The Applicability of Anomaly Detection Algorithms for Failure Prediction with 2D Mobile Robots

by

Lars Rhijnsburger

In partial fulfillment of the requirements for the degree of
Master of Science
at Delft University of Technology,
to be defended publicly on 08/01/2025.

Faculty: Mechanical Engineering
Department: Cognitive Robotics
Programme: Robotics

Mentors / Supervisors:	Yke Bauke Eisma
	Erik Vlasblom
	Alexis Siagkris-Lekkos
Graduation committee:	Joost de Winter
	Dimitra Dodou
	Renchi Zhang

An electronic version of this thesis is available at:
http://dx.doi.org/10.13140/RG.2.2.30081.98403

Keywords:

Failure Prediction
Anomaly Detection
Mobile Robots
Root cause Analysis
Unsupervised Learning

Abstract

With mobile robotics being applied for more and more complex applications, their autonomy should be preserved. While a lot of research is performed into the direction of failure prediction for autonomous processes or systems, the field of mobile robots has received less attention. Proactive failure prediction for mobile robots is a useful tool to prevent unwanted downtime and undesired damages. This work attempts to fill this research gap by showing the applicability of anomaly detection methods for failure prediction in the field of mobile robots. Specifically, we employ an unsupervised Variational Autoencoder to predict failures in the operational data from the Discovery Collector, a manure cleaning robot developed by Lely Industries. We elaborately showcase the feature engineering steps which yield the best performance, provide the performance of three general datasets, and state promising next steps for root cause classification which is enabled by accurate failure prediction. All in all, our work shows that the use of feature offsets, calculated from desired values compared to actual values, enhances the model performance tremendously. The provided datasets showcase F1-scores ranging from 0.64-0.76, showing the proposed solution is able to solve the failure prediction problem in the field of mobile robots, while highlighting the encountered limitations for future improvement.

The Applicability of Anomaly Detection Algorithms for Failure Prediction with 2D Mobile Robots

Lars Rhijnsburger
Cognitive Robotics (CoR)

Erik Vlasblom
Lely Industries N.V.

Alexis Siagkris-Lekkos
Lely Industries N.V.

Yke Bauke Eisma
Cognitive Robotics (CoR)

Delft University of Technology
December 19, 2024

Abstract — With mobile robotics being applied for more and more complex applications, their autonomy should be preserved. While a lot of research is performed into the direction of failure prediction for autonomous processes or systems, the field of mobile robots has received less attention. Proactive failure prediction for mobile robots is a useful tool to prevent unwanted downtime and undesired damages. This work attempts to fill this research gap by showing the applicability of anomaly detection methods for failure prediction in the field of mobile robots. Specifically, we employ an unsupervised Variational Autoencoder to predict failures in the operational data from the Discovery Collector, a manure cleaning robot developed by Lely Industries. We elaborately showcase the feature engineering steps which yield the best performance, provide the performance of three general datasets, and state promising next steps for root cause classification which is enabled by accurate failure prediction. All in all, our work shows that the use of feature offsets, calculated from desired values compared to actual values, enhances the model performance tremendously. The provided datasets showcase F1-scores ranging from 0.64-0.76, showing the proposed solution is able to solve the failure prediction problem in the field of mobile robots, while highlighting the encountered limitations for future improvement.

I. Introduction

Mobile robots are becoming more and more integrated into automation processes. While robotic arms dominated automation a few decades ago, advancements in artificial intelligence and control algorithms now enable the use of more complex mobile robotic solutions. Adeleke et al. recently stated that "machine learning algorithms will enable robots to adapt to changing production demands, optimize workflows, and perform complex tasks with greater autonomy and efficiency." ([1], p.907). Specifically for mobile robotics, Ullah et al. gives a nice overview about the current challenges in their work ([2], p.25-28). These challenges involve a wide range of technical and operational considerations, from navigation and environmental adaptation to system reliability and performance consistency. As mobile robots take on more and more roles in various industries, ensuring their reliable operation becomes essential. Among the current challenges, failure prediction stands out as a particularly complex problem missing a universal solution. Despite numerous research efforts (for example, [3], [4], [5], [6], [7]), it turns out to be difficult to reach a general solution for failure prediction, due to the diverse nature of robots and their operational contexts [8].

Related to failure prediction, is root cause analysis. Root cause analysis is the task to identify the fundamental source of failure to enable the development of targeted preventive solutions. When root causes can be detected exactly when they occur during robotic execution, failure prediction becomes a trivial task. Currently, one of the challenges of applying machine learning solutions to this problem is the acquisition and determination of ground truth labels for robotic failures. Labelling is a process that often demands extensive expert domain knowledge and a significant amount of time. Extending the current efforts into failure prediction and unsupervised root cause analysis eliminates this need for labelling. Besides, it offers substantial practical value for mobile robots, enabling proactive maintenance strategies and enhancing operational reliability.

Failure prediction starts by identifying irregular patterns in data streams [9]. Anomaly detection algorithms are a popular choice to determine when there are patterns in the data that do not conform to expected nominal behavior ([10], p.1). These algorithms can also be applied to failure prediction when an error is defined as "the situation in which a system deviates from the correct state, which may or may not eventually result in a system failure" ([9], p.5). Thus, for anomaly detection algorithms to be effective in predicting failures, there must be a detectable deviation from nominal data patterns before an error, as these algorithms rely on recognizing unusual or unexpected behaviors. Anomaly detection algorithms are already widely researched in many fields, like network intrusion [11], smart manufacturing [12], healthcare [13] and video surveillance [14].

Mobile robots require sensors and internal models to make sense of their environment, after which they take appropriate actions to reach some goal. Autonomous mobile robots behave very complex, which makes it hard for the current state-of-the-art anomaly detection algorithms to be able to learn the patterns of nominal

behavior and detect the deviations from it. However, these algorithms are particularly valuable because they can detect root causes as anomalies [15], thereby enhancing root cause analysis and ultimately contributing to better failure prevention strategies. OmniAnomaly, proposed by [16], is a promising unsupervised deep learning model with the ability to learn strong temporal dependencies in the data and capture stochastic variables, which are two aspects that are required in the context of this problem. This will be elaborated upon in Section III.B. Within this work, we will focus on an autonomous mobile robot, called the Discovery Collector, created by Lely Industries (see Figure 3), which works with precomputed path planning and by executing discrete actions. More details about the robot will be provided later in Section III.A.

We state the following research question: *"What predictive performance can be achieved using a state-of-the-art anomaly detection model for predicting and categorizing upcoming failures for a mobile robot, specifically the Discovery Collector made by Lely Industries*[1]*, and how do variations of the input feature types impact the effectiveness of this model?".*

By answering this question we will show that the autonomous mobile robot of this study exhibits consistent enough behavior. This consistent behavior allows the anomaly detection algorithm to learn the underlying nominal patterns in the behavior and to detect deviations from it in its sensor data streams. In turn, this provides the robot with a signal that can be used to prevent upcoming failures by executing failure prevention strategies which minimize the robotic system failures. Overall, this paper contributes the following:

- An assessment of the effectiveness of different feature types on the learning model performance with a feature ablation study.
- A performance evaluation of OmniAnomaly [16] for failure prediction on a mobile robot dataset. Failure prediction using anomaly detection for mobile robots is a novel field of application. The novelty of the research domain is highlighted by a systematic Scopus query [17], which yields only 4 related publications. For more details, see Section II.B.
- Preliminary insight into the relation between the learning model's input, output, and the underlying root causes of the predicted failures.

The rest of this paper is organized as follows. Section II will go over the state-of-the-art anomaly detection solutions, while highlighting the existing approaches and applicability to the field of mobile robots. Followed by Section III which gives the context of the problem, elaborates on the selected anomaly detection model and introduces the dataset used within this research. Section IV describes the experiments which will aim to answer the research questions defined above, followed by Section V which goes over all the obtained results, after which Section VI provides a discussion of the results. Finally, Section VII highlights the most important contributions this paper holds, answering and discussing the research questions stated above.

[1]The Discovery Collector is a 2D mobile robot with uncertainty in its odometry due to limited sensing, navigating over a predetermined route with a known map.

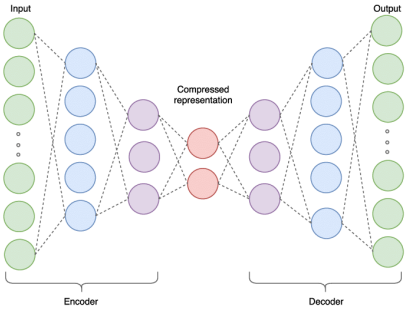


Figure 1: Example of an Auto Encoder architecture. Source: [22]

II. Related work

Anomaly detection is a popular tool for identifying unusual patterns or outliers in data. It has found widespread application across various fields, including cybersecurity, manufacturing, healthcare, and surveillance. On the other hand, little can be found about anomaly detection in the field of mobile robots. Within this section an overview of the existing anomaly detection applications is provided and the research gap of anomaly detection for mobile robotics is stated.

A. Applications of Anomaly Detection

The contexts of the fields of application of anomaly detection algorithms can differ greatly, resulting in approaches that are not universally applicable. A very common field of anomaly detection is network intrusion detection systems (NIDS). Anomaly detection systems within this field are tasked with monitoring network-related timeseries data. Their primary function is to detect when the network is under attack by an adversary. Yang et al. [11] identified 119 top-cited NIDS papers from 2021 and before in their survey. Other more recent approaches of NIDS include [18][19][20][21] which use graph neural networks, random forests, decision trees or generative adversarial networks (GAN) respectively to detect intrusions. Yang et al. [11] observed that many NIDS related anomaly detection approaches still rely on supervised algorithms and models, which requires the availability of ground truth labels. Consequently, they conclude that the way forward is to invest in unsupervised or semi-supervised approaches. These approaches only require little to no ground truth labels, which reduces the computational and resource-intensive process of manual data annotation and enables more scalable and adaptable models. However, Li et al. [21] conclude that, although their unsupervised model shows good performance, unsupervised learning approaches in the context of NIDS are rare and immature, with significant room for improvement.

While NIDS focuses on network security, another

popular application of anomaly detection is in real-time monitoring of smart manufacturing processes. The goal of anomaly detection in smart manufacturing is being able to detect unplanned production downtimes quickly in order to enhance production efficiency. [23], [24], [25] and [26] for example, all use Auto Encoder (AE) models to detect such anomalies. Other approaches in the field use GANs ([27], [28]), neural networks (NN) ([29], [30], [31]), or unsupervised clustering ([32]).

This shift towards unsupervised models reflects a broader trend in anomaly detection, where techniques like AEs, GANs, and clustering algorithms are gaining popularity due to their ability to learn from unlabeled data. AEs have become particularly popular for anomaly detection due to their ability to learn compact representations of normal behavior and identify deviations from it. Currently, AEs are most commonly used, and an example architecture can be seen in Figure 1. It shows the general structure of an encoder part, a smaller latent space representation in the middle, and a decoder part that tries to map the latent space representation back to the original input. Such models only require nominal data for training, and detect anomalies by failing to successfully map deviating behavior to and from the learned latent space representation. The popularity of unsupervised methods highlights the need for models that can adapt to changing environments and applications. Besides the fields discussed above, there are many more fields that work with anomaly detection algorithms. These fields include for example healthcare [13], video surveillance [14] and the Internet of Things (IoT) [33]. A more elaborate overview of relatively recent papers in various fields can be found in [34].

B. *Anomaly Detection for Mobile Robots*

In contrast to these well-studied fields, anomaly detection in mobile robotics has received relatively little attention. Examples of work that is performed so far in this field are [35], which apply anomaly detection to detect sensor and actuator misbehavior, [36] which use camera images to detect anomalies on the drivable plane in front of the mobile robot, and [37] which applies anomaly detection on positional information to detect deviations from a robot's planned trajectory. Azzalini et al. [37] interestingly follow up on the latter paper with [38] which does not only detect deviations from a robot's trajectory, but also classifies the type of deviation in an unsupervised manner using the latent space representation of the input data with a Variational Auto Encoder (VAE). [38] however did not work with mobile robots, but with nautical drones. They quantified the performance of their work by a False Positive Rate (FPR) of 0.0 on the nautical drone dataset, but did not quantify the latent space clustering of the different deviations. Nonetheless, the use of VAEs in their work demonstrates the ability of VAEs for clustering different types of anomalies or failures. Figure 2 shows an example of such latent space classification, found by Azzalini et al. in [38]. This highlights its potential application for root cause classification.

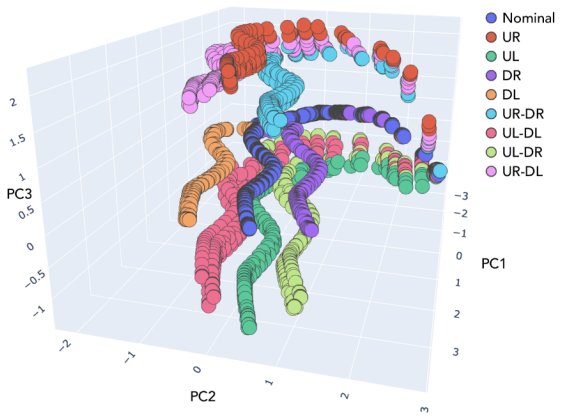


Figure 2: Latent space representation classification results, found by Azzalini et al. The abbreviations are all the different failure modes. Source: [38].

Furthermore, related to predicting upcoming failures, is [39] which proposed a proactive anomaly detection model trained on camera images and LiDAR scans. Papers related to mobile robot anomaly detection or failure prediction ([35], [36], [37], [39]) however, do not go into the potential root cause identification behind the detected anomaly or failure. The exception to this is [38] which does apply some form of classification. Moreover, [35] did mention they were not able to pinpoint potential root causes because their model aggregated the data in an irreversible manner. This limitation suggests a need for more interpretable unsupervised models in robotics, capable of not only detecting anomalies, but also providing insights into their origins.

III. Preliminaries

Before defining how the research into anomaly detection for mobile robots is performed, it is relevant to introduce the details surrounding the problem at hand. Below, an overview of the problem context is provided, along with the model used for the anomaly detection problem for mobile robots and the preliminary steps taken with the available data.

A. *Problem Context*

The mobile robot upon which this paper is based is the Discovery Collector[2], see Figure 3. The Collector is made by Lely, a company in the Netherlands focussed on robotic applications in the dairy industry. The Collector robot operates in a barn with milk cows, which is a challenging environment. The presence of manure, feed, and hay creates complex environmental conditions. Many external sensors are less reliable when they are dirty, and the floor traction varies significantly due to the manure on the ground. The robot navigates using a provided map, considered to be the ground truth map, and precomputed routes consisting of discrete actions. Discrete actions are for example straight actions

[2] https://www.lely.com/nl/discovery-collector/

Figure 3: The Discovery Collector robot made by Lely, which is tasked to autonomously clean up all the manure in a cow barn. Source: [40]

parallel to a wall, or a turn with a specific radius and angle. As the environment is so harsh for external sensors, the robot navigates mainly by using measures of the velocity of its two actuated wheels and two ultrasonic sensors, each orthogonal from the robot facing left and right, see also Figure 5. This limited sensing capability results in challenging localization, requiring the robot to occasionally bump into walls to recalibrate its position.

Moreover, due to the floor conditions being wet and covered in manure, it is not uncommon that the robot slips. Combining this with occasional incorrectly detected bumps or unreliable sensor information due to it being dirty, the robot sometimes localizes itself incorrectly, or starts or finishes actions prematurely. The robot is equipped with certain recovery actions which it can take when it detects it is no longer in the correct location. However, when this fails the robot goes into an erroneous state and will not move anymore until manual intervention. This is the failure that we are trying to predict using anomaly detection, with the goal of preventing the need for manual intervention after failure, thereby increasing the robot's uptime significantly. Preventing such failures can be achieved by learning nominal data patterns, in order to detect deviating data that precede failures. Despite the predetermined routes, this remains a challenging problem, due to the robot's limited sensing capabilities. The difficulty lies in distinguishing between critical uncertainty resulting in failure and temporary uncertainty from which the robot can recover on its own. This gray area, where the difference is often unclear to the human eye, is precisely where deep learning (DL) models can be highly effective if applied right [41].

B. Model Selection and Architecture

Like many other fields within anomaly detection, labelling data for mobile robots is still a difficult and time-consuming effort. As this problem's context is also quite complex, labelling would also require expert domain knowledge and a significant amount of time. Supervised and semi-supervised approaches are therefore less suited for this problem. Similar to the conclusion of Yang et al. [11], and following the general trend in the field of anomaly detection, the current work prefers an unsupervised model.

Another relevant aspect of the problem context is the input data format. Mobile robots, like many other robots and systems, produce multivariate timeseries. Due to the sequential order of actions the robot in this scenario takes, it is important to be able to track the robot's internal state over time. This indicates the need for a model with good temporal dependency, which enables the robot to not only consider its state based on current data, but rather evaluate its state based on a range of recent values. As an example, when the robot encounters a bump which it believes to be a wall, but is actually an unknown object on the route, it might prematurely take its next action, only to potentially encounter more unexpected bumps, or the absence of walls in certain places. Tracking the uncertain state of the robot from the initial early bump onwards for some time is therefore vital for the model to distinguish between nominal and anomalous behavior leading to a failure. The nominal behavior here is defined as the completion of a full route without any recovery behavior. Of course, the model should also be able to deal with uncertain actions that do not have negative consequences, as during normal behavior it is not uncommon to have some unexpected bumps here and there due to the presence of cows.

Furthermore, the model must be able to effectively capture the stochastic elements of the problem, as stochastic elements permeate this robot's problem context. Sensor measurements inherently contain noise, the external environment dynamically influences the robot's state at any time, and route execution is continuously challenged by localization uncertainties. These factors collectively create a need for a stochastic model. Rios et al. [42] highlight the two typical approaches. Namely, to model either the dynamical nature of the data using deterministic approaches, or to model the stochastic relations in the data using statistical approaches. Hereafter, they state that real-world systems produce a mixture of both types, emphasizing the need for a combined solution, which this problem also requires.

Overall, we require an unsupervised model which is able to learn nominal behavioral patterns in multivariate timeseries produced by a mobile robot. It has to consider both the strong temporal dependency of the data and the presence of stochastic variables. Su et al. [16] have introduced the OmniAnomaly model which does exactly all these things and more. Their model demonstrates good performance for multivariate timeseries anomaly detection, effectively handling datasets with stochastic variables and requiring temporal dependency insights. Compared to other state-of-the-art models using only temporal or stochastic approaches, OmniAnomaly achieves an average F1-score of 0.86 compared to 0.77 of the second-best model introduced by Hundman et al. [43], on the SMAP, MSL (both from NASA) and SMD (introduced in [16]) datasets. A more elaborate study into the usability of OmniAnomaly for

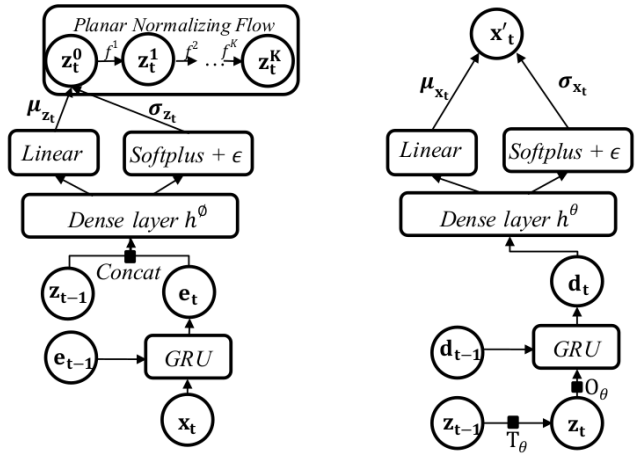


Figure 4: OmniAnomaly's architecture separated into two parts. The left is the encoder part of the VAE where $\mathbf{z}_t^K$ is the latent space representation. The right is the decoder part of the VAE which takes a sample $\mathbf{z}_t$ from the latent space distribution $\mathbf{z}_t^K$ and decodes it back to the original input $\mathbf{x}_t$. Source: [16]

the problem at hand is performed in an earlier literature study in [44].

OmniAnomaly uses Gated Recurrent Units (GRU) (a type of Recurrent Neural Network (RNN)) combined with a Variational Autoencoder (VAE) architecture to capture both temporal dependencies and the stochastic variables in the data. Besides the classical multi-Gaussian latent space representation of VAEs, OmniAnomaly enables learning non-Gaussian distributions for more complex data patterns by applying planar Normalizing Flows (NF). The full architecture of OmniAnomaly can be seen in Figure 4. The output of this model however, is not yet a prediction. The model's output is first compared to a threshold, which within OmniAnomaly is currently found by trying multiple different thresholds in a pre-determined range and choosing the optimal one. For a complete comprehensive description of OmniAnomaly, it is recommended to consult the work of Su et al. [16] itself. Furthermore, OmniAnomaly has two other aspects which suit the field of mobile robots well. First, a significant advantage of using a VAE inside the model is the fact that the loss is calculated as the sum of the loss on each individual input feature. This enables the extraction of the anomaly score contribution per feature. The anomaly score contribution per feature indicates how well the model was able to reconstruct each feature, which can be used for further root cause analysis, as the features contributing most to an anomalous output are potentially related to an underlying root cause. Second, the fact that the output of OmniAnomaly is given per time step, enables further investigation into the correlation between the input and output. As seen later Section V, combining this approach with positional information of the robot over time yields interesting results that contribute to the possibility of root cause classification of failures.

C. Data Definition

The quality of input data is crucial to construct good learning models. To maintain generalizability in anomaly detection for mobile robots, we focus on two generally available data streams: motor velocities and ultrasonic measures. These sensors provide promising indicators for failures without relying too much on robot-specific implementations. Within the experiments explained later in Section IV, several feature engineered data streams are also explored to see if they improve the model's performance. The additional engineered features are the following:

1. **Motor velocity offsets:** calculated by subtracting the motor velocity control setpoint and the actual measured motor velocity.
2. **Ultrasonic offsets:** calculated by subtracting the expected ultrasonic measure and the actual measure. The expected ultrasonic measurement is found by using a simple ray-trace model using the map provided to the robot, and the position on which the robot has localized, as can be seen in Figure 5. The used distance r is found with Equations (1)-(4) below.

$$d_{12} = |r_1 - r_2| \tag{1}$$
$$d_{13} = |r_1 - r_3| \tag{2}$$
$$d_{23} = |r_2 - r_3| \tag{3}$$
$$r = \begin{cases} \frac{r_1+r_2}{2}, & \text{if } d_{12} = \min\{d_{12}, d_{13}, d_{23}\} \\ \frac{r_1+r_3}{2}, & \text{if } d_{13} = \min\{d_{12}, d_{13}, d_{23}\} \\ \frac{r_2+r_3}{2}, & \text{if } d_{23} = \min\{d_{12}, d_{13}, d_{23}\} \end{cases} \tag{4}$$

In the equations above, r_i represents the distance found by a single ray-trace, and d_{ij} represents the distance between two ray-trace distances. The final used distance r is calculated as the average of the two most similar ray traces. While the ray-trace model yields good results for this project, it is important to state its limitations. The model is highly dependent on the accuracy of localization. When the robot is incorrectly localized, the obtained ray-trace intersections will not make any sense. Additionally, the ray-trace model assumes ideal ultrasonic reflections, which is not the case in practice. The use of this ray-trace model should therefore be carefully evaluated before use, as otherwise the model's predictions are related to anomalies in the used ray-trace model instead of related to the robot's execution.
3. **Flags:** discrete indicators for driving direction and left and right turning actions during a route. The value determination of each flag can be seen

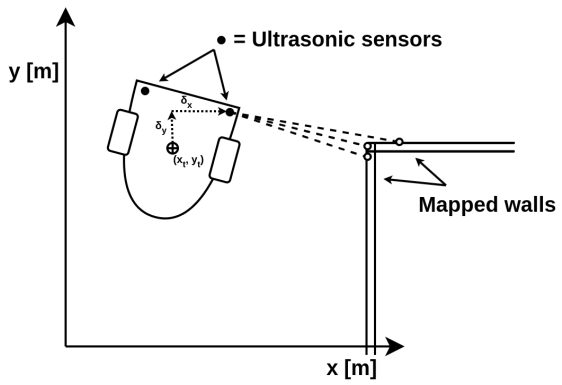


Figure 5: An illustration of the simple expected ultrasonic measure ray-trace model. The center position of the robot at each time t is known, as is the offset to both ultrasonic sensors. 3 ray-trace intersections to the closest wall in the map are then calculated of which the average of the two most similar ray-traces is used to prevent a noisy signal.

in Equations (5)-(7) below.

$$\text{direction\_flag} = \begin{cases} 1, & \text{if moving forward} \\ -1, & \text{if moving backwards} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\text{turn\_left\_flag} = \begin{cases} 1, & \text{if turning left} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$\text{turn\_right\_flag} = \begin{cases} 1, & \text{if turning right} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

From these signals 12 datasets are created with various combinations, with all input features normalized to ranges of $[0, 1]$ or $[-1, 1]$. The first nine datasets are smaller, derived from a single robot on a single route with only one underlying root cause before the failures. The single underlying root cause per dataset is also related to the input feature of the dataset. The smaller datasets are used for a feature ablation study [45], to evaluate the effectiveness of the various proposed input signals. The last three datasets are larger, incorporating data from six robots on multiple routes with multiple underlying root causes before failure, using the best performing input combinations as determined by the experiment described in Section IV.D, and its results in Section V.A. The performance of the model with the larger datasets demonstrates the model's ability to generalize across more generally representative input data. Table 1 provides a high-level overview of all datasets, including their composition, sizes, and amount of routes of each label. This dataset design allows us to investigate the contributions of each input type to predictive performance and the impact of combining multiple sensor streams. A more elaborate composition of the test set of each dataset can be found in Table 4 and Table 5 in Appendix B. All datasets consist of time steps which are equidistantly spaced. The original data from the robot is recorded with a frequency of 10Hz, which is then downsampled to 2Hz. This maintained the overall trends in the data, reduced the amount of noise within the datasets which influences the learning model negatively, and reduced the overall training time of the model. The moments before a failure itself, present in the available data, is purposely left out of the test datasets. The reason for this is that we do not care about behavior deviations during the failure, but rather the behavior before it. Finally, the specific definitions of 'anomalies', 'failures' and 'routes' in the context of this study are provided in the Section IV, as these concepts are fundamental to our approach.

IV. Methods

Within this section the methodology is provided for all performed research. It provides the exact definitions of all relevant parts of the research, clearly mentions all changes made to the OmniAnomaly model, and defines the approach used to get specific insights.

A. Definitions

It is crucial to clearly provide definitions of the important aspects used in this research. These definitions also determine some changes made to OmniAnomaly [16] to make the model work for the mobile robot of this study.

Firstly, the most important definition is what a failure is within this context. A failure within this study is defined by the situation where the robot is completely lost within its barn, and cannot return to its base station on its own. It requires manual intervention, and it will shut itself down in order to preserve battery charge for a user to drive it back to its base station. Before this failure the robot may have already executed several recovery actions, trying to find its way back on its route. When unsuccessful, the robot stops, which is the exact moment of failure. For this study, we disregard hardware failures and consider only failures as a result of its executed behavior, and its environment.

Secondly, there is the definition of an anomaly. Within this context, an anomaly is the deviation from nominal behavior potentially resulting in failure. This is a broad definition, as an anomaly can be the representation of various root causes. A root cause is the reason an eventual failure occurs. This can be a combination of different smaller events during a route, or one big event that on its own was enough to prevent correct execution. For example, a root cause in this context might be the ultrasonic sensor being obstructed by manure. Due to this, the robot cannot measure the distance to the wall on one side, and therefore eventually becomes more uncertain about its current location. The incorrect ultrasonic measures are the anomaly in this example. When this happens at a crucial part of a route, the localization might get so bad that the robot gets completely lost and starts to take the wrong actions. Finally, when the robot gets lost, does not manage to recover and decides to preserve battery and

Table 1: High-level overview of the datasets used within this paper.

#	Name	Dimensions	Train size (time steps)	Amount of train routes	Test size (time steps)	Amount of test routes
1	raw_motor_velocities	4	82,532	Nominal: 30	43,604	Nominal: 11 Anomalous: 5
2	motor_velocity_offsets	2				
3	raw_motor_velocities_with_flags	7				
4	motor_velocity_offsets_with_flags	5				
5	motor_velocities_combined_with_flags	9				
6	motor_velocities_combined_no_flags	6				
7	ultrasonic_raw	2	128,410	Nominal: 30	56,734	Nominal: 11 Anomalous: 4
8	ultrasonic_offsets	2				
9	ultrasonic_combined	6				
10	big_set_motor_velocities	6	528,846	Nominal: 178	238,856	Nominal: 60 Anomalous: 37
11	big_set_ultrasonic	6				
12	big_set_combined	12				

wait for manual intervention, a failure occurs.

In addition to the definition of an anomaly and failure, we have the routes of a robot. A route is a single round trip from the robot's base station[3] around the barn, ending back again at its base station. Any route execution that starts and ends at the base station without any recovery actions during the execution, is declared as a nominal route. If it does not get back to its base station, a failure must have occurred, and the route is declared anomalous. Importantly, this does not mean that all data points within an anomalous route are anomalous but rather that the route contains anomalies.

Only the data points contributing to the moment the anomaly threshold is exceeded are considered the anomaly. For example, a route execution might be normal for the first 20 minutes, after which several anomalous data points come up, and the model predicts an upcoming failure. As a result of this, the full route is declared anomalous, and the data points just before the moment of prediction are appointed as the anomaly, which is the embodiment of some underlying root cause. Ideally, when these anomalous data points are detected before the failure, this detection signal can be used to prevent the upcoming failure from happening at all.

B. *Changes to OmniAnomaly*

First, OmniAnomaly is adapted to predict failures on route-level instead of anomaly-level. Unlike the original approach that focuses on individual anomaly peaks, our method accounts for the nuanced nature of robotic route execution. During nominal operations, the robot can experience uncertainties without necessarily compromising the completion of the route. We introduce a sliding window approach to smooth the model's output, which allows us to distinguish between temporary uncertainties and persistent anomalies that indicate potential route failure. By calculating the average output over the last n minutes of execution, we can better detect when uncertainty progressively accumulates to a critical level. This approach offers two key advantages: it provides a more robust failure prediction mechanism in this context, and enables potential root cause identification by maintaining the temporal context of the model's output. The sliding window length was treated as a hyperparameter and optimized separately, with details provided in Section IV.D. Similar to OmniAnomaly [16], we still compare the adjusted output against an optimal threshold to predict failures. While an alternative approach might involve using the overall average output, the sliding window method preserves the critical temporal relationship between uncertainty and route performance which enables potential root cause identification.

Moreover, when detecting anomalies on route-level, the quantitative evaluation steps of OmniAnomaly had to change too. Instead of classifying each individual time step as a false or true positive or negative, we want to classify routes as such. Equations (8) and (9) below show how the model output at all time steps can be evaluated either using anomaly-level prediction or route-level prediction.

$$\text{anomaly\_level}(t_i, \theta) = \begin{cases} 1 & \text{if } t_i \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\text{route\_level}(T, \theta) = \begin{cases} 1 & \text{if } \exists t_i \in T : t_i \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

A full route execution T exists of multiple time steps t_i, which on anomaly-level are all individually compared to threshold θ or on route-level checked on the existence of any time step t_i exceeding θ. Note here the different scope of both functions: anomaly-level returns a prediction classification for each provided time step, and route-level returns a prediction classification for a full route. During the evaluation process it is thus required to have information about which time steps belong to which route.

Finally, the validation portion parameter, on which the training data is validated at each batch step, is lowered from 30% to 5%. This reduction is primarily motivated by computational efficiency. It represents a pragmatic choice rather than a rigorously tuned hyperparameter. The reduced validation set significantly decreased training time without compromising model performance, as shown later by the results in Section V.

C. *Hyperparameter Tuning*

The two hyperparameters that require separate tuning are the sliding window size and the z-space size. For both of these parameters, the process of tuning is described below.

[3] Within this context the base station is the charging station of the robot. In general, the base station simply indicates the robot uses the same start and end location.

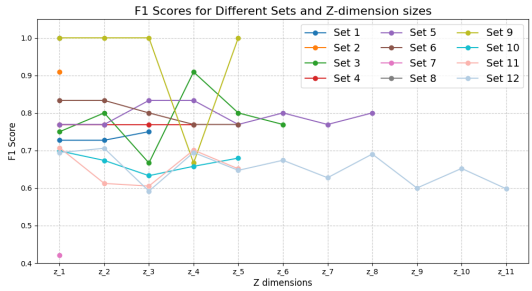


Figure 6: The F1-scores per dataset for multiple z-space sizes. Only scores are stated where the z-space size is smaller than the dimension size of that dataset.

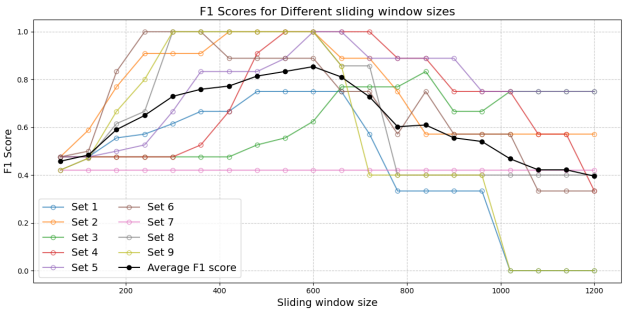


Figure 7: The F1-scores per dataset for each sliding window size, and the overall average per sliding window size.

Table 2: The selected z-space sizes for each dataset, along with the dimensions there originally are in the datasets and the obtained F1-score with this z-space size.

#	Name	Dimensions	Z-space	F1-score
1	raw_motor_velocities	4	3	0.75
2	motor_velocity_offsets	2	1	0.91
3	raw_motor_velocities_with_flags	7	3	0.67
4	motor_velocity_offsets_with_flags	5	3	0.77
5	motor_velocities_combined_with_flags	9	3	0.83
6	motor_velocities_combined_no_flags	6	2	0.83
7	ultrasonic_raw	2	1	0.42
8	ultrasonic_offsets	2	1	1.00
9	ultrasonic_combined	6	3	1.00
10	big_set_motor_velocities	6	1	0.70
11	big_set_ultrasonic	6	1	0.71
12	big_set_combined	12	2	0.71

Z-space The z-space size is determined first by evaluating the performance of each set for different sizes. Hereby setting the input for each dataset and all other hyperparameters to be independent variables, and the z-space hyperparameter to be dependent. The performance of each z-space size is evaluated with the F1-score, shown in Equation (10). The sliding window size for this hyperparameter study is chosen to be an arbitrary length of 5 minutes. For this tuning experiment, a few things should be kept in mind. First, the z-space should be strictly smaller than the dimension size in the dataset. Otherwise, the model can simply learn an identity transformation or a more complex representation, rather than discovering meaningful latent factors from the data itself. Having a latent space larger than the original feature space thus undermines the purpose of representation learning. Secondly, each set has its own optimal z-space size, which means all sets should be evaluated independently. Finally, there is a loss of visualization opportunity when a z-space of 4 and bigger is chosen (for example, plots similar to those in [38] are then no longer possible). Z-space sizes of 3 or less therefore have a slight advantage compared to higher sizes. The resulting F1-scores of this study can be seen in Figure 6. The final chosen z-space sizes can be seen in Table 2. There is one value that is selected at a suboptimal z-space size, which is set number 3. The obtained F1-score for this z-space is believed to be an outlier, which was later supported by the results shown in Table 3 where set number 3 performs similar to the other sets with flags.

Sliding Window For the sliding window size, the nine smaller datasets are used to determine the optimal size. Since this hyperparameter is model-wide, it is not necessary to tune this on the bigger three datasets as well. All window sizes between 30 seconds and 10 minutes with steps of 30 seconds were evaluated on the nine datasets based on their F1-scores (10). The z-space sizes for each dataset are already found at this time and are set to the values shown in Table 2. In Figure 7, the results of the sliding window tuning can be seen. The optimal sliding window size is 600 time steps, which is equal to 5 minutes as time step frequency for all datasets in this paper is 2Hz.

D. *Experiments*

With the problem context described in Section III, the concept definitions provided in Section IV.A and the model changes and hyperparameters stated in the Sections above, we are now able to perform research. There are four experiments described below and later evaluated in Section V.

Feature Engineering For this novel field for failure prediction, it is useful to show in detail which feature engineering steps are beneficial for the deep learning model's performance. The smaller datasets (sets 1-9), described in Section III.C are used in a feature ablation study [45]. 10 independent train and test runs are performed per dataset, where each dataset contains different feature combinations. The feature combinations can be found in Table 4. Sets 1-6 and sets 7-9 only contain data produced by the execution of a single route, where the failures in the anomalous routes are also the result of a single type of root cause. This is described in Section III.C and can also be seen in Table 5. The learning model and all its hyperparameters are kept constant, so the only variable under test is the composition of the datasets. The performance of the feature combinations is evaluated using the average F1-score, precision, and recall over all 10 runs per dataset, calculated as shown in the equations below. All in all, this full experiment will show the impact of the feature combination on the performance.

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{10}$$

$$precision = \frac{TP}{TP + FP} \tag{11}$$

$$recall = \frac{TP}{TP + FN} \tag{12}$$

Within the equations above, TP, FP, and FN represents the amount of true positives, false positives and false negatives respectively.

Performance Evaluation As failure prediction for mobile robots is a rather unexplored field, it is useful to evaluate the newfound performance within this field. The experiment with the smaller datasets yields a good feature combination for the used sensor input types. As is presented later in Section V.A, the found optimal feature combinations of both dataset types are dataset six and nine. With this optimal feature combination, three new datasets are created to show the current performance of failure prediction in the field of mobile robots. The three newly created datasets are more complicated than the smaller datasets. The datasets contain routes of multiple farms, driven by 6 Discovery Collector robots. The failures in the anomalous routes have various underlying root causes, which can be found in Table 5. Each big dataset is trained and tested 10 times independently in order to get more consistent results. The learning model's hyperparameters are kept constant, and the performance is measured using the average F1-score (10), precision (11), recall (12) and lead time per route. The lead time is the time difference in minutes between the prediction and moment of failure [46]. The time of prediction is the first occurrence of the sliding window average model output reaching above the threshold. If no failure is predicted, but the route does end in one, the lead time is set to 0. Besides these metrics, the performance of this setup also provides insight into the generalizability of the nominal behavior of the Discovery Collector robot. This is because the data in this experiment contains execution data of multiple routes, and is much more representative when compared to the datasets of performed feature experiments above. Besides the evaluation of the datasets with the aforementioned metrics, there are several other potentially interesting experiments to be performed, which have not been performed in this study, but are stated later in Section VI.D.

Heatmaps for Root Cause Analysis An experiment is also set up to investigate the possibility of root cause classification by using heatmaps. Currently, interpreting these heatmaps requires significant expert domain knowledge, which limits their immediate operational utility. The heatmaps are constructed from the output of the model. The mobile robot has a localization algorithm providing global pose information at each time step, and the anomaly detection model which provides output at each time step (before the sliding window application). These two data streams can be combined into a heatmap by applying a Kernel Density Estimator (KDE) [47], providing visual information of the aggregated anomaly score output matched to the locations of where the robot was in 2D space. To be able to use heatmaps for root cause classification, several hypotheses needs to hold. First, it is necessary that the heatmap's hotspots are not linked to the location of failure, but ideally match the location of occurring root causes. Second, to be able to effectively classify root causes, the heatmaps must be consistent enough or have some form of common pattern, in order to find significant differences in the spatial patterns between different root causes, allowing for both visual inspection and quantitative comparison through distance metrics. Third, the heatmaps or spatial patterns within the heatmaps for different root causes should be unique, as otherwise multiple root cause types would end up in the same category. Another use of the heatmaps is that the heatmaps of individual route executions can also be combined into an averaged heatmap, which potentially yields insights into risky or difficult parts of the route. However, drawing meaningful conclusions from these aggregated heatmaps remains challenging and currently depends heavily on expert interpretation.

V. Results

This section will present all the results gained from the experiments described in Section IV.

A. Feature Engineering Performance

The results of the 10 independent train and test runs of the smaller datasets can be seen in the first nine rows of Table 3. It is clear that the model is able to consistently get a high recall score, on average 0.98, while the precision generally falls a bit behind with an average of 0.88.

Table 3 also shows an improvement in performance for both the addition of the flags and the feature engineered offsets. I.e., the F1-score of dataset one of 0.83 improves to 0.89 in dataset three with the addition of flags, and to 1.00 in dataset two with the addition of the feature engineered offsets. When examining the flags in more detail however, it appears that dataset four with the flags performs slightly worse than dataset two without the flags. Both feature additions thus enhance the performance, but the offsets on their own perform better than when combined with the flags. As is mentioned in the description of the Feature Engineering in Section IV.D, datasets six and nine are selected for the bigger, more general datasets, even though they performed equally well as datasets two and eight. The reason for this is that the context used for training and testing of the smaller datasets is very simplistic (one type of route, one type of root cause, see also Table 5) and the bigger datasets require a more general understanding of the nominal behavior of the robot. Therefore, the feature combinations with both the raw and the offset values are presumed to be more reliable when applied on a more complex learning task.

B. General Dataset Performance

The performance of the bigger three datasets can be found in the lower three rows of Table 3. Notably, the scores are lower than the smaller nine datasets. Similar to the smaller datasets, the model is able to achieve a higher average recall of 0.73, and a somewhat lower

Table 3: Results of 10 independent train and test runs on the different datasets. Note that the metrics are the average values of all 10 runs, thus the precision and recall value in the table are not the exact values used to calculate the F1-score, as that metric is an average of 10 runs too.

#	Name	F1-score	Precision	Recall
1	raw_motor_velocities	0.83	0.79	0.92
2	motor_velocity_offsets	**1.00**	**1.00**	**1.00**
3	raw_motor_velocities_with_flags	0.89	0.92	0.90
4	motor_velocity_offsets_with_flags	0.99	0.98	**1.00**
5	motor_velocities_combined_with_flags	0.98	0.97	**1.00**
6	motor_velocities_combined_no_flags	**1.00**	**1.00**	**1.00**
7	ultrasonic_raw	0.44	0.28	**1.00**
8	ultrasonic_offsets	**1.00**	**1.00**	**1.00**
9	ultrasonic_combined	**1.00**	**1.00**	**1.00**
10	big_set_motor_velocities	0.64	0.61	0.74
11	big_set_ultrasonic	**0.76**	**0.78**	**0.75**
12	big_set_combined	0.72	0.75	0.71

average precision of 0.71. The model thus detects a slightly higher proportion of true positive predictions at the cost of introducing more false positive predictions.

Insight into the decision-making of the model is shown in Figure 9. These figures show examples of the offsets of each feature type, which are the most insightful features, next to the model output of the model trained with dataset 12, the dataset with both input types combined. The dark red boxes in the figures indicate the moment the sliding window averaged output of the model reaches above the threshold. The light red box shows the 5 minutes before the moment a failure is predicted, which is relevant as these 5 minutes are all time steps that contributed to the moment of the failure prediction. Note, that this is due to the application of a sliding window average over the output of the last 5 minutes. Figures 9a, 9b and 9c show the input of an anomalous route execution which eventually ends in failure. Each of these figures have a different underlying root cause, which can be recognized in the input offset data. Figure 9d shows a nominal route execution to put the observed values into perspective.

The average lead times of a prediction before a failure are shown in Figure 8. It can be seen that most of the routes have a relatively high lead time when compared to the average duration of each route in Table 5. The higher the lead time, the earlier the failure is predicted within the route on average.

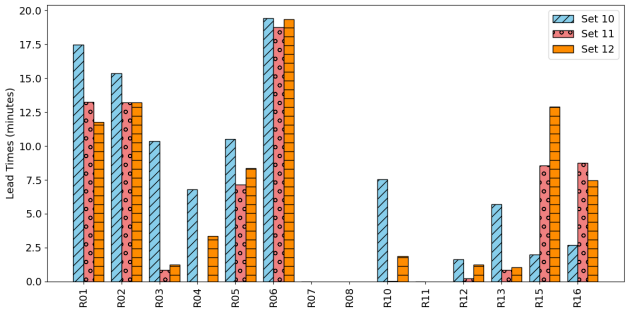


Figure 8: The average lead time before failure of the bigger datasets per route over all 10 test runs. Note that the bars that are zero, indicate false negatives on average for that route, as there was no lead time of the prediction before failure.

C. Heatmap Visualizations

Several created heatmaps can be found in Figure 10. These examples can be used to show which of the hypotheses stated in Section IV hold. Figures 10a, 10b and 10c show heatmaps of individual route executions, including the robot estimated failure location, the locations where the robot was in recovery mode, and the path the robot has taken over the route. The red hotspots on the heatmaps show the locations on the map where the model output is high. Figure 10d shows the aggregated average heatmap of multiple executions of the same route, providing a more general insight into the locations where the model output over multiple executions was high.

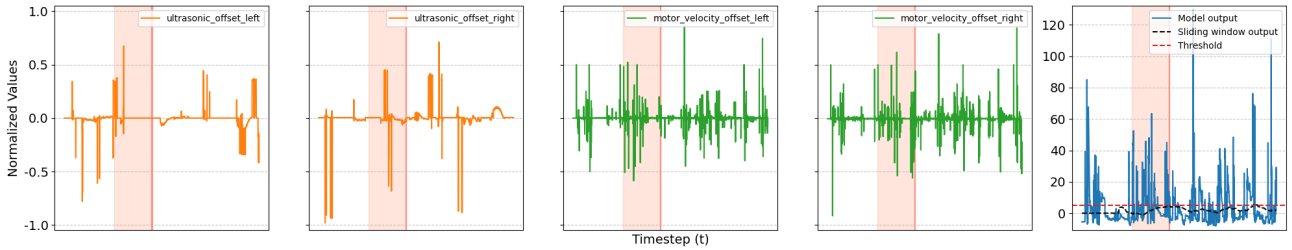
VI. Discussion

This section discusses how the research question, stated in Section I, has been addressed through the experimental results and subsequent analysis. The results show that while the current model's performance is not yet suitable for practical deployment, it provides promising insights for future research.

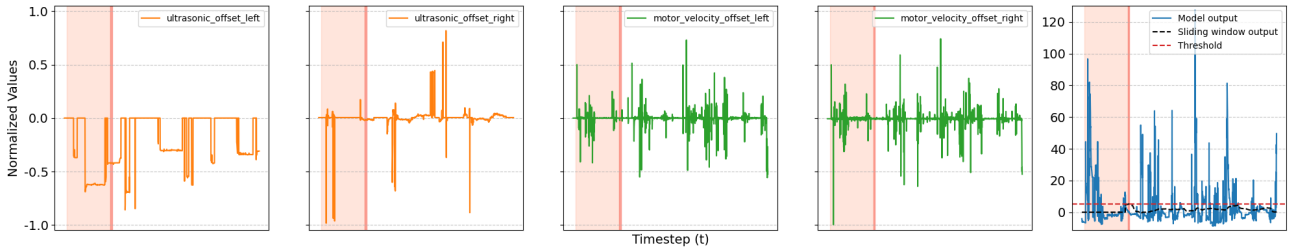
A. Feature Engineering Insights

As can be seen in the first nine rows of Table 3, both the addition of flags and offsets to the datasets improves the performance of the model in all scenarios. However, the combination of both feature engineered data streams does not perform better than the offset data streams on their own. This insight addresses the first contribution of the paper: assessing the effectiveness of two feature types on model performance.

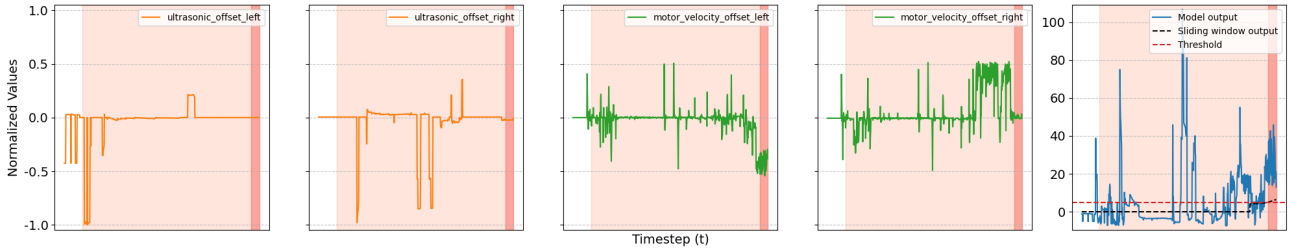
The insights of Figure 9 can be explained in more detail by a physical link that is found when examining the relation between the input and output of the model. An example of this is shown in Figure 11. Within this figure, the motor velocity offsets input and model output is shown from the test set of dataset six. As can be seen in the figure, the peaks of the model output in blue clearly overlap with sudden jumps in the motor velocity offsets fed to the model as input. This coincides with physical action where the robot changes driving direction or encounters unexpected bumps. In Figure 11, the multiple smaller peaks coincide with driving direction changes of the robot, whereas the larger peaks, inside the dashed red boxes, coincide with unexpected bumps during the route. The fact that the directional changes have smaller peaks compared to the unexpected bumps might be the result of the fact that directional changes are common within a route execution, thus the model might have incorporated this into its latent space representation as a possibility. Unexpected bumps on the other hand, are much more uncommon and ideally should not happen at all. Therefore, it seems likely that the latent space representation does not model for this uncommon moment during routes, and as a result, when it does occur, the model is less able to anticipate this and yields a higher out-



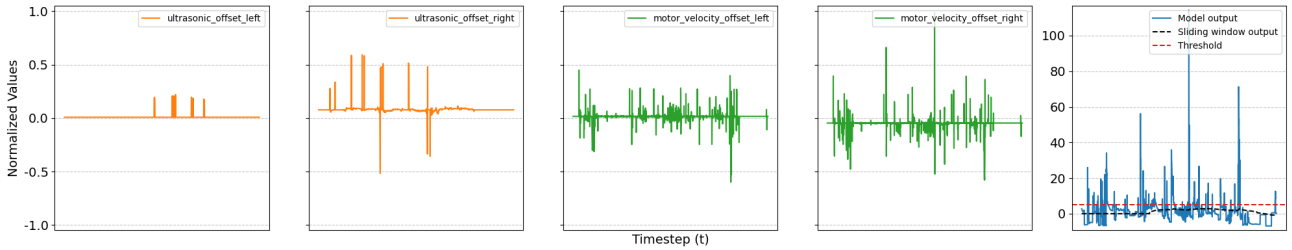
(a) Root cause: Lots of slip and unexpected bumps. Route: R2



(b) Root cause: Dirty ultrasonic sensor. Route: R2



(c) Root cause: Low obstacle blocking the route. Route: R15



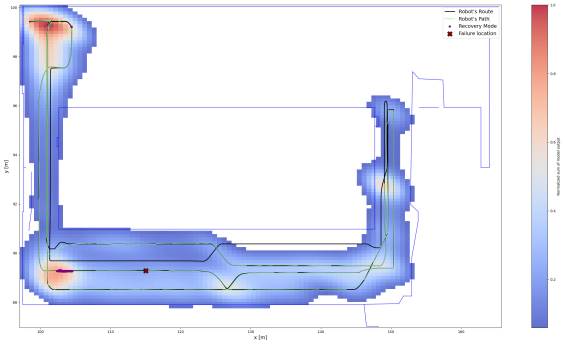
(d) Nominal route input and its output. No failure is predicted as the threshold is not exceeded. Route: R9

Figure 9: The input and output of a few scenarios. The dark red box indicates the moment the sliding window output exceeds the threshold and a failure is predicted. The light red box shows the sliding window contributing time steps before failure prediction. The details of each route can be found in Table 5.

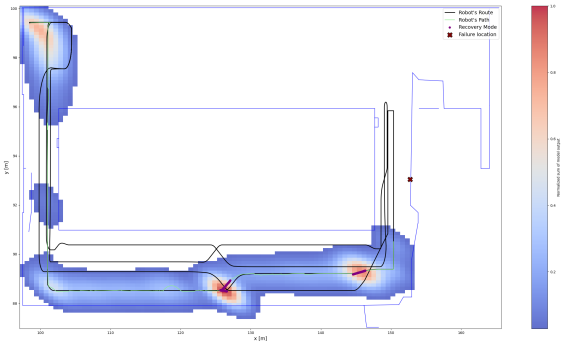
put. Note that this is a hypothetical assumption of the inner workings of the model, and that this is hard to confirm as the model is considered to be a black box [48]. To confirm this assumption, an experiment could be set up with a mobile robot dataset where the moments of bumps are known. This would allow for the physical link to be captured in a metric [49]. Nevertheless, the difference in the height of the output works in our advantage. Namely, directional changes are part of nominal behavior, and should not be detected as an anomaly, whereas the unexpected bump indicate wrong localization or obstacles, which should be detected as an anomaly as they are good indicators of upcoming failures.

This found physical link is not limited to dataset six or motor velocities specifically. The same link is present in all 12 datasets and both for ultrasonic measures and motor velocities. Moreover, the physical link between input and output yields a new hypothesis that is promising for future research. The model currently seems to not always be able to anticipate sudden state changes, but has seemingly learned in a way that directional changes are more common than unexpected bumps. This may indicate that the addition of other data streams, for example, indicating upcoming directional changes (like the combination of the distance driven and distance to drive for a current action), enables the model to learn more accurate latent space representations resulting in better anticipated sudden state changes. In turn, this hopefully results in an even bigger difference between nominal and anomalous model output, making the overall failure prediction problem easier.

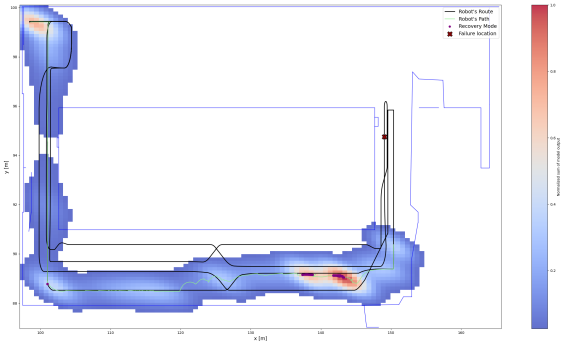
Section V.B provided the performance of the bigger, more general datasets in the bottom three rows of Table 3. These current results are not very suitable yet



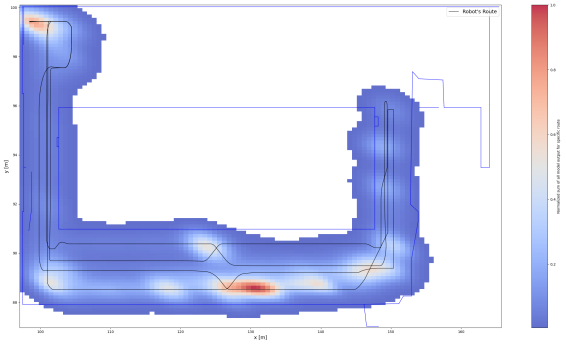
(a) Heatmap 1 with a fixed ultrasonic measure as root cause.



(b) Heatmap 2 with a fixed ultrasonic measure as root cause.



(c) Heatmap with wheel slip as root cause.



(d) Aggregated average heatmap of a specific route made from multiple heatmaps based on executions of this route.

Figure 10: Figures 10a, 10b and 10c show a heatmap of a single route execution, Figure 10d shows a heatmap of all executions on the same route, summed and averaged.

to be used in practice, as it would result in a lot of false alarms due to the low precision values, which ultimately does not result in more autonomy or uptime of the robot. The exact performance required for the model to be applicable, depends highly on the execution context of the robot at hand and the prevented

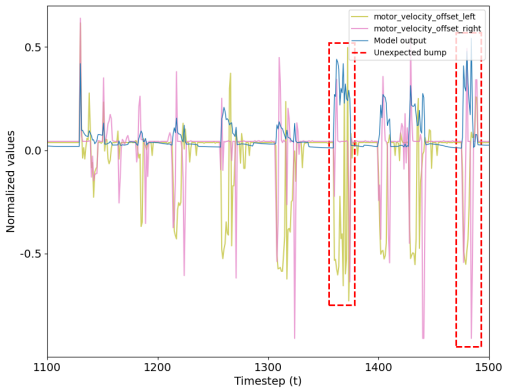


Figure 11: The output of the model overlaid with the motor velocity offset input streams at specific time steps on a route, highlighting the clear physical link between the input and the output of the model, and indicating the larger peaks at unexpected bumps.

amount of downtime as a result of the predicted failures. Altogether, the results do show the possibility of a working solution for the failure prediction problem. There are also several promising directions of improvement for this problem that have not been explored in this study. More about this in Section VI.D below.

Finally, a remark must be made about the lead times stated in Figure 8. The figure nicely shows that most failures in the dataset can be predicted ahead of time, but it does not link these predictions to actual underlying root causes. It might now be that the first moment a failure is predicted is a hallucination of the model that is not necessarily linked to the eventual failure. For example, when looking at Figure 9a, there are two moments the threshold is exceeded. The first one is shown in the figure in dark red, but near the end of the route execution, there is another. As there are no root cause timeframe labels, it is currently not possible to say which of the two (or both) is the actual root cause of the eventual failure. Currently, only the first occurrence of threshold exceedance is used, which might result in higher lead times. Despite the current ambiguity in pinpointing exact root cause timeframes, this analysis provides initial evidence of the model's ability to predict failures ahead of time, and calls for further research to obtain accurate lead times [46].

B. Root Cause Identification

Figure 9 provides a more elaborate insight into the workings behind the model. As explained earlier in Section V.B, Figures 9a, 9b, and 9c show the timeframe contributing to a failure prediction made by the model of three route executions with three different underlying root causes, and Figure 9d shows a nominal route execution without any predicted failure. The visualizations only go over three routes containing root causes, but they show a general trend in most of the correctly predicted route executions ending in failure. Figures 9a and 9c both show a failure where the motor velocities are indicative of the underlying root causes. In both

of these figures, the 5 minutes of the sliding window contributing to the predicted failure show anomalies in their motor velocity offset values as well. For Figure 9a there is a lot of jitter in the right wheel's motor velocity offset and an indication of an unexpected bump just before the predicted failure. In Figure 9c it is clearly visible in the right wheel's motor velocity offset that the target motor velocity is not achieved, as the green line is far from 0. Figure 9b ends in a failure where the ultrasonic sensors are indicative of the underlying root cause, namely a dirty ultrasonic sensor. This can also clearly be seen in the plot of the offset of the left ultrasonic sensor, where the offset is far from 0 in the light red sliding window before the failure. Interestingly, the failure in Figures 9a and 9b are predicted a long time before the actual failure occurs, which happens after the last time step visible in the figures. The model output of this initial anomaly is already detrimental enough such that the model predicted that the route execution would end in failure. Figure 9c however, predicts a failure only right before it actually happened. The underlying root cause of the failure in this case is the route of the robot being obstructed by some unknown object or cow. Even though generally there is less indication before the occurrence of such failure as compared to the dirty ultrasonic sensors or multiple unexpected bumps during a route, the model is still able to predict such failures before they happen.

C. Heatmap Interpretation

Since the heatmaps are visualizations of the model's output per route, and we found above that often the model is good at having higher model output at the moments of root causes, the heatmaps can be used to potentially visualize the problematic locations within a route. As Section V.C described, there are several hypotheses that need to hold to enable root cause classification by using the heatmap images. The first hypothesis, stating that the location of failure should not be coinciding with the hotspots in the heatmaps, holds. Figures 10a, 10b, and 10c show this, as the heatmap hotspots do not coincide with the location of failure shown in the figures. The second hypothesis however, does not hold. This can be concluded from Figures 10a and 10b. The two heatmaps in these figures are both based on a failure which have the same underlying root cause, determined by domain experts at Lely Industries. The root cause in this case is one of the ultrasonic sensors giving a fixed value. As the hotspot patterns are different for both of these heatmaps, clustering these images to the same cluster in order to more easily extract similar root causes seems unlikely. Moreover, root causes do not follow unique patterns. Figure 10c has a different underlying root cause, namely wrong localization due to wheel slip, but shows a similar hotspot pattern as Figure 10b. Consequently, root cause classification by using heatmaps is not so feasible for this study.

Nonetheless, as can be seen as well in Figures 10a, 10b and 10c, there is a connection between the hotspots in the heatmap and the locations where the robot executes recovery behavior. This confirms the ability of the model being able to detect deviations from nominal behavior, as recovery actions are not part of normal route execution. Besides, the heatmaps therefore provide an intuitive representation of potential problematic parts in a single route. Note however, that the hotspots are not only related to the problematic subsequent actions in a route, as the model output also takes into account the temporal context of execution in the dynamic environment. An object blocking the route for example, also results in higher model output (for example, as shown in Figure 9c earlier). The hotspots on their own therefore do not necessarily indicate problematic sections in the route. A solution to get a more general insight into the problematic sections of a route is to aggregate the heatmaps into the average heatmap of multiple executions of the same route. By averaging the values for all locations in the heatmap, the hotspots related to incidental root causes are averaged out, and the hotspots due to structurally difficult locations remain, as these hotspots are always in the same location. Such a heatmap is shown in Figure 10d. Since the problematic sections of a specific route are always in the same location, the aggregation of multiple heatmaps of that route will still show a higher model output in those locations, whereas the routes that show high model output related to environmental changes and obstacles averages out and are less prevalent in the resulting heatmap. There is however a big limitation to these insights, namely that there are no ground truths available for the datasets used within this work for the root cause locations or exact timeframes of the root causes. The gained insights in this section now rely on the expert domain knowledge of this particular robot, thus still require labelled data for quantification. More scientific confidence can therefore be gained by a future study using datasets where these ground truths are available for further evaluation. All in all, the current section shows how both the insights into root causes, and the insight into the heatmaps provide our third research contribution, providing preliminary understanding of the relationship between the learning model's input, output, and the underlying root causes of predicted failures.

D. Limitations and Future Research

Currently, this work provides a first investigation into the performance of failure prediction for mobile robot using anomaly detection. Its findings however, may be specific to this context and require further validation. Research is required to validate the broader applicability of failure prediction using anomaly detection in the field of mobile robots. A valuable study would be to perform a sensitivity analysis [50], for example similar to [51], who have extended first-order sensitivity analysis to generative models. Such study will result in a better understanding of the relation between the input

and the inner workings of the model. Another seemingly useful study is a Bootstrap analysis [52] in order to better differentiate between variance and shortcomings of the model's performance. While the current study did not resample datasets, the use of multiple independent train and test runs and averaging of performance metrics provides some protection against model variance. A Bootstrap analysis similar to [53] however, can strengthen this belief even more.

It is also worth mentioning separately that the performance of the algorithm in this work is relatively low compared to other works in the different fields of anomaly detection. For Network Intrusion Detection Systems (NIDS) the state-of-the-art performance has F1-scores of above 0.95, see for example [54] (p.9399) with an elaborate overview of the performances and the limitations of several works. The performance for smart manufacturing applications lies somewhat lower, more ranging between 0.80-0.95, see for example [27]. Both fields thus achieve much higher performance compared to the results of this study with F1-scores of about 0.64-0.76. Needless to say that the field of mobile robots is complex, and the particular context of this study has a very uncertain and dynamic nature. Besides, this study focused on the most promising parts of the study. For example, the feature engineering process, however valuable, can be extended greatly. There are also many more sensors present within the field of mobile robotics, and even with the existing feature combinations of this work, not all subsets have been evaluated.

Finally, something that received little attention in this work, is the latent space representation that comes with the use of Variational Autoencoders (VAE). As we understand more and more about VAEs, the more use is found for the encoded latent space representation. Specifically, the aforementioned paper by Azzalini et al. [38] achieved a lot by evaluating the latent space of their dataset, as mentioned earlier in the paper. They proposed a way to cluster different root causes by the samples taken from the latent space distributions, instead of finding root causes after decoding. Since good root cause analysis is one of the goals of anomaly detection, this is a huge achievement, and it can likely be extended to more fields. Su et al. with OmniAnomaly, as mentioned shortly in [16], also enable latent space visualizations, but have not attempted further classification within this space, which might be very insightful. Another option for potential root cause classification is investigating the correlation between the individual feature loss contributions and the root causes leading to failures. This is shortly mentioned as an option in the end of Section III.B, but is not investigated further within this research.

Another interesting and promising direction for this field, in line with the found physical link between the model's input and output, is the approach of majority voting, as proposed in [55]. Alshede et al. combine several models to improve parallel computational loads, while preserving the achieved performance with other state-of-the-art models. The approach of majority voting itself is very promising for the failure prediction problem of mobile robots. Mobile robots usually have multiple sensors or controlled variables, thus the same model can be applied on each individual sensor or variable, combined by a majority voting layer. This approach would enable better root cause analysis as well, as it gives the opportunity to identify the individual sensors which contributed to the final decision, without the need for manual evaluation of all input sensors.

VII. Conclusion

Within this work we successfully showed the applicability of anomaly detection models for the use of failure prediction in the field of mobile robots. The performance of this novel application is clearly stated by three general datasets, achieving F1-scores between 0.64-0.76. The process of selecting effective indicators of failure for the context of the robotic application used within this paper is stated in the results and explained in the discussion, and shows a promising advantage in the use of offsets, comprised of target or expected features subtracted from the currently measured features. At the same time the results also show an increasing performance with the use of discrete flags as indicators in the datasets. The performance however can still be enhanced further to enable satisfactory failure prediction in practice. The study therefore highlights multiple potential directions for further research. Namely, we identified a physical link between the input and output of the model, and the accompanied heatmap visualizations showcase its potential for identifying specific root cause locations during route executions. Furthermore, a change in the underlying model used for failure prediction in the direction of majority voting appears promising for the performance and has the bonus of enabling even more concrete root cause analysis. Moreover, this work clearly states the limitations in the research; the amount of feature combinations used to find the optimal feature combinations, the lack of ground truth timeframe labels of root causes for capturing the performance of root cause identification, and, being the first work in the field, missing extensive other work to compare to. All in all, this research lays the groundwork for more intelligent, proactive failure prediction in mobile robotics, enabling the transition from reactive maintenance towards predictive precision.

VIII. Acknowledgements

The authors acknowledge the use of Claude (Anthropic, 2024), a Large Language Model (LLM), to improve the readability and clarity of certain sections within this work. All intellectual content, analysis, and conclusions remain the authors' own work.

References

[1] A. K. Adeleke, D. J. P. Montero, K. A. Olu-lawal, and O. K. Olajiga, "Process development in mechanical engineering: innovations, challenges, and opportunities," *Engineer-*

ing Science & Technology Journal, vol. 5, no. 3, pp. 901–912, 2024.

[2] I. Ullah, D. Adhikari, H. Khan, M. S. Anwar, S. Ahmad, and X. Bai, "Mobile robot localization: Current challenges and future prospective," *Computer Science Review*, vol. 53, p. 100651, 2024.

[3] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, "Using a long short-term memory recurrent neural network (lstm-rnn) to classify network attacks," *Information*, 2020.

[4] A. Alvanpour, S. K. Das, C. K. Robinson, O. Nasraoui, and D. Popa, "Robot failure mode prediction with explainable machine learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 61–66, IEEE, 2020.

[5] A. Farid, D. Snyder, A. Z. Ren, and A. Majumdar, "Failure prediction with statistical guarantees for vision-based robot control," *Robotics: Science and Systems*, 2022.

[6] C. Gokmen, D. Ho, and M. Khansari, "Asking for help: Failure prediction in behavioral cloning through value approximation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2023-May, pp. 5821–5828, 2023.

[7] H. Ichiwara, H. Ito, and K. Yamamoto, "Real- time failure/anomaly prediction for robot motion learning based on model uncertainty prediction," *2024 IEEE/SICE International Symposium on System Integration (SII)*, pp. 376–381, 1 2024.

[8] A. Bala and I. Chana, "Intelligent failure prediction models for scientific workflows," *Expert Systems with Applications*, vol. 42, no. 3, pp. 980–989, 2015.

[9] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, pp. 1–42, 2010.

[10] V. Chandola, A. Banerjee, V. K. A. computing surveys (CSUR), and undefined 2009, "Anomaly detection: A survey," *dl.acm.org*, vol. 41, pp. 1–22, 2009.

[11] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102675, 2022.

[12] Q.-T. Nguyen, T. N. Tran, C. Heuchenne, and K. P. Tran, "Decision support systems for anomaly detection with the applications in smart manufacturing: a survey and perspective," in *Machine Learning and Probabilistic Graphical Models for Decision Support Systems*, pp. 34–61, CRC Press, 2022.

[13] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, "Deep learning for medical anomaly detection–a survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–37, 2021.

[14] H.-T. Duong, V.-T. Le, and V. T. Hoang, "Deep learning-based anomaly detection in video surveillance: A survey," *Sensors*, vol. 23, no. 11, p. 5024, 2023.

[15] Y. Yuan, J. Yang, R. Duan, I. Chih-Lin, and J. Huang, "Anomaly detection and root cause analysis enabled by artificial intelligence," in *2020 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2020.

[16] Y. Su, R. Liu, Y. Zhao, W. Sun, C. Niu, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. pp, pp. 2828–2837, 7 2019.

[17] Scopus, "Scopus Database Search on Mobile Robot Failure Prediction and Anomaly Detection," nov 2024. Search query: TITLE-ABS-KEY (("mobile robot*" OR "mobile robotics" OR "autonomous robot*" OR "robotic system*") AND ("failure predict*" OR "fault detect*") AND ("machine learning" OR "deep learning" OR "neural network*" OR "statistical model*") AND ("anomaly detect*" OR "outlier detect*" OR "novelty detect*" OR "deviation detect*" OR "rare event detect*")).

[18] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

[19] M. B. Pranto, M. H. A. Ratul, M. M. Rahman, I. J. Diya, and Z.-B. Zahir, "Performance of machine learning techniques in anomaly detection with basic feature selection strategy-a network intrusion detection system," *J. Adv. Inf. Technol*, vol. 13, no. 1, 2022.

[20] I. Ahmad, Q. E. Ul Haq, M. Imran, M. O. Alassafi, and R. A. AlGhamdi, "An efficient network intrusion detection and classification system," *Mathematics*, vol. 10, no. 3, p. 530, 2022.

[21] Z. Li, P. Wang, Z. Wang, and D.-c. Zhan, "Flowgananomaly: Flow-based anomaly network intrusion detection with adversarial learning," *Chinese Journal of Electronics*, vol. 33, no. 1, pp. 58–71, 2024.

[22] K. D. Garcia, C. R. de Sá, M. Poel, T. Carvalho, J. Mendes-Moreira, J. M. Cardoso, A. C. de Carvalho, and J. N. Kok, "An ensemble of autonomous auto-encoders for human activity recognition," *Neurocomputing*, vol. 439, pp. 271–280, 2021. Figure 2.

[23] A. L. Alfeo, M. G. Cimino, G. Manco, E. Ritacco, and G. Vaglini, "Using an autoencoder in the design of an anomaly detector for smart manufacturing," *Pattern Recognition Letters*, vol. 136, pp. 272–278, 2020.

[24] S. Abbasi, M. Famouri, M. J. Shafiee, and A. Wong, "Outliernets: Highly compact deep autoencoder network architectures for on-device acoustic anomaly detection," *Sensors*, vol. 21, no. 14, p. 4805, 2021.

[25] M.-A. Tnani, M. Feil, and K. Diepold, "Smart data collection system for brownfield cnc milling machines: A new benchmark dataset for data-driven machine monitoring," *Procedia CIRP*, vol. 107, pp. 131–136, 2022.

[26] K. S. Lee, S. B. Kim, and H.-W. Kim, "Enhanced anomaly detection in manufacturing processes through hybrid deep learning techniques," *IEEE Access*, 2023.

[27] H. Yan, J. Wang, J. Chen, Z. Liu, and Y. Feng, "Virtual sensor-based imputed graph attention network for anomaly detection of equipment with incomplete data," *Journal of Manufacturing Systems*, vol. 63, pp. 52–63, 2022.

[28] C. Park, S. Lim, D. Cha, and J. Jeong, "Fv-ad: F-anogan based anomaly detection in chromate process for smart manufacturing," *Applied Sciences*, vol. 12, no. 15, p. 7549, 2022.

[29] A. Albanese, M. Nardello, G. Fiacco, and D. Brunelli, "Tiny machine learning for high accuracy product quality inspection," *IEEE Sensors Journal*, vol. 23, no. 2, pp. 1575–1583, 2022.

[30] A.-E. R. Abd-Elhay, W. A. Murtada, and M. I. Yosof, "A high accuracy modeling scheme for dynamic systems: spacecraft reaction wheel model," *Journal of Engineering and Applied Science*, vol. 69, no. 1, p. 4, 2022.

[31] L. Kou, J. Chen, Y. Qin, and W. Mao, "The robust multi-scale deep-svdd model for anomaly online detection of rolling bearings," *Sensors*, vol. 22, no. 15, p. 5681, 2022.

[32] L. Scime and J. Beuth, "Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm," *Additive Manufacturing*, vol. 19, pp. 114–126, 2018.

[33] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[34] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[35] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Roboads: Anomaly detection against sensor and actuator misbehaviors in mobile robots," in *2018 48th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pp. 574–585, IEEE, 2018.

[36] H. Wang, R. Fan, Y. Sun, and M. Liu, "Applying surface normal information in drivable area and road anomaly detection for ground mobile robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2706–2711, IEEE, 2020.

[37] D. Azzalini, A. Castellini, M. Luperto, A. Farinelli, F. Amigoni, *et al.*, "Hmms for anomaly detection in autonomous robots," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*

(AAMAS 2020), pp. 105–113, IFAAMAS, 2020.
[38] D. Azzalini, L. Bonali, and F. Amigoni, "A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots," *IEEE Robotics and Automation Letters*, vol. 6, pp. 2985–2992, 4 2021.
[39] T. Ji, A. N. Sivakumar, G. Chowdhary, and K. Driggs-Campbell, "Proactive anomaly detection for robot navigation with multi-sensor fusion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4975–4982, 2022.
[40] Lely Industries, "Lely discovery collector." `lely.com`. Accessed: Oct 14, 2024 [Online]. Available: `https://www.lely.com/nl/discovery-collector/`.
[41] X. Liu, L. Faes, A. U. Kale, S. K. Wagner, D. J. Fu, A. Bruynseels, T. Mahendiran, G. Moraes, M. Shamdas, C. Kern, *et al.*, "A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis," *The lancet digital health*, vol. 1, no. 6, pp. e271–e297, 2019.
[42] R. A. Rios and R. F. De Mello, "Improving time series modeling by decomposing and analyzing stochastic and deterministic influences," *Signal Processing*, vol. 93, no. 11, pp. 3001–3013, 2013.
[43] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.
[44] L. E. Rhijnsburger and Y. B. Eisma, "Exploring the applicability of failure prediction for 2d mobile robots." Unpublished paper, 2024.
[45] S. Sheikholeslami, M. Meister, T. Wang, A. H. Payberah, V. Vlassov, and J. Dowling, "Autoablation: Automated parallel ablation studies for deep learning," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, pp. 55–61, 2021.
[46] A. Das, F. Mueller, C. Siegel, and A. Vishnu, "Desh: deep learning for system health prediction of lead times to failure in hpc," in *Proceedings of the 27th international symposium on high-performance parallel and distributed computing*, pp. 40–51, 2018.
[47] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
[48] T. Carraro, M. Polato, and F. Aiolli, "A look inside the black-box: Towards the interpretability of conditioned variational autoencoder for collaborative filtering," in *Adjunct publication of the 28th ACM conference on user modeling, adaptation and personalization*, pp. 233–236, 2020.
[49] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.
[50] I.-C. Yeh and W.-L. Cheng, "First and second order sensitivity analysis of mlp," *Neurocomputing*, vol. 73, no. 10-12, pp. 2225–2233, 2010.
[51] J. Wu, K. Plataniotis, L. Liu, E. Amjadian, and Y. Lawryshyn, "Interpretation for variational autoencoder used to generate financial synthetic tabular data," *Algorithms*, vol. 16, no. 2, p. 121, 2023.
[52] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994.
[53] D. N. Politis, "The impact of bootstrap methods on time series analysis," *Statistical science*, pp. 219–230, 2003.
[54] Y. K. Saheed, A. I. Abiodun, S. Misra, M. K. Holone, and R. Colomo-Palacios, "A machine learning-based intrusion detection for detecting internet of things network attacks," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9395–9409, 2022.
[55] H. Alshede, L. Nassef, N. Alowidi, and E. Fade, "Ensemble voting-based anomaly detection for a smart grid communication infrastructure.," *Intelligent Automation & Soft Computing*, vol. 36, no. 3, 2023.

Appendix

A. Features per Dataset

The exact features used within each dataset can be found in Table 4 below.

Table 4: The features present within each dataset, matching the dimensions of Table 1.

Feature \Set	1	2	3	4	5	6	7	8	9	10	11	12
motor_velocity_left	X		X		X	X				X		X
motor_velocity_right	X		X		X	X				X		X
motor_velocity_target_left	X		X		X	X				X		X
motor_velocity_target_right	X		X		X	X				X		X
motor_velocity_offset_left		X		X	X	X				X		X
motor_velocity_offset_right		X		X	X	X				X		X
ultrasonic_left							X		X		X	X
ultrasonic_right							X		X		X	X
expected_ultrasonic_left									X		X	X
expected_ultrasonic_right									X		X	X
ultrasonic_offset_left								X	X		X	X
ultrasonic_offset_right								X	X		X	X
direction_flag			X	X	X							
turn_left_flag			X	X	X							
turn_right_flag			X	X	X							

B. Detailed Dataset Composition

A more detailed composition of the test set of each dataset can be found in Table 5 below.

C. Hardware and Training Specifications

For this study, all experiments were conducted on the NVIDIA GP104GLM Quadro P3200 Mobile 6GB GDDR5 GPU. With all hyperparameters similar to OmniAnomaly or stated otherwise within this paper, the full training time for 20 epochs of the model was about 80 minutes for each of the first nine smaller datasets, and 24 hours for the individual last three bigger datasets.

D. Peaks-Over-Threshold

Su et al. [16] introduced an unsupervised threshold selection algorithm in their work, called Peaks-Over-Threshold (POT). As this study used the OmniAnomaly model on new datasets, a study into the performance of the POT algorithm on these new datasets is also performed. POT fits the tail portion of a probability distribution to a generalized Pareto distribution using only two parameters which are model-wide and can be set empirically. Within [16] they showcase that applying POT comes at the cost of a slightly lower F1 score (0.003 - 0.077 lower), but enables a fully unsupervised way of finding a good threshold for the model's output without the need of an optimal threshold selection process which requires labels for threshold performance evaluation. The lower quantile level is tuned separately for this study. It is found by multiplying two ratios. First, ratio R_1 is the amount of time steps exceeding the threshold within an anomalous route divided by all route time steps, averaged over all anomalous routes within the datasets. Second, ratio R_2 is the routes that are anomalous compared to all performed

Table 5: A detailed overview of the composition of all test datasets. The root causes present within the anomalous routes can be a single root cause or a combination of multiple. For example, the last row only holds one anomaly of this route, which is a combination of both stated root causes.

Datasets	Nr. of routes in set	Route index	Nr. of turn actions	Total actions	Distance (m)	Average duration (min)	Composition	Root cause(s)
1-6	1	R1	19	52	173	28	Nominal: 11 Anomalous: 5	• Incorrect bump detected
7-9	1	R2	33	76	284	37	Nominal: 11 Anomalous: 4	• Dirty ultrasonic sensor
10-12	16	R1	19	52	173	28	Nominal: 0 Anomalous: 5	• Incorrect bump detected
		R2	33	76	284	37	Nominal: 0 Anomalous: 10	• Wheel slip • Low obstacle blocking route • Dirty ultrasonic sensor
		R3	16	40	205	26	Nominal: 0 Anomalous: 3	• Incorrect longitudinal localization correction • Dirty ultrasonic sensor
		R4	26	68	232	30	Nominal: 0 Anomalous: 1	• Dirty ultrasonic sensor
		R5	20	54	169	23	Nominal: 0 Anomalous: 2	• Wheel slip • Incorrect orientation localization correction
		R6	40	91	249	31	Nominal: 0 Anomalous: 1	• Incorrect lateral localization correction
		R7	9	16	83	14	Nominal: 0 Anomalous: 2	• Incorrect wall detection • Incorrect longitudinal localization correction
		R8	7	17	85	13	Nominal: 0 Anomalous: 2	• Incorrect wall detection
		R9	4	18	107	14	Nominal: 23 Anomalous: 0	-
		R10	13	44	203	25	Nominal: 8 Anomalous: 1	• Low obstacle blocking route
		R11	15	38	131	19	Nominal: 9 Anomalous: 1	• Low obstacle blocking route
		R12	4	10	106	14	Nominal: 0 Anomalous: 1	• Dirty ultrasonic sensor
		R13	12	29	149	21	Nominal: 0 Anomalous: 1	• Wheel slip • Dirty ultrasonic sensor
		R14	31	72	176	23	Nominal: 14 Anomalous: 0	-
		R15	34	73	191	26	Nominal: 4 Anomalous: 6	• Low obstacle blocking route • Incorrect longitudinal localization correction • Incorrect orientation localization correction • Incorrect lateral localization correction
		R16	29	57	187	25	Nominal: 2 Anomalous: 1	• Incorrect bump detected • Incorrect longitudinal correction

Table 6: The obtained low quantile values of each dataset type for the Peaks-over-Threshold (POT) algorithm.

Set numbers	Dataset type	Lower quantile percentage
1-6	small datasets, motor velocity features	1.5%
7-9	small datasets, ultrasonic features	1.3%
10-12	big datasets	1.3%

routes. See also Equations (13), (14) below.

$$R_1 = \frac{\sum_{i=1}^{N_{anomalous}} \left(\sum_{j=1}^{T_i} \mathbf{1}\left[t_j \geq \theta\right] \right)}{N_{anomalous}} \tag{13}$$

$$R_2 = \frac{N_{anomalous}}{N_{total}} \tag{14}$$

$$R_3 = R_1 * R_2 \tag{15}$$

$N_{anomalous}$ is the number of routes that anomalous. N_{total} is the total amount of routes in the dataset. T_i is the total amount of time steps within route i, and t_j is a specific time step within a route. $\mathbf{1}$ is the indicator function of a particular time step exceeding the threshold θ. By multiplying these two ratios you get R_3 (15), an empiric probability of a time step being anomalous or not, which are the points we ideally want to be in the tail portion of the Pareto distribution.

Table 6 shows the obtained values for each dataset type. Note that we now set the level by using the optimal threshold found by using labels to quantify the results, which makes it supervised. Figure 12 shows the obtained POT F1-scores compared to the optimal F1-scores, both as an average of 10 individual train and test runs. Similar to what [16] concluded, the POT F1-score is always lower than the best F1-score. The difference in performance however, is bigger compared to what [16] found. Instead of a performance drop of about 0.003-0.077, this study found a performance drop of 0.08-0.58.

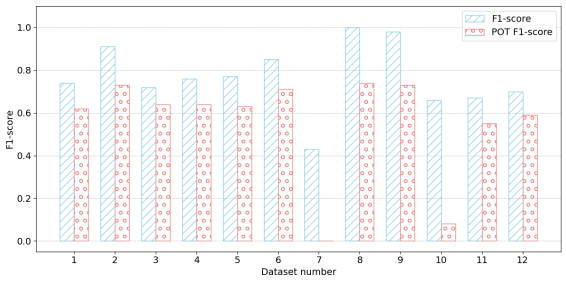


Figure 12: The F1-scores vs. POT F1-scores per dataset.

Some remarks that can be made based on this study are the following. First, there is a change in the evaluation approach used within this paper compared to how POT was initially constructed. Namely, on route-level instead of on anomaly-level. This has an impact on the way the peaks in the output are considered. Second, the datasets within this work are from a robot that encounters uncertainty, which is not directly captured within the data streams. The datasets used within [16] have a lot more features with the aim to describe the entire system's context.

Erratum to: The Applicability of Anomaly Detection Algorithms for Failure Prediction with 2D Mobile Robots

This is an erratum to: L.E. Rhijnsburger, E. Vlasblom, A. Siagkris-Lekkos, and Y.B. Eisma, The Applicability of Anomaly Detection Algorithms for Failure Prediction with 2D Mobile Robots, `https://resolver.tudelft.nl/uuid:a176832e-98e9-4851-b835-cb2291e63d31`, published 19 December 2024.

In the original version of the thesis, there is a discrepancy in the obtained results. During the evaluation process, a technical error caused route classifications to be based on partially overlapping model outputs from different routes. Specifically, when comparing a route's model output to the classification threshold, the evaluation incorrectly included data from adjacent routes. This may sometimes result in wrongful in misclassifications or unintentional correct classification, and thus in different F1, precision, and recall scores. This error affects the following elements of the thesis:

1. Table 2 in Section IV (Methods): Contains incorrect metric values
2. Figures 6 and 7 in Section IV (Methods): Display incorrect values
3. The hyperparameter selection based on these results

The evaluation mistake however is hereafter fixed within the paper and the classifications were evaluated correctly. The error was identified and corrected in subsequent parts of the thesis, ensuring that all later classifications were evaluated correctly. However, as the hyperparameter selection was based on the flawed evaluation process, the chosen parameters may not be optimal. While the overall research methodology remains valid, a new hyperparameter study is conducted with the corrected evaluation process. Should this yield different optimal parameters, all dependent experiments will be repeated and the results and conclusions will be updated accordingly. It is not expected to find significantly deviating hyperparameter values and consequently to find significantly different results as a consequence of these different hyperparameters.