würzburg-Schweinfurt in the Faculty of Computer Science and Business Information Systems to complete a course of studies in Computer Science

N. N.

Submitted on: 14.10.2024

Initial examiner: Prof. Dr. rer. nat. habil. Frank-Michael Schleif
Secondary examiner: M.Sc. Manuel Röder

Übersicht

Der Formula Student Wettbewerb fordert Hochschulteams auf der ganzen Welt heraus, einsitzige Rennwagen im Stil der Formel 1 zu entwerfen, zu bauen und in Rennen gegeneinander anzutreten. Technische Spitzenleistungen, Projektmanagement und Innovation stehen im Vordergrund dieses Wettbewerbs. In einem derart intensiven und dynamischen Umfeld ist die Zuverlässigkeit und die Leistung der Fahrzeuge von entscheidender Bedeutung, insbesondere bei Live-Rennveranstaltungen, bei denen die Fähigkeit, Sensorfehler so schnell wie möglich zu erkennen und auszubessern, erhebliche Probleme wie Disqualifikationen oder sogar Unfälle verhindern kann. Herkömmliche, manuelle Inspektionsmethoden sind oft ineffizient und anfällig für menschliche Fehler, insbesondere angesichts des häufigen Wechsels der Fahrzeugkomponenten, während der Entwicklungszyklen. Diese Arbeit konzentriert sich auf die Entwicklung und Evaluierung eines Systems zur Erkennung von Sensor Anomalien, dass speziell auf Formula Student-Rennwagen zugeschnitten ist, um die Fehlersuche für sensorbezogene Probleme zu verbessern. Das System soll Anomalien in den Sensordaten so schnell wie möglich erkennen und so den Zeitaufwand für die Erkennung, Identifizierung und Wartung reduzieren.

Abstract

The Formula Student competition challenges university teams around the globe to design, build and race against each other in single-seated Formula style cars, emphasizing engineering excellence, project management, and innovation. In such a competitive and dynamic environment, vehicle reliability and performance are critical, particularly during live racing events, where the ability to detect and diagnose sensor faults as fast as possible can prevent significant issues, such as disqualification or even crashes. Traditional, manual inspection methods are often inefficient and prone to human error, especially given the high frequency of changing components during live development cycles. This thesis focuses on developing and evaluating an anomaly detection system tailored to Formula Student racing cars, aimed at improving the fault diagnosis process for sensor-related issues. By addressing the key challenges of dynamic environments and non-stationary data, the system is designed to detect anomalies in the sensor data as soon as possible, thereby reducing the time needed for detection, identification, and maintenance.

Contents

1 Introduction

The Formula Student competition is a prestigious annual event where university teams from around the globe engage in the design, construction, and racing of single-seated formula style cars. Participating teams operate under the assumption that they are a manufacturer developing a prototype for evaluation in potential production. The competition aims to foster innovation and develop skills in engineering, project management, and teamwork among students. The teams are judged based on several dynamic and static events that evaluate the performance and efficiency of the vehicles. However success in the competition is not solely determined by having the fastest car but rather by presenting the best overall package consisting of construction, performance, financial planning, and sales strategy. This means participants have to tackle the various challenges associated with the development of a functioning prototype as well as considering the economic aspects associated with the automotive industry like cost planning and market value [19].

In practice this translates to the car needing to exhibit excellent driving characteristics, such as acceleration, braking and handling, while also being reliable, cost-effective and aesthetically appealing. The cars are first evaluated by a jury of experts from the motorsport, automotive, and supplier industries, where each team's car and sales plan is evaluated based on construction quality, cost planning, and sales presentation. The functional aspects of the vehicles are tested on the track in various dynamic performance events (see Figure 1.1), where students demonstrate how well their cars perform in a real-world environment [20]. Each of these events tests for specific features of the vehicle. Most of the dynamic disciplines have to be performed both manually by a human driver as well as by an autonomous driving system under the "Driverless" category of events. Each team is awarded a certain number of points based on how well their vehicle performs in each discipline and the team with the most overall points is declared the winner.

Designing a competitive car that meets all the requirements of the Formula Student competition is already a demanding task, made even more difficult by the limited time available to student volunteers, who must balance this with their regular academic commitments. During the off-season, teams dedicate countless hours to designing, testing, and optimizing the various components of their vehicles. This preparation is crucial, given the complexity and precision needed to build a competitive Formula Student racing car. The complex interactions between mechanical, electrical, and software components must be carefully considered to ensure both full functionality and high performance.

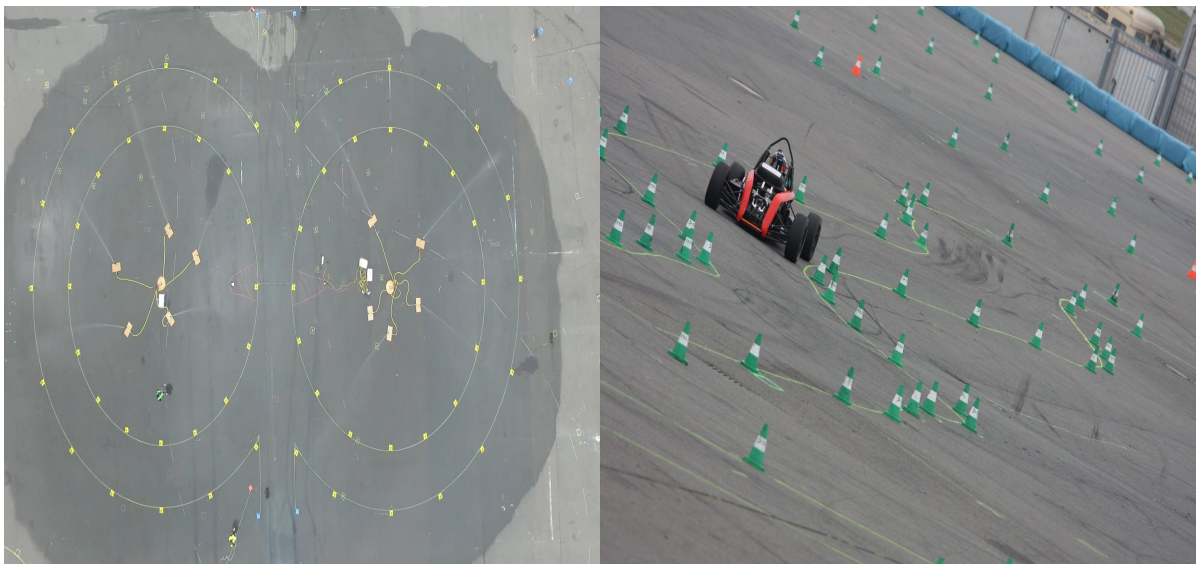


Figure 1.1: The tracks for two dynamic disciplines: Skidpad (left) and Autocross (right). The exact layout of the track for some disciplines changes annually [1, 63]

Despite rigorous testing and sometimes because of of it, some components may be installed incorrectly, start to degrade prematurely or fail to function from the very start. Detecting these issues as soon as possible is critical, as they can significantly impact the car's performance during competitions or even lead to devastating crashes. Malfunctioning sensors or sensors that record unexpected behaviour because of failing components are of particular importance here. One of the biggest problems during operation of the vehicle is the fact that faulty sensors or components often remain undetected until they cause noticeable issues. These issues may range from suboptimal performance in the best case and disqualification or a crash in the worst case scenario. Furthermore once a malfunction is actually recognized the cause needs to be identified as quickly as possible.

Detecting and identifying unexpected behaviour through data analysis is a challenging but important task that helps ensure system reliability across various industries. This unexpected behaviour is commonly referred to as an *anomaly* or *outlier*. Although these terms are often used interchangeably, they technically have slightly different definitions. A broad definition of an outlier is given by Hawkins as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" in [29]. The term outlier typically refers to data points that differ in their statistical properties when compared to the rest of the data, while anomalies usually describe instances or patterns that display unexpected behaviour in regard to their specific context [61]. However whether a data point should be classified as an anomaly or not is always highly dependent on the application context in which they occur.

Nowadays, anomaly detection techniques are employed extensively across various domains, including social media [5], early-onset cancer detection in healthcare [24], network intrusion detection in cybersecurity [39], fault diagnosis in industrial machinery [45] and even spacecrafts [22].

In the case of a Formula Student racing car, traditional, manual inspection methods are time-consuming and prone to human error, making them less effective, especially in the stressful conditions of a live competition. The objective of this thesis is to increase the efficiency and accuracy of the fault diagnosis process involved in the development and operation of Formula Student racing cars. Specifically, this thesis aims to enhance the safety and reliability of Formula Student vehicles by developing an anomaly detection system for sensor-related issues, capable of adjusting to the frequent changes in components and competition disciplines. The goal of the detection system is to detect faults as soon as they occur, thereby decreasing the time necessary for detection, identification and maintenance of malfunctioning components.

To this end, the following research questions are addressed in this thesis in no particular order:

- **Q1:** Which specific sensors in Formula Student cars are prone to malfunctions and what types of anomalies commonly occur?
- **Q2:** Which anomaly detection techniques are suitable for such an anomaly detection system, considering the types of common anomalies and the nature of the sensor data?
- **Q3:** How do different data characteristics impact the performance of the developed anomaly detection system?
- **Q4:** What potential weaknesses need to be addressed to enhance the system's real-world applicability and how can they be improved?

This thesis is structured as follows: The second chapter presents the preliminaries required for understanding anomaly detection in time series and streaming data. It introduces the different types of anomalies, the challenges involved in detecting them and explains the types of data relevant for this thesis. Chapter three examines the sensor data collected from Formula Student racing cars, focusing on the relevant sensors and their characteristics. It also describes common anomalies observed in the data. The fourth chapter reviews related work in the field of anomaly detection with a focus on techniques commonly utilized for time series data. Chapter five details the implementation of the proposed anomaly detection system and explains the rationale behind the chosen approach. The sixth chapter outlines the experimental setup used to evaluate the performance of the developed anomaly detection system. The seventh chapter presents the results and evaluation of the detection system and chapter eight summarizes the key

findings of the thesis.

2 Preliminaries

This chapter introduces the problem of anomaly detection in time series and streaming data, including the categorization of anomalies and the challenges associated with their detection. It also explores the relevant input data types for this thesis and highlights its key characteristics important for the task of anomaly detection.

2.1 Types of Anomalies

As mentioned in chapter 1, anomalies are data points or sequences that deviate significantly from the expected or normal behavior in a dataset. Depending on their nature, anomalies are typically classified into the following three categories [13, 14, 25, 57]:

2.1.1 Point Anomalies:

Point anomalies are the simplest type of anomaly. They occur when an individual instance deviates significantly enough from the rest of the data to be deemed anomalous. An example can be seen seen in Figure 2.1(a). A common source of point anomalies, particularly relevant to this thesis, is noise introduced by faulty sensors or measurement errors [16].

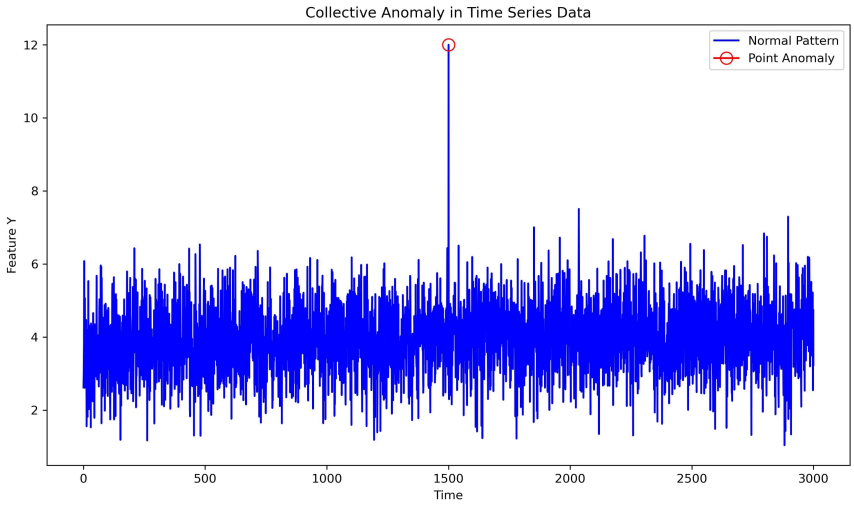
2.1.2 Collective Anomalies:

Collective anomalies describe a collective of related data instances that may appear normal by themselves but when viewed as a group, display anomalous behaviour with regards to the rest of the dataset. It is not important whether the individual data instances are anomalous or not, but rather if their occurrence as a collective is indicative of an anomaly. A straightforward example of a collective anomaly can be seen in Figure 2.1(b), where each individual observation is not anomalous individually but their consistent linearly increasing occurrence is.

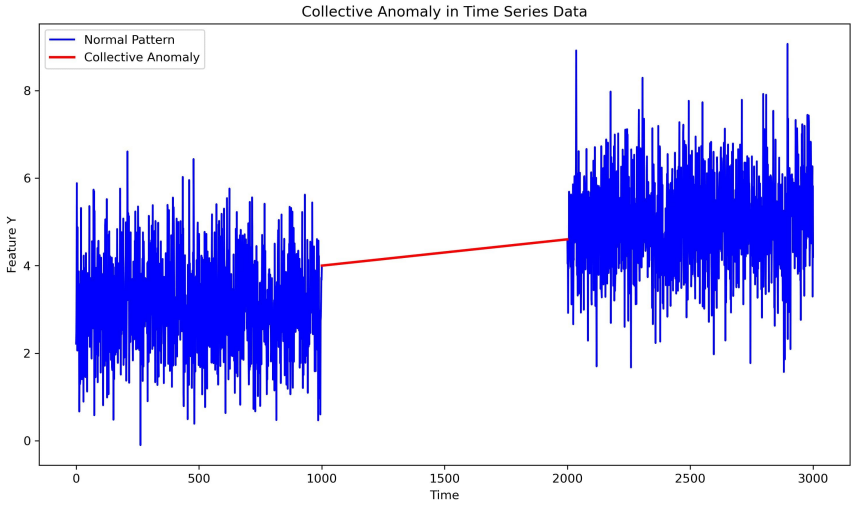
2.1.3 Contextual Anomalies:

A *contextual anomaly* occurs when a data instance appears anomalous only within a specific context around it. This context is determined by the structure of the data itself and in order to determine whether or not a data instance is anomalous in a specific context, both behavioural and contextual attributes have to be considered. As the name implies, contextual attributes determine the context in which a data instance occurs. For example, the most commonly used contextual attribute for time series and spatial datasets are time and space respectively. Behavioural attributes on the other hand describe the non-context related properties of a data instance and are often its primary characteristics [13]. For example, in Figure 2.1(c), which illustrates the average hourly electricity usage of households in a city, the behavioural attribute consists of the average electricity usage for a given hour, while the contextual attribute is the time at which data instances or patterns occur. The contextual anomaly in the illustration only becomes apparent when both features are taken into account. Note that power usage spikes regularly in the morning and evening but decreases during all other hours of the day. Thus the unusual spike around midnight (shaded in red) represents a contextual anomaly, even though such a spike would be considered normal during morning or evening hours.

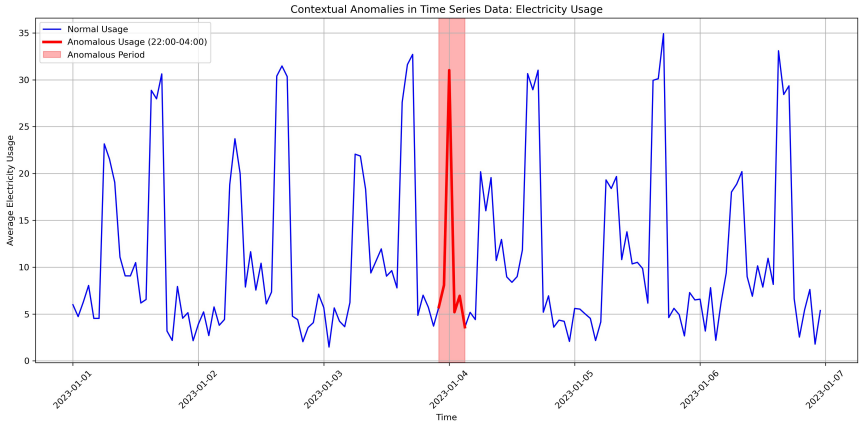
While these categories are not mutually exclusive, some restrictions apply. Since the relationships between individual data points are crucial for collective anomalies, they can only occur in data where instances are inherently related to each other, such as in time series or spatial data. Contextual anomalies, on the other hand, can only occur if such a context exists or is readily available, while point anomalies can occur in any type of data. Additionally, point anomalies and collective anomalies can be transformed into contextual anomalies if they are examined with respect to their context [13, 14]. [25] gives a simple example of this by adding the corresponding month to a set of temperature measurements as an additional feature and thus establishing a temporal context in the dataset. However, they also state that it may be necessary to derive one or multiple new features from the existing data in order to achieve such transformations in more complex scenarios.



(a) Point anomaly (red)



(b) Collective anomaly (red)



(c) Contextual anomaly (red)

Figure 2.1: Examples of different types of anomalies in simulated time series data

2.2 Anomalies, Noise and Novelties

Deviations from the expected behaviour are not always caused by anomalies and can also result from various other factors, such as noise introduced during data collection or previously unknown but legitimate events known as novelties.

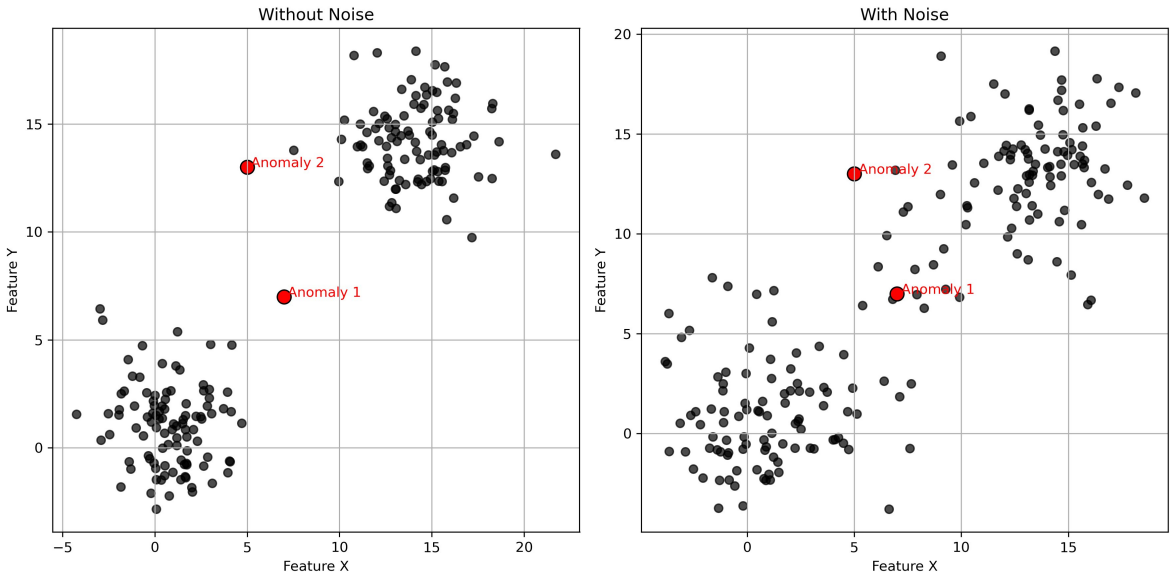


Figure 2.2: Illustration of the obscuring effect excessive noise can have on anomalies

Noise is generally an unavoidable part of any real-world measurement but can also be present in purely digital data. It is often a result of measurement errors, limited precision of measurement tools, environmental conditions or other issues during data collection or processing, which lead to random or irrelevant variations in the data. Depending on the amount of noise present in a dataset, it may be necessary to apply noise removal techniques or other measures, as high amounts of noise in a system have been shown to seriously impact the performance of some anomaly detection techniques [26]. For example, a high amount of noise can obscure true anomalies by making it difficult to distinguish them from normal data, while also causing noisy data points with large enough deviations to be mistakenly classified as anomalies. Additionally, applying noise removal techniques in conjunction with anomaly detection may lead to anomalies being mistakenly classified as noise and subsequently removed [3, 13]. An illustration of this issue can be seen in Figure 2.2. However, it should be noted, that depending on the context, the noisy data points shown in Figure 2.2, may simply be considered anomalous themselves if they are enough from the normal data clusters.

Novelties refer to new, previously unseen patterns that emerge in a dataset. Unlike anomalies, novelties do not necessarily indicate an error or a problem. Instead they

represent a change or shift in the underlying data distribution, whereas anomalies are considered to be generated by an entirely different distribution or mechanism. Novelties are therefore typically incorporated into the notion of the expected behaviour of the dataset in order to match this change [13]. Imagine, for example, a cybersecurity system monitoring network traffic within a company. If a previously unconnected department is added to the network, it may lead to an increase in traffic and altered usage patterns that were not present in the training data. While this change does not signify an anomaly, like a malicious attack, it requires the system to adapt to accommodate to the new normal.

The fields of novelty detection and anomaly detection are strongly related, but they differ in their notion of normal data and in how they handle unexpected behaviour [49]. The key difference, as noted by [13], lies in the "interestingness" of anomalies, meaning their real-life relevance in the application context. In the example of the aforementioned cybersecurity system, an anomaly detection system aims to determine whether the sudden spike in network traffic represents a security threat, while a novelty detection system aims to incorporate the change into its understanding of the expected behaviour. While a detailed exploration of this topic is beyond the scope of this thesis, further insights can be found in the works of [43, 49, 69].

2.3 Challenges in Anomaly Detection

The anomaly detection problem might seem simple at first glance. Since anomalies are defined as regions or instances of unexpected behaviour with regards to the rest of the data, it may seem like a straightforward approach to try to derive a notion of normal behaviour from the data and to categorise all instances that do not conform to this notion of normality as anomalies [13]. However, there are several factors that make this task more challenging than it may seem initially, as outlined in the following paragraphs [3, 13, 53].

Firstly, defining an accurate notion of normal behaviour, that covers all possible value ranges for all possible situations is very difficult. In many domains the normal behaviour of data is influenced by a number of external and context-related variables, such as environmental factors, the condition of hardware components, socioeconomic trends or a certain sequence of events. In the field of network intrusion for example, a user accessing a certain part of a network may seem completely normal on its own, however, if multiple failed login attempts occurred before and are followed by a large data transfer, this may signify an anomalous event. A change in any of these external influences or context-related factors, can change or shift what is considered normal behaviour. This can cause the normal behaviour to shift and evolve over time, which erodes the validity of the current notion of expected behaviour. Additionally, the border between normal and

anomalous behavior is completely rigid but rather continuous. As a result, observations near this boundary can be difficult to classify with certainty, as some anomalous points may appear normal and vice versa.

The similarity between noise and anomalies, as discussed in section 2.2, can complicate this issue even further. Furthermore the definition of a anomalies is heavily application- and domain-dependant and necessitates a thorough understanding of the application domain. Small deviations in network traffic may not be considered anomalous, while a similar increase in a generator's vibration could indicate a larger issue. This limits the generalizability of detection techniques, meaning that methods effective in one domain may be ineffective in others.

Another significant challenge is the lack of labeled data necessary for training and validation of many detection models. In many cases, labeled data either does not exist due to the inherent rarity of anomalies, because of a lack of knowledge about their properties or because labeled datasets are too expensive to create, as the labeling process is time-intensive and often requires to be done manually, by a human expert. Furthermore, it is often challenging to provide enough labeled data to cover all possible ranges of normal and anomalous behaviour.

All of these factors combined make anomaly detection a particularly difficult task, which is why most existing methods focus on specific use cases rather than universal solutions. The effectiveness of a particular detection technique is influenced by a variety of domain-specific factors, such as the purpose of the detection system, the constraints of application-domain, the types of anomalies to be detected, the characteristics of the data and the amount of available training data, especially labeled data [13, 53].

The works of [12, 17, 25, 61] have performed extensive benchmarks and evaluations of several established anomaly detection techniques for time series and streaming data, in order to provide guidelines for choosing the most suitable method depending on the specific use case. However, it should be noted that the lack of generalizability remains an active field of study.

2.4 Types of Input Data

As mentioned previously, the type of input data is one of the key aspects that determine the effectiveness of a anomaly detection techniques in a given scenario. Datasets are collections of data instances, where each instance consists of either a single attribute, referred to as *univariate*, or multiple attributes, referred to as *multivariate*. Multivariate datasets can be thought of as sets of multidimensional vectors or matrices, where each element in them represents the value of an individual attribute. The attributes them-

selves can be *binary categorical* or *continuous* in nature. It is possible for a multivariate instance to consist of multiple attributes of different types, however that pattern has to be consistent across all instances in the dataset. The relationship between data instances is also important, as instances can be either dependent or independent of each other. Examples for the former case include *sequence data*, *spatial data* and *graph data*. In sequence data, such as voice recordings, time series or text data, instances are linearly ordered. On the other hand, spatial data, such as satellite imagery or clime data, contains individual instances that are spatially related to their neighbouring instances. *Spatio-temporal* data is a mixture type of spatial and sequence data, where instances contain both temporal and spatial components [13].

2.4.1 Time Series Data

A time series is generated by continuous measurements over time and is defined as a collection of observations indexed by the timestamp of each observation. More explicitly, a time series is defined as:

$$X = \{v_1 = (x_1, t_1), v_2 = (x_2, t_2), \ldots, v_n = (x_n, t_n)\} \tag{2.1}$$

with $v_i = (x_i, t_i)$ being a data point x_i observed at timestamp t_i and n being the total amount of observations. Observations can be taken at regular or irregular intervals in which case the time series are referred to as evenly or unevenly spaced. Due to the continuous nature of observations in time series data, consecutive observations usually do not change abruptly and sudden shifts in the sequence can be indicative of anomalous processes. This concept, referred to as *temporal continuity* or *temporal dependence* by [3] and [14] respectively, is a valuable property for detecting anomalies in time series data. This property also implies that anomalies in time series data typically occur either as contextual or collective anomalies, as their temporal relationship with the adjacent instances or subsequences must always be taken into account [3, 27].

$$F_x(x^{1+\tau}, \ldots, x^{t+\tau}) = F_x(x^1, \ldots, x^t) \tag{2.2}$$

However, as mentioned in section 2.2, unexpected behaviour is not solely caused by anomalies. A change in the underlying distribution that generates the data is synonymous with a change in its statistical properties, such as mean, variance, or covariance. Whether or not the data distribution of a time series remains constant over time is described by the concept of *stationarity*. [14] defines a condition for strong stationarity in continuous stochastic processes $X = \{x_t\}_{t \in T \subset \mathbb{Z}^+}$, such as time series, as Equation 2.2 for any $\tau \in \mathbb{N}$. In simpler terms, a time series is stationary if the joint distribution of any

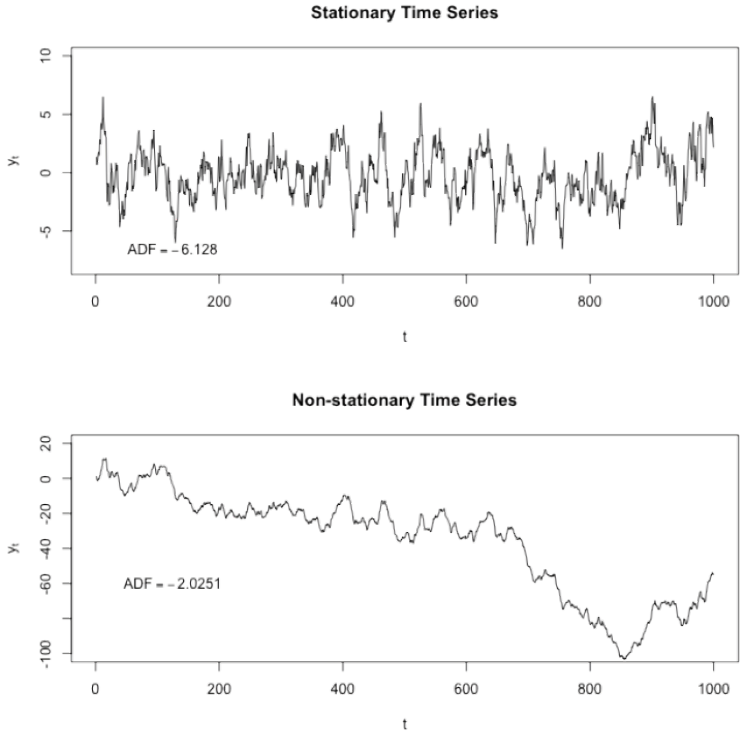


Figure 2.3: Plots of two simulated stationary and nonstationary time series [65]

set of observations is independent of when they are observed, meaning the underlying process generating the data remains unchanged. This implies that any subsets of data of a time series (from different periods of time) maintain similar statistical properties, as illustrated in Figure 2.3. Many anomaly detection techniques rely on their input data being stationary, as this makes time series easier to model and predict. However, this is often not the case for real-world scenarios, as they are influenced by a variety of factors that can cause the data distribution to evolve dynamically. If the applied detection techniques do not adapt to these changes, they become inaccurate and unreliable [14, 16].

Nonstationarity can have multiple sources, one of the most well-known being a phenomenon called *concept drift* [62]. Concept drift refers to persistent change in the values and trends of data over time, leading to models that depend on outdated historical data to become inaccurate. Concept drift can occur due to a variety of factors and may cause the data to change in sudden, gradual or periodically reoccurring ways [32]. In the case of Formula Student cars for example, the degradation of some components over time may cause gradual concept drift through the slow change in their behaviour that is recorded by the sensors. Another common source of nonstationarity is *seasonality*. Seasonality refers to recurring patterns that occur periodically and last for a fixed amount of time. These patterns alter the statistical properties, such as mean and variance, of the time

periods in which they appear, compared to those in which they do not, thus causing nonstationarity. Common examples of seasonality are factors, such as weather, holidays or marketing campaigns [34]. In contrast, *change points* refer to specific points in time, where the statistical properties of a time series change significantly. Unlike concept drift, change points represent abrupt, localized events to the normal state of a system rather than gradual changes over time. Change points are especially relevant in the Formula Student environment, as the frequent component changes during testing, updates to the software or the annually changing race tracks can all create significant shifts in the data [6]. Knowing whether or not a time series is stationary is an important factor for choosing an appropriate detection method. Fortunately, there are many readily available statistical tests and tools for determining if a time series exhibits stationarity, such as implementations of the *Augmented Dickey-Fuller* (ADF) test based on [27, 51] or the *Autocorrelation Function* (ACF) based on [11] among others.

Dimensionality is another important aspect of time series data. It is determined by the number of data attributes captured in each observation. Based on dimensionality, time series are categorized as either univariate or multivariate. As mentioned in section 2.3, the dimensionality of a time series is one of the main data characteristics that influences the complexity of the anomaly detection task and needs to be considered before choosing a method. In a univariate time series, each observation consists of a single data attribute recorded over time, making it relatively simple to establish and analyse the temporal continuity between each observation. In contrast, a multivariate time series consists of multiple attributes, captured at each time step. This generally introduces additional complexity, as both the temporal dependence and the correlations between the different attributes need to be considered, as certain anomalies might only be discernible through specific combinations of attributes. This raises computational costs and usually requires the use of more sophisticated analysis techniques [3, 14].

2.4.2 Streaming Data

While time series data typically involves the processing of entire pre-stored or batched historical datasets, where the entire time series is fully available from the start, *streaming data* refers to sequences of data points that are generated continuously and only arrive incrementally as they are created or transferred. Streaming data shares many of the previously mentioned challenges associated with time series data, such as temporal continuity, the presence of noise and nonstationarity but also provides a number of unique challenges [2, 18, 56].

One of the main difficulties in streaming data problems is the need for real-time analysis, also referred to as *online* processing or analysis. Since data arrives continuously, it must be processed immediately and is typically discarded after being processed. Certain parts of the data can be stored temporarily but the volume of data in streaming environments

is often too large to store permanently. Therefore, there is no access to the entire historical dataset and the availability of past observations is limited. Furthermore there is a frequent lack of labeled data in streaming environments. Streaming data typically stems from environments that generate large and continuous flows of information, such as real-time Internet of Things (IoT) sensors, social media, meteorological data or the stock market. These environments require constant adaptation to incoming data, as well as the efficient use of available computational and memory resources to ensure that the system can keep up with the potentially unbounded data streams [2, 18, 56].

3 Formula Student Sensor Data

This chapter provides an overview over the sensors relevant to the topic of this thesis. This includes an examination of the sensor data for some of its statistical properties, as well as an overview over known sensor-related anomalies in the Formula Student environment.

3.1 Relevant Sensor Data

The data for this thesis was provided by Mainfranken Racing team. It consists of sensor and software data that was collected in several testing runs over the course of the year and the Formula Student Germany 2024 competition in the Skidpad and Autocross disciplines. The data for each run is recorded and stored on a built-in laptop inside the vehicle in the form of Rosbag files and each these files contains 26 datasets of of different data generating sources.

The current iteration of the vehicle developed by the Mainfranken Racing team is equipped with five unique sets of sensors. The sensor data is transmitted over a Controller Area Network (CAN) bus onto the hard-drive of the built-in laptop. CAN is a robust and well established message based protocol and connected to the sensors using USB-to-CAN connector cables in order to minimise the risk of message-loss. While these sensors are not the only data sources in the vehicle's system, they are the only ones that monitor the state of the vehicle and of its hardware components directly, making them particularly relevant to this thesis. The properties measured by these sensors are as follows:

- **Steering angle:** The steering angle represents the tilt of the steering wheel in degrees, with zero degrees indicating the neutral position (no tilt). It is measured using an RFD-4000 rotary position sensor [47] installed directly onto the steering rod, which detects changes the orientation of the magnetic field of the steering rod.

- **Wheel speed:** The wheel speed measures amount of wheel rotations per second. This is done by using type SIEN proximity switches [55] that are attached to the

chassis of the wheels and small magnets attached inside of each wheel. The speed of each wheel is recorded individually by its own sensor.

- **Motor control:** The main engine and the steering motor are controlled and monitored by a C5-E motor control unit (MCU) [46]. The MCU receives instruction about how to adjust the velocity and the steering angle of the vehicle through software running on the laptop and accelerates or decelerates both motors accordingly.

- **IMU:** The *inertial measurement unit* (IMU) is a sensor that measures the linear acceleration, angular velocity and the orientation of the vehicle. These measurements are necessary to determine the position, velocity and orientation of the vehicle when operating in driverless mode. The Mainfranken Racing team uses a high-performance, high-accuracy Ekinox Micro inertial measurement unit [59] (IMU) for this task.

- **GPS:** The GPS is responsible for tracking the absolute position of the vehicle in terms of latitude, longitude and altitude. As GPS is inherently limited in its accuracy, it's data is used in conjunction with nearly all the other sensor data to derive a more accurate estimation of the vehicle's position.

- **LiDAR:** A *light detection and ranging* (LiDAR) is a sensor that creates mass point clouds of its environment by scanning it using the reflections of laser light. The Formula Student vehicle uses this sensor as part of its navigation and path finding system by detecting the position of the orange and blue coloured cones placed on each side of the race track.

A few of the sensors, namely the LiDAR [64], GPS and IMU, record spatio-temporal data, which, as mentioned previously, contains a spatial component in addition to the temporal component. The rest of the sensors record purely temporal data, with each of their measurements being indexed by its timestamp and being correlated with the previous observations. It's important to highlight that the LiDAR, GPS and IMU sensor data is not the primary focus for anomaly detection in this thesis. This is mainly due to the fact that their measurements are not directly tied to any of the hardware components and thus do not provide much insight into the hardware failures this thesis aims to detect. Additionally, the Mainfranken Racing team has not experienced any issues with these sensors in the current or previous seasons, which is in stark contrast to the sensors monitoring the steering angle, wheel speed and the engine as well as the steering motor, which are prone to frequent failures and incorrect installation. While the anomaly detection techniques discussed in chapter 5 are applicable to the IMU, LiDAR and GPS data as well, the primary focus will be on the three more hardware related and failure-prone sensors.

In addition to the sensor data, the system of the vehicle generates a large amount of data directly on the laptop itself through various software processes. These processes are essential to ensure proper functionality of the vehicle, especially during its driverless mode. This mainly includes data resulting from and necessary for *simultaneous localization and mapping* (SLAM), object recognition and velocity estimation algorithms as well as diagnostic and log files. There are two main issues with this software generated data that make it unsuitable for the hardware focused anomaly detection system this thesis aims to develop. Firstly, the calculations involved in the generation of this data are very complex and often utilise the data of multiple sensors in combination. This makes it difficult to interpret any anomalies found in them and to trace them back to their concrete cause. Secondly, it is difficult to determine whether or not an anomaly in the software generated data is the result of calculation errors or a genuine issue with the hardware. It is not guaranteed that an anomaly in the sensor data will cause an anomaly in the software generated data, as there is a chance for it to be obfuscated during the calculation process or to be overshadowed by the data of other sensors that operate normally. Furthermore, as the aforementioned processes have proven to be very reliable, no desire for anomaly detection in this data has been expressed by the Mainfranken Racing team, which is also why there is no available data on typical anomalies in this data.

3.2 Data Characteristics

The initial plan for the anomaly detection system proposed by the Mainfranken Racing team was to continuously process and analyse sensor data in real-time, as it was generated. However, after reviewing the available computational resources of the on-board laptop, which are taken up almost entirely by the aforementioned complex software processes, this approach was reconsidered. Interfering with or slowing down these critical processes can lead the car to drive off the track or crash and has been an issue before. Instead, the decision was made to apply time series anomaly detection techniques to the stored datasets in the breaks in-between runs. Shifting the focus from continuous analysis of streaming data to time series analysis offers two main advantages. First, the analysis can be performed on the on-board CPU during breaks in-between runs, when the essential subroutines are paused and more processing power is available. Alternatively, the analysis can be performed on an external computer after transferring the files. This approach increases the available computational resources and allows the use of a broader range of analytical techniques. Second, having access to the entire dataset of a complete run provides a more comprehensive view of the overall data.

Table 3.1 gives an overview of the most important properties for the analysis of each sensor dataset. Only real datasets of successful runs without abnormal incidents or non-sensor-related issues, such as hitting cones, the vehicle driving off the track for unknown

reasons or mechanical braking failures, were included in the calculation of this data in order to keep the properties as close as possible to normal behaviour. This limits the total amount of available normal datasets to three for both the Autocross and Skidpad disciplines. A selection of the normal sensor data is illustrated in Figure 3.1.

Each data point in the datasets consists of at least one feature and the Unix timestamp of the measurement. The measuring interval of every sensor is consistent over time, meaning the time series of each sensor is evenly spaced. Note that, while the measuring intervals across all sensors are equal at ten milliseconds, the average length of each sensor dataset varies not only from run to run and in between disciplines but also from sensor to sensor. The difference in between runs simply stems from the varying duration the car requires to complete one lap and the recording process being started and stopped manually. However, the difference in between sensors is a result of them starting and stopping their measurements at different times. Specifically, the wheel speed and steering angle sensors start transmitting their data when the vehicle is at a standstill as well as before starting and after finishing the lap. The MCU on the other hand, only does so once the engine is started and stopped. This results in the extended linear segments at the beginning and end of the steering angle and wheel speed datasets, as shown in Figure 3.1, with higher variability occurring in the middle sections. Additionally, it's important to note that the non-zero values transmitted by the wheel speed sensors during this linear period occur because, if the wheel's rotational speed is less than once per second, the sensor continues to transmit its last recorded value instead of zero.

The dimensionality of each dataset is determined by its relevant features, which refers to the attributes that are necessary for accurately modelling the behaviour of the respective time series and are affected by anomalies. All of these relevant features are numerical in nature. Some of the properties in each dataset consist of meta data, redundant timestamps or measurements that are zero or of a constant, unchanging value for the entirety of the observation. Since these measurements hold no informational value for the analysis of their respective time series, they are discarded before processing and are not accounted for in the "Relevant Features" row of Table 3.1. Additionally, the timestamp is not included in this metric as well, as it is present in each dataset.

As mentioned briefly in subsection 2.4.1, the Augmented Dickey-Fuller (ADF) test is a test used to determine whether or not a time series is stationary. Describing the specifics of the test would go beyond the scope of this thesis but in summary, the test checks for the presence of a unit root, which is a feature of a time series that makes it nonstationary. The null hypothesis of this test is that the data has a unit root, meaning that it is nonstationary. It calculates the probability of the null hypothesis applying and expresses it as a single value. A low result (typically below 0.05) indicates that a time series is stationary, while higher values indicate nonstationarity. The implementation used to acquire the results in Table 3.1 can be found in [58]. The ADF test results for each of the wheel speed sensors were averaged across each normal dataset. The entirety of the test results were averaged across all available normal datasets for each of the two

disciplines.

Datasource	Steering Angle	Wheel Speed	Engine	Steering Motor
Relevant Features	1	4	1	1
Measuring Interval	10ms	10ms	10ms	10ms
Average Length Autocross	23695	23695	14273	14273
Average Length Skidpad	23055	23055	5256	5256
Average ADF Autocross	0.06	0.57	0.06	0.07
Average ADF Skidpad	0.23	0.74	0.1	0.16

Table 3.1: Summary of sensor data characteristics. "ADF Autocross" and "ADF Skidpad" refer to the average results of the ADF test for each discipline.

The ADF test results shown in Table 3.1 demonstrate that the wheel speed data is clearly nonstationary across both Autocross and Skidpad datasets. All other sensors show nonstationary behavior as well, with their test results exceeding or bordering on 0.05. Overall the data produced by the Skidpad runs achieved higher ADF results across the board than its counterpart, implying stronger nonstationary behaviour. However, it is important to note that the results were not consistent across the datasets within each discipline. While some datasets achieved values far exceeding the threshold of 0.05, others of the same sensor and discipline did not. This inconsistency can be attributed to the nature of the Formula Student competition, as the layout of the racetracks is not static and and the vehicle's performance varying slightly from run to run. As a result, stationarity can not be assumed and needs to be considered in the selection process of suitable anomaly detection techniques.

In summary, the sensor data of the vehicle can be classified as a mix of multivariate and univariate, evenly spaced, nonstationary time series data.

3.3 Anomalies in Formula Student Sensor Data

The available data on anomalies in the vehicle's sensor data is sparse due to their inherent rarity and the difficulties involved in their detection. The Mainfranken Racing team experienced three major sensor related incidents in the time frame of this thesis and was able to provide the datasets for two of them.

The first incident was caused by a short lag of the laptop of unknown origin, which lead to a communication error between it and the MCU. The MCU did not receive any new information about how to adjust the steering angle in this time frame and the car consequently drove off the track. Unfortunately, this issue is impossible to detect from the sensor data alone, as instead of a discrepancy between the steering motor acceleration

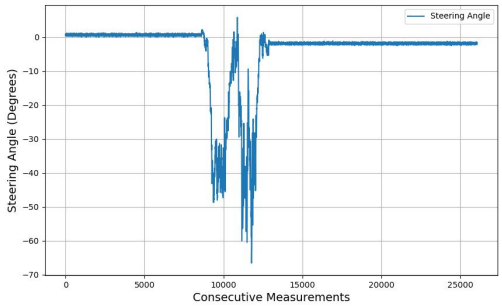
and the resulting change in the steering angle, it was caused by the absence of new instructions to the motor itself. This means that the observations of these sensors show no deviations from their expected behaviour and are thus unable to detect this anomaly. Additionally the transmission and storage of the sensor data to the laptops hard drive was not interrupted, which means that there is no unexpected change in the length of the datasets, which could be used as an indicator for the presence of this lag.

The second incident involved damage to the encoder of the steering motor, which, similarly to the first incident, lead to it not being able to receive new instructions from the MCU properly. During normal operation there is a clear correlation between the steering commands from the MCU and the change in the actual steering angle, as even very small changes in the steering command should result in large changes to the steering angle, as seen in Figure 3.2. However, during this incident the steering motor was unable to receive or interpret these instructions correctly and the steering angle barely changed, effectively locking the steering in place. The recording process for this incident seems to have been stopped prematurely, so only 600 consecutive observations are available.

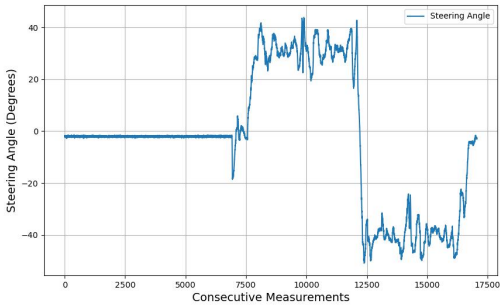
The third type of sensor issue is more common and less severe but still affects performance. It results from incorrect installation of the steering angle sensor on the steering rod, causing random flipping of smaller values (around five degrees) from positive to negative and vice versa. While this is usually not enough to cause the car to leave the track, it degrades performance and is difficult to detect without careful examination of the sensor data. Since no dataset for this anomaly is available, it will be tested using manually altered data.

Based on these incidents, it appears that the steering-related sensors, specifically the steering angle sensor and the MCU, are the most prone to failure. Regarding the types of anomalies discussed in section 2.1, the third issue (random flipping) would be classified as a contextual anomaly if isolated data points are affected. If a sequence of values is flipped repeatedly, it would qualify as both a contextual and collective anomaly instead, as such rapid and opposing variations do not normally occur in the steering angle data, which typically follows a upward or downward trend. However, this trend is less pronounced in the Skidpad discipline, as shown in Figure 3.1. Detecting this anomaly is particularly challenging when the flipping only occurs infrequently and on individual data points.

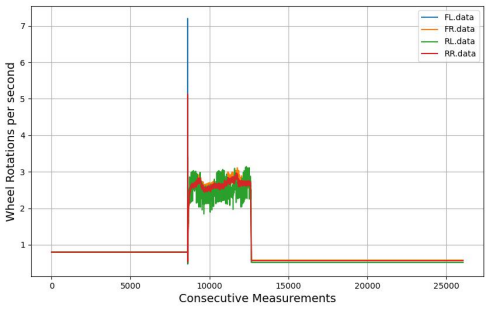
The incident involving the encoder damage falls under the category of a collective, multivariate anomaly, as the relationship between the MCU steering commands and the resulting steering angle data is key in detecting it. The multivariate nature and the small value range of the MCU steering commands make this anomaly especially challenging to detect. Additionally, an interesting pattern emerges in the wheel speed data, as rapid increases in wheel rotations often produce point outliers, as seen in Figure 3.1(c) and Figure 3.1(d). These outliers are not anomalous in nature and are likely caused by brief instances of tire slippage when the vehicle loses traction during sudden accelerations, as they only occur at points of strong increases in wheel rotation.



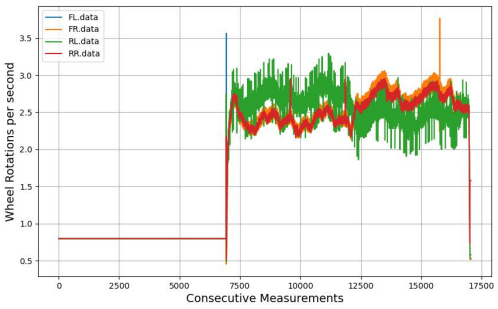
(a) Autocross steering angle



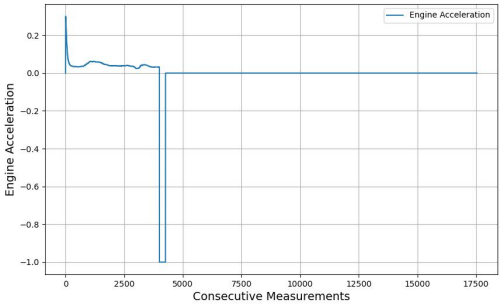
(b) Skidpad steering angle



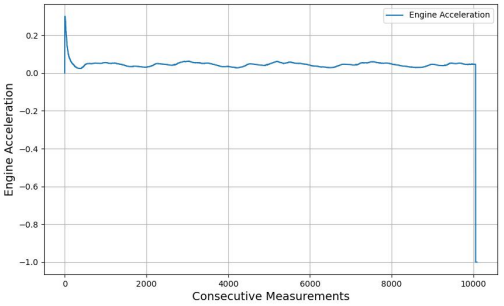
(c) Autocross wheel speed



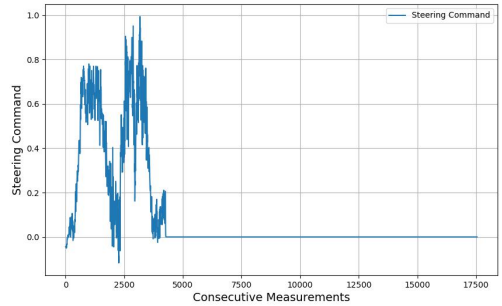
(d) Skidpad wheel speed



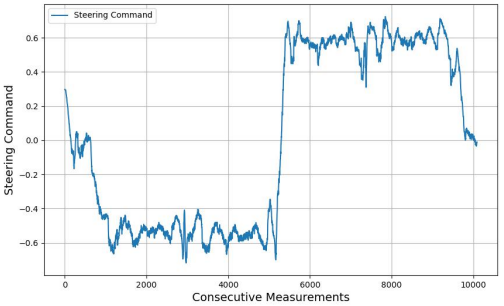
(e) Autocross engine acceleration



(f) Skidpad engine acceleration



(g) Autocross steering command



(h) Skidpad steering command

Figure 3.1: Examples of the time series data produced by each sensor. Each of the features in Figure 3.1(c) and Figure 3.1(d) represents the rotational speed of one of the four wheels

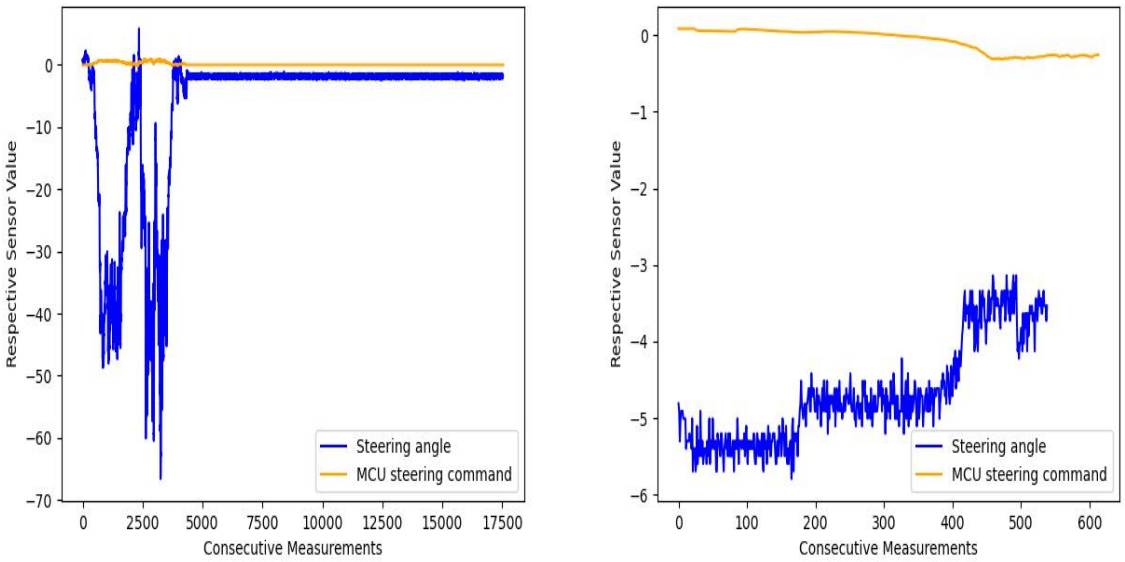


Figure 3.2: Left: Overlapped measurements of steering angle sensor and steering commands during a normal Autocross run. Right: Overlapped measurements of the same sensors during a run with a damaged steering motor encoder.

4 Related work

The detection of anomalies is a task with diverse applications and plays a vital role in many industries. As research on this topic reaches back as far as the 19th century, the field has amassed a vast amount of research over the years, which lead to the development of a many different anomaly detection techniques for a variety of different domains. Thorough overviews over different approaches to anomaly detection in a variety of data can be found in the works of [13] and [31]. Additonally, [7] provides a comprehensive review of widely-known anomaly detection techniques for time series data. However, as mentioned in section 2.3, there is still no universal solution to this problem, as the performance of these techniques is generally at least somewhat dependant on their intended application domain [13]. The following sections provide an introduction and overview of some of the most well-known approaches to the anomaly detection problem. Additionally, a special focus is placed on techniques relevant to the topic of this thesis, that being anomaly detection for time series data with a limited or no amount of labeled data.

4.1 General Distinctions

4.1.1 Availability of Labeled Data

As mentioned in section 2.3, the nature of the available data plays a crucial role in anomaly detection and the effectiveness of different techniques. Just like in traditional machine learning, the availability of labels determines which methods are most applicable. Depending on the availability of labels in a given dataset, anomaly detection problems are typically categorized as either *supervised*, *semi-supervised* or *unsupervised anomaly detection*, as illustrated in Figure 4.1 [13, 25].

Supervised anomaly detection relies on datasets where each instance is fully labeled as either normal or anomalous. These labeled instances serve as a reference for classifying new data points. While similar to traditional supervised machine learning, the key difference is that the normal and anomalous classes are often highly imbalanced. The major limitation of this approach is that it requires prior knowledge of all possible normal and anomalous behaviors, which is unrealistic in many real-world applications due to the

emergence of novelties and previously unseen anomalies in the data. Semi-supervised anomaly detection also relies on labeled datasets but only requires labels for the normal data. This allows a model to estimate the patterns of normal behaviour and subsequently identify anomalies by detecting deviations from these learned patterns. Lastly, unsupervised anomaly detection requires neither training datasets nor labels. Techniques in this category are able differentiate between normal and anomalous instances through the intrinsic properties and the structure of the data itself. These techniques are very flexible, as no labeling process is necessary. Additionally, most semi-supervised techniques can be used with unlabeled data as well by using a part of the unlabeled data as normal training data. However, as noted by both [25] and [13], the examined data needs to contain a much larger proportion of normal to anomalous instances for these techniques to be effective.

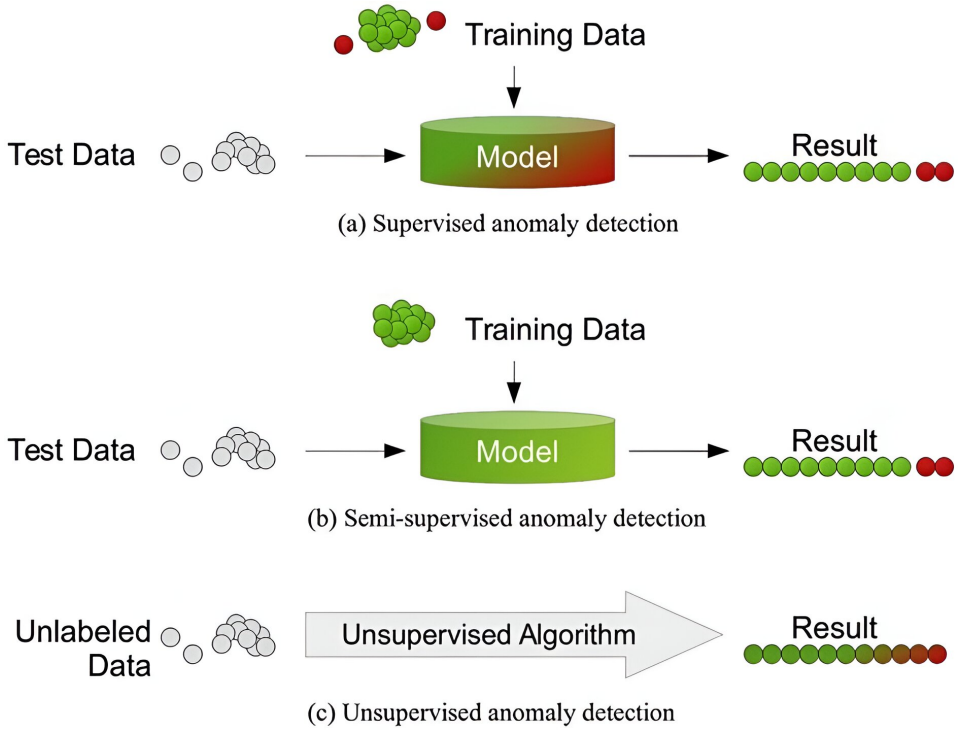


Figure 4.1: Categorization of anomaly detection problems depending on the availability of labels [25]

4.1.2 Types of Detection Results

Anomaly detection techniques typically report their results in the form of either a score or a binary label for each of the examined data points or sequences. In the case of a score, this represents a metric quantifying the degree of anomalous behavior exhibited by a specific instance or sequence. This score can be reviewed manually or compared

against a predefined threshold to determine if it signifies an anomaly. Additionally, this method allows possible outliers to be ranked depending on the height of their score, where only the points with the highest scores may be considered to actually anomalous. With binary labels, the technique classifies each instance or sequence as either anomalous or normal, offering a straightforward indication of whether or not an anomaly has occurred. Most techniques that return binary labels use anomaly scores with the aforementioned threshold to achieve this classification. While this reporting method does not allow a direct way of examining the degree of anomalous behaviour, it can be influenced through parameter changes [3, 13].

4.2 Statistical Approaches

Statistical anomaly detection techniques assume normal data is generated by an underlying statistical distribution with certain statistical properties, while anomalous data is not [13]. In order to differentiate normal data points from anomalous ones, a statistical model is fitted onto the normal data and the probability of a specific data point being generated by this model is used to determine whether to classify it as an anomaly. Statistical anomaly detection techniques are categorised into parametric and non-parametric techniques depending on the assumptions they make about the underlying data distribution [13, 15, 31].

Parametric techniques assume that the normal data is generated by a known parametric distribution $f(x, \Theta)$, such as a Gaussian distribution, with x describing an observation and Θ being the parameters of said distribution, which are estimated from the normal data [13]. One of the simplest parametric techniques of this type is often referred to as the *3σ* rule, commonly used in quality control of products [54]. Here the mean μ and the standard deviation σ are estimated from the available data, using techniques like *Maximum Likelihood Estimation* (MLE). All observations that fall outside the range of $\mu \pm 3\sigma$, which encompasses 99.7% of all observations for Gaussian distributions, are deemed to be outliers. However, as noted by [15], this approach is dependent on the mean and standard deviation being consistent over the entirety of the distribution, meaning the data being stationary. Additionally both parameters are sensitive to the presence of outliers. To address the latter issue, more robust methods like the *Median Absolute Deviation* (MAD), are commonly used instead. It is calculated as:

$$\text{MAD} = c \cdot \text{median}\left(|x_i - \text{median}(x)|\right), \quad i = 1, \ldots, n \tag{4.1}$$

where n is the amount of observations in a dataset, *median(x)* is the median of these observations and c is a constant determined by the assumed distribution [28, 50]. Some

distance based approaches, such as the k-nearest-neighbour method also fall into this category, as they require distribution specific parameters to be set beforehand [31].

Auto Regressive Integrated Moving Average (ARIMA) [9] models are a popular unsupervised, regression-based approach for anomaly detection in non-stationary, univariate time series data [18]. An ARIMA model consists of three components, an auto-regressive (AR) model that predicts future values based on the preceding ones, an integrated component (I) that applies differencing to the time series to make it stationary if necessary and a moving average (MA) model that tracks the prediction errors (residuals) of past values. By utilizing both past observations and their residuals, ARIMA models are able to learn the temporal dependencies in the data. For anomaly detection, ARIMA models are trained on normal data and used to forecast future values. The difference between the predicted and actual values is then used to calculate an anomaly score [14].

ARIMA models have been applied successfully for anomaly detection in network traffic [48, 68] and credit card fraud detection [44]. The work of [66] examine the effectiveness of a variation of the ARIMA model that considers multivariate time series data in its forecasting called *ARIMAX* in traffic data. Their evaluation concludes that ARIMAX provides better forecasting results than its univariate variations but also introduces more complexity in model specification and computational overhead. There are several variations of ARIMA models that extend its capabilities further, such as predictions for time series data with seasonal variations [8]. However, a clear disadvantage of ARIMA-based models is their reliance on several parameters to be set beforehand based on knowledge about the data distribution. Additionally, ARIMA models struggle to forecast data that exhibits non-linear pattern [71].

Unlike their counterpart, non-parametric techniques do not assume the statistical properties of the underlying distribution beforehand and determine them based on the available data instead. Common examples include techniques based on the use of kernel functions or histograms. For the latter, a histogram is built on the normal data and new instances are checked to see whether or not they fit into it. This method can be extended to multivariate data by building a histogram for each feature, comparing new instances based on how well its attributes fit into their respective histograms them and aggregating the results into a single score [13]. However, as outlined by the works [18] and [13], this approach does not take into account the relationship between the different features themselves and is thus limited in its suitability to multivariate anomaly detection. [31] make a further distinction between non-parametric and semi-parametric techniques. Semi-parametric techniques make use of local kernel-based methods, such as using multiple Gaussian mixture models to estimate the normal data instead of assuming a global distribution for the entire dataset. Data points in regions of low density are then flagged as outliers. Modern adaptations of specifically Gaussian mixture models have been applied successfully in novelty detection to identify cancerous growths in mammograms, albeit with a high amount of false-positive results [60].

Statistical techniques offer reliable results for anomaly detection, if the examined data coincides with the assumptions made about the underlying distribution. Parametric techniques are generally computational efficient and simple to implement but require reliable knowledge about the data distribution to be effective, while non-parametric methods are more flexible in this regard but more computationally expensive [31, 42]. The main disadvantage of these approaches is that the assumptions about the underlying data distribution often do not apply for real world and especially high dimensional data [18].

4.3 Distance Based Techniques

Distance based techniques make no assumptions about the data distribution and instead define the distance between data points or clusters as a similarity metric to discern between normal and anomalous data. The distance metrics used are not universal and can vary depending on the nature of the data or the approach of a particular method. For univariate, continuous data the Euclidean distance is commonly used, while for the multivariate case, the *Mahalanobis* distance is a popular choice. The Mahalanobis distance for a given data point is calculated as:

$$d_M(x, \mu) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)} \tag{4.2}$$

where x is the data point, μ is the mean of the distribution and Σ^{-1} is the inverse of the covariance matrix of the distribution. By considering the covariance matrix of the observations the Mahalanobis distance is able to account for correlations between the multivariate variables. However, the calculation of the covariance matrix requires to iterate over all instances of a dataset, which is computationally expensive for large and high dimensional datasets [25, 31].

A common approach to identify distance based outliers is through the identification of the neighborhood of an instance, where it is assumed that normal data points occur closer to other normal data points and thus form denser neighborhoods than outliers [13]. The *local outlier factor* (LOF) proposed by [10] is a well established unsupervised anomaly detection technique utilizing this approach. In LOF an anomaly score for each data point is calculated based on the Euclidean distance to its k-nearest neighbours, while also taking into account the distance between those neighbours and their k-nearest neighbours. This allows the technique to determine the local density of a data point, while also determining the density of the region said point appears in. The correct value for the k parameter is critical to ensure good detection results when using this algorithm, as noted by [25]. Due to the number of necessary calculations the original LOF algorithm is computationally expensive and scales poorly with large datasets as

well as high dimensional data [13, 17, 31]. To this end, more computationally efficient variations of the original method, such as the method proposed by [35], have been developed, which calculates anomaly scores for micro-clusters instead of individual data points, in exchange for more false-negative results. It should be noted that LOF can but is not typically used for anomaly detection in time series data, as the approach doesn't explicitly focus on the temporal elements. For this purpose, other distance-based techniques, such as *dynamic time warping* (DTW), which measures the similarity between two sequences that may vary in speed or length, are more commonly used instead [36].

4.4 Data Driven Techniques

Machine learning and deep learning techniques have proven to be effective tools in modeling different kinds of data, as they are able to adapt to unseen and complex patterns [31] but often in exchange for long training times, especially for large and high dimensional datasets. There are a large amount of different approaches utilizing various machine learning models for anomaly detection in time series data alone. The works of [12, 14] have provided an extensive overview over some of the most established ones. For this work particularly, unsupervised or semi-supervised approaches focused on time series and sensor data are of special relevance.

The work of [23] uses *one-class support vector machines* (OC-SVM), which are based on *support vector machines* (SVM), a popular supervised learning algorithm used for classification and regression tasks. SVMs work by finding a hyperplane that best separates different classes in the feature space but OC-SVMs do not require labeled data for the anomaly class. [23] use this approach for intrusion detection in the sensor networks of smart cities. They compare their approach against the performance of other anomaly detection techniques, including those based on the Mahalanobis distance and LOF and demonstrate that it is able to outperform their detection rate by 56%.

[67] presents a more complex approach by combining multiple anomaly detection techniques into a unified system to detect anomalous sensor behavior in autonomous vehicles in real-time. Specifically, it integrates a *convolutional neural network* (CNN), a deep learning model well-suited for analyzing spatial or temporal data by using multiple layers of filters to detect inherent patterns, with a sliding window mechanism over the data. This is combined with an adaptive *Kalman filter*, a statistical tool used to estimate the state of dynamic systems by filtering for noise and inaccuracies. This combined system is then used to analyze data from multiple redundant sensors built into the vehicles to identify when one of them exhibits unexpected behavior. The combination of these methods results in improved detection performance compared to using any of the individual subsystems alone.

Sequential machine learning models, such as *recurrent neural networks* (RNN) and the *long short-term memory* (LSTM) networks based on them have shown promising results for uni- and multivariate time series anomaly detection. The work of [21] demonstrates that LSTMs and *gated recurrent units* (GRU) outperform ARIMA models in their experiments on traffic flow prediction. [33] utilized an LSTM-based *autoencoder* to detect anomalies in multivariate data to detect faults in production equipment. An autoencoder is a type of neural network trained to encode input data into a compressed representation and then reconstruct it from that representation. By learning to reconstruct normal data patterns, autoencoders can identify anomalies as deviations from the learned patterns. They propose an architecture, where both the encoder and decoder each consist of one layer of LSTM units and two stacked dense layers on top of the decoder. They compare their approach against approaches based on CNN as well as a combination of DTW and k-nearest neighbour classification and manage to achieve better detection results by about 10%. Similarly positive results were reported in the work of [38], where a LSTM-based autoencoder achieved better detection results than a standard autoencoder in the anomaly detection of vibration data of an electric motor.

5 Implementation

5.1 Rationale

The nature of the sensor data and the types of known anomalies, outlined in section 3.3, impose several requirements on the selected detection system. First, the system must be able to handle varying degrees of non-stationarity depending on the dataset. Second, it must be able to detect multivariate anomalies and leverage the temporal components of time series data to manage correlations between instances, as the sensor data exhibits strong autocorrelation.

Furthermore, the system needs to be flexible and robust enough to accommodate the introduction of new components or changes to existing sensors during the testing and development cycle, without requiring excessive effort or domain knowledge to adapt to these changes. Given the limited availability of data on known anomalies, the approach must also be able to operate in a semi-supervised or unsupervised manner.

Additionally, the system must handle the inherent unpredictability between different runs, as the time series data shows significant variations in properties such as length, stationarity, and identifiable patterns. Considering these factors, the architectures presented in section 4.4 seem promising, as they have been successfully applied to time series anomaly detection, particularly in the context of sensor data. Additionally, as these approaches are data driven, they do not require distribution-specific knowledge or external parameters to be set, once a fitting architectural configuration has been found. This makes them more adaptable to new data-generating sources or sensor changes introduced during the development cycle of Formula Student vehicles, compared to, for example, ARIMA models. However, a notable disadvantage of these models is their high training time, along with the necessity for fine-tuning hyperparameters to achieve optimal performance.

While a detailed quantitative analysis of several potentially suitable techniques, some of which are mentioned in chapter 4, is beyond the scope of this thesis, an attempt was made to employ a SARIMA/SARIMAX-based approach. Unfortunately, the results of this method were unsatisfactory and outperformed by the LSTM autoencoder approach described later in this section. The SARIMA/ARIMAX model struggled to account for

the variations between datasets. This is likely due to the inconsistent and dynamically evolving nature of the sensor data, where patterns and their occurrence are influenced by external factors, such as track conditions and vehicle performance. Finding consistent parameters across different sensors and datasets proved challenging, which negatively impacted prediction results. While there are implementations that automate the selection of certain parameters, they still require some distribution-specific knowledge, particularly concerning seasonal patterns and multivariate relationships.

Based on these considerations, this work implements and evaluates an LSTM autoencoder architecture, inspired by the approach proposed by [40], using the provided datasets. This architecture was chosen for its ability to handle dynamic, multivariate time series data and its effectiveness in detecting anomalies without requiring distribution-specific knowledge or manual parameter setting.

5.2 LSTM

LSTM networks are variations of traditional RNNs that was introduced in 1997 in the work of [30]. They are typically used for task such as handwriting, speech recognition or machine translation and also find increasing use in time series and sequence prediction. The original RNNs suffer from the vanishing gradient problem and have difficulties when it comes to retaining information over longer periods of times. LSTM units were developed to address this issue and have been successfully utilized in a variety of techniques for time series anomaly detection due to their ability to retain information over long periods of time. This allows them to be especially effective at modeling temporal contexts as they are able to learn long-term dependencies present in the data. [18, 38, 71]. A LSTM unit is comprised of four components, a cell state, a forget gate, an input gate and an output gate, as illustrated in Figure 5.1. These gates control what information is discarded, updated, or passed to the next time step.

LSTMs process their input sequentially over multiple time steps. Unlike in a traditional feed forward neural network, where each input is treated independently, the output of a LSTM is dependant on the output of the previous time step h_{t-1}, also referred to as the hidden state and it's cell state C_{t-1}. A LSTM cell processes the input vector $x = (x_1, x_2, x_3, \ldots, x_t)$, where t represents the time steps, sequentially into the output vector $y = (y_1, y_2, y_3, \ldots, y_t)$ time step by time step. The following paragraphs describe this process for an input of a single time step.

First, the forget gate determines how much information of the previous time step should be retained for the current time step by using the sigmoid function, which outputs a value between 0 and 1:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.1}$$

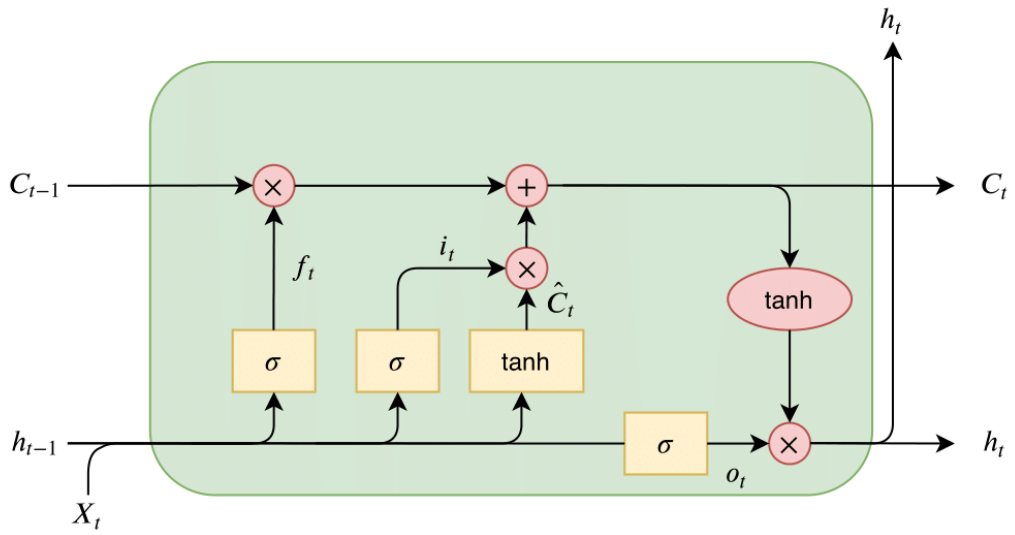


Figure 5.1: Structure of a LSTM cell [41]

where f_t is the forget gate's output, h_{t-1} is the output from the previous time step, x_t is the input at the current time step, and W_f and b_f are the weights and biases of the forget gate. Next, the input gate determines, which parts of the cell state should be updated with new information from the current input x_t and the previous hidden state h_{t-1}:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.2}$$

where i_t represents the input gate's output, W_i represents the weight matrix, and b_i is the bias vector of the input gate. Then the cell update state $\tilde{C}_t$, which represents new information from the current input that could be added to the cell state, is calculated using the tanh function:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5.3}$$

where W_C is the weight matrix, and b_C is the bias vector of this layer. In the next step the outdated cell state from the previous time step C_{t-1} is updated. This is done by combing the output of the forget gate from earlier f_t with the current, outdated cell state C_{t-1} and the cell update state $\tilde{C}_t$ combined with the result of the input gate i_t, as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.4}$$

This equation determines how much information of previous time steps should be discarded and how much new information from the current input should be remembered by the cell. In the last step the output of the output gate o_t is combined with the current cell state C_{t-1} using the tanh function to produce the output of the LSTM cell for the current time step h_t as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.5}$$

$$h_t = o_t * \tanh(C_t) \tag{5.6}$$

where W_o is once again the weight matrix, and b_o is the bias vector of this layer. This complex mechanism is performed for every time step of the input vector and enables LSTMs to effectively manage long-term dependencies. The output of the previous time step h_{t-1} can be thought of as the short-term memory of the cell and the cell state C_{t-1} as the long-term memory [72].

5.3 LSTM Autoencoder

In practice, LSTMs are often arranged in layers or even stacked layers in order to increase their ability to capture complex patterns and behaviour in data. The approach implemented for this thesis is based on the LSTM autoencoder architecture proposed by [40], which consists of two layers of LSTM units of the same size, which act as the encoder and decoder. Additionally, a linear layer is placed on top of the decoder layer. The following paragraphs will give an overview of the core idea and process behind this approach. For a more detailed exploration of the topic, the reader is referred to the original paper [40].

The model is trained on unlabeled normal data and uses a sliding window to obtain subsequences over a larger time series. Each of these subsequences is fed into the encoder layer time step by time step. Once the last time step has been processed, the encoder has compressed the input sequence into a fixed length vector representation in the form of its hidden state. An illustration of this process is shown in Figure 5.2 for a sequence of three elements over three time steps, where $x^{(t)}$ are the individual inputs of the subsequence $X = (x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(t)})$ at time step t and $h_E^{(t)}$ is the hidden state of the encoder at that same time step. In this approach, the compression effect typical for autoencoders is achieved by only passing the hidden state from the encoder to the decoder after the encoder has processed the entire subsequence. This idea behind this compressed representation of the input sequence is to help the network learn the impor-

tant underlying patterns in the behaviour of the normal training data, while discarding less relevant ones. This hidden state, illustrated in Figure 5.2 as $h_E^{(3)}$, is then copied to the decoder and used as its initial hidden state so that $h_E^{(3)} = h_D^{(3)}$.

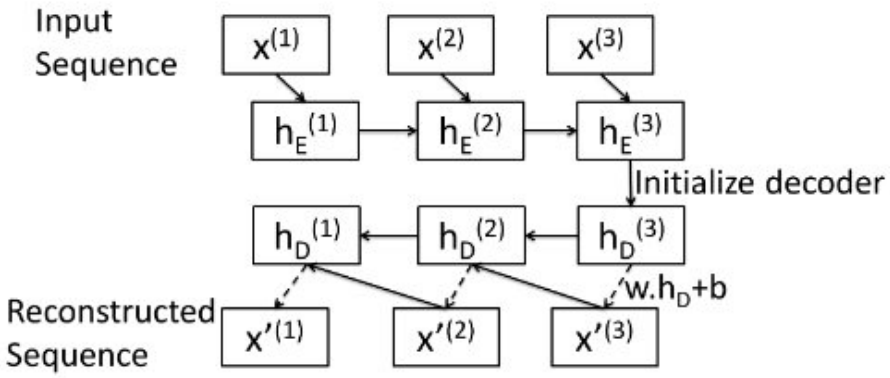


Figure 5.2: Initialization of the Decoder using the last hidden state of the Encoder [70]

The decoder then reconstructs the input sequence time step by time step but in reverse in accordance with the sequential encoding process. The output for each time step is passed through the linear layer behind the decoder to produce the predicted output for a single time step. This prediction is then used as input to the decoder to make the prediction for the next time step until the entire sequence is reconstructed. As no prediction is available for the first time step of the reconstructed sequence, the initialized hidden state of the decoder $h_D^{(t)}$ is used as input to the linear layer instead. The autoencoder is trained to minimize the sum of the *squared euclidean distance* between the input and output sequence as:

$$\sum_{i=1}^{L} \left\| x^{(i)} - x'^{(i)} \right\|^2 \tag{5.7}$$

where L is the total number of time steps or entries in the subsequence, $x^{(i)}$ is the input value at time step i, and $x'^{(i)}$ is the predicted value at time step i. After the training process, two validation datasets are used to enable the calculation of an anomaly score. The validation dataset v_N contains normal data and is passed into the autoencoder to calculate the reconstruction error for each point in v_N as:

$$e^{(i)} = |x^{(i)} - x'^{(i)}| \tag{5.8}$$

These error vectors are used to estimate the mean μ and the covariance matrix Σ of a Gaussian distribution using Maximum Likelihood Estimation. These parameters are used to calculate an anomaly score $a^{(i)}$ for each data point in future predictions. For

multivariate data, this anomaly score is calculated using the squared Mahalanobis distance measure, mentioned in section 4.3. As the original paper does not provide an equation for univariate data, this implementation uses the *z-score*, which is an established alternative for univariate data instead [13]. The z-score uses the variance σ of the error vectors, instead of Σ:

$$a^{(i)} = \begin{cases} \frac{|e^{(i)} - \mu|}{\sigma} & \text{if univariate} \\ (e^{(i)} - \mu)^T \Sigma^{-1} (e^{(i)} - \mu) & \text{if multivariate} \end{cases} \tag{5.9}$$

The second validation dataset v_A can be populated with known anomalies if any are available and normal data points. The anomaly scores of this dataset are used to learn an optimal anomaly threshold using the F_β score, described in the following equation:

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{\beta^2 P + R} \tag{5.10}$$

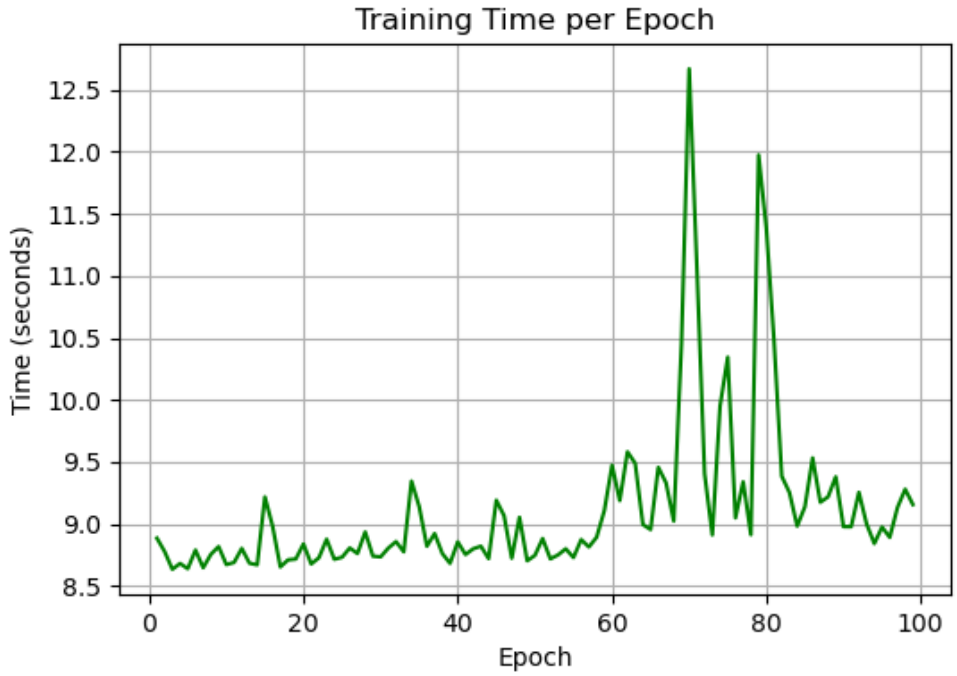
where P and R are the precision and recall metrics respectively. For the calculation of both metrics the anomalous data points form the positive class, while normal instances are the negative class. The equations for precision and recall are described in chapter 6. The advantage of this approach is that the β value can be set manually to either value precision or recall higher depending on the circumstances. Finally, if a single data point in a sequence exceeds the anomaly threshold, the entire window is labeled as anomalous.

5.3.1 Data Preprocessing and Hyperparameters

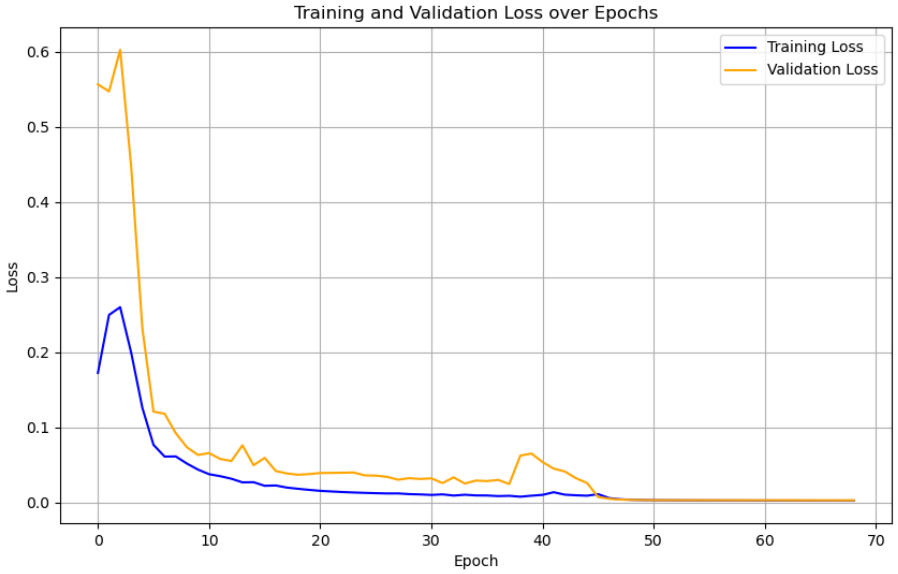
The implemented approach closely follows the one outlined in [40], with a few modifications. The best results for univariate datasets were achieved using a two-layer LSTM autoencoder, with 128 LSTM units in both the encoder and decoder. For multivariate datasets, increasing the layer size to 256 improved performance, though further increases did not yield additional benefits. The Adam optimizer from the PyTorch framework, with an adaptive learning rate starting at 0.001, consistently provided the best results. The model was trained with mini-batches of size two and a normal validation dataset used for early stopping.

One notable deviation from [40] is that in this implementation, the sliding window size and the predicted subsequence length can be set independently, rather than being equal. This proved particularly effective in boosting the detection rate for the "Flip" anomalies, as described in section 3.3, especially when a shorter prediction length was used in combination with a larger sliding window. The model averages around nine

seconds of training time per epoch, however this depends on the size of the sliding and prediction window. Figure 5.3(a) and Figure 5.3(b) show a typical example the training time per epoch as well as the loss curves for the train and validation dataset of the steering angle sensor.



(a) Training time per epoch



(b) Example of the loss curve of the LSTM Autoencoder during training on the univariate steering angle sensor data

6 Experimental Setup

6.1 General Setup

In order to achieve the most accuracy and to be able to trace faults back to specific components, a separate model was trained for each sensor, with a maximum of 100 epochs. The long linear parts of the time series data for the steering angle and the wheel speed sensors, mentioned in section 3.2 were cut out out off the time series before training and evaluation, as they only contain redundant data from when the car is standing still. The cut-off point is determined from the timestamp of the first measurements from the MCU, as it coincides exactly with the start of the section with high variability, meaning when the vehicle actually starts performing the run. The β value in the F_β score was set to 1.0 for all tests to keep the measurements comparable and give equal amounts of priority to precision and recall, even though in practice a smaller value may be advisable, due to the inherent rarity of anomalies. As the available datasets vary in length, the performance of the models will be evaluated using three standard evaluation metrics, namely precision, recall and the F1-score [14].

They are defined as follows:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{6.1}$$

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{6.2}$$

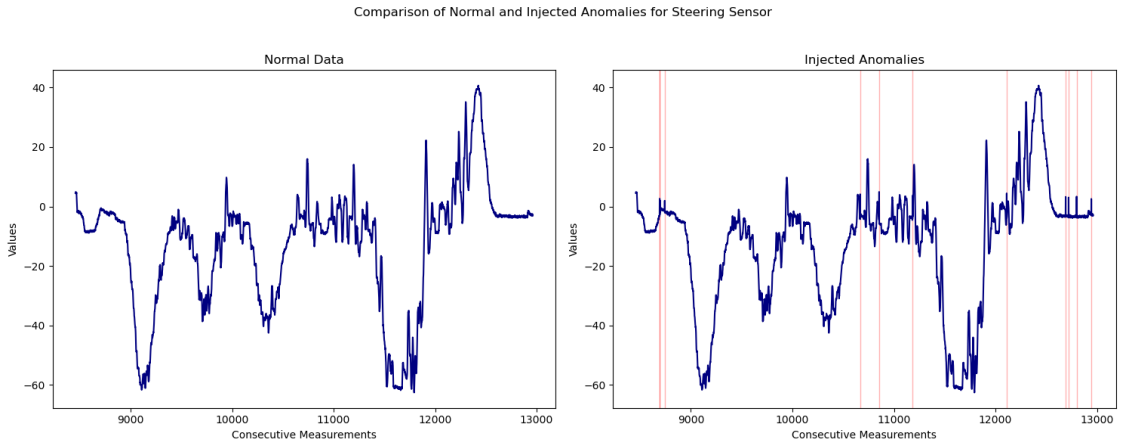
$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

The anomalous validation dataset v_A was populated sparsely with some of the randomly inserted steering angle anomalies to mirror the real world scenario, where little to no data is available on known anomalies.

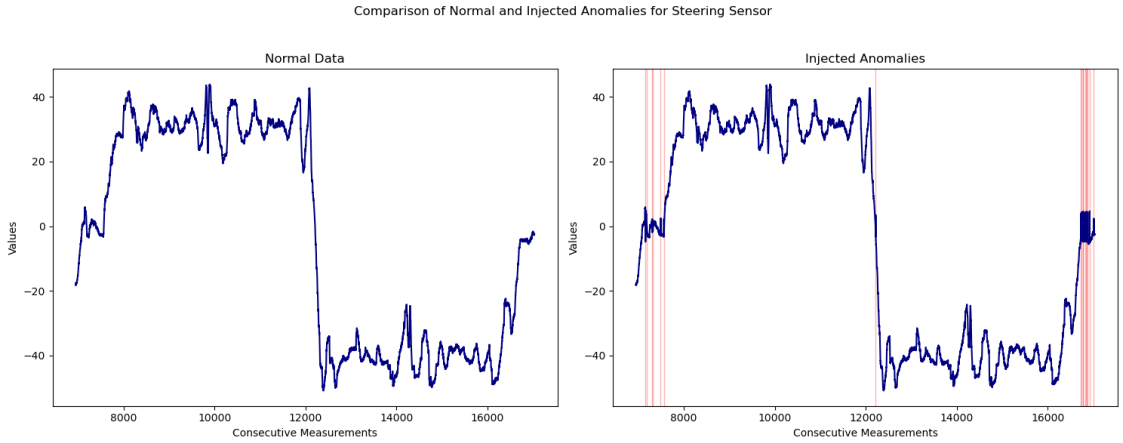
Multiple tests were performed for the steering angle anomaly and the encoder damage

mentioned in section 3.3. The steering anomalies represent contextual point anomalies and were randomly injected into the datasets with varying percentages in total amount of the eligible data points, ranging in steps from 1%, 5%, 10% to 20%. Through dataset study it was determined, that the maximum difference between subsequent values in the steering angle is around 2 degrees, while the maximum of the values affected by this anomaly are limited to a maximum of 5 degrees according to the reports from the Mainfranken Racing team. For this reason, only values in the ranges between 1.2 to 5 and -1.2 to -5 were flipped. The offset of 0.2 was chosen to account for the noise present in the data and to provide a small but clear separation window between normal and anomalous instances. All predictions utilized a model configuration of an input sliding window of 50 time steps, a prediction window of three time steps and one layer of 128 LSTM units for the encoder and decoder respectively. To measure the reaction of the model to the different patterns and characteristics present in the data for each discipline, a separate model was trained for both the Autocross and Skidpad datasets. The anomaly injections and the measurements were repeated ten times for each of these scenarios, over all of the available normal datasets for both disciplines. Due to the access to only six confirmed normal datasets, where two are used during training and validation each, the data from other runs with relatively harmless incidents, such as the car running over and dragging along traffic cones on the side of the track, were used as well, as no obvious adverse effects on the relevant sensor data could be observed. A comparison between some of the injected and the corresponding normal datasets for the steering angle anomaly is displayed in Figure 6.1.

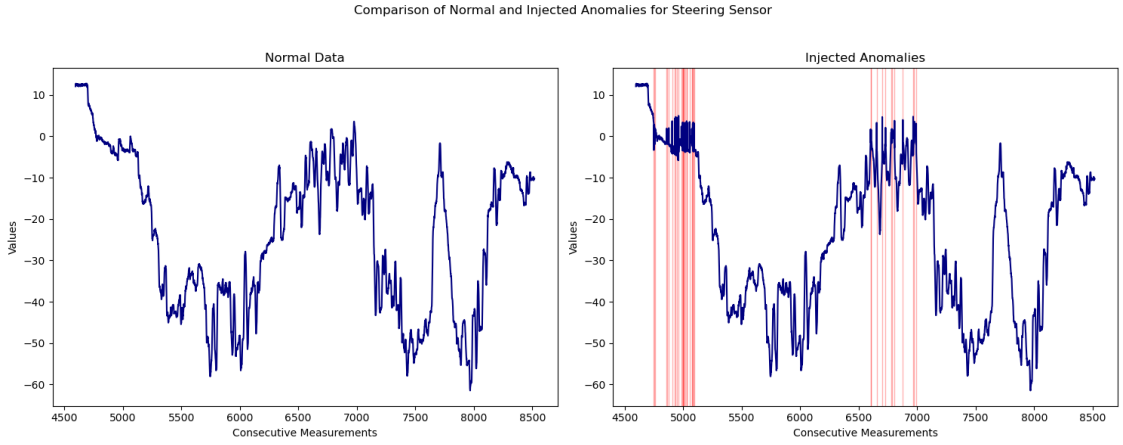
Varying amounts of random Gaussian noise were introduced to the datasets populated with 10% of the steering angle anomaly, to simulate the gradual degradation of a sensor and measure the robustness of the model. The detection of the encoder damage was tested using the real dataset from the incident. Training was performed by combining measurements from the steering angle sensor with MCU steering commands from normal runs. Additionally, the model's effectiveness in detecting potential contextual anomalous subsequences was also evaluated.



(a) Anomalies injected into 1% of the eligible data points



(b) Anomalies injected into 5% of the eligible data data points



(c) Anomalies injected into 10% of the eligible data points

Figure 6.1: Left: Examples of the normal time series of steering angle sensor. Right: The same time series after being injected with differing amounts of steering angle flip anomalies in the eligible value ranges.

7 Results and Evaluation

7.1 Detection of Steering Angle Anomalies

Anomaly Amount	AC Precision	AC Recall	AC F1	SK Precision	SK Recall	SK F1
1%	0.33	0.58	0.41	0.41	0.71	0.52
5%	0.44	0.47	0.45	0.49	0.68	0.56
10%	0.52	0.57	0.53	0.61	0.74	0.66
20%	0.69	0.66	0.67	0.64	0.84	0.72

Table 7.1: Detection results with varying amount of "Flip" Anomalies depending on the eligible data points. AC stands for the datasets of the autocross discipline and SK stands for those of the Skidpad discipline.

The detection results for the univariate contextual steering angle anomalies, as described in section 3.3, are displayed in Table 7.1. Both models show improved performance as the anomaly density increases, with the worst results observed at an anomaly content of 1%. This trend is likely because regions with higher anomaly density produce larger reconstruction errors. As the autoencoder moves across these regions during the sliding input window, the anomalous values are repeatedly included, causing a significant deviation from the normal data that the models were trained on. This, in turn, increases the reconstruction error after decoding.

Precision scores are consistently lower than recall across all measurements, with a mean precision of 0.50 for Autocross and 0.54 for Skidpad datasets. In contrast, recall tends to be higher, averaging 0.57 for Autocross and 0.74 for Skidpad. The higher recall in the Skidpad datasets can likely be attributed to the more consistent patterns resulting from its eight-shaped race track, as illustrated in Figure 7.3(b) and Figure 1.1. On the other hand, the Autocross data is more variable due to the different runs, which likely introduces greater complexity into the training and detection process, as shown in Figure 7.3(a) and Figure 7.3(c). This variability may hinder the model's ability to learn consistent patterns in the data, adversely affecting performance.

While the relatively low precision and recall scores, particularly for the Autocross datasets, indicate a significant number of false positives and false negatives, these results should be considered in the context of the anomaly type and the detection outcomes on

normal datasets. First, the distinction between normal and anomalous behavior for this sensor anomaly is particularly challenging, due to the limited range of possible values that can be affected by the value flip. This makes a higher rate of prediction errors expected.

Second, false positive results predominantly occur in the immediate vicinity of actual anomalies, thus still indicating the anomalous behaviour fairly accurately. A typical example of this can be seen in Figure 7.2(b), where a number of contextual point anomalies were inserted into the time series of the steering angle sensor data. This is likely caused by the aforementioned sliding window, moving over the anomalous data points multiple times during prediction. Additionally, as mentioned in section 5.3, the autoencoder labels an entire window as anomalous if the anomaly score for any point within the window exceeds the threshold. This interpretation is reinforced by the low false positive rate observed when passing the same time series, without injected anomalies, through the model and comparing the results, as displayed in Figure 7.2(a), which shows very few false positive detection results.

7.1.1 Influence of Noise on Detection Results

Amount of Noise	AC Precision	AC Recall	AC F1	SK Precision	SK Recall	SK F1
None	0.52	0.57	0.53	0.61	0.74	0.66
Low	0.56	0.42	0.46	0.57	0.73	0.64
Moderate	0.56	0.42	0.46	0.55	0.66	0.60
High	0.36	0.53	0.36	0.14	0.68	0.24

Table 7.2: Detection results with varying amount of "Flip" Anomalies depending on the eligible data points. AC stands for the datasets of the autocross discipline and SK stands for those of the Skidpad discipline.

To evaluate the robustness of the detection system and quantify the influence of noise on the detection rate, varying amounts of random Gaussian noise were introduced to the same datasets previously injected with 10% steering angle anomalies, as discussed in the previous section. The results of these experiments are summarized in in Table 7.2.

The level of noise introduced to each data point was determined by the standard deviation of the Gaussian distribution from which the noise values were sampled. For the row labeled "Low", a standard deviation of 0.1 was used, resulting in small variations where most noise values fell between -0.1 and +0.1, with a few outliers extending to approximately ±0.3. For the "Moderate" setting, the standard deviation was increased to 0.5, leading to a broader spread of noise values between approximately -0.5 and +0.5, with some larger outliers reaching up to ±1.5. In the "High" setting, a standard deviation of 1.0 was applied, introducing more significant fluctuations, with most noise values

between -1.0 and +1.0 and some extreme values extending up to ±3.0. This allowed for a structured comparison of the models' performance under increasing levels of noise and its impact on the detection accuracy.

For low levels of noise, the results for the Skidpad datasets remain largely unchanged, with only minor drops in precision and recall. However, the Autocross datasets experience a significant drop in recall, suggesting that predictions for the steering angle in this discipline are sensitive to even small amounts of noise not present in the training data. This can likely be attributed to the irregular patterns in this dataset, as discussed in section 7.1. This further suggests that the model has some issues in learning the underlying patterns in the data properly, leading to some degree of overfitting.As the level of noise increases to "Moderate," both precision and recall see a minor decrease in the Skidpad results, while the Autocross results remain unaffected. This indicates that the model has learned the Skidpad data more effectively and is not as sensitive to moderate deviations from the training data. However, a further increase in noise to "High" leads to a drastic drop in precision for both models, although recall remains relatively stable.

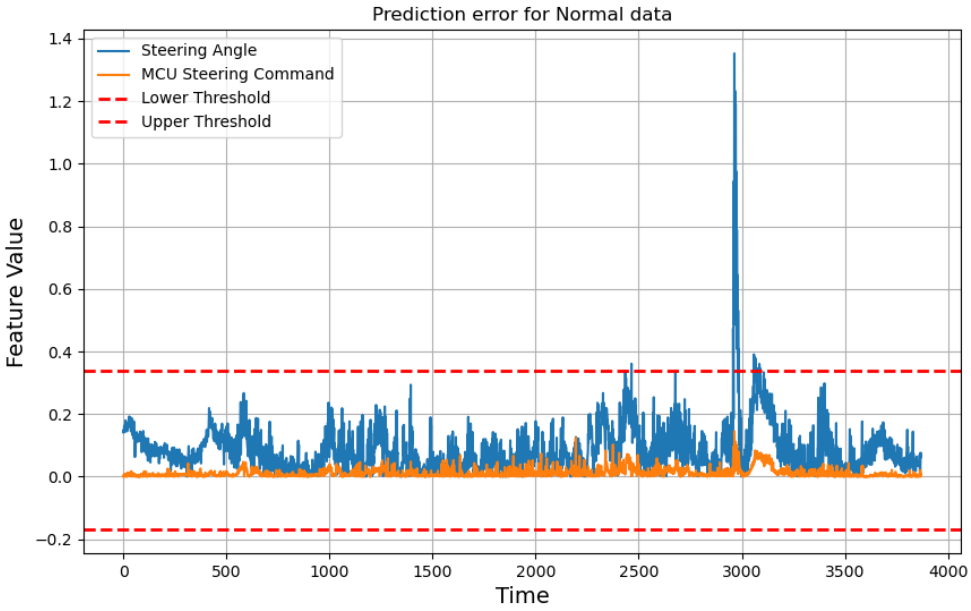
These results suggest that the Autocross model is particularly sensitive to noise, with a clear decrease in performance even at lower noise levels. The model trained on the Skidpad datasets showed more robustness to low and moderate amounts of noise, however the high noise scenario led to a significant drop in detection performance across both datasets.

7.2 Detection of Encoder Damage

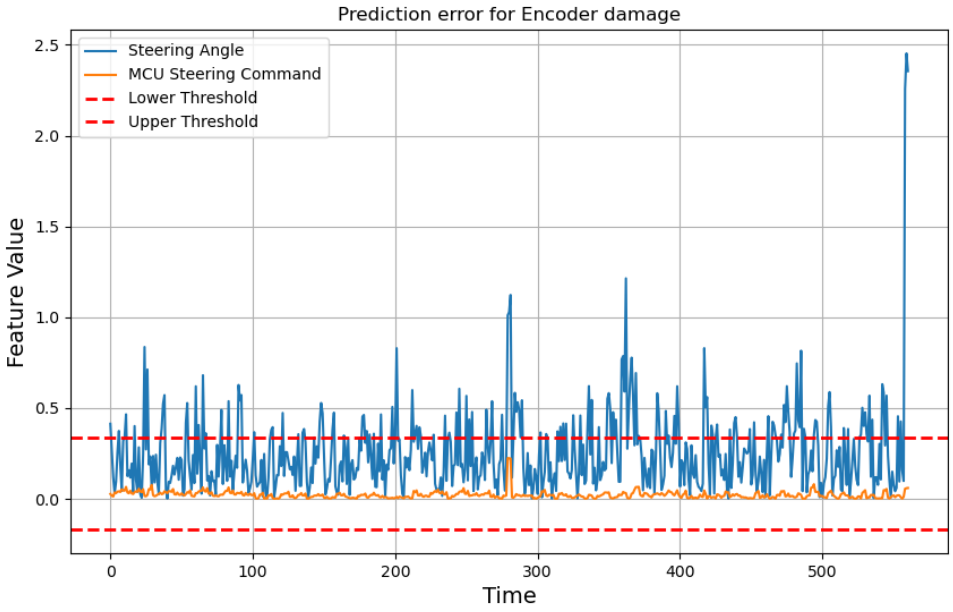
Utilizing the anomaly scores provided by the F_β score for the detection of the encoder damage anomaly proved difficult, as the anomalous behaviour is not indicated by any changes in the data but rather the near absence of them. This is difficult to model using manually labeled anomalies, as the behaviour considered anomalous in this context, is entirely normal if the encoder is functioning correctly. Instead, the 3σ rule, mentioned in section 4.2, is calculated on the prediction errors of normal datasets that were not used during training or validation. The upper and lower thresholds acquired by doing so are plotted over the prediction errors for each tested dataset and provide a clear visual identification of the anomalous behaviour. This method was tested across all available datasets and consistently proved effective in detecting the anomaly.

Figure 7.1 provides an example of the results, demonstrating the model's ability to reliably detect the abnormal pattern and learn the relationship between both variables. This approach allowed for reliable detection of this collective anomaly and demonstrates the model's ability to learn the relationship between both variables, despite the lack of

explicit changes in the data typically indicative of anomalous behaviour.



(a) Prediction errors for an undamaged steering motor encoder



(b) Prediction errors for the dataset of the damaged steering motor encoder

Figure 7.1: Comparison of the prediction errors between a normal and the steering motor encoder damage dataset.

7.3 Contextual Anomalous Subsequences

As discussed in section 2.1, contextual anomalies, particularly anomalous subsequences, pose a greater challenge compared to other types of commonly occurring anomalies. During testing, it was found that both models were able to detect contextual anomalous subsequences relatively accurately when using a prediction window of five time steps or fewer. However, increasing the size of the sliding input window led to higher false positive rates, especially in regions with strong gradients between consecutive data points. Despite several attempts to improve detection performance by altering model configurations, such as varying layer sizes, reducing the batch size, increasing the sliding input and prediction windows, adjusting the learning rate and changing the scaling of the data, no significant improvements were observed. However, as even the relatively small deviations from normal behaviour caused by the steering angle anomaly, discussed in section 7.1, are able to be detected fairly accurately, this points towards the autoencoder struggling to learn the more intricate, long-term temporal patterns in the data when larger sequences are involved.

In contrast, the original work by [40] applied this approach to detect contextual anomalous subsequences with greater success, using larger prediction windows. A possible explanation for this discrepancy lies in the nature of the datasets. Three of the five datasets used in the original work are publicly available [37] and a few key differences stand out. Firstly, the datasets used in [40] were larger in both size and number of individual samples, offering the model more training data. These datasets include records of power demand, ECG data and sensor readings from a valve in the space shuttle. More importantly, these datasets contain highly regular, repeating patterns in both shape and duration, unlike the high variability and irregular patterns observed in some of the Formula Student vehicle's sensor data, even within data collected by the same sensor over the same discipline. This suggests that the LSTM autoencoder architecture, while effective in detecting anomalies in datasets with repeating patterns, may not be well-suited for detecting longer, contextual anomalous subsequences in the highly variable, irregular and limited data generated by the Formula Student sensors.

7.4 Possible Optimizations

The LSTM autoencoder architecture based on the approach by [40] delivers fairly accurate detection results for the known types of anomalies encountered in Formula Student sensor data examined in this work. However, the evaluation of the detection results open up two main areas for further optimisation. Firstly, the system delivers a high amount of false positive detection results in the immediate vicinity around actual anomalies. While this does not hinder the detection of the anomalies themselves, as shown in Figure 7.2, it

may complicate the interpretation of the plotted detection results. Secondly, the system struggles to detect contextual anomalous subsequences in their entirety for larger sliding windows.

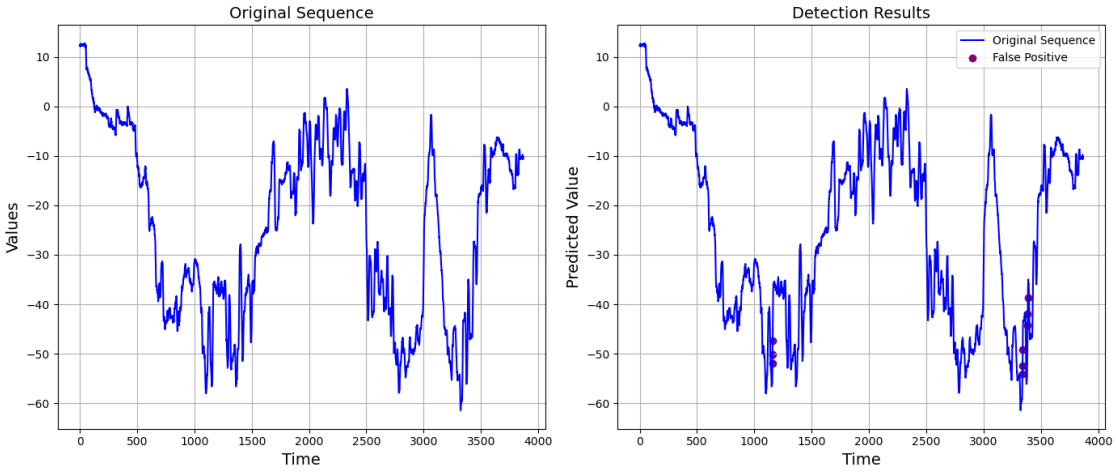
Both of these issues primarily stem from the same source. That is, the high variability and irregular patterns in the sensor data and the autoencoder's inability to learn these patterns properly across longer time frames. This limits the current implementation of the approach to longer sliding input and shorter prediction windows, which causes the high false positive rate by moving over anomalous data points more than once. A simple way to help improve on this issue may be through an increase of the available training data to more accurately portray the most common patterns that may be encountered during operation of the vehicle. While relatively simple, this approach is not guaranteed to improve detection results, as the environment of the Formula Student vehicle is highly dynamic and future changes to the vehicle's components or the track may introduce new, unknown normal and anomalous patterns not included in the training data.

A more promising solution to this issue may be to modify the architecture of the LSTM autoencoder itself to better suit the high variability environment of Formula Student racing. One possible technique that may help achieve this is through changing the current offline learning setup to an online learning one, as demonstrated by [52]. They propose an unsupervised approach with a model that integrates RNNs for anomaly detection and concept drift adaptation in time-series data in an online fashion. Their approach employs multi-step predictions using historical data. The prediction errors are used to both detect anomalies and to update the model, ensuring adaptation to concept drift in real time. Their approach manages sudden, gradual and continuous concept drift without manual intervention on a variety of non-stationary data. However, they note that their model suffers from a similar issue of persisting false positive results after actual anomalies, similar to the current LSTM autoencoder implementation, when employing multi-step predictions.

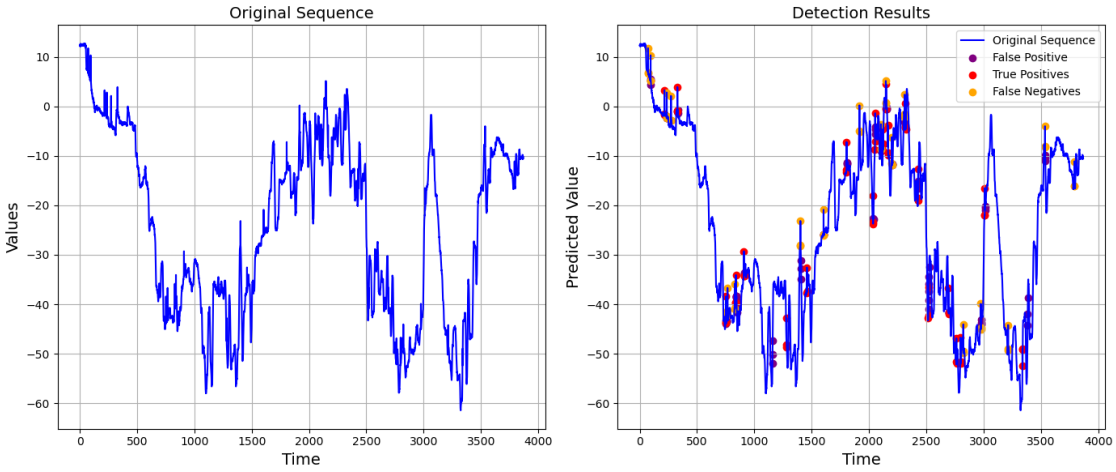
An alternative to this approach can be found in the well-established *Hierarchical Temporal Memory* (HTM) algorithm based on the work of [4]. HTM is designed specifically for streaming as well as time series data and continuously learns temporal patterns from the input data. HTM works in a unsupervised fashion and employs a two step anomaly detection process, where a raw anomaly score is calculated based on the prediction error, followed by an anomaly likelihood score, which helps reduce false positive results. Additionally, the online learning capabilities of both approaches could prove highly valuable for real-time anomaly detection during vehicle operation. This would eliminate the need for manual user intervention and enable the system to autonomously detect and respond to sensor or component faults before they escalate into more serious issues.

Implementing these techniques or adapting the existing model in a similar way could significantly improve the system's ability to detect anomalies in the sensor data, potentially in real-time or even preemptively. The online learning approach of both methods

may allow the system to better adapt to the high variability and irregular temporal patterns typical in racing environments. This would reduce the risk of false positives and enhance the system's reliability and detection accuracy. Additionally, it would minimize the need for manual intervention, even as track conditions and vehicle behavior evolve over time, ensuring continuous and efficient monitoring.

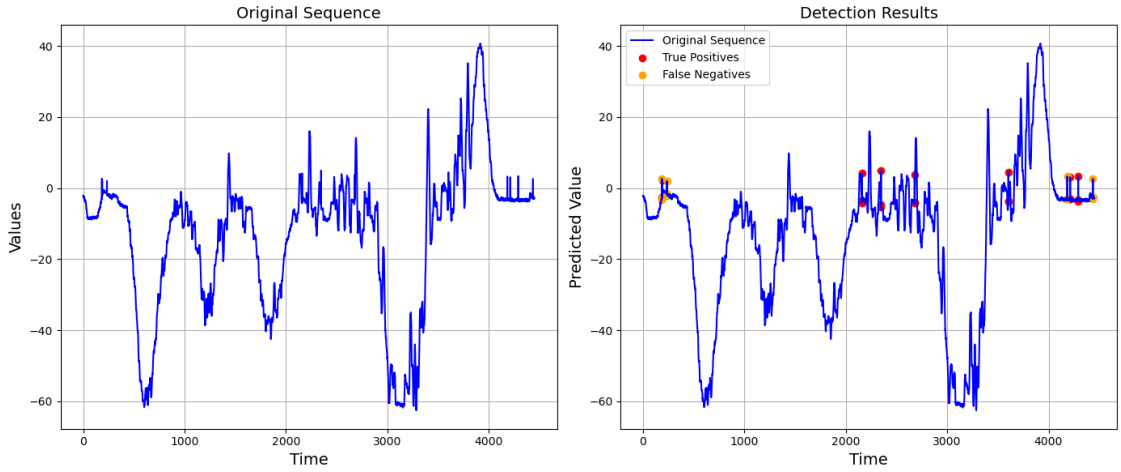


(a) Normal sequence without anomalies and its detection result

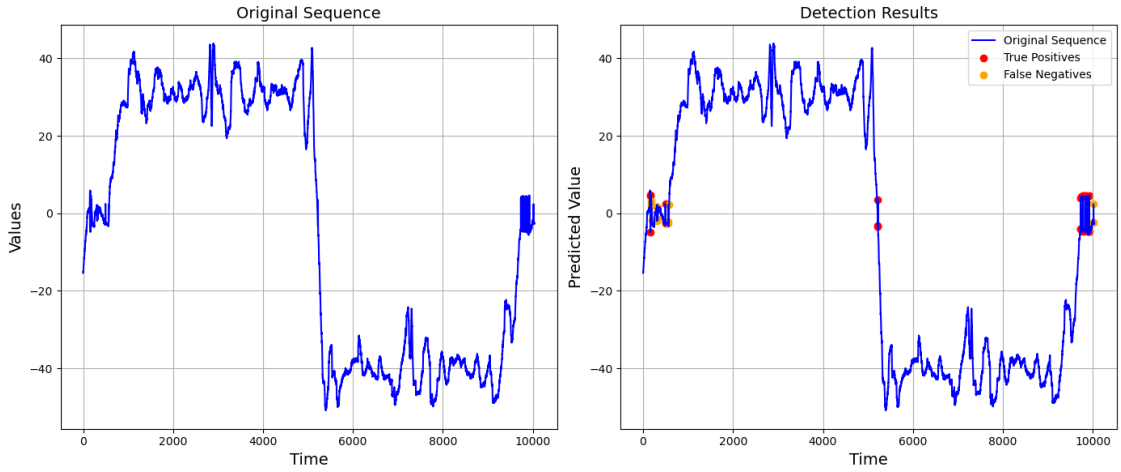


(b) Anomalous sequence with contextual point anomalies and its detection result

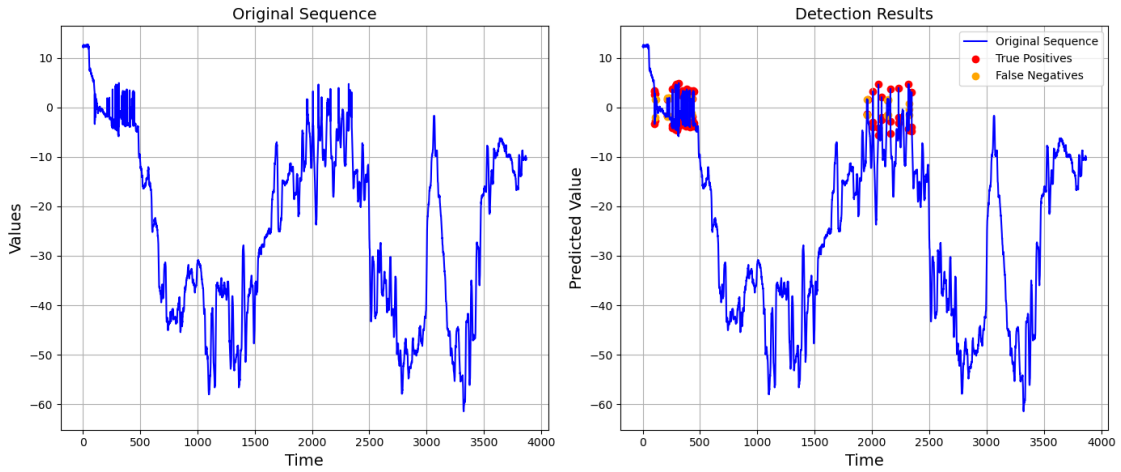
Figure 7.2: Typical example of occurrence of false positive detection results around the immediate area of actual anomalies.



(a) Detection result for 1% anomaly content in the eligible data points in Autocross steering angle data.

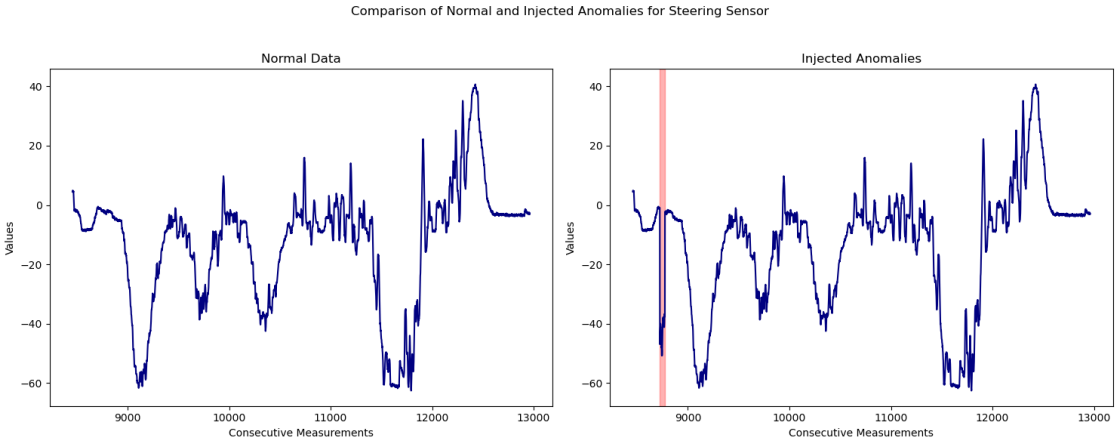


(b) Detection result for 5% anomaly content in the eligible data points in Skidpad steering angle data.

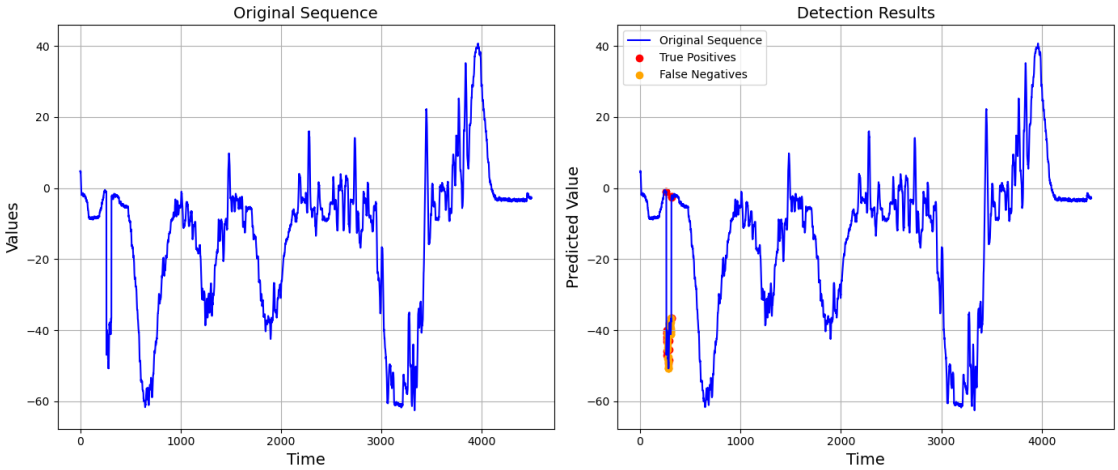


(c) Detection result for 10% anomaly content in the eligible data points in Autocross steering angle data.

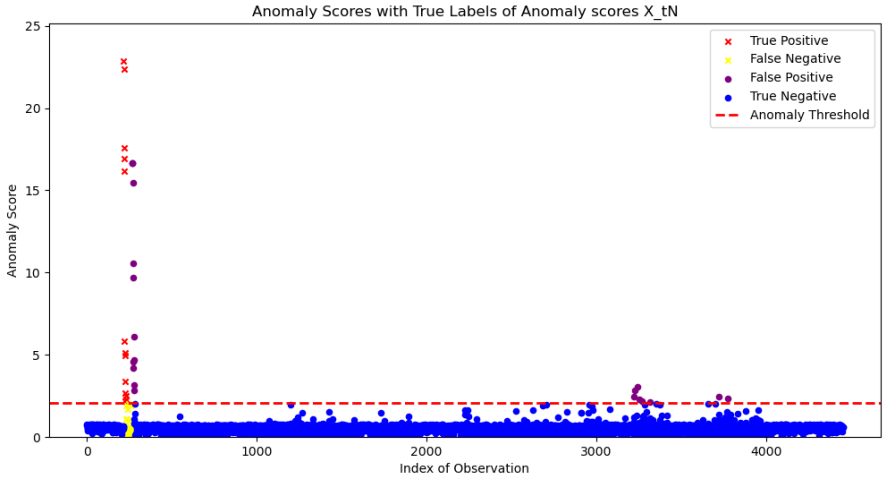
Figure 7.3: Left: Steering angle data that was injected with the specified amounts of steering angle anomalies. Right: The detection result for that time series.



(a) Example of the injected contextual anomalous subsequences



(b) Detection result for the anomalous subsequence



(c) Anomaly scores for each data point plotted over the anomaly threshold

Figure 7.4: Figure (a): Steering angle data that was injected with an anomalous subsequence over 50 consecutive measurements. Figure (b): The detection result for that same time series. Figure (c): The anomaly scores for each data point in the time series.

8 Summary

This thesis aims to address the challenges of anomaly detection in Formula Student racing cars through the development and evaluation of an anomaly detection system suitable for detecting sensor-related anomalies. The primary goal of the system is to enhance the efficiency, accuracy, and reliability of fault diagnosis for sensor-related issues. The high variability in the sensor data, along with the frequent changes in vehicle components and track conditions, necessitates a robust anomaly detection system that can adapt to this evolving environment.

To achieve this, the relevant sensor data was analyzed for its statistical properties and common anomalies were identified and categorized based on their characteristics. A suitable anomaly detection technique was then implemented and assessed for its ability to detect both common and other types of anomalies. The evaluation also considered the influence of various data characteristics on the detection results. Weaknesses in the system were identified and possible improvements to enhance the system's real-world applicability were discussed.

The study began with a detailed analysis of the sensor data collected from the Mainfranken Racing team during Formula Student competitions and test runs. The characteristics of the sensor data, including its multivariate, non-stationary and highly variable nature, posed significant challenges for traditional anomaly detection methods. Several incidents, such as damage to the steering motor encoder and incorrect installation of the steering sensor, provided real-world examples of common, domain-specific anomalies that needed to be detected to improve vehicle performance and safety. Based on the determined anomalies, the steering angle sensor in combination with the MCU proved to the most prone to failure, producing contextual and collective anomalies, respectively

Multiple anomaly detection approaches, including statistical methods, distance-based techniques, and machine learning models were considered based on their suitability to the problem. However, the final implementation focused on an LSTM autoencoder architecture due to its ability to model complex temporal dependencies and detect anomalies in dynamic, multivariate time series data. A statistical approach, based on the SARIMA-model was attempted as well but was outperformed by the LSTM autoencoder.

The model was trained on normal sensor data and evaluated using a combination of real and synthetic datasets with known anomalies. The influence of noisy environments,

differing amounts and types of anomalies on the detection results were examined. The results demonstrated that the LSTM autoencoder effectively detected both contextual point and collective anomalies, but struggled in some cases with false positive detection results and the accurate prediction of longer subsequences.

In conclusion, the proposed LSTM autoencoder-based anomaly detection system showed promising results in identifying faults in Formula Student racing cars' sensor data. However, further improvements, such as reducing the false positive rate and enhancing the system's ability to handle irregularities in the data, are necessary to increase its real-world applicability. Future work could explore more complex, hybrid approaches that combine the strengths of multiple techniques or models, as well as investigating methods for real-time anomaly detection during live vehicle operation.

List of Figures

List of Tables

Appendix

Bibliography

[1] [FormulaStudentTV]. *FSG 2016 Skidpad Overlay Video [Video file]. Youtube*. 2016. URL: https://www.youtube.com/watch?v=c-Ta7uYycEo. [Online;Accessed 01/09/2024].

[2] Charu C. Aggarwal. "On abnormality detection in spuriously populated data streams". In: *Proceedings of the 2005 siam international conference on data mining*. SIAM. 2005, pp. 80–91.

[3] Charu C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2017.

[4] Subutai Ahmad and Scott Purdy. *Real-Time Anomaly Detection for Streaming Analytics*. Version Number: 1. 2016.

[5] Talha Mahboob Alam and Mazhar Javed Awan. "Domain analysis of information extraction techniques". In: *International Journal of Multidisciplinary Sciences and Engineering* 9 (2018), pp. 1–9.

[6] Michele Basseville, Igor V. Nikiforov, et al. *Detection of abrupt changes: theory and application*. Vol. 104. prentice Hall Englewood Cliffs, 1993.

[7] Ane Blázquez-García et al. *A review on outlier/anomaly detection in time series data*. arXiv:2002.04236. 2020.

[8] G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Revised. Oakland, California: Holden-Day, 1976, pp. 238–242.

[9] G. E. P. Box and David A. Pierce. "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models". In: *Journal of the American Statistical Association* 65.332 (Dec. 1970), pp. 1509–1526.

[10] Markus M. Breunig et al. "LOF: identifying density-based local outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104.

[11] Peter J. Brockwell and Richard A. Davis. *Time series: theory and methods*. Springer science & business media, 1991.

[12] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[14] Kukjin Choi et al. "Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines". In: *IEEE Access* 9 (2021), pp. 120043–120065.

[15] Dhruv Choudhary, A. Arun Kejariwal, and Francois Orsini. "On the runtime-efficacy trade-off of anomaly detection techniques for real-time streaming data. CoRR". In: *arXiv preprint arxiv:1710.04735* (2017).

[16] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. "Anomaly detection for IoT time-series data: A survey". In: *IEEE Internet of Things Journal* 7.7 (2019), pp. 6481–6494.

[17] Rémi Domingues et al. "A comparative evaluation of outlier detection algorithms: Experiments and analyses". In: *Pattern recognition* 74 (2018), pp. 406–421.

[18] Laura Erhan et al. "Smart anomaly detection in sensor systems: A multi-perspective review". In: *Information Fusion* 67 (2021), pp. 64–79.

[19] Formula Student Germany GmbH. *Formula Student Germany*. 2024. URL: `https://www.formulastudent.de/about/concept`. [Online;Accessed: 28/08/2024].

[20] Formula Student Germany GmbH. *Formula Student Germany Disciplines*. 2024. URL: `https://www.formulastudent.de/about/disciplines`. [Online;Accessed: 28/08/2024].

[21] Rui Fu, Zuo Zhang, and Li Li. "Using LSTM and GRU neural network methods for traffic flow prediction". In: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2016, pp. 324–328.

[22] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. "An approach to spacecraft anomaly detection problem using kernel feature space". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 401–410.

[23] Victor Garcia-Font, Carles Garrigues, and Helena Rifà-Pous. "A Comparative Study of Anomaly Detection Techniques for Smart City Wireless Sensor Networks". In: *Sensors* 16.6 (June 13, 2016), p. 868.

[24] Muhammad Umer Ghani, Tariq Mahmood Alam, and Faris Hayder Jaskani. "Comparison of Classification Models for Early Prediction of Breast Cancer". In: *2019 International Conference on Innovative Computing (ICIC)*. 2019, pp. 1–6.

[25] Markus Goldstein and Seiichi Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PloS one* 11.4 (2016), e0152173.

[26] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[27] James D. Hamilton. *Time series analysis*. Princeton university press, 2020.

[28] Frank R. Hampel. "The influence curve and its role in robust estimation". In: *Journal of the american statistical association* 69.346 (1974), pp. 383–393.

[29] Douglas M. Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.

[30] S Hochreiter. "Long Short-term Memory". In: *Neural Computation MIT-Press* (1997).

[31] Victoria Hodge and Jim Austin. "A survey of outlier detection methodologies". In: *Artificial intelligence review* 22 (2004), pp. 85–126.

[32] T. Ryan Hoens, Robi Polikar, and Nitesh V. Chawla. "Learning from streaming data with concept drift and imbalance: an overview". In: *Progress in Artificial Intelligence* 1 (2012), pp. 89–101.

[33] Ruei-Jie Hsieh, Jerry Chou, and Chih-Hsiang Ho. "Unsupervised Online Anomaly Detection on Multivariate Sensing Time Series Data for Smart Manufacturing". In: *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*. 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA). IEEE, Nov. 2019, pp. 90–97.

[34] S. Hylleberg. "General introduction". In: *Modelling Seasonality*. Ed. by S. Hylleberg. Oxford: Oxford University Press, 1992, pp. 3–14.

[35] Wen Jin, Anthony K. H. Tung, and Jiawei Han. "Mining top-n local outliers in large databases". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. San Francisco California: ACM, Aug. 26, 2001, pp. 293–298.

[36] Rohit J. Kate. "Using dynamic time warping distances as features for improved time series classification". In: *Data Mining and Knowledge Discovery* 30.2 (Mar. 2016), pp. 283–312.

[37] E. Keogh, J. Lin, and A. Fu. "HOT SAX: efficiently finding the most unusual time series subsequence". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. 2005, 8 pp.-.

[38] Fadhila Lachekhab et al. "LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor". In: *Energies* 17.10 (2024).

[39] Aleksandar Lazarevic et al. "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2003, pp. 25–36.

[40] Pankaj Malhotra et al. *LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection*. Version Number: 2. 2016.

[41] órir Mar. *Insight into LSTM*. 2022. URL: `https://thorirmar.com/post/insight_into_lstm/`. Accessed: 2024-10-09.

[42] Markos Markou and Sameer Singh. "Novelty detection: a review—part 1: statistical approaches". In: *Signal Processing* 83.12 (Dec. 2003), pp. 2481–2497.

[43] Marc Masana et al. "Metric learning for novelty and anomaly detection". In: *arXiv preprint arXiv:1808.05492* (2018).

[44] Giulia Moschini et al. "Anomaly and Fraud Detection in Credit Card Transactions Using the ARIMA Model". In: *The 7th International Conference on Time Series and Forecasting*. International Conference on Time Series and Forecasting. MDPI, July 16, 2021, p. 56.

[45] Ayesha Munawar, Praveen Vinayavekhin, and Giovanni De Magistris. "Spatio-Temporal Anomaly Detection for Industrial Robots through Prediction in Unsupervised Feature Space". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 1017–1025.

[46] Nanotec. *C5-E – Motor controllers/drives*. 2023. URL: https://www.nanotec.com/eu/en/products/1764-c5-e-controller-for-stepper-motors-bldc. [Online;Accessed: 23/09/2024].

[47] Novotechnik. *RFD-4000 Series Rotary Sensor Datasheet*. 2023. URL: https://www.novotechnik.de/uploads/tx_extprodfind/39_DS_120_RFD-4000_Ratiometrisch--de.pdf. [Online;Accessed: 23/09/2024].

[48] Eduardo H. M. Pena, Marcos V. O. de Assis, and Mario Lemes Proenca. "Anomaly Detection Using Forecasting Methods ARIMA and HWDS". In: *2013 32nd International Conference of the Chilean Computer Science Society (SCCC)*. 2013, pp. 63–66.

[49] Marco A.F. Pimentel et al. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215–249.

[50] Peter J. Rousseeuw and Christophe Croux. "Alternatives to the median absolute deviation". In: *Journal of the American Statistical association* 88.424 (1993), pp. 1273–1283.

[51] Said E. Said and David A. Dickey. "Testing for unit roots in autoregressive-moving average models of unknown order". In: *Biometrika* 71.3 (1984), pp. 599–607.

[52] Sakti Saurav et al. "Online anomaly detection with concept drift adaptation using recurrent neural networks". In: *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. CODS-COMAD '18. Goa, India: Association for Computing Machinery, 2018, pp. 78–87.

[53] Kamran Shaukat et al. "A review of time-series anomaly detection techniques: A step to future perspectives". In: *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 1*. Springer. 2021, pp. 865–877.

[54] W. A. Shewhart. "Economic Quality Control of Manufactured Product [1]". In: *Bell System Technical Journal* 9.2 (Apr. 1930), pp. 364–389.

[55] *SIEN Proximity Switch Datasheet*. 2023. URL: https://www.festo.com/de/en/p/proximity-switch-id_SIEN/. [Online;Accessed: 23/09/2024].

[56] Jonathan A. Silva et al. "Data stream clustering: A survey". In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–31.

[57] Xiuyao Song et al. "Conditional anomaly detection". In: *IEEE Transactions on knowledge and Data Engineering* 19.5 (2007), pp. 631–645.

[58] Statsmodels Developers. *statsmodels.tsa.stattools.adfuller*. [Online;Accessed: 2024-10-06]. 2024.

[59] SBG Systems. *Ekinox Micro Documentation*. 2024. URL: https://support.sbg-systems.com/sc/hp/latest/documentation-resources/ekinox-micro-documentation. [Online;Accessed: 2024-09-21].

[60] L. Tarassenko. "Novelty detection for the identification of masses in mammograms". In: *4th International Conference on Artificial Neural Networks*. 4th International Conference on Artificial Neural Networks. Vol. 1995. Cambridge, UK: IEE, 1995, pp. 442–447.

[61] Félix Iglesias Vázquez et al. "Anomaly detection in streaming data: A comparison and evaluation study". In: *Expert Systems with Applications* 233 (2023), p. 120994.

[62] Gerhard Widmer and Miroslav Kubat. "Learning in the presence of concept drift and hidden contexts". In: *Machine learning* 23 (1996), pp. 69–101.

[63] Wikipedia contributors. *Formula Student Germany Autocross — Wikipedia, The Free Encyclopedia*. 2024. URL: https://de.wikipedia.org/wiki/Formula_Student_Germany%5C#/media/Datei:FSG_Autocross_Kroeger.jpg. [Online;Accessed: 01/09/2024].

[64] Wikipedia contributors. *Lidar — Wikipedia, The Free Encyclopedia*. 2024. URL: https://de.wikipedia.org/wiki/Lidar. [Online;Accessed: 23/09/2024].

[65] Wikipedia contributors. *Stationary process — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/wiki/Stationary_process#/media/File:Stationarycomparison.png. [Online;Accessed: 25/09/2024].

[66] Billy M. Williams. "Multivariate Vehicular Traffic Flow Prediction: Evaluation of ARIMAX Modeling". In: *Transportation Research Record: Journal of the Transportation Research Board* 1776.1 (Jan. 2001), pp. 194–200.

[67] Franco van Wyk et al. "Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (2020), pp. 1264–1276.

[68] Asrul H. Yaacob et al. "ARIMA Based Network Anomaly Detection". In: *2010 Second International Conference on Communication Software and Networks*. 2010, pp. 205–209.

[69] Salisu Wada Yahaya, Ahmad Lotfi, and Mufti Mahmud. "A consensus novelty detection ensemble approach for anomaly detection in activities of daily living". In: *Applied Soft Computing* 83 (2019), p. 105613.

[70] Sultan Zavrak and Murat Iskefiyeli. "Flow-based intrusion detection on software-defined networks: a multivariate time series anomaly detection approach". In: *Neural Computing and Applications* 35.16 (June 2023), pp. 12175–12193.

[71] G.Peter Zhang. "Time series forecasting using a hybrid ARIMA and neural network model". In: *Neurocomputing* 50 (2003), pp. 159–175.

[72] Minghu Zhang et al. "Data-Driven Anomaly Detection Approach for Time-Series Streaming Data". In: *Sensors* 20.19 (2020).

Declaration on oath

I hereby certify that I have written my bachelor thesis independently and have not yet submitted it for examination purposes elsewhere. All sources and aids used are listed, literal and meaningful quotations have been marked as such.

N. N., 14.10.2024

Consent to plagiarism check

I hereby agree that my submitted work may be sent to PlagScan (www.plagscan.com) in digital form for the purpose of checking for plagiarism and that it may be temporarily (max. 5 years) stored in the database maintained by PlagScan as well as personal data which are part of this work may be stored there.

Consent is voluntary. Without this consent, the plagiarism check cannot be prevented by removing all personal data and protecting the copyright requirements. Consent to the storage and use of personal data may be revoked at any time by notifying the faculty.

N. N., 14.10.2024