Anomaly detection on a hybrid kinematic machine

Henrik Paldán

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6188
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2022

Abstract

Detecting anomalies is a promising and current research subject that can have useful applications, for example in the field of robotics. In this thesis, anomaly detection is investigated using a hybrid kinematic machine, which is a pick-and-place robot that excels at moving objects at high speed and with great reach. Three different types of anomalies have been chosen to be studied in this thesis; collision, the robot dropping an object, and weight offset between the expected weight that the robot is carrying and the actual weight being carried.

The data were gathered from the robot control system, as well as sensors on the robot when the robot was moving in a specified trajectory cycle. Then, the previously mentioned anomalies were introduced to the robot.

The anomaly detection was conducted by using different anomaly detection models which have certain characteristics. The models were first trained using normal data. Then the trained models and a devised threshold value were used to evaluate how well the models were able to detect the anomalies.

The results are promising, especially for collision detection and weight offset detection. Detecting an object being dropped, however, seems more challenging. The experiments also indicate that a good model of the robot dynamics is of great importance when detecting anomalies. The results also indicate that the most important features for detecting anomalies are the torque data from the control system and data from an accelerometer at the endpoint of the robot. The most promising models for anomaly detection are the local outlier factor model and the autoencoder, which is a type of artificial neural network.

Further work could investigate more varied anomalies that are harder to detect with more advanced models, while also focusing on the models being computationally fast enough to be applicable in a real-time system.

Acknowledgements

I would like to give a big thank you to Björn Olofsson, my supervisor at LTH, for always taking the time to give input, feedback and answer all questions that arose during this work. Another big thanks to Olle Hedbrant, my assistant supervisor from Cognibotics for always having good input regarding the robot and its application. I'm also grateful to all the other people at Cognibotics that helped me both practically and by answering all kinds of questions during my time working on this thesis. Finally, thank you Katja Eftring for great support, both emotionally and by supplying coffee and sweets when things were dire.

Contents

1

Introduction

1.1 General introduction

Anomaly detection means to identify rare groups of data or individual data points that deviate significantly from the majority of the data in a dataset. A thorough review of anomaly detection can be found in the article [Chandola et al., 2009]. In the case of a moving robot that could be any number of things such as the mechanics being broken, the robot not functioning properly, or that it has collided with something. It could also be more closely related to the specific task of one robot.

The hybrid kinematic machine (HKM) [Cognibotics, 2022] shown in Figure 1.1 is the robot on which this project is carried out. It specializes in pick-and-place, meaning it excels at picking up and moving objects at high speed with long reach. In this context three different types of anomalies have been chosen to be investigated; collisions on the robot, the robot dropping an object that it is carrying, and offset between the expected weight carried by the robot and the actual weight carried.

The benefits of identifying these anomalies are in the case of collisions that it can improve the reaction time for the robot when a collision occurs and therefore reduce the damage to the robot or other objects in the environment. Improved collision detection could also make the robot safer for humans to be close to. Detecting if the robot has dropped an object can be used to notify an operator that something has gone wrong. Weight offset in what the robot is carrying could indicate that the robot has not picked up an object or picked up the wrong object than it is supposed to, which also indicates that something has gone wrong in its normal procedures which should be notified to an operator.

Figure 1.1 Illustration of the HKM robot.

1.2 Academic relevance of the problem and previous research on the subject

Anomaly detection in multivariate time series is a current research field that has gained much of attention recently, mostly because big datasets are becoming more easily available. The possible application areas are broad from fault detection in the industry to fraud detection in finance [Audibert et al., 2022].

In the field of robotics, a common task is to detect collisions, largely because of the increasing use of cobots and the safety requirements that come with them. Traditionally collisions are detected by offset in expected and measured torque based on models [Min et al., 2019], but recently a wide variety of methods most concerning some sort of machine learning have been researched. Some examples of different approaches are using vibration analysis in detecting a collision [Min et al., 2019], using neural networks as a virtual torque sensor [Czubenko and Kowalczuk, 2021], unsupervised collision detection with normal data training [Park et al., 2022], and using accelerometer values that already are installed to be used in reducing link vibrations [Pucher et al., 2019]. This project is closely related to several of the above-mentioned examples and papers on these topics but using both available data from the control system such as reference and feedback position, speed, acceleration, and torque together with acceleration data from the robot endpoint in a machine-learning context seems to be unexplored. There is also limited research in

detecting other anomalies than collisions, such as the robot dropping an object and offset in expected weight.

1.3 Problem description

The purpose of this thesis is to investigate if it is possible to identify anomalies that occur on the HKM robot using the data from the sensors and the control system that is available. Three different types of anomalies have been chosen to be experimented with, collisions with an object, dropping off an object, and offset between expected and carried weight.

The models used have different characteristics and utilize different aspects of what characterizes an anomaly which leads to another interesting question to explore, what model is most proficient at detecting anomalies. The research questions for this thesis are:

- What types of anomalies are possible to detect on the HKM using the data available?
- What model is most proficient in detecting anomalies in this setting?
- Which type of data is most significant in detecting anomalies?

Delimitations. An important aspect of an anomaly detection model for it to be applicable in a real time-system, is the computation time. This is especially relevant for critical events like collisions. There are many aspects that affect this, such as what hyperparameters that are used and the available hardware. Since this thesis does not include an actual test on the real system, the computation time is an aspect that is not covered in depth in this thesis.

1.4 Outline of thesis

Chapter 2 gives an overview of the Hybrid kinematic machine that anomaly detection is performed on. It also describes the data available from the system to detect anomalies. In Chapter 3 theory regarding time series and anomaly detection is presented together with a general description of the anomaly detection models and the evaluation metric used to evaluate the performance of the models.

Chapter 4 explains the method for performing the experiments on the robot as well as data management. The analysis that has been done on the gathered data with the models is also explained.

In Chapter 5 the result is presented and analyzed for each experiment conducted and Chapter 6 consists of a more in-depth discussion about some aspects of the result. Finally, in Chapter 7 a conclusion and a summary of the result are given as well as suggestions for further work.

2

Background

This chapter presents a more thorough background on the robot the project is conducted on as well as what data are available from the system.

2.1 The Hybrid kinematic machine (HKM)

The hybrid kinematic machine [Cognibotics, 2022] used in this project is a robot developed at the company Cognibotics that specializes in an area called pick-and-place, meaning that it excels at picking up and placing objects which is a common task in especially e-commerce. The robot has a unique and patented design and stands out from other robots with its long reach and speed. The long reach and speed are because of that the majority of the weight is centered on the base and not on its moving parts. A drawback with this is that it creates flexible modes in the structure, which are suppressed with a control design that uses a model of the flexible modes, the motor torque as control signal and data from the endpoint accelerometer as feedback.

2.2 Available data

The control system for the HKM is developed at the company, which means that all feedforward and feedback signals used to move the robot are available. The signals used in this project from the control system are the feedback and feedforward values for the position, velocity, and acceleration for all the motors and the torque for three out of four motors. The feedforward values are calculated using a mathematical model of the dynamics of the HKM.

The data from the accelerometer at the endpoint of the robot described in the section above are also directly available to use.

Finally, there are also sensors integrated into the gearboxes attached to two of the motors; these sensors measure the acceleration in the gearboxes. The gearboxes are at the center of the robot and while all values used in the control system have a sampling rate of 1000 Hz the sensors have a slower sampling rate of 200 Hz. In total 35 features are used in training the models.

3

Theory

In this chapter, a brief introduction to time series and anomaly detection is presented. There is also an overview of the different types of strategies for detecting anomalies and the underlying theory of each of the models used. Lastly, the evaluation metric, the F1-score, used to evaluate the performance of the models is presented.

3.1 Time series

The following definition is taken from [Audibert et al., 2022].

Definition 3.1.1 (Time series)**.** *A time series is defined as a sequence of data vectors* $T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, $\mathbf{x}_t \in \mathbb{R}^m$, $\forall t, 1 \leq m$ *that are measured in time where* t *is the specific time point for that measurement. In the special case when* $m = 1$ *it is a univariate time series and otherwise, it is a multivariate time series*

In the case of this thesis, the data gathered can be defined as a multivariate time series. This data are presented in Chapter 2.2 about available data.

There are several different ways of detecting anomalies in time series, the ones that are used in this thesis are presented in the chapter about anomaly detection models. An important concept in anomaly detection on time series is utilizing a time frame window as a data point. This allows methods that are not specifically designed for time series to exploit that data in time series usually are related in time. Instead of only using data from a single time point as a data point, we take a frame that includes previous data in time and call this a data point instead. We then consider whether this frame is an anomaly or not. Mathematically, a frame in a time series is presented as follows.

Definition 3.1.2 (Time frame window)**.** *We define a time frame window* w_t *with length* K *such that* $w_t = \{x_{t-k+1}, x_{t-k+2}, ..., x_t\}, K \leq t$.

A time series can then be described as a sequence of windows $W = \{w_1, w_2, ..., w_T\}$.

A multivariate time series with T time points and m dimensions which we rewrite as a series of time frames will have $T - K + 1$ time frames because we cannot use the first K time points since we lack previous values to insert into the frame. However, each frame will have a dimension of size mK, because for each dimension we include previous values K time steps back [Audibert et al., 2022].

3.2 Anomaly detection

An anomaly is defined as a data point that does not conform to a well-defined notion of normal behaviour [Chandola et al., 2009]. This is exploited in this thesis by collecting a large amount of data when the robot is behaving normally to teach a model this normal behavior. As for anomalies, there are several different types of anomalies to account for, the most important ones are presented below [Chandola et al., 2009]. In Figure 3.1 all types of anomalies are presented graphically.

Definition 3.2.1 (Point anomalies)**.** *A point anomaly is when a single data point is classified as an anomaly. This is the simplest type of anomaly and also the type subjected to most research [Chandola et al., 2009].*

An example of a point anomaly is fraud where a person's spending would increase significantly from how much the person normally spends.

Definition 3.2.2 (Contextual anomalies)**.** *Contextual anomalies are when the value of data in itself does not appear out of the ordinary from other data but the context in which it is in makes it an anomaly.*

To determine contextual anomalies one needs attributes that give structure to the data. In time series, time is a typical contextual attribute [Chandola et al., 2009]. An example of a contextual anomaly is snowing which is considered normal beaviour (at least in many places) during winter but something strange during the summer.

Definition 3.2.3 (Collective anomalies)**.** *Collective anomalies are a group of data points that do not stand out separately but form an anomaly when put together. To have collective anomalies one needs a relation between data points. In time series the data points are related by time [Chandola et al., 2009].*

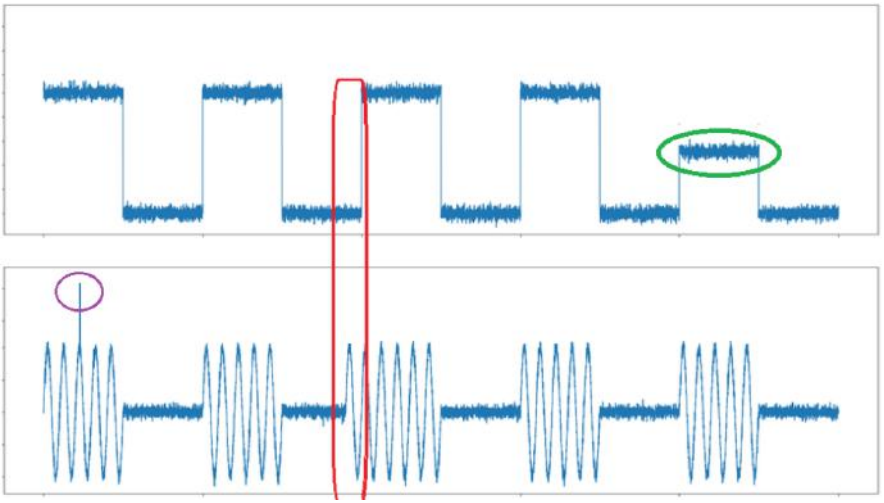

Figure 3.1 Illustration of different types of anomalies. A point anomaly (purple), contextual anomaly (red) and collective anomaly (green) [Audibert et al., 2022].

Data labels and anomaly detection

It is generally hard to obtain labeled data for anomalies. By its nature anomalies are rare compared to the rest of the data and can be unpredictable in the way they manifest. In this thesis, something called semi-supervised anomaly detection is conducted. Semi-supervised means that data labeled as normal are available and since all anomalies are inflicted superficially, we can safely assume that no anomalies occur when normal data gathering is taking place. The approach in this case is to build a model with the available normal data and then consider data points that do not conform to this model as anomalies [Chandola et al., 2009].

Output of anomaly models

Two main types of output can be generated by anomaly detection models: scores and labels. The score is a value for each data point of how much of an anomaly the data point is, and labels are a direct label of if a data point is a normal point or an anomaly point. Given some information about the data, it is possible to go from a score output to a label output, for example, if it is known in advance that a certain ratio of the data are anomalies.

In the experiments in this thesis, it is hard to know exactly when the system reacts to the anomaly, and therefore to label the data to be an anomaly or a normal data point the scoring system is used. A high score that happens in close proximity to a reference value of an anomaly event is considered a detected anomaly [Chandola et al., 2009].

3.3 Anomaly detection models

This section introduces the models that are used to detect anomalies in the dataset. There are different strategies to detect anomalies, the presented models are each a representative of a specific type of strategy. The selection of models is inspired by the paper [Audibert et al., 2022], which compares more traditional statistical models with machine learning and deep learning models in detecting anomalies in multivariate time series. Also, the hyperparameters, meaning manually tuned parameters, are presented for each model.

The local outlier factor (LOF) model

The local outlier factor method belongs to a category of methods called neighbourhood-based methods [Audibert et al., 2022]. Neighbourhood-based methods investigate the neighborhood of each data point to determine whether it is an anomaly or not. The principle is that an outlier would be more isolated compared to normal data points. The approach makes it easy to assign a score for each point [Audibert et al., 2022].

The key concepts and the equation for the local outlier factor model is presented in the following way:

Definition 3.3.1 (k-distance of object p)**.** *The k-distance is a distance $d(p,o)$ between objects $p,o \in D$ such that:*
(i) for at least k objects $o' \in D \setminus \{p\}$ it holds that $d(p,o') \leq d(p,o)$
(ii) for at least $k-1$ objects $o'' \in D \setminus \{p\}$ it holds that $d(p,o'') < d(p,o)$

The k-distance of object p will further on be notated as $k_d(p)$. The k-distance could be explained as the radius in a sphere that encloses at least the $k-1$ closest objects and is equal to the distance to the k:th closest object [Breuniq et al., 2000]. Then there may be several objects that are of equal distance as the k:th object from object p.

Definition 3.3.2 (k-distance neighborhood of object p)**.** *Given the previously defined $k_d(p)$, the k-distance neighborhood is the set of all objects q such that*

$$N_{k_d(p)}(p) = \{d(p,q) \leq k_d(p)\} \quad and \quad \{q \in D \setminus \{p\}\} \tag{3.1}$$

These objects are called the k-nearest neighbours of p.

Definition 3.3.3 (Rechability distance of an object p with respect to o)**.** *The reachability distance is defined in the following way:*

$$reachability\_distance_k(p,o) = \max\{k_d(o), d(p,o)\}) \tag{3.2}$$

Definition 3.3.4 (Local reachability density of an object p)**.** *Local reachability is a way of measuring a local density in the dataset. The definition is:*

$$\sigma_k(p) = \frac{|N_{k_d}(p)|}{\sum_{o \in N_k(p)} reachability\_distance_k(p,o)} \tag{3.3}$$

where the local reachability density is notated as $\sigma_k(p)$ [Breuniq et al., 2000]. In equation (3.3), $|N_{k_d}(p)|$ is the absolute value of $N_{k_d}(p)$. Now all pieces of the local outlier factor have been defined and all that is left is to put them together. The local outlier factor is defined in the following way:

Definition 3.3.5 (Local outlier factor of object p)**.**

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{\sigma_k(o)}{\sigma_k(p)}}{N_k(p)} \tag{3.4}$$

Equation (3.4) outputs an anomaly score for each point p. It can be described as the ratio between the average local reachability density of k points close to p and the local reachability density of point p. So, if the local reachability density of p is low and the local reachability density of points close to p is high, then the anomaly score for point p is high. This way we get a distance-based anomaly detection model that adjusts how much of an anomaly a distance is based on the neighborhood of the point [Breuniq et al., 2000].

Hyperparameters. The only hyperparameter to tune for the model is k, the number of points close to point p that are being compared to p [Breuniq et al., 2000].

The isolation forest (iForest) model

Isolation forest is an isolation-based type of anomaly detection. Isolation-based means that the model relies on the fact that anomaly points in a dataset typically have two characteristics. They are generally much fewer than normal points and their attributes typically differ significantly from other points. These characteristics make anomaly points easy to isolate [Liu et al., 2012].

This is exploited by the isolation forest model by partitioning data using several binary search trees and then based on the number of partitions required to isolate a data point determining if the data point is an anomaly or not [Liu et al., 2012]. The procedure for the iForest algorithm is explained here.

Training phase. The algorithm divides the dataset into subsets of data. For each subset, a feature from the data is selected and a binary tree is used to partition the data points in the subset based on the value of this specific feature. The binary tree continues to divide the subset until all points are either isolated or all datapoint in one partition has the same value (so that the group cannot be divided further). The same procedure is done with several binary trees, each dividing a randomly selected subset of the data based on a randomly selected feature. All of these binary trees and the values they use to split the data are saved [Liu et al., 2012].

Evaluation stage. The dataset to be investigated is then used on the selection of trees that were previously created by partitioning the training data. The evaluation of a single data point works as follows. The data point is evaluated in all binary trees created in the training phase, and the average value of the path length taken to

isolate the data point in each tree is saved. This mean value is then fed into equation (3.5) which is used to account for normalisation and the subsampling size:

$$s(x,\Psi) = 2^{-\frac{E(h(x))}{c(\Psi)}} \tag{3.5}$$

In equation (3.5), $E(h(x))$ is the average path length for data point x, Ψ is the subsampling size that is used for each binary tree, $c(\Psi)$ is the average path length based on Ψ which works as a normalisation factor and finally $s(x,\Psi)$ is the anomaly score [Liu et al., 2012].

Hyperparameters. The hyperparameters for the isolation forest method are Ψ, the size of the subsample that is used for each tree, and t the number of trees used in the isolation forest [Liu et al., 2012].

The autoencoder (AE) model

The autoencoder is a type of feedforward artificial neural network which first reduces the dimension of the data, called the encoder part, and then expands the data back to the original number of dimensions, called the decoder part. It works as an outlier detector by training on normal data which leads to that the model outputs low loss on normal data, while anomalous data that the model have not trained on will give a larger loss value [Ohlsson and Edén, 2021].

Artificial neural network. An artificial neural network is a model inspired by neuroscience that aims at getting a certain input x to produce a desired output y. This desired output is learned by giving the network training data with a known output and then progressively improving the network. There are several different types of neural networks but only the feedforward network will be covered here since the autoencoder is based on the feedforward network [Ohlsson and Edén, 2021].

The feedforward network consists of layers of nodes, the first layer is called the input layer where the input data go in and the last layer is called the output layer where the derived output y comes out [Ohlsson and Edén, 2021].

The node is the core of a neural network, and a single node consists of a linear function and an activation function. The linear function is linear, and the activation function is typically non-linear. Because of the nonlinearity in the activation function, a large enough neural network can approximate any function according to the universal approximation theorem [Goodfellow et al., 2016].

An illustration of the components of a single node in an artificial neural network is presented in Figure 3.2 and in equations (3.6) and (3.7). The variables in equation (3.6) and (3.7) are the same as are shown in Figure 3.2, ω_k are the weights of a single node, x_k is the input that can be the input data if the node is in the input layer, otherwise it is the output from the previous layer, b is the bias for the node, ϕ is the activation function used in this node and y is the output from the node [Ohlsson and Edén, 2021].

$$a = \sum_{k=1}^{K} \omega_k * x_k + b \tag{3.6}$$

$$y = \varphi(a) \tag{3.7}$$

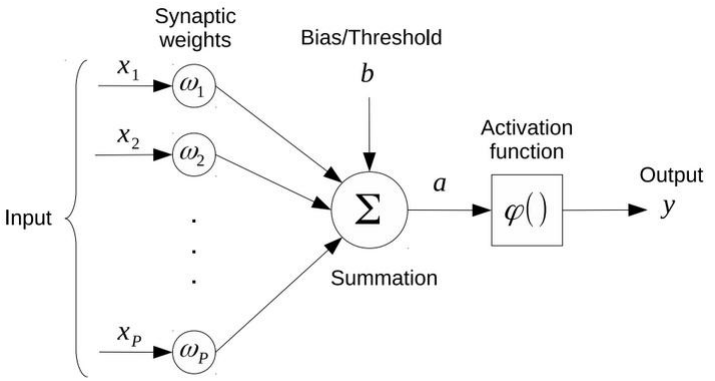

Figure 3.2 Descriptive overview of a node in a neural network [Ohlsson and Edén, 2021].

There are many different types of activation functions that are conventionally used, the ones that are being used in this thesis are the tanh and rectified linear unit (ReLu) functions defined in Table 3.1.

Table 3.1 The common activation functions in neural networks tanh and ReLu.

$\varphi(a) = \tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$
$\varphi(a) = \mathrm{ReLu}(a) = \max(0,a)$

The goal when training an artificial neural network is to tune the parameters ω_k and b for each node. This is done by defining a loss function $L(\omega, b)$ and then finding parameters ω and b that minimize $L(\omega, b)$. Commonly this is done by exploiting

the gradient of the loss function $L(\omega, b)$. The loss function $L(\omega, b)$ is also a way to measure how well our model performs. With a training set we have a known output d_n given an input x, which we then compare to the actual output of the model y_n. A common loss function that is also used in this thesis is the mean squared error defined as

$$L(\omega, b) = \frac{1}{N} \sum_{n=1}^{N} (d_n - y_n)^2 \tag{3.8}$$

where N is the total number of data points in the dataset [Ohlsson and Edén, 2021].

Autoencoder. The autoencoder is a type of feedforward network with a distinct design of the network. It takes an input and then through one or several layers reduces the number of nodes to a bottleneck that is smaller than the number of input nodes, this part is called the encoder part. Then it increases the number of nodes again through one or several layers until it has the same number of nodes as the input layer, this part is called the decoder part. An illustration of an autoencoder neural network can be seen in Figure 3.3.

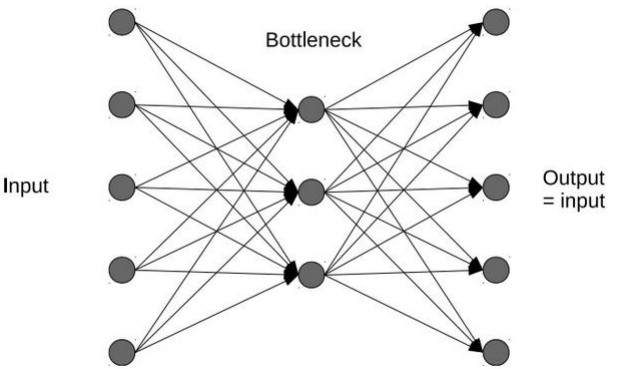

Figure 3.3 Illustration of an autoencoder artificial neural network [Ohlsson and Edén, 2021].

In the context of an autoencoder the loss function described in equation (3.8) becomes:

$$L(w, b) = \frac{1}{N} \sum_{n=1}^{N} (x_{in} - x_{out})^2 \tag{3.9}$$

where x_{in} is the input to the network and x_{out} the output [Ohlsson and Edén, 2021].

An autoencoder can be used for several things, one of them being the detection of anomalies in a dataset. With semisupervised learning where normal data are available for training, the neural network can train on that dataset and optimizes on

reconstructing normal data. This will, in turn, result in the loss function becoming small for normal data. Then if the network is given anomalous data on which it has not trained the loss will be higher so the anomalous score here is the loss for the network [Ohlsson and Edén, 2021].

Hyperparameters. There are many ways of constructing an artificial neural network. The only thing granted in an autoencoder network is that the size of the input layer should be the same as for the output layer and that the middle layer should be smaller. Other than that, the number of layers and the number of nodes in each layer can be adjusted. There are also other parameters such as the activation functions, the optimization method, and batch size that are more centered around the training performance that also can be adjusted [Ohlsson and Edén, 2021].

The vector autoregression (VAR) model

A vector autoregression model is a model that aims to model and forecast multivariate time series based on statistics. The model takes previous values of all features and then predicts the next value. The number of previous values that are being used is called lag and is up to the creator of the model. A model of p:th order looks the following way:

$$\mathbf{y_t} = c + A_1 y_{t-1} + A_2 y_{t-2} + ... + A_p y_{t-p} + e_t \tag{3.10}$$

where c is a constant vector working as the bias for the model, the matrices A_n are time-invariant matrices, e is an error term, and $\mathbf{y_t}$ is the next predicted value in the time series [Petropoulos et al., 2022].

A time series process has to fulfill two conditions to be modeled with VAR, the first one is that there has to be a correlation between the different features in the time series, otherwise they are just individual time series. The second one is that because the matrices A_n are time-invariant the time series has to be stationary, meaning that their underlying statistical properties do not change over time [Petropoulos et al., 2022]. The first condition, that there is a correlation between time series, can be checked by using a cointegration test. The cointegration test is not covered in this thesis but can be read about in [Johansen, 1991]. The second condition that the time series has to be stationary can be checked by using a Dickey-Fuller test, which is also not covered in this thesis but is described in the paper [Dickey and Fuller, 1979].

Hyperparameter. The only hyperparameter for a VAR model is the number of previous data points used to predict the next value, which is also called lag. There is a test that can be used on different lags to determine which one works best, the test is called the Akaike information criterion and is presented in [Akaike, 1974].

The one class support vector machine (OCSVM) model

A one-class support vector machine is a special kind of support vector machine that is specifically designed to identify outliers [Schölkopf et al., 2000]. The method is

domain-based and tries to create a boundary around all normal points in a feature space and then classify all points outside of the boundary as outliers [Schölkopf et al., 2000].

Support vector machine (SVM). A regular support vector machine is a classifier that aims at creating a boundary between all classes of data in a feature space. It does this by maximizing the distance between the closest data point from each class and putting the boundary at that distance. SVM is typically a linear classifier but by projecting the data to higher dimensions using a kernel, one can also create non-linear boundaries [Lindholm et al., 2022a]. The kernel used in this thesis is the radial basis function (RBF) kernel [Lindholm et al., 2022a].

The one-class support vector machine differs from typical SVMs by only having one class that represents the normal data points. So the model aims at creating a boundary as close as possible around the normal data points and all points that are outside of that boundary are considered outliers [Schölkopf et al., 2000].

Hyperparameters. According to the original paper about the one class support vector machine [Schölkopf et al., 2000], the algorithm has a built-in factor ν that sets a threshold for how many datapoints in the training set that are anomalies and should be outside of the boundary. The RBF kernel also has a parameter γ that decides how much influence a single training example has [Lindholm et al., 2022a].

3.4 Evaluation metrics

To objectively evaluate a model, an evaluation metric is needed. A common one that is also used in this thesis is the F1-score. To understand the F1 score one needs to start with the following key concepts:

True positive (TP). True positive is when an anomaly occurs, and the model can detect it.

False positive (FP). A false positive is when no anomaly occurs, but the model still signals that an anomaly has happened.

False negative (FN). A false negative is when an anomaly occurs, but the model is not able to detect it.

Intuitively based on these definitions, many true positives and few false positives and false negatives is an indicator that a model performs well [Lindholm et al., 2022b]. Furthermore, precision and recall are used to weigh true positives and false positives as well as true positives and false negatives. These concepts are presented next.

Precision. Precision measures the ratio of relevant instances and the total number of instances. Precision is defined by the following expression:

$$precision = \frac{TP}{TP+FP} \tag{3.11}$$

Recall. Recall, also known as sensitivity, describes the division between relevant instances retrieved and the total number of instances. Recall is defined by the following expression:

$$recall = \frac{TP}{TP+FN} \tag{3.12}$$

F1-score. Recall and precision are both metrics that could be misleading if one does not have the full picture. As an example related to the topic of this thesis, a high precision but a low recall could mean that the system is too sensitive and would likely to signal a collision or other anomaly when none have happened. A system with a high recall but low precision would instead mean that it rarely gets it wrong when signaling an anomaly but that it also misses a high rate of anomalies [Lindholm et al., 2022b]. Both are of course unwanted properties of an anomaly detection model. Therefore there is another metric called the F1-score which takes both of these into account and balances them. The F1-score is defined in the following way

$$F_1 = 2\frac{precision \cdot recall}{precision + recall} \tag{3.13}$$

The F1 score returns a value between 0 and 1 where 0 means that no anomalies have been detected and 1 means that the model detects all anomalies and gives zero false alarms [Lindholm et al., 2022b]. In the rare case that the model does not detect a single correct value, the F1-score will be a value divided by zero and therefore undefined.

4

Method

This chapter is divided into three parts, the first one describes the method of data collecting and how the experiments were carried out. The second part describes the data management, the procedure used to define when the anomalies occur based on reference values and how a threshold value was calculated for the different experiments. The third part describes how the analysis was done using the models on the data.

4.1 Data gathering

Since the robot is running on programmable logic controllers (PLCs) and software from Beckhoff, an automation company, all data gathering was done with the Beckhoff software TwinCat scope [Beckhoff, 2022]. The data gathering for the training data were recorded in intervals of 20 minutes and these datasets were then divided into two 10-minute sequences each to be able to export them. After that they were exported as a comma separated values (csv) file, then read into the Julia programming language [Bezanson et al., 2017] as a DataFrame datatype. All data gathering was done with a 1 millisecond sample period.

Collision detection

Training data gathering. The training data for the collision tests were gathered by creating a trajectory for the robot which fulfilled two criteria: the first one was that it should be a trajectory similar to its real application, so a pick and place cycle, and the second criterion was that it should excite all joints of the robot to get a good overall representation of its movements. Since for collisions, the speed and therefore the force of the robot is of great interest this trajectory was run at three different speeds. The robot controller is designed in a way that a maximum velocity is set when executing a movement. The maximum velocities used were 2.7 m/s, 1.8 m/s and 0.9 m/s. The robot was then run for 4 hours for each velocity to gather a sufficient amount of data.

Testing data. The obstacle was created using aluminum bars, some foam, and duct tape. The obstacle also had an accelerometer attached to it that was connected to the Beckhoff software together with all other data to be used as reference data to detect when the collision takes place. A close look at the obstacle can be seen in Figure 4.1. The first 100 collisions for each velocity were recorded at the endpoint and the tool of the robot. For these 100 collisions the resistance, meaning the tension of the screws attaching the obstacle, as well as the height of the obstacle were varied to get a variation in the collision data. An overview of the experiment setup can be seen in Figure 4.2.

Figure 4.1 Close look at the construction of the collision obstacle. The red circle encircles the whole collision object and the red arrow points at the accelerometer used to get a reference signal.

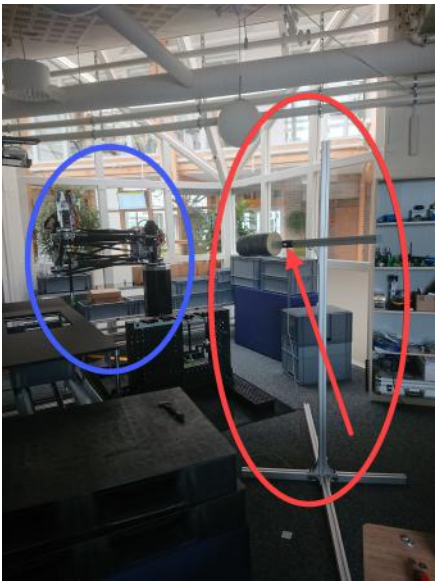

Figure 4.2 Overview picture of the collision obstacle and the robot. The blue circle encircles the HKM, the red circle encircles the collision object and the red arrow points at the accelerometer used to get a reference signal.

After the collisions at the endpoint of the robot were performed, the obstacle was reconfigured to record collisions closer to the center of the robot. Then 40 collisions were recorded in the same way for each velocity at different points between the endpoint and the center of the robot. This setup can be seen in Figure 4.3.

Figure 4.3 Overview picture of the reconstructed collision obstacle for inner collisions and the robot. The red circle encircles the reconstructed collision object and the red arrow points at the accelerometer that gives the reference signal.

In total 140 collisions were recorded for each velocity, 100 at the endpoint of the robot and 40 closer to the center of the robot.

Object drop detection

Training data. The data for the object drop experiment was gathered in a similar way as the collision data. A path that is both realistic as to the robot's area of use and excites all joints was produced. The difference here is that the robot is moving a package back and forth along its path. The robot is equipped with vacuum suction that it uses to pick up objects.

For this data the velocity is of less importance and was therefore chosen to be constant. Instead, the weight of the object that is moved around is of more interest. Two different weights were used, 2.5 kg and 7.5 kg, the weights were chosen because the robot's nominal payload is 2 kg and 2.5 kg was close enough to it and its maximum payload is 7.5 kg. The velocity for the respective weight was 1.8 m/s for 2.5 kg and 1.35 m/s for 7.5 kg. The reason different velocities were used is because the data for a 2 kg weight were gathered first, and the robot was unable to keep a grip on the heavier 7.5 kg object at that velocity so it had to be lowered.

The same package was used for both tests, so the volume is constant for both tests. A picture of the package can be seen in Figure 4.4. A complete overview of the experimental setup can be seen in Figure 4.5. Data were gathered for two hours for each weight and then split up into datasets of 10 minutes in the way described at the beginning of the chapter.

The suction cups kept dropping the carbon box with the 7.5 kg weight so during the last 20 minutes of data gathering a plastic lid was attached to the top of the box to get a better grip.

Testing data. The test data were gathered by using the same path as for the normal data with the object but then the vacuum was switched off manually at different instances of the cyclic path. For the heavier load of 7.5 kg a plastic lid was used for all data gathered. This plastic lid was so well attached to the vacuum cups so that it sometimes took a long time to let go of the object. Because of this, a blowout function was also used on some of the datasets, this blowout function makes the robot drop the object faster. The vacuum system also logs the vacuum pressure which was used as a reference signal to when the robot drops the object.

Figure 4.4 Close look at the object used for drop experiment, in this picture the lid used for the heavier load of 7.5 kg is on.

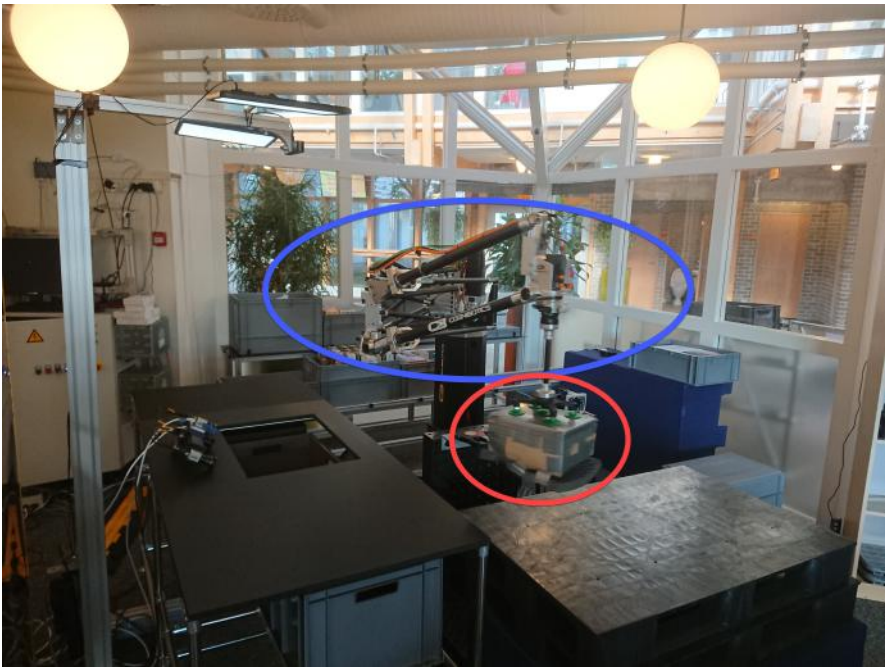

Figure 4.5 Overview picture of the robot carrying the object used in object drop experiments. The blue circle encircles the HKM robot, and the red circle encircles the object used for the experiment.

Weight offset detection

The control system that plans the trajectory of the robot uses a mathematical model of the robot dynamics to calculate the feedforward signal. With this comes the option to take into account the weight and center of mass of any object attached to the robot or carried by it to keep the model of the system closer to the real physical system. In this way, the control error is minimized. This experiment was conducted to see whether it is possible to detect deviations between the adjusted weight of an object and the actual object weight. The experiment here is that the adjusted weight is set to a certain weight while the real robot does not carry any weight at all.

Training data gathering. The training data were gathered by attaching a test object of a predetermined weight on the robot and adding the weight of the test object into the model of the robot dynamics that is used when computing a feedforward signal to the controller. Then a trajectory that realistically simulates a pick-and-place cycle and excites all the joints of the robot was chosen and the robot ran this trajectory for 2 hours while data were gathered on all parameters. Two different test objects with a weight of 2 kg and 7.5 kg, respectively, were chosen since this is the

nominal and the maximum payload of the robot. The objects can be seen in Figure 4.6 and Figure 4.7. The maximum velocity of the robot was chosen to be 3.6 m/s for the weight 2 kg and 3.15 m/s for the weight 7.5 kg. The reason to using these maximum velocities were to use as high maximum velocity as possible without risking breaking the robot since this is how the robot would be used in a practical setting.

Testing data gathering. The data gathering for data used in evaluating the performance was done a bit differently in this experiment than the previous two. For this experiment, two different types of test data were collected, firstly anomalous data were gathered by having the robot to move along the same trajectory as for the training data and with the same weight setting but without any object attached. So, the weight offset between the model used in the control system and the real system was 2 kg and 7.5 kg, respectively. The second type of test data were gathered in the same way as for the training data with the object attached. This way the anomalous data were be used to investigate if the models can detect if it is an anomaly and the data gathered the same way as the training data were used to see if the robot detects any false positives. Both types of data were gathered for 2 hours, respectively. The data were then split into small parts that correspond to one trajectory cycle, for the experiment with 7.5 kg it was roughly 4.5 seconds and with the 2 kg weight it was roughly 2.6 seconds.

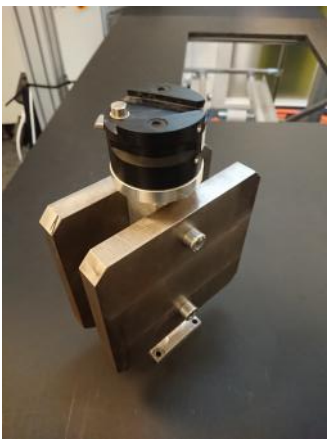

Figure 4.6 7.5 kg weight used in the weight offset experiment.

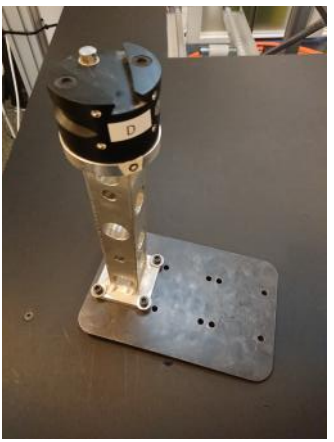

Figure 4.7 2 kg weight used in the weight offset experiment.

4.2 Data management

The data were processed using the Julia programming language and packages available in Julia, mainly the package DataFrames [Bezanson et al., 2017].

Preprocessing

For the training and evaluation of all the models except the VAR model, the data were normalised by using pre-computed values to get the mean value of the training data to 0 and the standard deviation to 1. The pre-computed values were derived using all training data for each experiment and each subcategory. This means that for example for collisions the normalisation was done separately for each maximum velocity and for the drop experiment and weight offset experiment the normalisation was done for each weight separately. The values used were calculated by using all available training data for each test, even if only a subset of the data were used in the training of the models.

Reference values

To evaluate whether a model can detect an anomaly or not one needs to know for sure when the anomaly event is happening. For this, a reference signal was used to extract the relevant information. In this section the reference values for the collision and the drop experiment are explained. For the weight offset no reference signal is used, instead normal data were used to compare whether the model would give a higher anomaly score to anomalous data than to normal data.

Collisions. The object that was used for colliding with the robot had an accelerometer attached to it. The data from the accelerometer were stored the same way as the data for the system. The accelerometer data, acceleration along the x, y, and z -axes were all put together into a single vector with a root mean square (RMS) value for each time point. The root mean square value is calculated for each time point with the following equation

$$\chi_n = \sqrt{\frac{1}{3}(x_n^2 + y_n^2 + z_n^2)} \tag{4.1}$$

where χ_n is the RMS value for a single time point, n is the specific time point and x_n, y_n and z_n are the accelerometer values in each spatial direction from the accelerometer. A value is calculated from the RMS vector for each time point by first calculating the difference between the average of the previous ten values and the current value and then caluclating the absolute value of this, this is the final reference value used. Then a threshold value was identified for the collisions so that when the calculated value was above that threshold value it would imply a collision. The same threshold value was used for all collisions, except for the inner collisions where more disturbance occurred in the accelerometer because of the new position so a slightly higher threshold value had to be used. The RMS value from the accelerometer is shown in Figures 4.8 and 4.9, the red line seen in the figures is an indicator that the method described in this section has identified a collision.

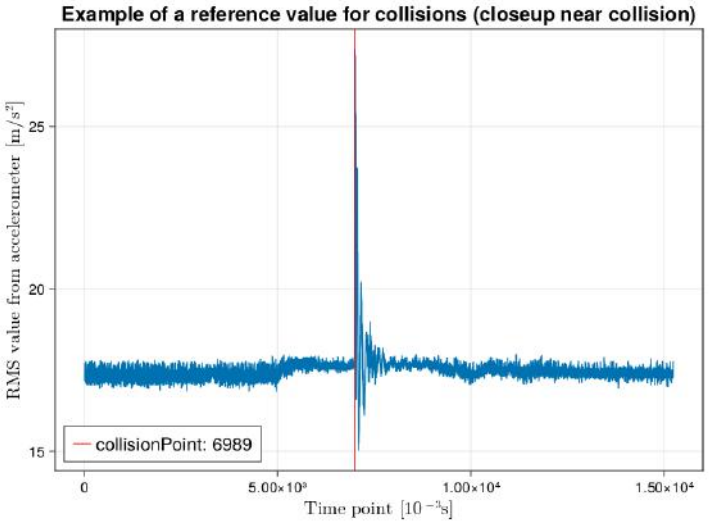

Figure 4.8 This is the RMS value calculated from the accelerometer attached to the collision object during a collision.

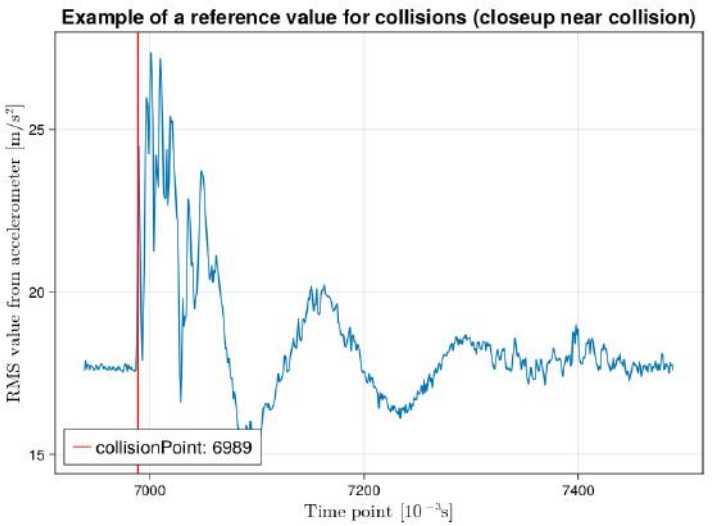

Figure 4.9 This is a closeup near the collision of the RMS value of the accelerometer data from the sensor attached to the collision object.

Object drops. The robot picks up an object by using vacuum and suction cups. On the device that is used to create a vacuum there is a vacuum sensor that measures the pressure when picking an object. This value is stored together with the other signals for the control system. When an object is starting to fall the pressure drops and when the object has fallen off completely the vacuum value is zero.

If an object is held tightly by the suction tool it might take a few seconds from that the vacuum is turned off until the object drops. It is also uncertain when exactly the anomaly would be registered by the anomaly detection models, when the vacuum is turned off or when the object has completely fallen off from the suction tool. Therefore, two indicators were used instead of one. The first one is when the pressure value drops below a threshold value derived by taking the lowest pressure value from a dataset of normal data which was 10 minutes. The other indicator was set to when the pressure value is zero, which indicates that the object has let go from the suction tool completely. These two indicators are marked in the figures with a red line shown in Figure 4.10. For some test cases, a function that blows out air was used to make the object drop almost instantly. This can be observed in Figure 4.11 where the first and second indicators occur almost at the same time.

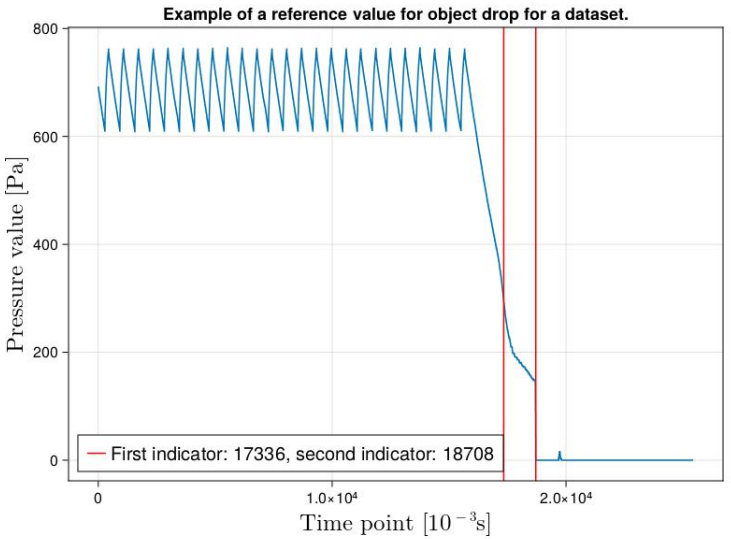

Figure 4.10 This is how the data from the vacuum sensor on the robot look like when the vacuum device is turned off normally and no blow out is used.

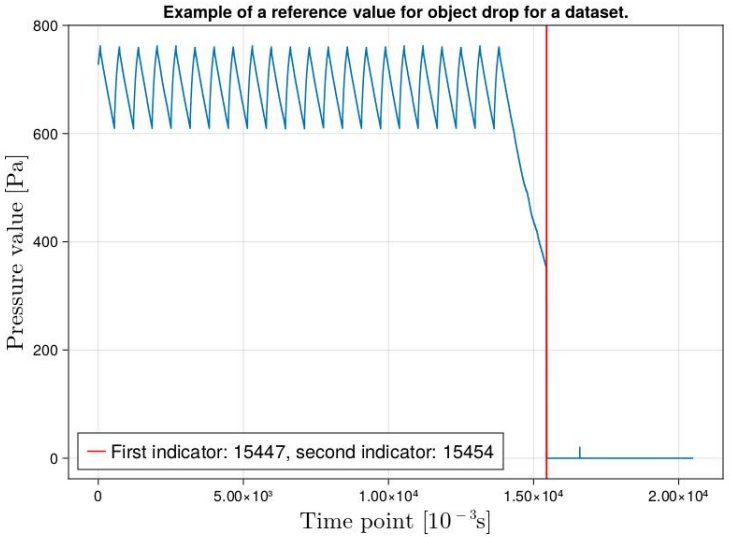

Figure 4.11 This is the reference value from a test data where blow out is used to drop the object faster.

Performance evaluation of the models

To objectively evaluate the model, a procedure with a set of rules was used. The main task is to get a threshold value of the anomaly score that is used to decide whether a datapoint is an anomaly or not. The method to calculate the threshold value was derived with the goal to maximize the number of true positive values and minimize the number of false negatives and that a perfect model should produce a perfect F1-score.

The collision data.

1. First, the model was used on a dataset taken from the training data where the robot operates normally. The size of that dataset was set equal to the size of the training data. The highest anomaly score output when using the model on this dataset was saved.

2. Then 30 percent of the testing data with anomalies was set aside. The current model was used to get an anomaly score for each time series. Then all these time series with an anomaly score were iterated through and for each time series the highest value within 500 milliseconds after a collision was saved. Then, from all these values the lowest one was selected.

3. Then finally, the mean value between the two saved values was calculated as the threshold value.

4. Then, the model was used on the remaining test data (70 %) to convert each datapoint into an anomaly score. If an anomaly score above the threshold value occurs within 500 milliseconds after the collision it counts as a true positive, if it does not occur it counts as a false negative. If the anomaly score is above the threshold value anywhere else other than after a collision it counts as a false positive.

5. All the true positives, false positives and false negatives were then used to calculate the precision, recall, and F1 score for that model.

The drop data.

1. A first value was computed by first using the model on a dataset taken from the training data where the robot operates as normal. The size of that dataset was set equal to the size of the training data. The highest anomaly score output was saved.

2. Then 30 % of the test data with anomalies was set aside and the current model was used to turn each datapoint into an anomaly score. From these time series with anomaly score the highest value within an interval that was set between 500 milliseconds before the first indicator and 500 milliseconds after the second indication was saved. From all these values the lowest one was used.

3. Finally the mean value between these two saved vales was calculated as the threshold value.

4. Then the model was used on the remaining anomaly data (70 %). If an anomaly score above the threshold value occurs within the interval 500 milliseconds before the first indicator and 500 milliseconds after the second one it counts as a true positive, if it does not occur it counts as a false negative. If the anomaly score is above the threshold value anywhere else other than after an object drop it counts as a false positive.

5. All the true positives, false positives and false negatives were then used to calculate the precision, recall and F1 score for that model.

The weight offset data. For the weight offset test an equal amount of normal and anomaly data were used to evaluate the performance.

1. A first value was found by using the model on a training dataset of 5 minutes with normal data and saving the mean value of the anomaly score of this dataset.

2. A second value was found by using the model on an anomaly dataset of 5 minutes and saving the mean value of the anomaly score of this dataset.

3. Finally the mean of these two values was calculated as the threshold value.

4. This threshold was then used together with the model to evaluate the remaining datasets.

5. Step 4 was done by dividing both normal data and anomaly data into small datasets that represent one movement cycle. For the 7.5 kg weight it was approximately 4.5 seconds and for the 2 kg weight it was approximately 2.6 seconds.

6. Then the model was used to get an anomaly score for the small datasets and the mean value of this anomaly score was calculated.

7. The mean anomaly score was then compared to the previously calculated threshold score for both normal datasets and anomaly datasets.

8. If a normal dataset was classified as anomalous by the mean anomaly score being higher than the threshold value it was registered as a false positive, if it instead was classified as a normal data point it was registered as a true positive.

9. If an anomaly dataset was noted as anomalous by the mean anomaly score being higher than the threshold value it was registered as a true positive and if it was not classified as an anomaly it was registered as a false negative.

10. Finally the F1 score was calculated from all the true positives, false positives, and false negatives.

4.3 Data analysis

In this section the different tests conducted on the data gathered are described. The models one class support vector machine, isolation forest and local outlier factor comes from a julia package called OulierDetection.jl [Muhr et al., 2022]. The vector autoregression model comes from the python package statsmodels [Seabold and Perktold, 2010]. Finally, the autoencoder model was created using a package for artificial neural networks in Julia called Flux [Innes et al., 2018].

Comparison between models

When comparing the models to investigate which one performs best, the same data were used on all the models. Because some models were considerably time-consuming only 5 minutes of recorded data were used for each test. This data were randomly selected from the total data available for all experiments.

For all models except the vector autoregression model, several different hyperparameters were used in an interval which was chosen so that the performance of the model started to decrease when increasing or decreasing the hyperparameter values

and so that the training time was within reasonable bounds. For the vector autoregression model, instead the Akaike information criterion was used which measures the prediction error for a given model. Different sizes of previous datapoints between 1–30 were used on the training data for each test and the model with the number of previous values that gave the best result was used.

A comparison was made between the best performance for each model and the result from using only the torque control error (see the following section). This was done for all three experiments using the same data and the same evaluation rules. For the collision test the data gathered from the inner collisions were not included. No windowing of the data was used here because of the considerable increase in training time.

Anomaly detection using the torque control error. To evaluate the benefit of using an anomaly detection model compared to using only the data already available from the control system, another method was also used in the comparison between models. This method was to use the control error from the torque data instead of an anomaly score derived from an anomaly detection model. The control error was computed to be the absolute value of the difference between the feedback and feedforward values of the torque in the control system.

The procedure for evaluating the models described in the previous section was followed with the only difference that the control error was used instead of an anomaly score.

The reason why using the torque control error can be a good method in anomaly detection is because when everything works as intended the mathematical model of the robot dynamics will be close to what is happening, but when an anomaly occurs the mathematical model will no longer fit as well with what the physical system of the robot and therefore the control error is likely to increase.

This method is from here on referred to only as the control error.

Expanded performance evaluation on the autoencoder

In the previous test, a comparison between models was done with a small amount of data because of the high computational time for some of the models. Because of this, another test was performed where a larger amount of data were used. This test was only performed on the collision data but the data from the inner collisions were also included. However, the inner collision data were only used to verify the model, no change was done in computing the threshold value.

For this test, only the autoencoder model was used because it was able to utilize the graphical processing unit instead of the central processing unit when performing calculations and therefore had considerably more computing power.

For this test, 60 minutes of data were used on a few different types of autoencoder networks and the best one was selected for comparison with the result using only the control error. No windowing was used in this test either because of the large expansion of data when using windows.

Feature evaluation using feature groups

One more test, referred to as the feature evaluation test was performed in order to evaluate the importance of different features when detecting anomalies. The features were divided into six different groups which can be seen in Table 5.2. Then an autoencoder model was trained on the data, first on all available features and then using only five of the six groups of features in order to investigate the decrease in performance when excluding one group. This was done only on the data from the collision experiment. In total 60 minutes of training data were used in the training and no windowing was used.

Table 4.1 Overview of the grouping of features when evaluating importance of different features.

Feature groups
Group 1. Motor acceleration data
Group 2: Endpoint accelerometer data
Group 3: Motor positional data
Group 4: Motor velocity data
Group 5: Gearboxes accelerometer data
Group 6: Motor torque data

5

Result and analysis

In this chapter, the result from the experiments described in Chapter 4 is presented. First off is the comparison between the different anomaly detection models. Then follows the test in which the autoencoder model was trained with more training data are compared with the control error data on the collision dataset. Finally, the test comparing different feature groups is presented. Each result is followed by a short analysis.

5.1 Comparison between anomaly detection models

In this section, the result from the comparison between models is presented. The cases of collision detection, object drop detection, and weight offset detection are presented separately. For each test, the F1-score for all models is presented together with the result of using only the control error for the torque data instead of using an anomaly score derived from a model. The control error follows the same rules in computing a threshold value and evaluating the performance as for the anomaly detection models. The evaluation score using the control error from the torque data is referred to as the control error.

Collision detection

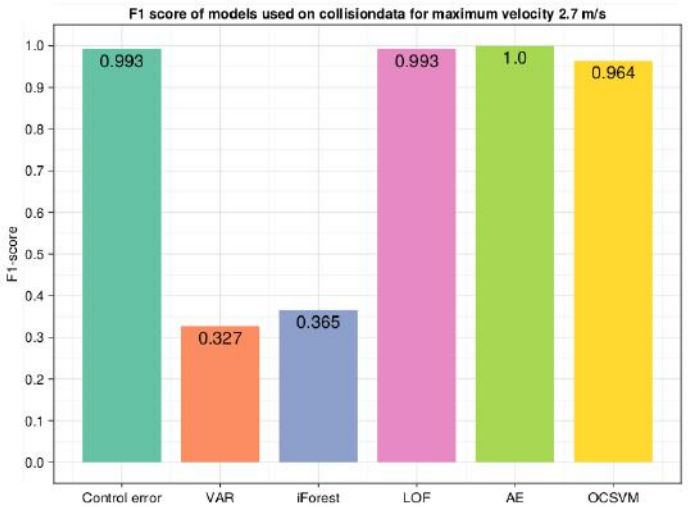

Figure 5.1 F1 score for collision test with maximum velocity 2.7 m/s.

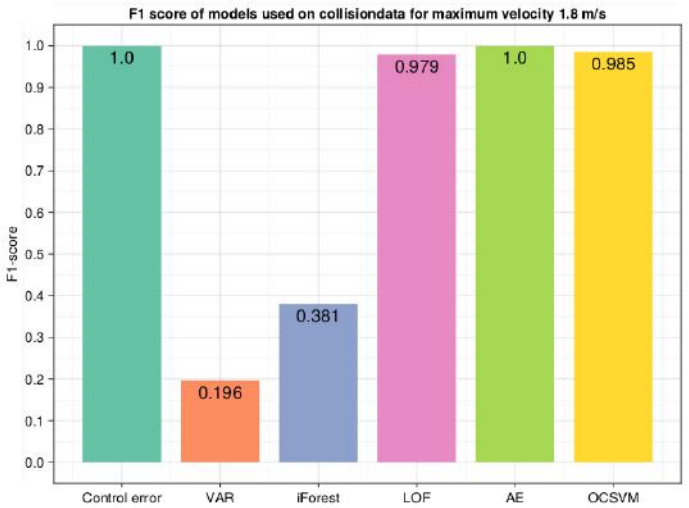

Figure 5.2 F1 score for collision test with maximum velocity 1.8 m/s.

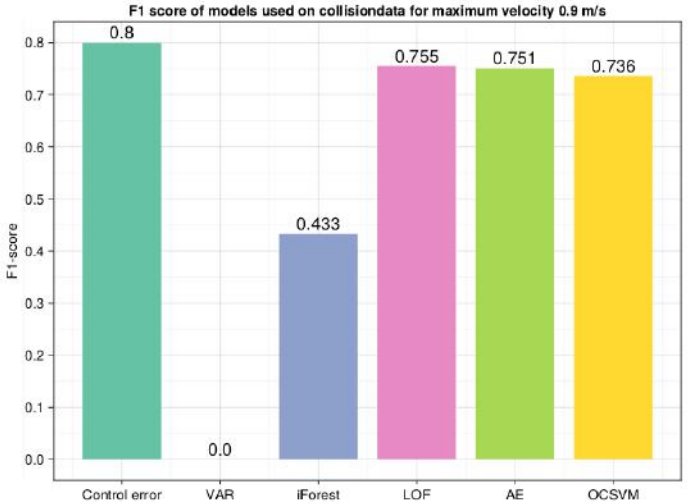

Figure 5.3 F1 score for collision test with maximum velocity 0.9 m/s.

Analysis. The result shown in Figures 5.1, 5.2 and 5.3 gives a clear indication that the system is affected by the collision and that it is possible to detect both using only the control error of the torque and with the anomaly detection models. The high performance using only the control error shows that there is a larger error in the control system than usual during a collision. It also seems like there is a positive connection between the maximum velocity of the robot and the likelihood to detect a collision. This seems intuitive because a higher velocity also means a greater impact on the robot during the collision.

The performance of using the control error is slightly higher at a maximum velocity of 1.8 m/s than at 2.7 m/s. A possible reason for this is that at higher velocity, dynamic effects not included in the mathematical model of the robot such as elasticity in the joints plays a larger role and therefore increase the control error. A higher velocity could also increase the risk of values becoming very off for some data points since the overall energy in the system is higher.

Even though the result is promising, collision detection is a critical anomaly to detect since it concerns the safety of both the robot as well as people close to the robot. Therefore, the requirement for the performance of a model detecting collisions is higher than for the other types of velocities and it would need to be even better, also for a low velocity to be practically useful.

Object drop detection

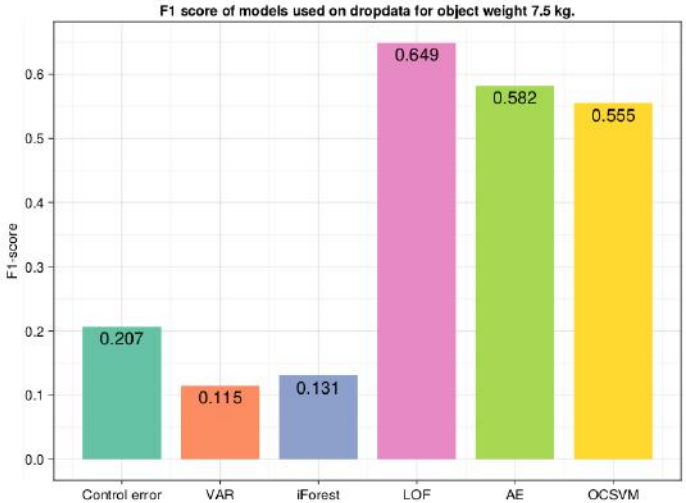

Figure 5.4 F1 score for models on the dropping object experiment for 7.5 kg weight.

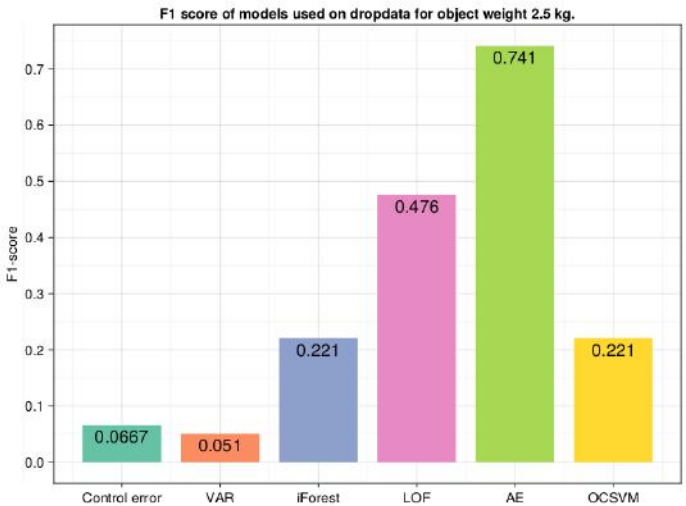

Figure 5.5 F1 score for models on the dropping object for 2.5 kg weight.

Analysis. The result shown in Figure 5.4 and 5.5 indicates that it is possible to detect when the robot unexpectedly drops an object. The task seems significantly harder than the collision test, however. In particular the performance of the control error is drastically lower.

A possible explanation for the low performance of the control error is because of something during the gathering of data for this experiment, the model of the robot did not include the weight of the object being carried. The assumption was that including the weight would not have a significant impact on the control system, but based on the performance it seems that was not the case. Also, figures of the control error during the drop detection test show that the control error is lower after the object has been dropped than before. This fact makes it hard for the control error to identify the anomaly. However, the result for a heavier object in Figure 5.4 shows a higher evaluation score than for a lighter object shown in Figure 5.5. Assuming that the disturbance of dropping a heavier object is true, would indicate that the event of an object being dropped can still be noticed by the system.

The fact that the weight of the objects was not included in the model has likely decreased the performance of the other models as well since they also use the feedback and feedforward torque values as features. This is an indication that for optimal performance in detecting anomalies, great care needs to be taken to ensure that the model is good.

Another interesting aspect is that the best performing models perform better on the 2.5 kg weight than on the one with 7.5 kg. A possible explanation here is again because of how the data were gathered. A carbon box was used as an object when collecting data and for the test with 7.5 kg, the vacuum was unable to carry the box because the carbon box surface not being stiff enough. Because of this, a plastic lid was attached for the test on 7.5 kg which was not used for 2.5 kg. The plastic lid had the unintended consequence that the time between the vacuum pressure was turned off and the robot dropping the object was considerably longer than for the test with the 2.5 kg weight. This in turn might have dampened the effect of the object dropping and therefore made it harder to detect.

Weight offset detection

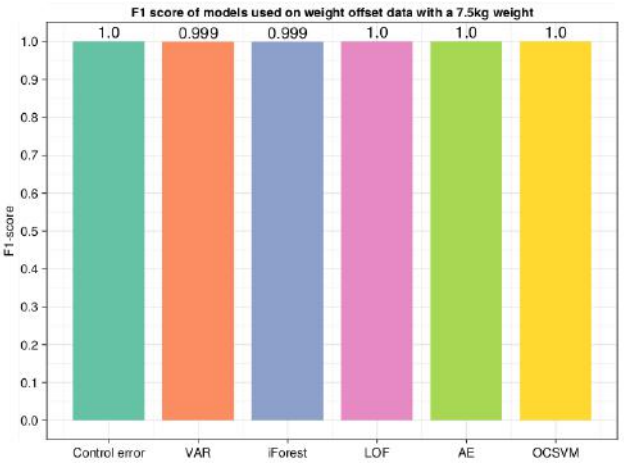

Figure 5.6 F1 score for model on weight offset test for a 7.5 kg weight.

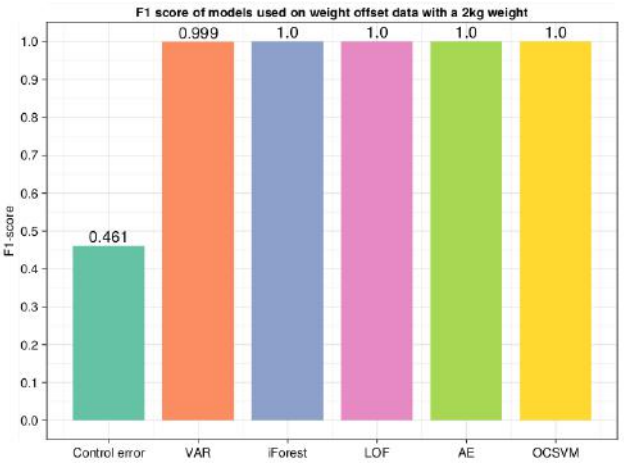

Figure 5.7 F1 score for models on weight offset test for a 2 kg weight.

Analysis. The results in Figures 5.6 and 5.7 show that all models perform exceptionally well in detecting whether the set weight on the model corresponds to the real system or not. One possible explanation of the good results is that in calculating

the threshold value and in evaluating the performance the mean value of one trajectory cycle is used. This could mean that unintended outlier data points do not give such a big impact on the evaluation as might be the case in the previous tests. The data gathering is also done differently in this experiment compared to the other two. With the other two many short time series of a few seconds where the anomaly occurs were recorded, this increases the risk of an error happening when starting and stopping the recording. In this experiment instead a long time series of 10 minutes was recorded and then afterwards divided into small parts which reduces the risk of error.

An interesting part of the result is that using only the control error does not give a better result than the anomaly detection models, this is especially interesting since a weight offset should give the biggest impact on the control error data compared to the other features. A possible explanation is that a difference in weight settings for the model also has a large effect on the features other than the control error, such as more vibrations at the end effector that gets picked up by the models. The conclusion seems to be that having a small weight attached to the robot increases the control error, even if the weight is included in the model of the dynamics. This could be because a weight partly impacts the system in a way that is not included in the model and therefore increasing the control error compared to having no weight attached at all.

Finally, looking at Figure 4.7 we see that the lighter weight has a larger gravity point offset than the heavier weight. The adjustment that is made on the model by adding this weight should take this into account but perhaps it still creates a larger control error than anticipated.

5.2 Expanded evaluation of the autoencoder model compared to the control error

An additional evaluation was done only on the collision experiment data but including the data from the inner collisions, which the previous test on model comparison did not. This means that this result can not be directly compared to the previous one. However, since this test includes collisions on other parts than the end point of the robot this is a more conclusive evaluation of how well the models can detect a collision with the robot.

Table 5.1 F1 score of the autoencoder model and control error on collisions and inner collisions at all velocities.

F1 score result for control error and autoencoder on collisions experiment		
	Control error	Autoencoder
2.7 m/s	0.836	0.9
1.8 m/s	0.824	0.865
0.9 m/s	0.654	0.667

Analysis. The result shown in Table 5.1 shows that the autoencoder model performs slightly better on this test when compared with the control error. This could be an indication that a bigger training set can further improve the model and outperform using only the control error, but it could also be that other features than the feedforward and feedback values from the torque are more important in detecting collisions that occur closer to the center of the robot.

5.3 Feature importance for detecting anomalies in collision detection

Figures 5.8, 5.9 and 5.10 represent the score from the autoencoder when removing one feature group for the collision experiment data. The feature groups are presented in the method section and shown here again in Table 5.2.

Table 5.2 Overview of the grouping of features when evaluating importance of different features.

Feature groups
Group 1. Motor acceleration data
Group 2: Endpoint accelerometer data
Group 3: Motor positional data
Group 4: Motor velocity data
Group 5: Gearboxes accelerometer data
Group 6: Motor torque data

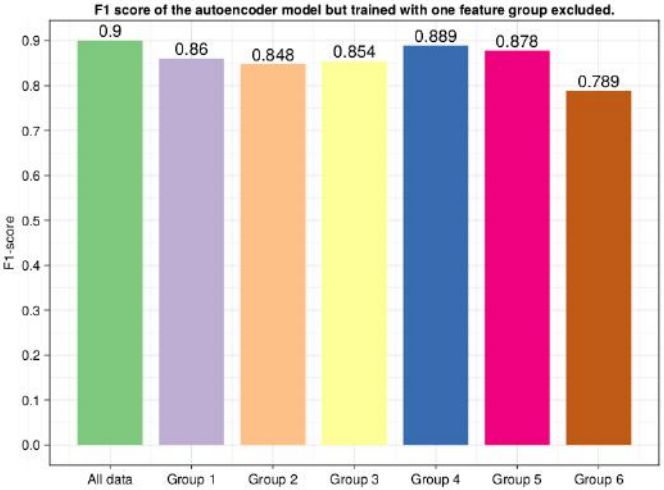

Figure 5.8 Evaluation of the significance of different feature groups when detecting collisions at maximum velocity 2.7 m/s. The x-axis label indicates the feature group excluded.

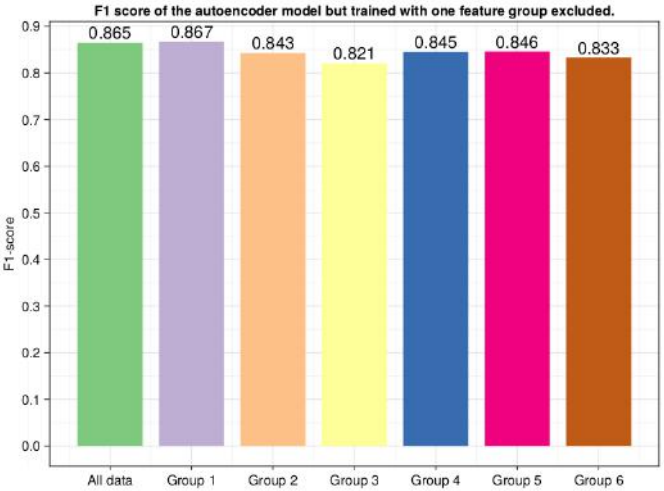

Figure 5.9 Evaluation of the significance of different feature groups when detecting collisions at maximum velocity 1.8 m/s. The x-axis label indicates the feature group excluded.

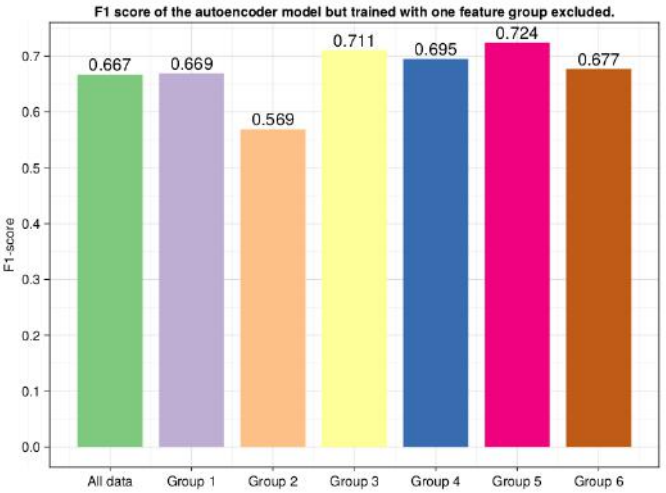

Figure 5.10 Evaluation of the significance of different feature groups when detecting collisions at maximum velocity 0.9 m/s. The x-axis label indicates the feature group excluded.

Analysis

The evaluation score is overall even when excluding a feature group, so one should be careful making too large conclusions based on this test. At least it seems like no single feature group is solely responsible for detecting collisions.

In Figure 5.9, which shows the evaluation for a maximum velocity of 1.8 m/s, the result is similar and these variations could just be a result of the natural randomness that occurs when training an artificial neural network.

At a higher maximum velocity of 2.7 m/s shown in Figure 5.8 we see a slightly lower result when excluding the torque data, this could imply that this feature group is slightly more important when detecting anomalies at a high velocity. This could be because the endpoint accelerometer is noisier at a high velocity and less useful in detecting collisions and a higher velocity means a greater impact which should influence the torque data more.

At a lower maximum velocity of 0.9 m/s shown in Figure 5.3, we see that the evaluation score is higher when removing some feature groups, the difference is not high so it is hard to conclude from this but it could be because it is generally more difficult detecting collisions at a lower velocity. Therefore, removing the feature groups that contribute the least in detecting collisions namely group 3, the motor positional data, group 5, the gearbox accelerometer data, and perhaps also group 4 the velocity data improves the model in detecting anomalies.

Another interesting aspect with the result shown in Figure 5.3 is that the model

performs worse when removing the endpoint accelerometer data. This could be because at a lower velocity there are no large vibrations that need to be dampened which means that there is less noise in the data and that the accelerometer data are more sensitive than for example the motor torque data. This means that the endpoint accelerometer data are more important than the others in detecting a collision at lower velocities.

6

Discussion

This chapter contains a more in-depth discussion of some interesting questions based on the previously presented result. First, a deeper investigation of the difference between the models is presented, and why they produce so different results in detecting anomalies. Then the question of whether it is necessary at all to use anomaly detection models and not simply use the control error from the control system is investigated more deeply. Lastly, uncertainties and potential sources of error in this study are presented.

6.1 Comparison between models

The two superior models for detecting the anomalies studied in this thesis are the local outlier factor and the autoencoder.

The principle of the local outlier factor is that it compares the data points close to a data point to determine its anomaly score. A problem with this is that all data points used in the training stage need to be saved for when detecting the anomaly. It also seems like the more training data that are used, the longer it takes to evaluate each data point. This could prove to be a problem in a real application where memory and computing time are limited, especially if the task becomes more complex and more training data than were used in this thesis are needed.

The autoencoder on the other hand also has a constant evaluation time regardless of how much training data it has trained on because the only thing that happens with more training data are better-tuned weights in the artificial neural network. One potential problem with the autoencoder is that it would not be complex enough to be able to detect anomalies when the anomaly detection task becomes more complex with for example more variation in trajectory and movement of the robot, a possible way to counteract this is to expand the number of layers used but this comes with the drawback of a longer computation time in the evaluation stage so a compromise would have to be found here.

It is hard to pinpoint exactly why some of the models perform so poorly compared to the best-performing ones. One explanation is of course that the hyperpa-

rameters were not well enough tuned. Figures 6.1, 6.2, and 6.3 show the anomaly score for different models on the same dataset, a collision with maximum velocity 1.8 m/s. In Figure 6.1, which depicts the anomaly score for an autoencoder model, we see that the anomaly score after the collision is considerably higher than other values. For the isolation forest in Figure 6.2 on the other hand, it can be seen that the model is able to detect the collision but also that the model does not score that much higher than other data that are to be considered normal data. The conclusion here is that this model is not complex enough to encompass all the normal data and therefore has difficulties distinguishing these from anomalous data. The VAR model on the other hand in Figure 6.3 also reacts to a collision but reacts considerably more to other data points. This also shows that the model is not sufficiently tuned to what data points count as normal data. Another aspect could be that some models are better at identifying what features are most important in detecting anomalies. In Figure 6.2 it seems like the anomaly score follows a pattern that could be the positional data from the robot or other data that is cyclic. This pattern is non-existent in Figure 6.1 with the anomaly score for the autoencoder where it gives all normal data points roughly an equal score. Thus, it encompasses all data in a trajectory cycle more or less equally normal and the isolation forest model does not.

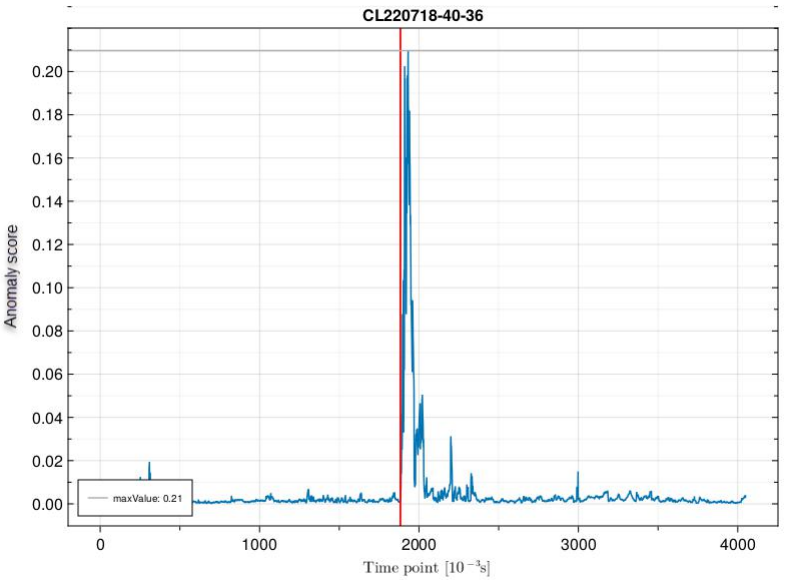

Figure 6.1 Anomaly score for the dataset CL220718-40-36 using the autoencoder model. The red line is the indication from the reference signal that the collision has occurred.

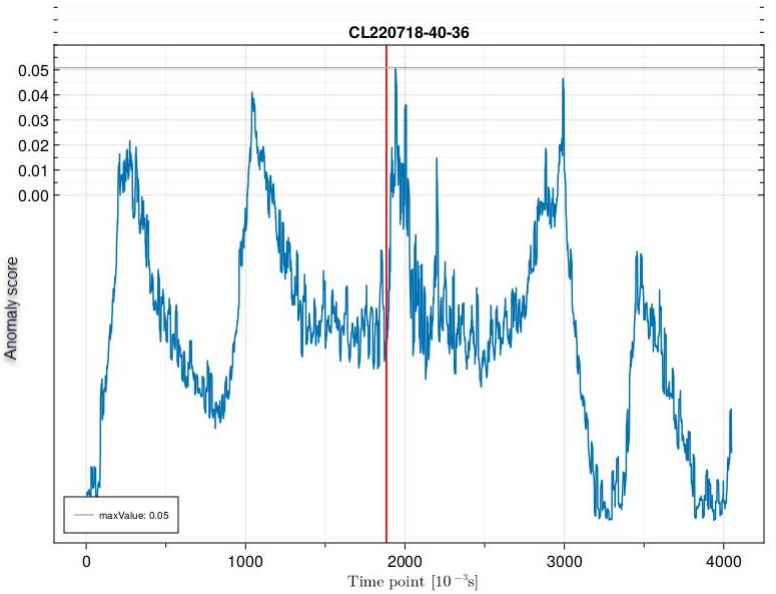

Figure 6.2 Anomaly score for the dataset CL220718-40-36 using the isolation forest model. The red line is the indication from the reference signal that the collision has occurred.

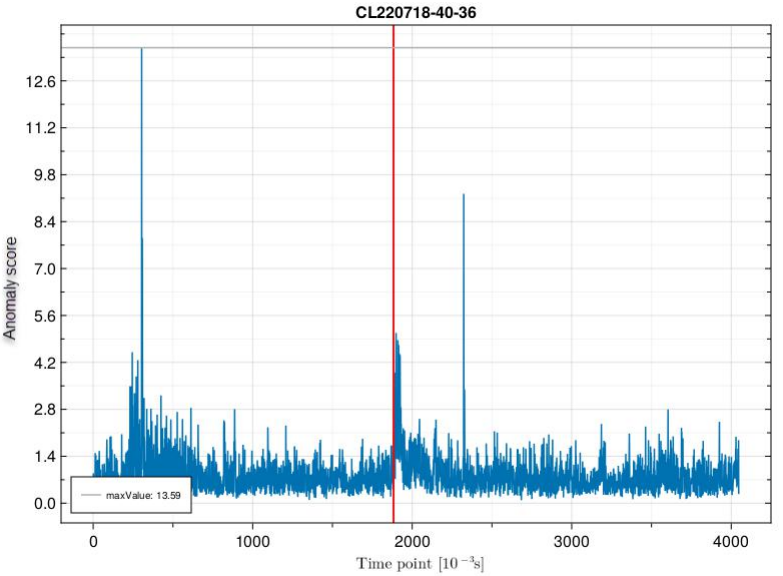

Figure 6.3 Anomaly score for the dataset CL220718-40-36 using the vector autoregression model. The red line is the indication from the reference signal that the collision has occurred.

6.2 Is an anomaly detection model necessary?

An interesting question is whether an anomaly detection model is necessary at all or if one can simply use the control error to identify anomalies. The result from the collision detection test shown in Figures 5.1, 5.2 and 5.3 shows that the performance of using only the torque control error is even better than the other models.

The result from the object drop experiment in Figures 5.4 and 5.5 can not be used to conclude that the control error is ineffective at detecting object drops since the model of the robot dynamics was off. It shows, however, the importance of having a representative model of the system to effectively use the control error. If the control error would be used in a practical setting it would be imperative that the model of the robot dynamics is as good as possible, and accounts for the payload.

Several of the experiments performed indicate that using the control error performs worse when the anomaly could be considered harder to detect, for example in detecting weight offset and at object drop detection for a 2.5 kg weight compared to a 7.5 kg weight, and the result for the collision experiment at a lower velocity is also worse for using the control error. This is especially interesting since Figure 5.10 seems to indicate that the end-point accelerometer data are more useful in detecting

more subtle anomalies.

Finally, the result from the expanded test with the autoencoder that used more data, which is presented in Table 5.1, shows that the autoencoder is on equal footing with using the control error when given more training data.

To summarize, the control error data are an important part of detecting anomalies, and using only the control error without any anomaly detection model works decently in detecting anomalies but it requires that the model being used is good. Also, when the anomalies are more subtle it seems like other data play a larger role compared to the control error in detecting anomalies.

6.3 Uncertainties and sources of error

The method used for finding a threshold value that decides whether a data point is an anomaly or not is rather blunt. Because it selects the highest value from the reference dataset and the lowest value from a set of testing data it ironically becomes quite susceptible to outliers. The basic principle of the method used to select a threshold value is that a perfect model will give a perfect score but, likely, a more advanced method for selecting a threshold value that uses some form of averaging of data would improve the evaluation score for the models.

Since it is not possible to know exactly when an anomaly occurs, in the process of detecting whether the model has successfully detected an anomaly or not, an interval around where the anomaly has occurred is used. The interval is chosen by a careful study of how the system reacts to the anomaly but could be too generous which would imply that false positives would be wrongly labeled as true positives if they occur within that interval. Also, there is no way of knowing if something other than the anomaly triggers a false positive inside of the interval, this will count as a true positive only because it is within the aforementioned interval, even if the high anomaly score is not because of the anomaly. This is why the evaluation score presented only is an indication of how well the models perform and should not be taken as the true performance of a model.

All data are recorded by manually making the robot move, adjusting the velocity, adjusting all settings in the software, and making a program record the data, this opens up the possibility that some of the data might not have been recorded properly. During the collision data gathering, some collisions have been recorded with a tool equipped on the robot and others have been recorded without a tool to prevent damage to the robot. These small modifications might have slightly altered the physical robot from the model used in the control system. Possible errors in recording data may have affected the result and may account for some of the false positives of the models. Since the steps where an error occurs happens at the beginning and end of a data recording, this has most likely only affected the test data and not the training data since there are so many short instances of data recordings for the test data compared to the recording of the normal data.

The object drop experiment was done with two different weights which is the main interesting aspect but because of that, the 7.5 kg object kept dropping unintentionally at the higher velocity and a lower velocity had to be used in data gathering for the drop of the heavier object. This extra variable might affect the comparison between the two different experiments. The fact that a plastic lid was used with the heavier object might also have affected the result and made it more difficult in detecting this anomaly. This might have affected the comparison in detecting object drop with different weights since several factors were involved but the evaluation of each anomaly is not affected, and the weight difference is still a major difference between the two experiments.

7

Conclusion

The overall result of this thesis is that the system; the robot, and its sensors are affected when being introduced to external anomalies and that it is possible to detect them using anomaly detection models. This is also possible to some extent, using only the torque data from the control system.

The result from the comparison between models shows good results in detecting weight offset and collisions. The result from the object drop experiment is not as good but this might be because the model is not being adjusted for the weight carried during the experiment.

The models showing the most promise from the ones studied are the local outlier factor model and the autoencoder model. Out of these two, the local outlier factor model has some drawbacks such that the computation time in detecting an anomaly increases with more training data, which could lead to a problem when using the model in a real-time application. The autoencoder performs well and does not have a computation time in detecting anomalies that depends on the amount of training data.

Based on the feature group evaluation presented in the result and analysis chapter the most important feature groups to use in detecting anomalies are the feedback and feedforward torque data from the control system and the end-point accelerometer data normally used to reduce vibrations in the robot. It seems like the end-point accelerometer is most important to detect collisions at a lower velocity, which is a more subtle anomaly than a higher velocity impact. The torque data, on the other hand, seems more important in detecting anomalies at higher velocities. However, the result only gives small indications of this and there might also be a more intricate interplay between the different groups of features that are not captured in this study.

Future work

Because of the positive result in this thesis a natural progression is to increase the complexity of the anomaly to be detected with for example more variation in the trajectory and velocity. It would also be interesting to find the lower limit for when it becomes harder to detect collisions, object drops and weight offset.

Furthermore, the goal is to make an anomaly detection system practically viable. This means that more focus needs to be put on implementation. For an anomaly detection model to be able to be implemented in a real-time system focus need to be put on the time complexity of the model and this needs to be balanced with the performance of the model. The autoencoder was one of the best performing models in this thesis which is promising and there are also more advanced variations of the autoencoder like the convolutional variational autoencoder that seems to perform well in a real-time setting [Chen et al., 2020]. Another model with similarities to the autoencoder is the convolutional neural network (CNN) model which also shows promise in real-time anomaly detection [Wyk et al., 2020] and could be investigated further.

Bibliography

Akaike, H. (1974). "A new look at the statistical model identification." *IEEE Transactions on Automatic Control* **19**:6, pp. 716–723. ISSN: 0018-9286. DOI: 10.1109/TAC.1974.1100705.

Audibert, J., P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga (2022). "Do deep neural networks contribute to multivariate time series anomaly detection?" *Pattern Recognition* **132**, p. 108945. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.108945.

Beckhoff (10, 2022). *Te1000, twincat 3 engineering*. Version 3.1. URL: https://www.beckhoff.com/en-en/products/automation/twincat/texxxx-twincat-3-engineering/te1000.html.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2017). "Julia: a fresh approach to numerical computing". *SIAM review* **59**:1, pp. 65–98. DOI: https://doi.org/10.48550/arXiv.1411.1607.

Breuniq, M., H.-P. Kriegel, J. Sander, and R. Ng (2000). "LOF: Identifying density-based local outliers." *SIGMOD Record (ACM Special Interest Group on Management of Data)* **29**:2, pp. 93-104 –104. ISSN: 01635808. DOI: 10.1145/335191.335388.

Chandola, V., A. Banerjee, and V. Kumar (2009). "Anomaly detection: a survey." *ACM Computing Surveys* **41**:3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

Chen, T., X. Liu, B. Xia, W. Wang, and Y. Lai (2020). "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder." *IEEE Access* **8**, pp. 47072–47081. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2977892.

Cognibotics (2022). *Hkm product information*. https://cognibotics.com/wp-content/uploads/2022/04/200669-HKM-1800-Product-sheet-rev-AD.pdf. (Visited on 2022-12-08).

Czubenko, M. and Z. Kowalczuk (2021). "A simple neural network for collision detection of collaborative robots." *Sensors* **21**:12, p. 4235. ISSN: 14248220. DOI: 10.3390/s21124235.

Dickey, D. and W. Fuller (1979). "Distribution of the estimators for autoregressive time series with a unit root". *JASA. —Journal of the American Statistical Association* **74**. DOI: 10.2307/2286348.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org (visited on 2022-10-01).

Innes, M., E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah (2018). "Fashionable modelling with flux". *CoRR* **abs/1811.01457**. arXiv: 1811.01457. URL: https://arxiv.org/abs/1811.01457.

Johansen, S. (1991). "Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models". *Econometrica* **59**:6, pp. 1551–1580. ISSN: 00129682, 14680262. DOI: https://doi.org/10.2307/2938278.

Lindholm, A., N. Wahlström, F. Lindsten, and T. Schön (2022a). *Machine learning : a first course for engineers and scientists.* Cambridge University Press, pp. 192–196. ISBN: 9781108919371.

Lindholm, A., N. Wahlström, F. Lindsten, and T. Schön (2022b). *Machine learning : a first course for engineers and scientists.* Cambridge University Press, pp. 86–88. ISBN: 9781108919371.

Liu, F. T., K. M. Ting, and Z.-H. Zhou (2012). "Isolation-based anomaly detection". *ACM Transsactions on knowledge discovery from data* **6**:1. ISSN: 1556-4681. DOI: 10.1145/2133360.2133363.

Min, F., G. Wang, and N. Liu (2019). "Collision detection and identification on robot manipulators based on vibration analysis." *Sensors* **19**:5. ISSN: 14248220. DOI: 10.3390/s19051080.

Muhr, D., M. Affenzeller, and A. D. Blaom (2022). "Outlierdetection.jl: a modular outlier detection ecosystem for the julia programming language". *arXiv preprint arXiv:2211.04550*.

Ohlsson, M. and P. Edén (2021). *Introduction to artificial neural networks and deep learning*. Course manuscript in EXTQ40 at Faculty of Engineering, Lund University.

Park, K. M., Y. Park, S. Yoon, and F. C. Park (2022). "Collision detection for robot manipulators using unsupervised anomaly detection algorithms". *IEEE/ASME Transactions on Mechatronics* **27**:5, pp. 2841–2851. DOI: 10.1109/TMECH.2021.3119057.

Petropoulos, F., N. Kourentzes, and F. Ziel (2022). *International Journal of Forecasting* **38**:3. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.11.001.

Pucher, F., H. Gattringer, and A. Müller (2019). "Collision detection for flexible link robots using accelerometers". *IFAC-PapersOnLine* **52**:16. 11th IFAC Symposium on Nonlinear Control Systems NOLCOS 2019, pp. 514–519. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.12.013.

Schölkopf, B., R. Williamson, A. Smola, J. Shawe-Taylor, and J. Piatt (2000). "Support vector method for novelty detection." In: *Advances in Neural Information Processing Systems*. 12 - Proceedings of the 1999 Conference, NIPS 1999. (1)Microsoft Research Ltd., pp. 582-588 –588. ISBN: 0262194503.

Seabold, S. and J. Perktold (2010). "Statsmodels: econometric and statistical modeling with python". In: *9th Python in Science Conference*.

Wyk, F. van, Y. Wang, A. Khojandi, and N. Masoud (2020). "Real-time sensor anomaly detection and identification in automated vehicles." *IEEE Transactions on Intelligent Transportation Systems* **21**:3, pp. 1264–1276. ISSN: 1524-9050. DOI: 10.1109/TITS.2019.2906038.

A

Hyperparameters used for models

In this appendix the hyperparameters for the best performing models are presented. The best performing models are the ones used when the result is presented in Chapter 5. The hyperparameters for each model are presented in the Chapter 3 but a short summary is given here.

Table A.1 Hyperparameters for each model used in this thesis.

Anomaly detection model	**hyperparameters**
LOF	Number of points close to point p: k
iForest	Subsample size: Ψ and number of trees: t
Autoencoder	Layer size and activation function: φ
VAR model	Previous datapoints used in prediction: lag
OCSVM	Anomaly ratio: ν and RBF kernel parameter: γ

A.1 Comparison between anomaly detection models

Collision detection

Maximum velocity 2.7 m/s.

Table A.2 Hyperparameters used in result for the test collision at maximum velocity 2.7 m/s.

Model	Hyperparameter values
LOF	k : 600
iForest	Ψ : 1500, t : 1500
Autoencoder	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
VAR model	lag : 3
OCSVM	ν : 0.01, γ : 0.2

Maximum velocity 1.8 m/s.

Table A.3 Hyperparameters used in result for the test collision at maximum velocity 1.8 m/s.

Model	Hyperparameter values
LOF	k : 600
iForest	Ψ : 3000, t : 1500
Autoencoder	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
VAR model	lag : 3
OCSVM	ν : 0.01, γ : 0.2

Maximum velocity 0.9 m/s.

Table A.4 Hyperparameters used in result for the test collision at maximum velocity 0.9 m/s.

Model	Hyperparameter values
LOF	k : 600
iForest	Ψ : 3000, t : 750
Autoencoder	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
VAR model	lag : 3
OCSVM	ν : 0.01, γ : 0.2

Object drop detection

Object weight 7.5 kg.

Table A.5 Hyperparameters used in result for the test object drop at object weight 7.5 kg.

Model	Hyperparameter values
LOF	k : 600
iForest	Ψ : 3000, t : 750
Autoencoder	Layer size: (35, 40, 40, 30, 20, 30, 40, 40, 35), φ: relu
VAR model	lag : 4.
OCSVM	ν : 0.01, γ : 0

Object weight 2.5 kg.

Table A.6 Hyperparameters used in result for the test object drop at object weight 2.5 kg.

Model	Hyperparameter values
LOF	k : 300
iForest	Ψ : 3000, t : 3000
Autoencoder	Layer size: (35, 40, 30, 20, 30, 40, 35), φ: relu
VAR model	lag : 4
OCSVM	ν : 0.01, γ : 0.2

Weight offset detection

Weight offset 7.5 kg.

Table A.7 Hyperparameters used in result for the test weight offset with a 7.5 kg offset.

Model	Hyperparameter values
LOF	k : 200
iForest	Ψ : 750, t : 750
Autoencoder	Layer size: (35, 33,31 29, 28, 29, 31, 33, 35), φ: tanh.
VAR model	lag : 3.
OCSVM	ν : 0.01, γ : 0

Weight offset 2 kg.

Table A.8 Hyperparameters used in result for the test weight offset with a 2 kg offset.

Model	Hyperparameter values
LOF	k : 50
iForest	Ψ : 3000, t : 3000
Autoencoder	Layer size: (35, 33,31 29, 28, 29, 31, 33, 35), φ: tanh
VAR model	lag : 3
OCSVM	ν : 0.01, γ : 0.2

A.2 Expanded evaluation of the autoencoder model compared to the control error

Collision at maximum velocity 2.7 m/s.

Table A.9 Hyperparameters used in result for the test expanded evaluation of autoencoder model compared to the control error, for collision at maximum velocity 2.7 m/s.

Model	Hyperparameter values
Autoencoder	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh

Collision at maximum velocity 1.8 m/s.

Table A.10 Hyperparameters used in result for the test expanded evaluation of autoencoder model compared to the control error, for collision at maximum velocity 1.8 m/s.

Model	Hyperparameter values
Autoencoder	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh

Collision at maximum velocity 0.9 m/s.

Table A.11 Hyperparameters used in result for the test expanded evaluation of autoencoder model compared to the control error, for collision at maximum velocity 0.9 m/s.

Model	Hyperparameter values
Autoencoder	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh

A.3 Feature importance for detecting anomalies in collision detection

Because this test was done only with the autoencoder model the first table will instead show which feature group that was removed.

Collision at maximum velocity 2.7 m/s.

Table A.12 Hyperparameters used in result for the test feature importance for collisions at maximum velocity 2.7 m/s.

Removed feature group	Hyperparameter values
All data	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh
Feature group 1	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh
Feature group 2	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 3	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 4	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 5	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh
Feature group 6	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh

Collision at maximum velocity 1.8 m/s.

Table A.13 Hyperparameters used in result for the test feature importance for collisions at maximum velocity 1.8 m/s.

Removed feature group	Hyperparameter values
All data	Layer size: (35, 28, 21, 14, 21, 28, 35), φ: tanh
Feature group 1	Layer size: (35, 28, 21, 14, 21, 28, 35), φ: tanh
Feature group 2	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 3	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 4	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 5	Layer size: (35, 28, 21, 14, 21, 28, 35), φ: tanh
Feature group 6	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh

Collision at maximum velocity 0.9 m/s.

Table A.14 Hyperparameters used in result for the test feature importance for collisions at maximum velocity 0.9 m/s.

Removed feature group	Hyperparameter values
All data	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 1	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 2	Layer size: (35, 28, 21, 14, 21, 28, 35) , φ: tanh
Feature group 3	Layer size: (35, 33, 31, 29, 28, 29, 31, 33, 35), φ: tanh
Feature group 4	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh
Feature group 5	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh
Feature group 6	Layer size: (35, 31, 28, 24, 28, 31, 35), φ: tanh

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
December 2022

Document Number
TFRT-6188

Author(s)
Henrik Paldán

Supervisor
Olle Hedbrant, Cognibotics, Sweden
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Anomaly detection on a hybrid kinematic machine

Abstract
Detecting anomalies is a promising and current research subject that can have useful applications, for example in the field of robotics. In this thesis, anomaly detection is investigated using a hybrid kinematic machine, which is a pick-and-place robot that excels at moving objects at high speed and with great reach. Three different types of anomalies have been chosen to be studied in this thesis; collision, the robot dropping an object, and weight offset between the expected weight that the robot is carrying and the actual weight being carried.

The data were gathered from the robot control system, as well as sensors on the robot when the robot was moving in a specified trajectory cycle. Then, the previously mentioned anomalies were introduced to the robot.

The anomaly detection was conducted by using different anomaly detection models which have certain characteristics. The models were first trained using normal data. Then the trained models and a devised threshold value were used to evaluate how well the models were able to detect the anomalies.

The results are promising, especially for collision detection and weight offset detection. Detecting an object being dropped, however, seems more challenging. The experiments also indicate that a good model of the robot dynamics is of great importance when detecting anomalies. The results also indicate that the most important features for detecting anomalies are the torque data from the control system and data from an accelerometer at the endpoint of the robot. The most promising models for anomaly detection are the local outlier factor model and the autoencoder, which is a type of artificial neural network.

Further work could investigate more varied anomalies that are harder to detect with more advanced models, while also focusing on the models being computationally fast enough to be applicable in a real-time system.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-67

Recipient's notes

Security classification

http://www.control.lth.se/publications/