F3

Faculty of Electrical Engineering
Department of cybernetics

Bachelor's Thesis

Anomaly detection in robotic assembly process using force and torque sensors

Aleš Trna
Study program: Cybernetics and robotics

May 24, 2024
Supervisor: Ing. Martin Macaš PhD.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Trna Aleš** Personal ID number: **507289**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Anomaly detection in robotic assembly process using force and torque sensors

Bachelor’s thesis title in Czech:

Detekce anomálií při robotické montáži založená na signálech z čidel síly a momentu síly

Guidelines:

1. Investigate the topic of anomaly detection in multidimensional time series, focusing on methods suitable for application to time series data collected from force and force moment sensors. Evaluate these methods with regard to their applicability in online detection, continuous learning, and edge computing implementations.
2. Based on the findings from the research, propose at least two anomaly detection methods and apply them offline to the time series data acquired from a delta robot involved in the assembly process. Utilize data provided by the team leader and gather any necessary additional data from the Testbed for Industry 4.0.
3. Implement online anomaly detection for the proposed methods, ensuring real-time monitoring during the assembly process.
4. Develop and implement a robust offline testing procedure for the proposed anomaly detection methods. Evaluate the effectiveness of these methods and analyze the advantages they bring to the detection process.

Bibliography / sources:

[1] Su, Weixing, et al. "AI on the edge: a comprehensive review." Artificial Intelligence Review 55.8 (2022): 6125-6183.
[2] Situnayake, Daniel, and Jenny Plunkett. AI at the Edge. " O'Reilly Media, Inc.", 2023.
[3] Huyen, Chip. Designing machine learning systems. " O'Reilly Media, Inc.", 2022.

Name and workplace of bachelor’s thesis supervisor:

Ing. Martin Macaš, Ph.D. Intelligent Systems for Industry and Smart Distribution Networks CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **01.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Martin Macaš, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgement / Declaration

I would like to thank my supervisor Ing. Martin Macaš PhD. for his guidance and valuable advice during my work on this thesis.

I would also like to thank Votěch Hanzlík, for creating an interface to apply the developed methods for a real-world application in the CTU Testbed for Industry 4.0.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

In Prague, May 24, 2024

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 24. května 2024

Abstrakt / Abstract

Tato bakalářská práce se zabývá problematikou detekce chyb při robotické montáži za pomoci metod používaných pro detekci anomálií. Tyto metody používají data ve formě časových řad získaných ze senzorů na chapadlu robota. Celkem byly navrženy tři architektury metod, které mouhou být použity v různých konfiguracích podle aplikace. Metody byly testovány na úloze detekce chyb v robotickém procesu, ale zároveň byl kladen důraz na obecnost, aby je bylo možné použít in na jiné úlohy z oblasti detekce anomálií v časových řadách.

Navržené metody jsou schopny zpracovávat signály v celé délce (po dokončení zkoumaného procesu) i v průběhu daného procesu, proto je možno metody používat i pro predikci v reálném čase. Navržené metody byly důkladně porovnány podle standardizovaného systému testování. Prioritou při návrhu tohoto systému bylo co nejspravedlivější zhodnocení navržených metod. Tohoto cíle bylo dosaženo důkladným testováním metod pomocí několika typů křížové validace. Zajímavým přínosem této práce je porovnání statistických metod a metody založené na hlubokém učení. Metody byly přidány do PyPI databáze v rámci knihovny ctuFaultDetector, kterou je možno nainstalovat pomocí příkazu pip. Největší výhodou implementovaných metod je možnost optimalizovat metodu pro kritérium (pravděpodobnost detekce, pravděpodobnost falešného poplachu, přesnost, vážená přesnost), které může být pro každou aplikaci různé.

Klíčová slova: Detekce anomálií, Klasifikátor, Statistické metody, Hluboké učení, Detekce chyb

Překlad titulu: Detekce anomálií při robotické montáži založená na signálech z čidel síly a momentu síly

This thesis discusses the problem of fault detection in industrial robotic assembly process utilising the methods used for anomaly detection. These methods rely on data in the form of time series collected from the sensors located on the end effector of a robot. In total, three different method architectures were designed, which can be used in multiple configurations based on the application.

The proposed methods were not only designed to work on the case study of this thesis, but rather to be universally applicable to any time series anomaly detection task. The proposed methods are able to process both the signals in their whole length and the partial signals and therefore are able to make prediction during the run of the process in real time. All the methods were compared to one another in a standardized testing system. Our main priority when designing this testing system was to create an as fair assessment of the method performance as possible. This was achieved through testing the methods thoroughly through multiple types of crossvalidation. An interesting part of this testing process was the comparison of the classical statistical anomaly detection methods, and the deep learning method. The methods were added to PyPI database as ctuFaultDetector library which can be installed using a pip command. The biggest advantage of our implementation is that the methods can be optimised for a desired criterion (true positive rate, true negative rate, accuracy and a new proposed metric called skewed accuracy), which can be different for each use case.

Keywords: Anomaly detection, Classifier, Statistical methods, Deep learning, Fault detection

Contents /

Tables / Figures

Chapter 1
Introduction

Quality control is very important task in assembly process theory and its automation is crucial part of making the industrial processes more efficient. The tedious task done by human operators can be substituted by independent, automatic methods. The aim of this thesis is to develop a reliable method for detecting anomalies in an assembly process based on time series data from the sensors on the end effector of a Delta robot in the `Testbed for industry 4.0` laboratory at the Czech institute of informatics, robotics and cybernetics (CIIRC) under Czech Technical University in Prague (CTU). After researching the techniques, three method architectures were developed to execute this task. Additionally, the methods operating on data streams were built on two of these architectures.

The two of the developed methods could be categorized as classical statistical methods and one method is based on deep learning. The comparison between these two approaches is very interesting since in recent years there is a strong surge in usage of deep learning methods, including usage on tasks which classical methods may solve better.

The primary motivation of developing these methods was the specific application in CIIRC Testbed lab, however they were designed to be applicable to similar time series anomaly detection and classification tasks. These methods were implemented in python and the library is available through installing the PyPI ctuFaultDetector package using `pip` command. The main benefit of this package is that the methods are not available only as a raw form to evaluate them or to use them in our application, but are implemented in an intuitive way to be easily built-on and integrated into other projects.

It is also important to mention that even though the name of the thesis suggest, that the primary motive is to detect anomalies within the signals, The main objective is to detect faults in the process, on which we use the anomaly detection techniques. There is therefore an assumption that the fault in the outcome of the process is caused by an anomaly within the signal.

1.1 Related work

Detection of anomalies in any type of data is an often solved problem. In the continuous processes time series data is often used. In recent years the research has shifted from classical methods to the methods based on deep learning. In this section, experiments which resemble our work will be discussed. No papers addressing the exact same problem were found in the review of related literature, but the ones below share either the general topic of anomaly detection in time series, the topic of assembly process fault detection or the architecture of methods used in these, is similar to the methods used in our case.

In 2021, researchers from the university of Trnava published an article ([2]) about detecting the anomalies in the operation of steel bearings in conveyor belt which is a

part of an assembly line, which made the assembly process halt. The team used neural networks to avoid manually tuning the data. Their research came to a conclusion that a great amount of data is needed for this approach to work.

In 2019, researchers from the university of Padova proposed a detector of abnormal time series in refrigerant insertion into refrigerant cabinet during refrigerator assembly process ([3]). They used a CNN-MLP Bayesian neural network architecture and achieved an f_1 score of 0.743.

In 2021, a team from the polytechnic university in Tomsk [4] developed a simple algorithm for detecting faults of an industrial manipulator. The method is based on mean, minimal and maximal normal actuator currents. The downside of their approach is that only abnormal processes, which signals which breach these thresholds are detectable.

In 2024, a German team created a dataset for detecting failures in object handover between a human and a robot [5]. They also published two methods which they tested on the dataset. This dataset consists of annotated video, force-torque and joint states for successful and failed robot to human and human to robot handovers. They used neural networks and found out that the annotated video is essential for classification, but force-torque sensor is very beneficial. They reached maximum accuracy of 71%.

In [6] two unsupervised approaches were developed - a one class support vector machine and an autoencoder to detect collisions of collaborative robots. The results presented by this paper were exceptional, but the nature of the task upon which is the anomaly detection computed is quite different. In another article pursuing the collisions of collaborative robots the researchers used force and torque sensor placed in the robot's bed plate and used predetermined threshold combined with moving average filter for avoiding noise to detect collisions [7].

Biegel et al. used multiple methods including principal component analysis, dense autoencoder and autoencoder neural network based on LSTM to detect faults in sheet metal forming process for automotive industry [8]. They published outstanding results on all of the methods they examined, however the nature of the dataset used in this article was simpler than the one used in this thesis. Particularly interesting is that they managed to train the autoencoder neural networks to very high accuracy even with a relatively small training dataset.

In [9] (survey paper) many other state-of-the-art methods for detecting faults in industrial processes are listed. They focus mainly on the applications of deep learning methods in the industrial fault detection, but also review the traditional approaches. Another survey paper [10] shows the different types of approaches to the data-stream-driven (online) anomaly detection. Multiple approaches to online learning are discussed.

In 2023, a Slovakian team published an article of using a similarity metric used for time series called the dynamic time warping [11] for anomaly detection in the method they developed. They tested their model on three different datasets. They aimed their efforts at making the model learn from predictions with human assisted continual learning and using human in the loop paradigm.

1.2 Motivation

There are several ways of approaching a problem of quality control in assembly process. Among the ones used commonly in industry is visual approach (a camera that monitors the process). One of the main disadvantages is that camera can see the process only from one scene. If it is necessary to control the process from more angles, either more cameras need to be used or the product or the camera must be moving which

is expensive and complicated. There are also many industrial processes, in which the usage of camera is not possible because of poor visibility or other reasons.

In this section 1.2.1 we will discuss briefly the two approaches to solving the problem. The more in-depth description, with respect to the dataset provided in the appendix, is written in section 2.1. In the following sections 1.2.2-1.2.5 a description of certain machine learning concepts and motivation for their use in this thesis is described.

1.2.1 The problem solved in the thesis

The purpose of this thesis is to design a method for identifying poorly assembled wheels from the force and torque sensors located on the end effector of a delta assembly robot. The method used before developing the methods from this thesis worked by analyzing the photographed assembled wheels after the assembly process was completed. The primary drawback of this approach is, that the camera solely captures the wheels from an overhead perspective. This leaves a room for a lot of undetectable anomalies.

Anomaly type	Recognizable from Camera	Recognizable from Force-Torque
Failure to pick up tire	✓	✓
Tire does not seal the rim at the top	✓	✓
The tire is defected (visible)	✓	✓
The tire is defected (not visible)	×	✓
Tire does not seal the rim at the bottom	×	✓
The rim is defected (not visible)	×	✓
The rim is defected (visible)	✓	✓
Tire is too elastic	×	✓
Tire is too stiff	×	✓
Wrong sized tire/rim ($\geq\approx 5\%$)	✓	✓
Wrong size of tire/rim ($\leq\approx 5\%$)	✓	×

Table 1.1. Types of anomalies and their detectability by the different approaches to solutions.

The most common types of anomalies are listed in table 1.1 along with the detectability by each method. All of these anomalies were encountered when collecting data for training. Some anomalies were artificially evoked, when creating the dataset. The manufacturing of artificial anomalies is discussed in chapter 2.3.4.

1.2.2 Data-stream-driven anomaly detection

Many time series processes can benefit from real time interaction. In the case of anomaly detection in an assembly process, real time interaction can be beneficial, for example in aborting process to prevent collision and thus destroying expensive tools or materials. Another benefit could be making the assembly process more efficient, because when encountered defected parts, the robot will not perform the whole process of assembly, but will rather abort the process, in real time, just when anomaly is defected. In the application described in this thesis, the amount of saved time may not be as significant, because the process is rather short and anomalies are by definition

uncommon, but in other cases, when the monitored process is longer the time saving benefits could be significant. The biggest advantage of data-stream-driven anomaly detection in our case is protection of valuable end effector of the robot. The fingers of the `gripping tool` end effector of the delta robot are 3D printed from expensive material and when collision happens, there is a risk of breaking them. Preventing collision is a big enough motivation to try analyzing the signals from data stream. Throughout this thesis, the methods with capability to operate in real time on data streams are named `online methods`.

1.2.3 Edge AI

Edge AI can be described as implementing AI models on local "Edge" devices. These can be sensors, processors, IoT devices, computers or any other devices that are directly next to the monitored process, whence the input data comes from [12]. Among the positives of this approach is real time online data processing or independence from cloud infrastructure. On the other hand, the local devices which host the AI models do not possess the computing power and memory storage of cloud systems. When designing a method for an edge device, it is important to know the parameters of the device so method of appropriate time and memory complexity can be implemented.

For the purpose of deploying the model on the edge device the code was encapsuled as a Docker container. Because of the fact that, the specific edge device for application is still not known, this approach was used. The dockerization and deployment methods were developed by Vojtěch Hanzlík [1].

1.2.4 Continual learning

In literature, continual methods are defined as methods capable of updating the parameters of the model after the model has been put to production (not only from the training process) [13]. The methods proposed in this thesis can really benefit from such an approach, since parameters of the data can shift over time due to different calibration of the robot, the environment in which the process takes place and other factors. Continual learning is implemented using passive or active learning. In passive learning, the model updates its parameters solely from the incoming data, whereas in active learning the model can be updated from the human intervention. As an example of using human intervention in classifiers, stating ground truth of a sample predicted with high uncertainty can be utilised. All methods proposed in this thesis are designed to not be dependent on human interactions, but in some of the methods, active learning via human in the loop (HITL) paradigm pictured in figure 1.1 can be used.

1.2.5 Supervised vs unsupervised learning

Machine learning algorithms can be divided into two types based on the training process and the nature of the dataset. These are supervised and unsupervised learning algorithms. In literature the definition of these terms is often vague, so it is important to define both of those for our usage.

The supervised learning algorithms rely on labeled training dataset. On the other hand the unsupervised learning algorithms use only raw data without labels during the training process. Instead of learning from labelled examples unsupervised algorithms find hidden patterns within the data and make their predictions based on them.

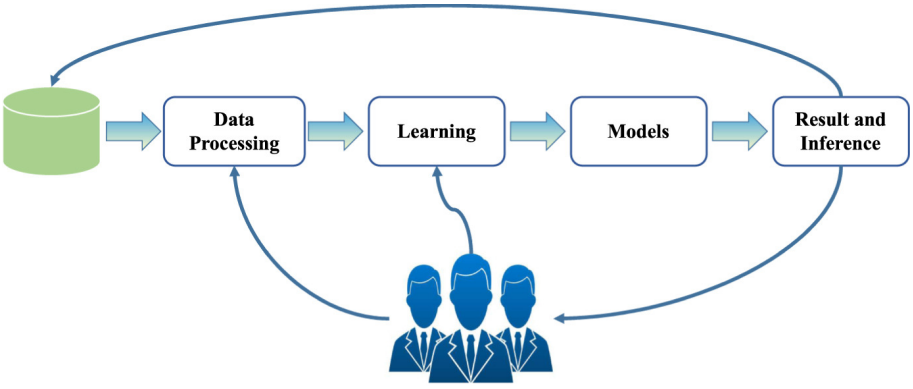

Figure 1.1. Schema of human in the loop (HITL) data processing pipeline paradigm adapted from [14]

The supervised methods in this thesis use the dataset labeled with ground truth labels. Even though the anomaly detectors used in this thesis (with an exception to the method described in section 4.1) are one class classifiers, and therefore could be named unsupervised, we use the labels to train these one class classifiers only on the non-anomalous signals. The `anomaly score` (outcome of the one class classifier) of anomalous signals is then used to set the threshold of anomality. Therefore, we use a ground truth information about anomalies, and call these methods supervised.

On the other hand, the unsupervised learning algorithms we used, are the usual one class classifiers trained without the use of labels. The threshold of anomality is set based on the estimation of the ratio of non-anomalous samples within the dataset (`success ratio`). The unsupervised methods are trained on all the data, and the part of the most deviating data from the model parameters (set as $1 - SR$, where SR is the success ratio) is predicted as anomalous. After this, the classifiers are retrained on the data which remained in the training dataset and the threshold of anomality is tuned based on these predicted labels.

Supervised models generally outperform unsupervised ones, which was also demonstrated by this thesis, however unsupervised models offer the significant advantage of not requiring labeled dataset, which in some cases may be challenging and expensive to obtain.

Chapter 2
Dataset

The goal of this thesis is to design a fault detection system for an assembly process in the CIIRC Testbed for Industry 4.0 laboratory. Therefore, the models developed are trained and tested on data acquired from this facility. Even though solving the problem of anomaly detection for this very specific process is the main motivation for developing the presented models, the origin and nature of the data is not really important. The methods are designed to be applicable to any time series data.

In this chapter the origin of the dataset used for training and testing the methods is described.

2.1 Process

The kinematics used for the robot used in the case study of this thesis is called the delta robot. This type of architecture is used when high speed and precision is needed, but comes at the cost of reduced operating space or higher cost of the architecture [15]. The delta robot can be seen in the figure 2.1.

Figure 2.1. Delta robot in CIIRC Testbed for Industry 4.0 lab adopted from [15].

The delta robot performs an assembly process of a wheel, which is a task in a production line at CIIRC Testbed lab with the goal to manufacture a model of a car. A wheel consists of the rim and the tire. One process involves assembling four wheels on a pallet, which can be divided into these four stages:

1. The position of rims and tires placed inside a rectangle bounded by *Aruco* labels is identified
2. The "suction tool" is used for picking up the rims and move them onto the pallet for assembly
3. The "tire gripping tool" is used for picking up tires and put them on the rims
4. This pallet of four assembled wheels continues to next station of the assembly line

The assembly process fails during parts one, two and four of described process in less than 1% according to the engineers at CTU Testbed, which is acceptable. However, the fail rate during the process of putting the tires on the rims is higher and this process needs a control mechanism for fail checking. The example of correctly and poorly assembled wheel can be seen in figure 2.2.

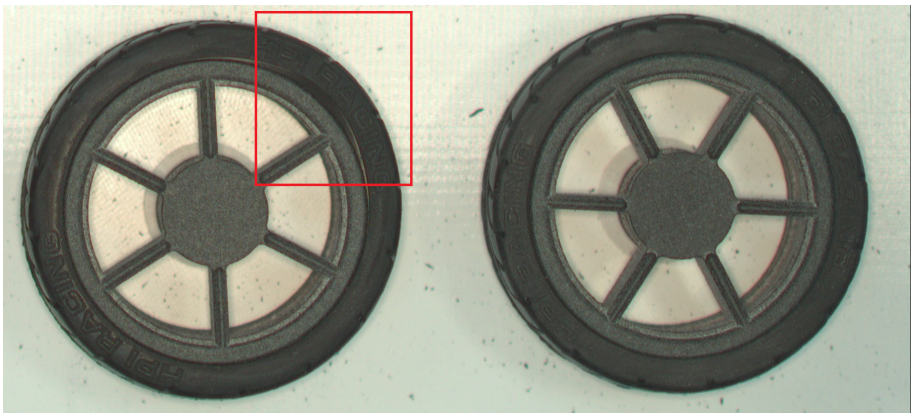

Figure 2.2. Comparison of poorly (left) and correctly (right) assembled wheel. The left tire is not tightly attached onto the rim. This is highlighted in the red rectangle.

2.2 The nature of the data

The end effector of the delta robot is equipped with SCHUNK Mini58 force/torque sensor ([15], [16]), which can measure these quantities in all three dimensions. The output of the sensor is constantly streamed through Profinet and can be read via OPCua protocol [1].

All the signals within any dataset used for training our methods need to be multivariate time series with the same dimensionality. The dataset for our case study contains six-dimensional signals, where dimensions are the three forces and three torques in the directions of the x, y and z axes. For the methods to work it is necessary that the columns within the matrix for each dimension are fixed[1].

As has been mentioned in one process run, four wheels are assembled on the pallet. We need to process the data and pick out the four individual processes of equipping the rim with a tire. To identify this an `identifier` variable is also read when measuring the data. When the process of putting the tire onto a rim starts, this variable changes from zero to the value of the process id. In the context of the rest of the thesis, the

[1] In terms of implementation, all the methods are designed to take a list of tuples *($n \times 6$ numpy array/pandas DataFrame, label)*, for the supervised methods and *($n \times 6$ numpy array/pandas DataFrame, anything (usually None))* for the unsupervised. Both the offline and online predicting methods expect an $n \times 6$ *numpy array* with data from each dimension of the signal being in one fixed column of the matrix.

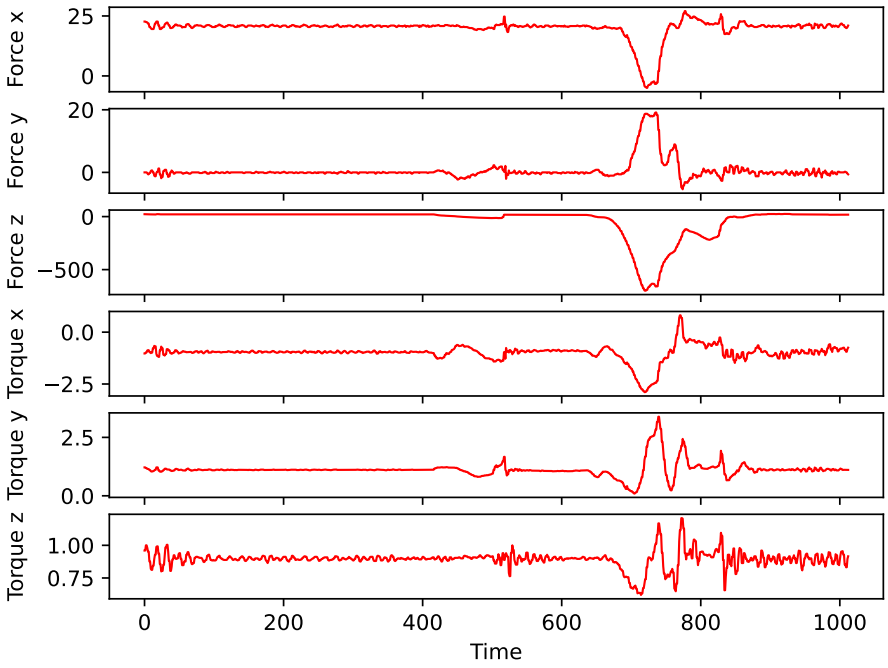

Figure 2.3. Sample process signal of putting the tire onto the rim

terms `signal`, or `time series` refer to the signal read during an individual process of putting tire onto a rim. Therefore, four signals are measured during each pallet cycle.

The example of data acquired from monitoring a process of equipping the tire onto a rim can be seen on picture 2.3. The distribution of the normal and anomalous processes within one measuring day for a particular dimension can be seen in the histogram in the figure 2.4. To create this figure, the value of each signal was sampled in time and the values of the signal were quantized. The value of each point (the z axis in the figure) is the number of samples which fall into each quantized block. The red values are a result of anomalous signals, the blue of the non-anomalous signals.

The data from the sensor is read in constant intervals, so the timestamps do not need to be saved with the data itself, and if they are necessary, they can be recomputed from the sampling period and the index of the sample.

2.3 Data acquisition and preparation

At the start of this thesis we were given one hundred samples of the process by the CIIRC Testbed team, however for the successful development more samples were needed. Thus, several measurements of data were performed. In the end, 524 samples of the process were collected, although not all of them are labeled properly. The number of samples with in-depth labeling containing the type of anomaly is 292. The number of correctly binary-labeled (true/false) processes is 392. All signals from this dataset are plotted in figure 2.5. From the first glance, it can be seen that the dataset consists of a lot of fault types, which exhibit varying degrees of anomaly. The results of the measurement process are visualized in figure 2.6. The ways of labeling used for data in each day are listed in the table 2.1.

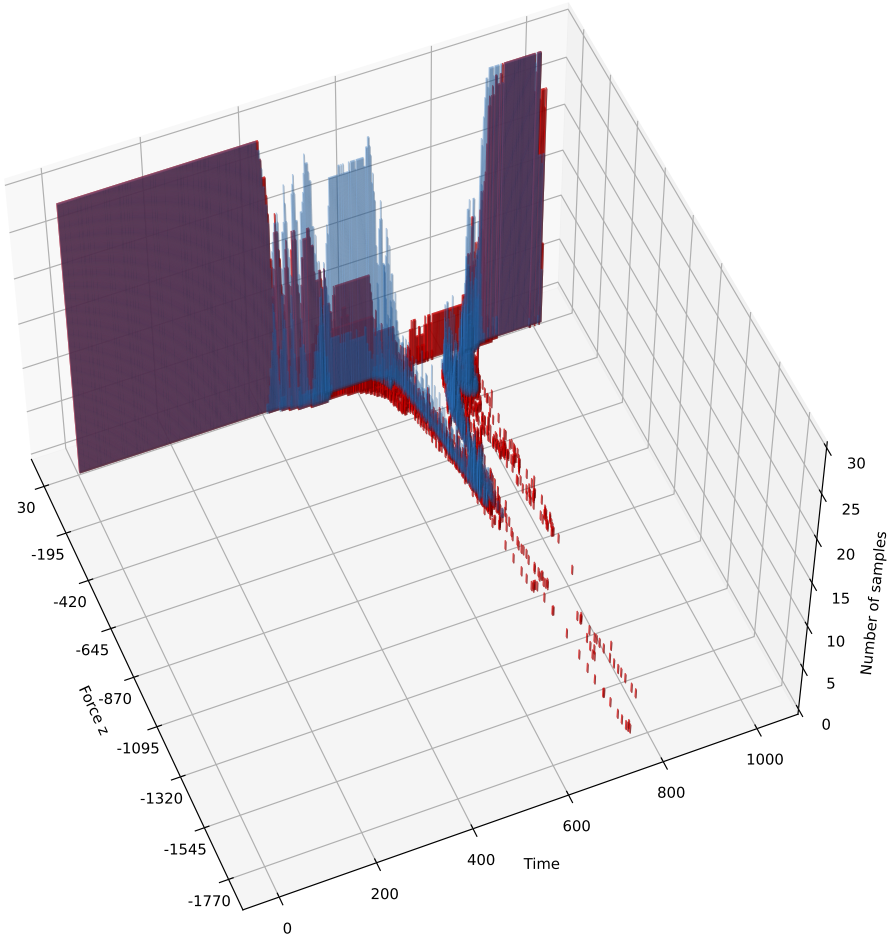

Figure 2.4. Histogram of values of force in the z axis for correctly executed and fault processes. The blue histogram comes from the correctly executed processes, the red histogram comes from the fault processes. The data in this histogram comes from 14.03.2024 measuring day, since the number of successful and anomalous processes is the same in this measuring day.

2.3.1 Data measurement

To measure and acquire data a script that communicate with robot was created. This interface was created by Vojtěch Hanzlík in [1]. The script interacts with OPCua protocol and reads and stores data from the robot in the form of *numpy array*.

The dataset we used for experiments consists of data from four measuring days. In the dataset a problem of worsening the quality of tires was encountered. The tires used in the assembly process are designed to be put on the rims only once, however at the time of creating the dataset, only a few prototype tires were available. Thus, the tires had been reused a great number of times, which has negatively impacted their quality. During some measuring days, the quality of tires was not monitored properly, even though the labeling of data from these sessions exists, for the purpose of preventing false labeling, this data was used (and is available) only as unlabeled.

2.3.2 Labeling the dataset

When using supervised methods, the data must be labeled in order to train the model. Several labeling methods were used for the measured data. The first basic technique,

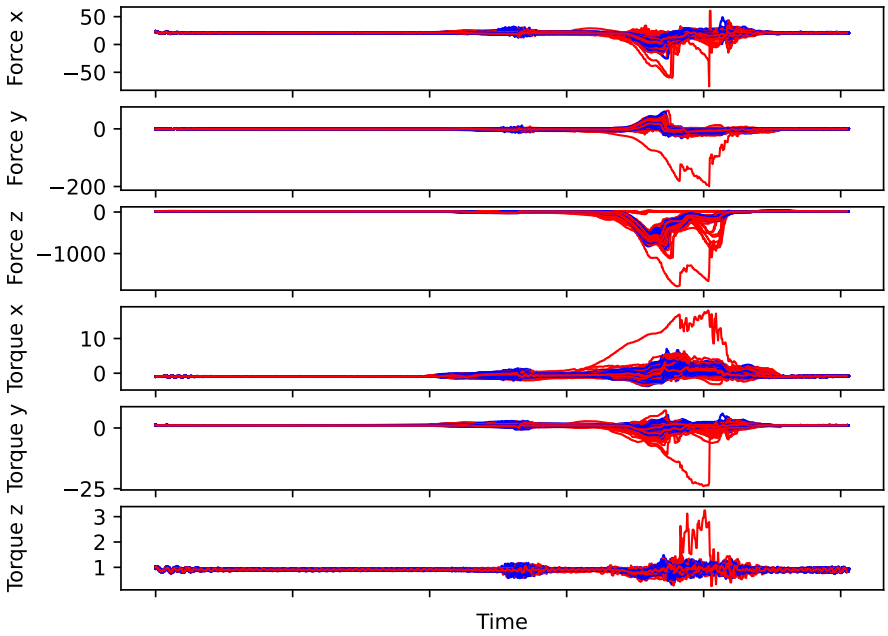

Figure 2.5. A visualization of all the processes from the dataset used for evaluating the methods (392 signals). Blue color represents a correctly executed process and red color represents anomalous process

which is available for all measurements was that the labeling was performed by a convolutional neural network (CNN) developed by Artem Moroz at CIIRC Testbed lab. After the assembly process ends, the result in assembled wheels on a pallet is photographed from an overhead perspective. The photo is then downsampled and used as an input to this CNN. The network returns a True/False label.
As was mentioned in chapter 1.2, the main drawback of this labeling method is that many types of anomalies are undetectable by this method, for example the one in figure 2.7. That is why for some measurements human labeling was used. The labeling done by human is not binary, but rather classifies the process either as successful or as:

- Failure to pick up tire
- Defected tire
- Defected rim
- Tire does not seal the top of the rim
- Tire does not seal the bottom of the rim
- Tire does not seal the top and bottom of the rim
- Other

This extended labeling was done in order to train models to diagnose specific anomaly type. Even though such models were not developed in this thesis, in the future we plan to experiment with this concept. The labels were available only for offline methods. The labeling for evaluating the online methods is only semi-supervised and is described in section 5.2.1.

The results and the distribution of the labeling for each measuring day can be seen in the figure 2.6. Additional info for each measurement day can be seen in the table 2.1.

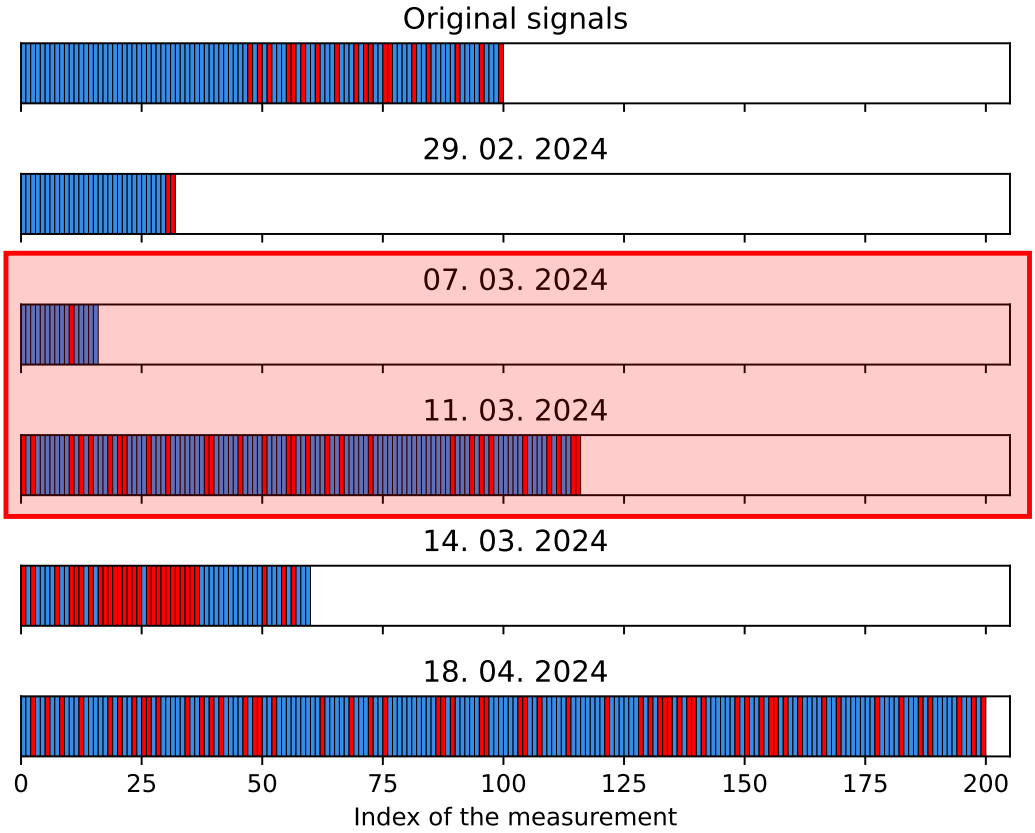

Figure 2.6. Schema of the dataset measurement process. Each subchart represents a day of measurement and each little block represents a sample. Blue color represents a correctly executed process and red color represents anomalous process

a) b)

Figure 2.7. Example of an insufficient assembly of a wheel, which is undetectable by the camera-based method. In the subfigure a), the wheels look assembled correctly, but the side view in subfigure b) shows that one wheel is assembled correctly and the other poorly. The fault is highlighted in the red rectangle.

2.3.3 The invalidity of the part of the dataset

During the measuring process, it was found that the data from two measuring days are unusable. These two days are the 07.03.2024 and 11.03.2024 (highlighted by the red rectangle in the figure 2.6). During these two days the measurement was conducted with a fault and the `ground truth` labels do not match the reality. During these days the measurements were conducted with tires that were already really worn, some of them almost in the state of the tire in figure 2.8 b). Those measurements were not

Measuring day	CNN labels	Human labels	Extended human labels	Artificial anomalies	Used in experiments
Original	✓	×	×	✓	✓
29.02.2024	✓	×	×	×	✓
07.03.2024	✓	×	×	×	×
11.03.2024	✓	Wrong	×	✓	×
14.03.2024	✓	✓	✓	✓	✓
18.04.2024	✓	✓	✓	×	✓

Table 2.1. Detailed information about each measuring day. The `Wrong` refers to the fault in measuring on this day, caused by a defected tires when measuring which was not properly labeled. `Artificial anomalies` refers to the artificially manufactured anomalies explained in the section 2.3.4.

properly labeled, but we kept the original labels, since at the time we thought that these are the last tires available to us. A few days later another set of tires was given to us, and we could do more proper measurements. Because of this, the measurements from these two days were discarded from the training dataset. They can still be found in the dataset provided in the appendix.

a) b)

Figure 2.8. Example of a normal and a defected tire. An undamaged tire should look like the one in the subfigure a). The rubber of the tire in the subfigure b) is torn on the inside of the tire which can be seen in the red rectangle.

2.3.4 Physical augmentation of the measurement

One of the greatest problems in anomaly detection is to have enough anomalous training data. That is why several measurement augmentation techniques were used. Among those was the usage of poorly manufactured rims or tires (rims of different sizes/shapes or with defects and tires from poor quality rubber). These techniques were applied because problems like these can be encountered in the real process. The deliberate generation of anomalous data to ensure a good representation of them within the training dataset is a common practice in the industry.

It is important to bear in mind, that artificially created anomalies can cause problems when the model is put to production. The most obvious problem a model can face is that the artificial anomalies do not accurately simulate the naturally occurring anomalies. Another instance of this can be overfitting the model for a specific type of anomaly. The examples of the augmentation can be seen in figures 2.8-2.10.

Figure 2.9. Tires of different stiffness used as anomalous data. The left tire is 3D printed and is very stiff, the tire in the middle is a different type of a tire and is a bit stiffer than the normal tire and the tire in the right is less stiff than the normal tire. The tire with normal stiffness can be seen in figure 2.8, subfigure a).

Figure 2.10. Examples of different defects on the rims used as anomalous data. The rims in the red rectangle are defected, the rim in the green rectangle is an example of a correctly manufactured rim.

Chapter 3
Statistical methods

In this chapter the statistical methods for solving the problem are discussed. We propose two architectures of methods, which serve as a basic logic block, upon which tailored methods for a specific use case can be built. Each of these architectures has its own solution for both supervised and unsupervised learning, is capable of continual learning, and can operate on the data stream and make predictions in real time of the process. For both architectures of methods, the principle, training and approach to all variants built upon the idea of given architecture is presented. The biggest advantage of our statistical methods in comparison to deep learning methods and other approaches is the ability to easily train the model in a specific way. This means that our methods are capable of being trained to the specific minimal value of the true positive rate (TPR), true negative rate (TNR) or to maximise accuracy (ACC). All of these metrics are explained in the section 5.1. In addition to these we introduce our custom metric called skewed accuracy (sACC), which serves a similar purpose as the f-score and based on the input parameter defines the ratio of importance of correct classification of positive and negative samples. The way of computing the skewed accuracy is in the equation (1).

$$sACC = \frac{\theta \cdot \frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{\theta + 1} \tag{1}$$

where $\theta \in [0, \infty]$ is the importance ratio between the positively and negatively classified samples. TP is the number of true positives, TN is the number of true negatives and FP is the number of false positives and FN is the number of false negatives.

Throughout our work, a term smart skewed accuracy is used. This refers to the skewed accuracy, where θ is set as:

$$\theta = \frac{N}{P} \tag{2}$$

where N is the number of negative samples in the training dataset, and P is the number of positive samples in the training dataset.

It is possible to train the deep learning based models so that they satisfy these criteria, for example through oversampling the dataset or using loss functions with different weights for different classes, but the training process is not as straightforward and the result is uncertain. In general, this chapter focuses on explaining the principle of the developed methods rather than the implementation, which is explained in the `Jupyter notebook` provided with the code in the appendix of this thesis.

3.1 Error aggregation from n-σ

The error aggregation from n-σ method (shortened to n-σ or deviation classifier) is based on the common statistical anomaly detection method of counting the deviation from mean. Specifically, the method is based on three sigma control limits described in

[17]. The principle is to count the number of timestamps, in which the studied signal deviates from the precomputed mean (μ) over the signals from by more than $n\sigma$, where n is preset multiplier and σ is the standard deviation of the given timestamp. This classifier assumes that the signals from anomalous processes have greater number of timestamps, which deviates from the mean than the signals from successfully executed processes.

3.1.1 Training the classifier

Supervised approach

An euclidean mean and standard deviation of time series dataset is computed for all signal dimensions in each time stamp from the non-anomalous samples. The mean signal matrix (shaped $t \times d$, where d is the dimensionality of the signal and t is the maximum length of a signal within the training dataset), and the variance signal matrix with the same shape are used as model parameters. The time series within the training dataset may be different lengths so the mean and variance for the specific timestamp is always computed only from the signals which contain it. The classifier should be trained on complete signals to avoid variability in the update rates across different timestamps that can occur with continual learning.

From the computed mean and variance a non-anomalous zone is defined. This non-anomalous zone is defined as the region within the n-th multiple of standard deviation from the mean signal. Visual representation of this can be seen as the blue zone in figures 3.1 and 3.2. The n is a parameter of the classifier which is determined before the training process. In this thesis we solely use $n = 3$, because it is commonly used in the statistical process control literature [17].

All the signals from training set (including the anomalous samples) are compared with the model parameters and the number of timestamps where the signals exceed the $\mu \pm n\sigma$ threshold. These values are saved for normal and fault processes.

As mentioned, before starting the training process it is possible to select a criterion for which the detector is optimised on the training set. In the next step this is where this functionality is used. The detector finds such value called `anomaly threshold` which satisfies the required criterion. For example if we trained the detector with required minimal TPR of 0.8, a value smaller than 80% of number of anomalous timestamps of signals from fault processes, which as a second criterion achieves the best accuracy is selected. This `anomaly threshold` is the third and final model parameter. Given the criterion is minimal TNR of 80%, the same procedure is followed in the opposite direction - a number greater than 80% of number of anomalous timestamps of signals from successfully executed processes with best possible training accuracy is selected. The maximum accuracy criterion finds a value for anomaly threshold for which the accuracy metric reaches its maximum, and for skewed accuracy, the argument of maximum of skewed accuracy function is found.

Unsupervised approach

If the labels are not available, the unsupervised approach is applied. The mean is computed from all samples. Then, a percentage estimate of samples with greatest euclidean distance from the computed mean is removed and labeled as anomalies. In the literature this is usually called fault ratio, in our methods we use the opposite term success ratio, which is described as the number of successful processes within the training dataset and can be computed as $1 - FR$, where FR is the fault ratio.

The success ratio can be estimated based on prior knowledge of the frequency of anomalies in the process, or it can be set to a predefined value. In the case of implementation used in this thesis the value is approximately 10%. A word approximately is used because the real removal is done in two steps of removing the five percent of the samples with the greatest distance from mean. From experience this performs better, than removal of samples in one step.

After the removal of these assumed anomalies, the mean and standard deviation are computed, and these values serve as the model parameters. The rest of the training is the same as the supervised approach.

3.1.2 Prediction mechanism

The prediction mechanism of this method is pretty straightforward. A number of timestamps the studied signal exceeds the range of $\mu \pm n\sigma$ is computed and if the number is greater than the anomaly threshold, the signal is predicted as anomalous. If the signal is smaller or equal to the anomaly threshold, the signal is predicted as non-anomalous. In figures 3.1 and 3.2 a visual representation of classification using 3-σ method can be seen. The light red background signifies a breach of standard deviation from the mean signal by more than three times in any dimension at that timestamp. The dark red background signifies such breach in current dimensions. The tested signal is displayed in black, while the bounds and the normal range of the signal is in blue.

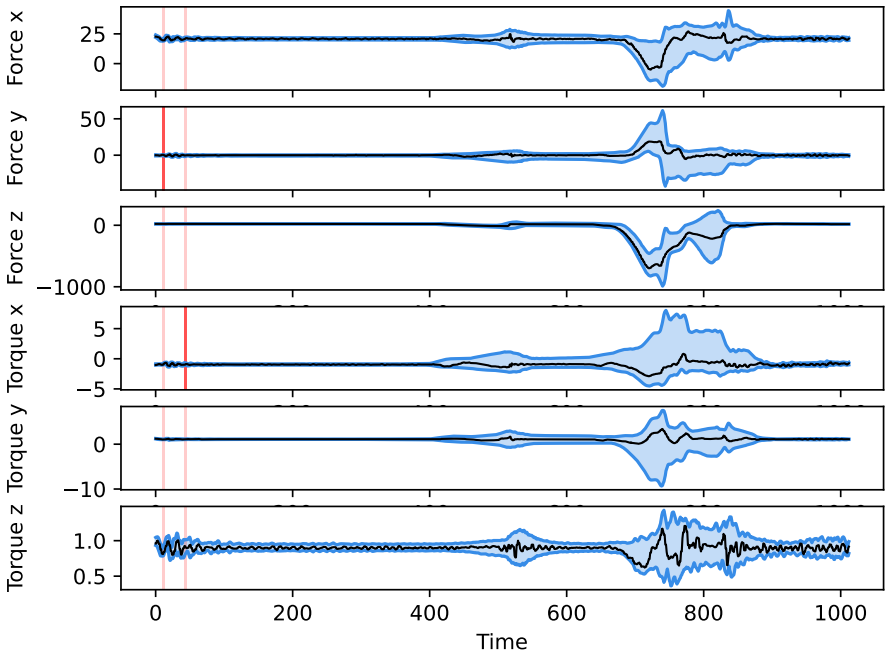

Figure 3.1. Successfully executed process signal (failure to pick up tire) classified by 3-σ method. The normal signal range is the blue area in each dimension, the signal being predicted is the black line, light red area indicates an anomaly in the timestamp in any dimension and dark red indicates an anomaly in the timestamp and the dimension.

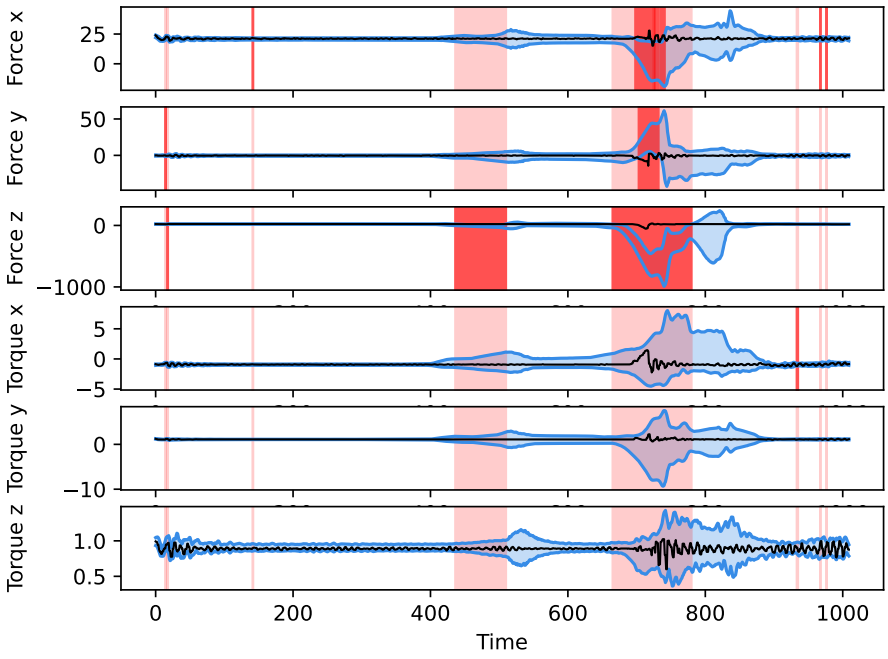

Figure 3.2. A signal coming from a fault process (failure to pick up tire) classified by 3-σ method. The normal signal range is the blue area in each dimension, the signal being predicted is the black line, light red area indicates an anomaly in the timestamp in any dimension and dark red indicates an anomaly in the timestamp and the dimension.

3.1.3 Continual learning

Continual learning is utilised by online update of the mean and variance prototype signals, which saves allocated memory and time. Instead of keeping all the samples in memory and computing mean and variance signals each time new sample comes to classifier, which would be very demanding on the memory and that is not desirable especially when implementing the detector for the edge device, the new mean signal is computed via incremental mean update:

$$\mu_i = \mu_{i-1} + \frac{x_i - \mu_{i-1}}{s} \tag{3},$$

where μ_i is the new mean signal, μ_{i-1} is the previous mean signal, x_n is the new sample signal to be predicted and s is the number of samples. The standard deviation (variance) signal is computed via incremental variance update:

$$\sigma_i = \sqrt{\frac{s\sigma_{i-1}^2 + (x_i - \mu_{i-1})(x_i - \mu_i)}{s}} \tag{4},$$

where σ_i is the new standard deviation signal, σ_{i-1} is the previous standard deviation signal, μ_i is the new mean signal, μ_{i-1} is the previous mean signal, x_i is the new sample signal being predicted and s is the number of samples. The indexing in both of these methods refers to old and new signals, not elements within the signal.

These two incremental updates are always computed together during the same process, so the dataset mean values used in the equation for incremental variance update are directly pasted after computing them via incremental mean update. The equations (3) and (4) were adapted from [18].

3.1.4 Application on data stream

We introduced two ways of making the method work on partial signals. For the first, naive way no additional training is required (the offline training for evaluating the whole signals is sufficient), and the sample is simply evaluated to the part of the prototype (mean) of the same length, the number of anomalies is counted, and this number is multiplied by the ratio of length of the whole mean signal to the length of the evaluated prototype. Since there are no labels for when the anomalies happen in signals, the positive and negative predictions are measured differently. The way of evaluation is explained in section 5.2.1. Because of this, it may be beneficial to train the online detector with a different requirement than for the one operating solely on the whole signals, which is why it was implemented. To satisfy the online criterion a model parameter which multiplies the number of anomalies for partial signals can be trained. The principle of tuning this parameter to a desired TPR, TNR, accuracy or skewed accuracy for online approach (which can operate on data streams) is the same as with tuning these criteria in the offline approach (which operates solely on the whole signals).

3.2 Distance based feature classifier

The distance-based feature classifier (shortened to feature classifier, feature method) leverages time-series clustering methods, which are often computationally and memory intensive, to encode high-dimensional signals into low-dimensional feature vectors, which enables fast prediction. The feature vectors are easy to operate with, and a conventional classifier can be used for the anomaly detection. The implemented solution can work both as supervised or unsupervised and may be fully independent, but can also utilise Human in the loop (HITL). In essence, the nature of the features is not important. In literature some computed signal parameters are often used for similar methods, But for the sake of this thesis the features are distances to precomputed `prototypes`. A time series prototype is a term often used in time series clustering applications describing an `average` signal computed from a time series dataset. Note that `average` in this context does not describe mean signal (although mean signal can also be a prototype), but rather a representative sample for a set of time series. In this thesis the DBA barycenter [19] and euclidean barycenter (mean signal) are used as signal prototypes. The computation of the DBA is very computationally expensive, however the prediction is very fast.

We assume, that the distances to the specific barycenter are normally distributed among the successfully executed signals. The parameters of the normal distributions of the features are estimated, and signals are classified based on the Mahalanobis distance to this distribution. For solving the second part of this method (feature vector classification) a Gaussian classifier was used in the implementation, but other types of classifiers also work. For example support vector machine, k-nearest neighbors, artificial neural network and others.

3.2.1 Feature extraction

Dynamic time warping

Dynamic time warping (DTW) is an algorithm that has been introduced by [20–21]. The implementation used in this thesis is described in [19].
DTW is used to get an accurate measure of similarity between two time series. Unlike norms, which cannot measure the distance between time series of different lengths without tricks such as padding, DTW is capable of it.

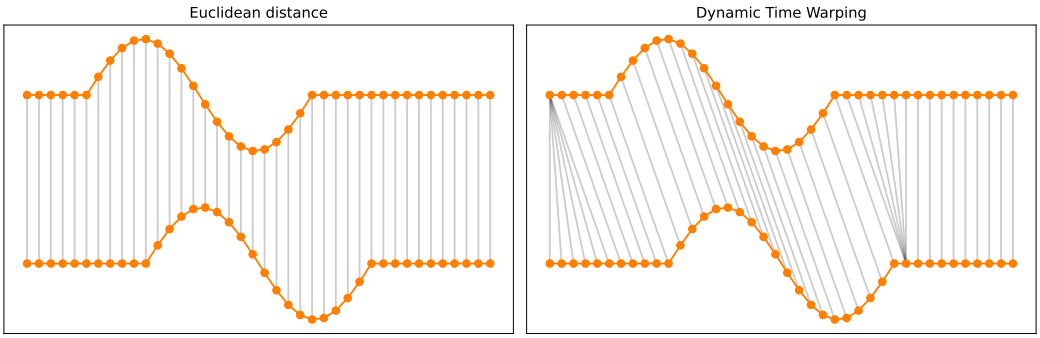

Figure 3.3. Comparison between aligning the indices for measuring the distance by euclidean distance and DTW. To generate this figure a code from [22] was used.

The principle of the algorithms is described in the following paragraph. Given two time series $A = \{a_1, a_2, ..., a_m\}$, $B = \{b_1, b_2, ..., b_n\}$, each point from time series A is paired with a point from time series B and vice versa. The pairing must respect the following properties:

- The first element from the time series A is paired with the first element from the time series B
- The last element from the time series A is paired with the last element from the time series B
- The indices of paired points (i_k, j_k) from both time series are not decreasing:

$$1 \leq i_k \leq m, \; 1 \leq j_k \leq n$$
$$i_{k-1} \leq i_k \leq i_{k+1}$$
$$j_{k-1} \leq j_k \leq j_{k+1}$$

The cost of the optimal alignment can be recursively computed by

$$D_{(A_i,B_j)} = d(a_i, b_j) + \min\{D(A_{i-1}, B_{j-1}), D(A_i, B_{j-1}), D(A_{i-1}, B_j)\}$$

with exponential complexity. Thanks to dynamic programming we can turn the problem to $O(n \cdot m)$ [19]. The result of such pairing can be seen in figure 3.3.

The time series used in this thesis are not univariate. Because the time is the same for all dimensions of the time series, dependent DTW (DDTW) [23] needs to be used, which minimises the distance for all dimensions in the same time sample.

Dynamic time warping barycenter averaging

Dynamic time warping barycenter averaging (DBA) is an algorithm that was published in [19]. It solves the problem of creating a representative prototype from a time series dataset. This can be useful in applications like clustering based time series tasks or time series classification tasks. In this thesis we use it to create a prototype for one component of a feature vector.

3.2.2 Training the classifier

Supervised approach

In the supervised approach labeled dataset is required to train the model. In general the training algorithm is described as follows:

- The normal data (labeled as non anomalies) are selected from the dataset
- Prototypes are computed from the non-anomalous samples
- Distances to the prototypes for each signal from the training dataset are computed and saved as feature vectors. These can for our case be seen in figure 3.4.
- A mean value of feature vector of the non-anomalous samples is found
- Mean and variance is computed for each metric (feature vector component) among the non-anomalous samples
- A multivariate Gaussian distribution ($G(\nu, \Sigma)$), where ν is the mean feature vector and Σ is the covariance matrix between the components of the feature vectors is constructed.
- An ideal confidence interval for the classifier is computed based on the given criterion. The criteria (TPR, TNR, accuracy, skewed accuracy) are described in the beginning of the chapter 3.

The ideal confidence interval of the distribution which approximates the feature vectors of successfully executed processes is tuned in a way that it optimises the criterion. As of this day, there are four criteria for which the classifier can be optimised. These are true positive rate, true negative rate, accuracy and skewed accuracy. To set the confidence interval a quantile which best satisfies the condition of χ^2 distribution with the degrees of freedom equal to the dimensionality of the feature vector is examined for evaluation. The Mahalanobis distance to the multivariate Gaussian distribution model parameter is computed for all the samples in the training set and is compared to the χ^2 quantile value. As with the n-σ method, if we want to train the classifier with minimum TPR rate of 0.8, a χ^2 quantile which satisfies this condition is found. If there is no such quantile (for the example where the signals of the dataset cannot be assumed as normally distributed), the closest possible value to our requirement is found and set as the `confidence interval` model parameter. The Gaussian classifier used by this method is inspired by [24].

Unsupervised approach

In the unsupervised approach no dataset labels are needed. At first The prototypes are computed from all the samples from the training dataset. For this technique to work well, it is necessary for the anomalies to be sparsely represented in the training dataset. After the computation of the prototypes, the features for all the training signals are computed. A set percentage of the furthest laying signals from the prototype (norm of the feature vector) is deemed anomalous. This can be done multiple times, depending on the requirements for false positivity. After this the supervised approach is applied to the newly labeled dataset.

3.2.3 Prediction mechanism

The prediction function intakes the signal and computes the selected distances to the prototypes from which the feature vector is built. The feature vector is then evaluated by the trained Gaussian classifier. - A Mahalanobis distance to the multivariate normal distribution model parameter is computed. The Mahalanobis distance is then compared to the trained `confidence interval` value. If the distance is greater than this confidence interval, the signal is predicted as anomalous, in the other case it is deemed non-anomalous. The equation for computing the Mahalanobis distance and doing the prediction is in the equation (5).

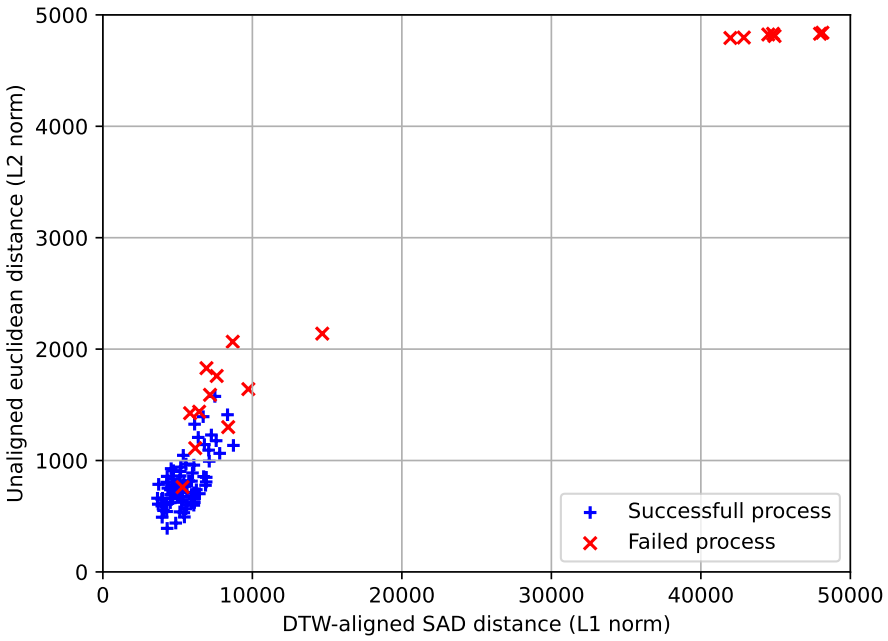

Figure 3.4. Feature space in which the classification is performed. The samples in the picture come from the original training set from CIIRC Testbed lab.

$$(f-\nu)\Sigma^{-1}(f-\nu)^T \tag{5}$$

where f is the row feature vector of the studied signal, ν is the mean row feature vector of the multivariate normal distribution model parameter, Σ^{-1} is the inverse of the covariance matrix of the multivariate normal distribution model parameter and $(.)^T$ is the operation of transposition.

3.2.4 Application on data streams

Training

A naive solution based on our offline approach for training the classifier to work on data streams would be to compute the multivariate normal distribution model parameters for signals of all lengths up to the whole signal. To prevent this time and memory highly demanding computation a mean `error signal` for all features is computed. The process of training the detector to work on the data streams is explained in the following article.

Given a set of training data of whole signals, the prototypes are computed as explained in 3.2.2. For each signal from a successfully executed process (or from a process deemed successful by detectors trained with an unsupervised approach), an error is calculated at each timestamp for each feature. These values are combined across all timestamps to form a vector, which we call the `error signal`. An error signal is a distance to the "shortened" prototype for example if a time series of l timestamps is given, it is compared to the part of prototype of the length l. Even though this operation is also time demanding, it is not even near the complexity of computing the mean and covariance matrix for each timestamp. The result of this operation can be seen in figure 3.5. These error signals are then scaled between the values 0 and 1 (so that the error can be expressed as a percentage of the error of the whole signal) and stored. Because of the time complexity of this solution, we use randomly selected n

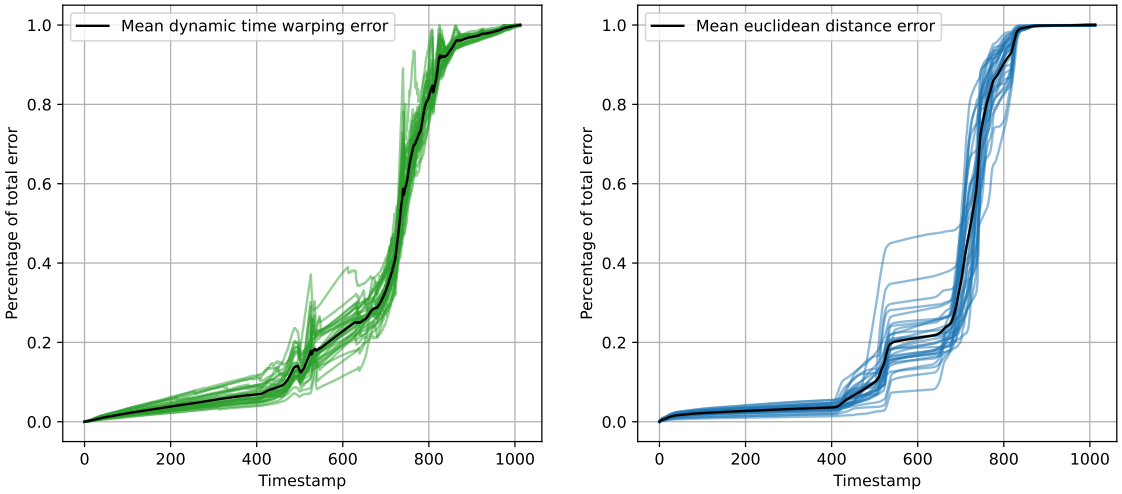

Figure 3.5. Feature error signals training data and computed mean

non-anomalous samples from the training dataset. This is beneficial when dealing with datasets containing hundreds or greater number of signals.

Prediction mechanism

When evaluating the data stream time series, distances to parts of prototypes of the same length are computed and put into a feature vector. These distances are scaled by the inverse value of the error percentage and the result is evaluated by the feature classifier of the whole signals. This outputs real time evaluation of the signal. A definitive evaluation is not implemented, but it can be built easily based on this solution. For instance when evaluating real signal in production on a data stream, the whole process can be deemed anonymous, when the signal is evaluated as anonymous for n consecutive times and then the process may be interrupted.

3.2.5 Continual learning

In this model the continual learning updates either the prototypes, the classifier parameters or both, although from experience, it is better not to update the prototypes. Because in the case of DTW barycenter, if the signals are not computed from the whole training set, but rather only as a weighted barycenter the method tends to quantize the outcome resulting in different shape of the signal and thus greater DTW distance. This is not the problem with mean euclidean signal because weighted average in each timestamp can be adjusted constantly.

In the case of updating the classifier parameters, this can be more beneficial. When the model is adaptable (unfrozen), the feature vectors of signals to be predicted are saved in memory. This is acceptable even on edge devices, as one feature vectors consists in our case of two floats in comparison to saving one signal, which is in our case approximately 6000 floats. The outcome (label) of the predictor is also saved and model can update its parameters according to the number of predictions.

This can be extremely effective when using human-supervised continual learning approach described in the following subsection. If the feature vectors are saved in the memory, the classifier can also be retrained to satisfy a different criterion.

3.2.6 Expert classification - human in the loop

Although this method is designed to be able to operate independently, human help can be beneficial especially in the training stage. The training can be extended by doing human-supervised continual learning. This is done by executing the process with adaptable (unfrozen) model, evaluating the physical outcome and reporting the labeled results back to the model. The parameters of the model are updated not based on the prediction of the model, as with continual learning but rather based on the label given by the human. Human intervention can also be beneficial when the sample is near the anomaly threshold and the risk of inaccurate prediction could be higher.
Another way of using expert knowledge for this model is if the data can be divided into separable clusters. Even though this phenomenon does not exist within the dataset used in this thesis, it could be beneficial for applications like quality measurement score which is often used in the industry. If there are multiple classes of separation, prototypes and distribution of features is computed for each cluster. The clustering can be trained from labeled dataset, or by clustering algorithms. In this thesis k-means clustering algorithm was used for semi-supervised division into k clusters [1].

k-Means clustering

k-Means clustering is an algorithm that has been proposed by [25]. It is one of the most if not the most used clustering algorithms. The algorithm itself has several ways in which it can be performed. The form described below comes from the original paper [25]. The algorithm starts with randomly selecting k points. Then iteratively a new point is added to each group and the mean of each group is adjusted. This process is repeated until convergence.
When not dealing with large datasets, the approach of selecting the means first and iteratively comparing all points to each mean can be more usable. This approach is also used in algorithm in this thesis. The process iteratively continues by labeling the points to each new mean and is stopped, when convergence is reached. The most used metric for k-Means clustering algorithm is norm based distance (usually L2 norm), but in theory any metric can be used. As an example, dynamic time warping is often used [19, 11]. When dealing with time series, other variations of the algorithm such as k-medioids clustering (the mean is not really a mean, but rather prototype, which minimises selected distance to samples from the same cluster [26]).

[1] The word semi-supervised is used because the k-means clustering is an unsupervised algorithm, but our models are not capable of figuring out the ideal number k of clusters. This has to be inputted as a parameter beforehand.

Chapter 4
Deep learning approach

Another approach to classification can be utilising artificial neural networks (ANN). When working with time series a number of architectures can be used. Among those one can name recurrent neural networks (RNN), long-short term memory neural networks (LSTM), transformer architecture or convolutional neural networks (CNN). The greatest disadvantage of using neural network for any classification task is that for satisfactory results a great amount of data is needed (when using conventional approach this data also needs to be labeled). When training a neural network for anomaly detection task this problem becomes even worse, because of the (usual) lack of anomalies in the training data. The upside of using neural network is, that with the right training it can be learned to solve very complex tasks. Within the dataset used in this thesis, there are several samples, that even though are labeled as anomalies by ground truth, are not detectable by proposed statistical methods, because of the fact, that the distance of the signal is not far from the prototype. Nevertheless, the signal is anomalous for a reason and its invalidity should be detectable. In this chapter the goal is to design an ANN, which is able to accurately detect these subtleties.

4.1 Anomaly detection using ANN with LSTM core

The deep learning based method we selected for doing such task is a LSTM neural network connected to multilayer perceptron. The LSTM core learns the normal course of a signal and the perceptron decides based on the outcome if the signal is anomalous or not. This classifier-like architecture was selected, because the plan was to develop a method, which could not only classify the anomaly, but also its type. Unfortunately this functionality was not achieved maybe because of the lack of training data in this thesis, however in the future we may try to solve this problem [1].

4.1.1 Multilayer perceptron

Multilayer perceptron (MLP) is a type of ANN which is based on the unit called a "perceptron". In [27], a perceptron is described as a unit which outputs a linear combination of the inputs and returns `1` or `-1` if the result is greater than certain value. The perceptron however can use any non-linear activation function on its output [28].

[1] The plans are discussed more in depth in section 6.1.

4.1.2 Long-short term memory

LSTM is an type of neural network, built upon the ideas of RNN, which attempts to solve the *vanishing-exploding gradient problem* (VEGP). This is a problem often encountered when training RNN, because the weight of the feedback part of the RNN multiplies the output of the previous elements of the input and thus the value is either very big (*exploding gradient problem*) or nearly zero (*vanishing gradient problem*). Because of this fact simple RNNs are very hard to train (the longer the input sequence the harder), since when using backpropagation when training the neural network, the gradient is either too big or too small, hence the training step is either too short or too long thus it is hard for the optimization method to find the local minimum [29].

These networks tend to outperform RNNs, but the solution of VEGP is still not perfect [30]. However, in most applications it allows the network to "remember" context accurately (particular when dealing with shorter input sequences) and reduce the risk of catastrophic forgetting. Introduced in 1997 by [31] it is still used in many deep learning applications, particularly in time series analysis, even though in recent years it is being substituted by the transformer architecture. The principle of the LSTM memory block can be seen in figure 4.1.

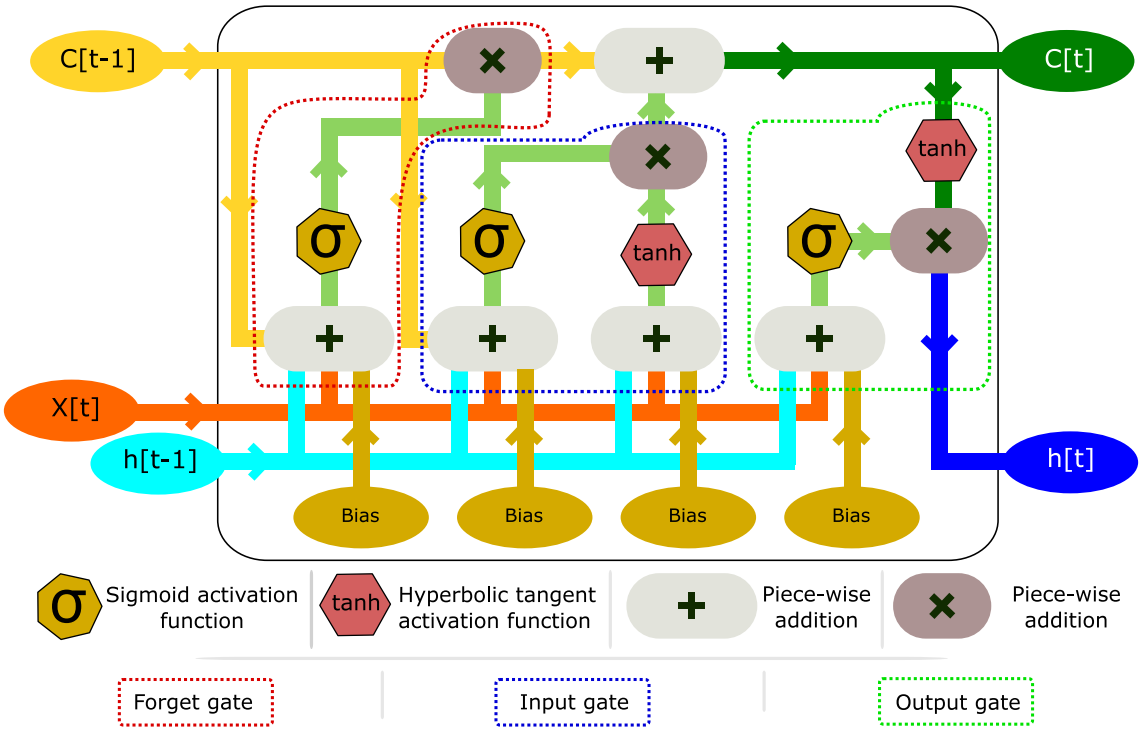

Figure 4.1. Schema of the LSTM core block

Sigmoid activation function

$$\sigma(y) = \frac{1}{1 + e^{-y}} \quad [27] \tag{1}$$

The domain of the sigmoid function is $[-\infty, \infty]$
The range of the sigmoid function is $[0, 1]$

Hyperbolic tangent activation function

$$\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad [32] \tag{2}$$

The domain of the hyperbolic tangent function is $[-\infty, \infty]$
The range of the hyperbolic tangent function is $[-1, 1]$

The "C" stream is called the cell state and represents the long term memory of the network. The "h" stream is called the hidden state and represents the current outcome and the short term memory of the network. The "X" stream is the incoming signal. The LSTM network can be decomposed on three subsections called gates. The first gate named the `forget gate`, which through sigmoid nonlinearity connects the sum of the short term memory, decides the amount of long term memory that is prevailed. The sigmoid activation function's range is a number between zero and one and its outcome is multiplied with the value of the cell state, the input and forget gate bias directly with the long term memory, which allows to reduce the influence of the long term memory (cell state) - hence named the `forget gate`. The second gate is named the input gate, and it combines the input transformed by sigmoid and hyperbolic tangent nonlinearity. This is then added to the long term memory of the network. The last part of the LSTM NN is the `output gate`. The output gate combines the output of the short term memory transformed by the sigmoid function and long term memory transformed by the hyperbolic tangent activation function.

4.1.3 Architecture

When designing the network, the idea was not only to detect anomalies, but also classify the anomalies by type. That is why the architecture of this network is similar to one used in classification. As mentioned above, the network consists of the two parts. The first part is the LSTM NN, which has three hyperparameters which can be modified. The dimensionality of the inputs (in the case of used dataset 6), dimensionality of the vector of the hidden state and number of LSTM layers. The output of the network in this case is the hidden state. After all of the signal goes through the LSTM part of the network, the last hidden state of the network serves as an input to the second part of the network - the MLP. The hyperparameters of the perceptron (activation function, number of hidden layers, ...) can be tuned as well. The dimensionality of the output of the MLP part can be also modified. Between each hidden layer of the MLP there is a sigmoid nonlinearity. The schema can be seen in the figure 4.2.

If the outcome is put through a softmax function, the result is the deemed probability of prediction for the model. The argument of maximum of the output vector is the resulting evaluation of the signal.

4.1.4 Training process

The implementation of this neural network was done using python's `pytorch` and `pytorch_lightning` library. This method does not use the time series in the whole length, only a partial signal of the assembly process. The input time series which is therefore classified is a matrix of shape 150×6, where the first dimension is the value in time and second dimension is the data coming from the other sensors. After the

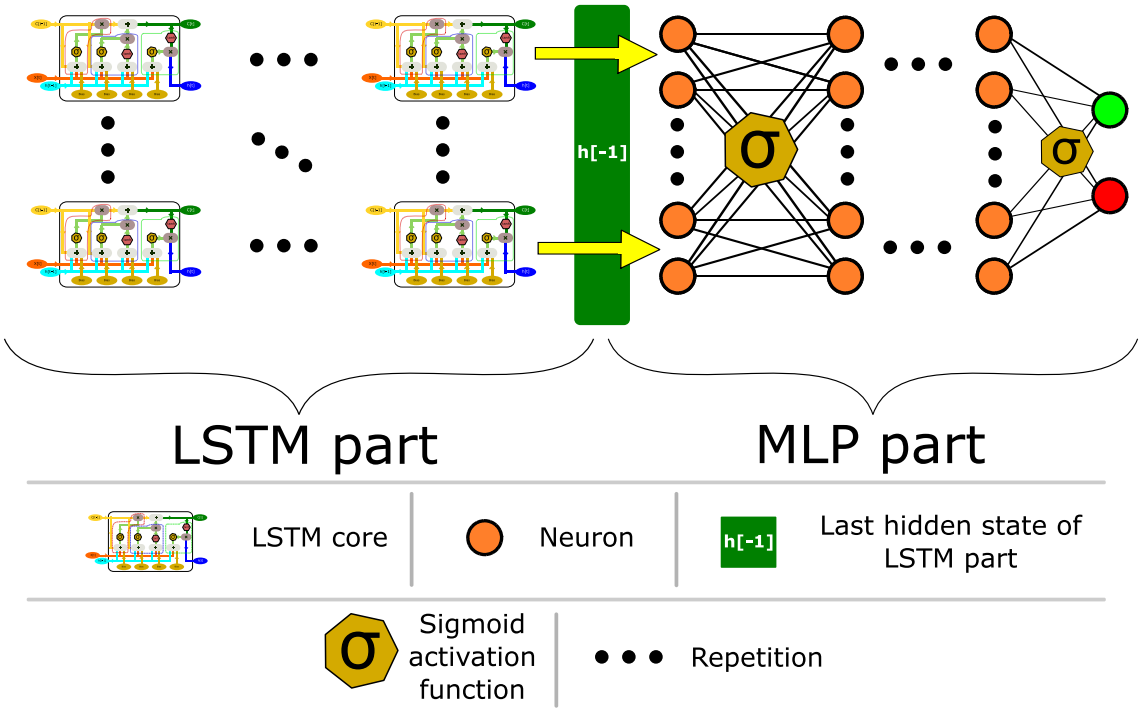

Figure 4.2. The principle of the architecture of the LSTM-MLP neural network

preparation of the data, the neural network is trained by the standard backpropagation algorithm. The training process uses the ADAM optimizer ([33]).

There are two possible loss functions which are used in our model. The first one is the cross entropy loss (CE) and the second one is the weighted cross entropy loss (referred later in this thesis as WL). The usage of the weighted cross entropy loss has its advantages and disadvantages. The main problem that this technique is supposed to solve is the unbalanced number of different classes within the dataset. The weighted cross entropy loss lets the model to train itself in a way that the loss is multiplied by different coefficient for different classes. This approach is particularly useful in our case, since anomaly detection is specific with not having many anomalous samples to train on. Among the drawbacks of this method is that the successful training of the model may be more difficult, and require even larger dataset. The weights of the loss are computed from the training set through the formula (3).

$$\mathbf{W} = \frac{\mathbf{w}}{\sum_{i=1}^{n} \mathbf{w}_i \cdot \min(\frac{\mathbf{w}}{\sum_{i=1}^{n} \mathbf{w}_i})} \tag{3}$$

Where $\mathbf{w}$ is the vector of occurrence of the classes in the dataset, n is the dimensionality of $\mathbf{w}$ - number of classes and $\mathbf{W}$ is the resulting vector of the weights.

For training an implementation of backpropagation algorithm in `pytorch` is used. During the training of the model, the best achieved state of the weights (with minimal loss) is always saved.

4.1.5 Prediction mechanism

This method's prediction process is really straightforward. First, it is ensured that the weights of the NN are freezed. The signal in the form of *numpy array* is then converted

to *pytorch tensor* class, The same preparation step as with the signals of the training set is done (extracting the signal of length 150) and the result is fed to the forward pass of the neural network. As mentioned, the outcome can be either binary (True/False) when using the argument of maximum on the outcome of the NN, or in probabilistic confidence when using the softmax function. The use case of this feature could be to monitor the values of confidence, and if the model outputs a value of confidence in the prediction lower than certain threshold another method of evaluation (e.g. another system, human operator) is used for confirming the prediction.

Chapter 5
Experiments

For consistent and reliable evaluation of presented methods a standardized system of testing was developed. The quality of anomaly detectors is often evaluated by the same metrics used as when evaluating classifier performance. However, the metrics hold different levels of importance. The anomaly detectors proposed in this thesis are evaluated in a way common in literature. In the first section of this chapter, we discuss the evaluation of offline methods, which detect anomalies across the entire signals. The second section then examines data stream-driven methods that are capable of operating on partial signals. The supervised and unsupervised approaches are compared to one another. For getting the most accurate performance metrics of the classification, k-fold crossvalidation (CV) is used. The folds in this case are used in two ways.

Shuffled crossvalidation

In the first case the dataset is randomly split into training and testing set k times in a way that each signal is used exactly once in the testing set. This technique is the conventional way of doing the k-fold CV. The testing window set size should always be the same and since our evaluation dataset consists of 392 samples, the size of the window was chosen to be 56. This number is ideal, because it is a divisor of the number of samples, so the windows can be the same size for all the folds. Furthermore, in literature the ratio of test samples is usually 10 or 20% of the whole dataset so approximately 14.2% is an appropriate value. In literature this is called random k-fold CV, we will call this shuffled CV.

Day-based block crossvalidation

The second way of CV used is day-based block CV where the dataset is divided by day in which the measurement took place. In every fold data from one day is used as testing set, and the detector is trained on the rest of the dataset. Although this may result in a lower performance estimate of the classifiers in comparison (depending on the nature of the dataset) to the classic random approach, it may be beneficial to measure its performance in this way because it is less optimistically biased and better simulates the real deployment of the model. When the model is put into production for an industrial application, the measurement of the data for training the model is often expensive since this usually means, that the robot on which the model runs cannot perform its task in a usual way, so usually the model is tuned on data from as few days as possible.

The unsupervised methods were tested on the same dataset as the supervised and the known labels were used just for evaluation the performance of the methods.

Inverse crossvalidation

To examine the influence of the training dataset size, an additional non-conventional way of testing the models was introduced which we called `inverse crossvalidation`. To simulate the smaller training dataset, the training and testing sets were interchanged. The classical k-fold CV is done to estimate the performance of the model trained on the

whole dataset. The `inverse crossvalidation` probably cannot be used in this way, because the testing datasets contain the same data in multiple folds. Nevertheless, the training is still done in a way that the training and testing sets are strictly disjunct.

In summary cross-validation was conducted in four distinct methods for the receiver operating characteristic (ROC) analysis (section 5.1.1) and in two methods for other measurment in section 5.1.2. These are day-based CV, day-based "inverse" CV, 7-fold CV and "inverse" 7-fold CV for ROC experiments. For the accuracy part the "inverse CV" was not used because the training of the neural networks would not make sense with such a small dataset.

5.1 Performance metrics

When evaluating classifier the most basic and the most important metrics are the true positive ratio (TPR) and false positive ratio (FPR). These are derived from the number of correctly and incorrectly detected positive samples, correctly and incorrectly classified negative samples. To better understand these terms a `confusion matrix` is used. It can be seen in the figure 5.1. The rows of the matrix corresponds to the output of the classifier/detector and the columns corresponds to the ground truth of the measurement. The equations (1)-(6) are the formulas to compute false alarm ratio (false positive ratio), sensitivity (recall, true positive ratio), specificity, accuracy and f-measure respectively.

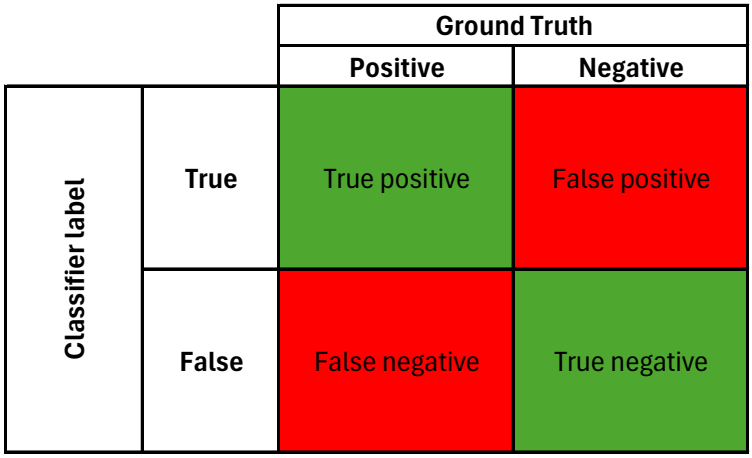

Figure 5.1. Confusion matrix

$$\text{FPR} = \frac{FP}{FP + TN} \tag{1}$$

$$\text{TNR} = \frac{TN}{FP + TN} \tag{2}$$

$$\text{Sensitivity} = \text{Recall} = \text{TPR} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{4}$$

$$\text{Accuracy} = \frac{TP + TN}{P + N} \tag{5}$$

$$f_\beta = (1+\beta^2) \cdot \frac{TP}{(1+\beta^2) \cdot TP + \beta^2 \cdot FN + FP} \tag{6}$$

Where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives. These can be seen in the confusion matrix. The P is the number of positives in general ($TP + FN$) and N is the number of negatives in general ($TN + FP$). The β in the f-measure is the parameter of importance between precision and recall. when using the f_β measure, the recall is β times more important than precision. The equations were adopted from [34–35]

5.1.1 Receiver operating characteristic and AUC

Receiver operating characteristic curve

The receiver operating characteristic curve is one of the most widespread tests for evaluating classifiers. [34]. It puts into a context a True positive rate and false alarm rate. The ROC curve is a 2D graphical representation of these two measures. On the x axis of the chart lies the false positive rate, whereas on the y axis lies the true positive rate. The ROC curve is constructed by tuning the parameters of the model and measuring these two rates. These directly measured points from the dataset are known as `accuracy points`. These are then usually linearly interpolated for nice visual representation, but the true value in between the two accuracy points is uncertain, however it is known that it lies within the rectangular area which sides are parallel to the axes and is defined by two accuracy points in its diagonally opposing corners. This can be seen in the figure 5.2. The described linear interpolation corresponds to the "expected" line in this figure.

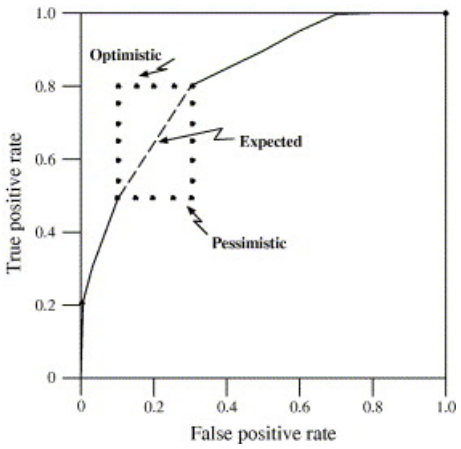

Figure 5.2. The unknown values of the ROC curve and the bounding rectangle. The most optimistic and most pessimistic outcome of real values of the ROC curve are the edges of this rectangle. The line marked as `expected` is the linear interpolation between the two accuracy points which was used in generating the ROC curves in this thesis. This image was adopted from [34].

Area under curve

The often used measure related to the ROC curve is the area under curve (AUC). It stands for the area under the ROC curve. The measurement of this can be done in several ways, and in our case trapezoidal numerical integration of the curve was used.

Usage and problems

All the ROC curves and AUC values in the chapter 5 are measured on the testing dataset which is a disjunct set to the training (the model has never seen the testing signals during training). Its distribution might be different from the distribution of the training dataset, upon which the models are learned. That is why the model may not be able to achieve all outcomes of the ROC curve by self-tuning. The ROC curve is just a representation of all the possible decision based on the tunable threshold, but the achievability by self-tuning is not guaranteed. This is especially apparent with unsupervised models, for which the estimation of the percentage of anomalies is crucial for working properly. The ROC curves were generated in two ways. In the first way directly by setting the specific parameter (aggregated anomaly threshold for n-σ method and confidence interval for distance based feature method) to a spectrum of values and measuring the outcome on the testing set. The other way was to make the detector tune to multiple exact values of TPR. In each fold the classifier was trained with a criterion of TPR of the multiplies of ten in percentage between zero and one. In the ROC curves generated by testing the unsupervised models by self-tuning the TPR parameter, the curve was unfortunately not an increasing function. This was especially apparent when trained during the day-based block CV. Some interesting cases of the curves generated by autotuning the parameter were therefore plotted in the form of a scatter chart. Each color in those scattered ROC curves plots represents one fold of the CV. The figures 5.5 - 5.4 show the ROC curves of offline 3-σ method. The unsupervised ROC curves were generated with a success ratio (estimation of the non-anomalous ratio) of 0.9.

5.1.2 Accuracy performance evaluation

In the figures 5.11 and 5.12 there is a result of the accuracy, true positive rate and true negative rate testing of the offline methods by the shuffled 7-fold crossvalidation and the day-based block crossvalidation. The unsupervised methods were tested on two different success ratios (estimated number of anomalies within the training set) - these being 64% and 90% and the optimization criterion for all was maximum accuracy (ACC). In anomaly detection tasks, the rate of catching the anomalies is often more important than the pure accuracy, therefore the models were also tested when trained with optimisation on the different reasonable values of skewed accuracy for statistical methods or weighted loss for the deep learning based methods. These were (ACC - general accuracy, sACC - smart skewed accuracy, 2sACC - skewed accuracy with skew parameter of 2 (the classification of the positive samples is twice as important as the negative samples), WL - weighted loss, 2WL - weighted loss where the classification of the positive samples is twice as important as the negative samples). The supervised methods were optimised for criterions in the parentheses in the figures 5.11, 5.12 for offline methods and in the figures 5.16 and 5.17 for methods operating on data streams.

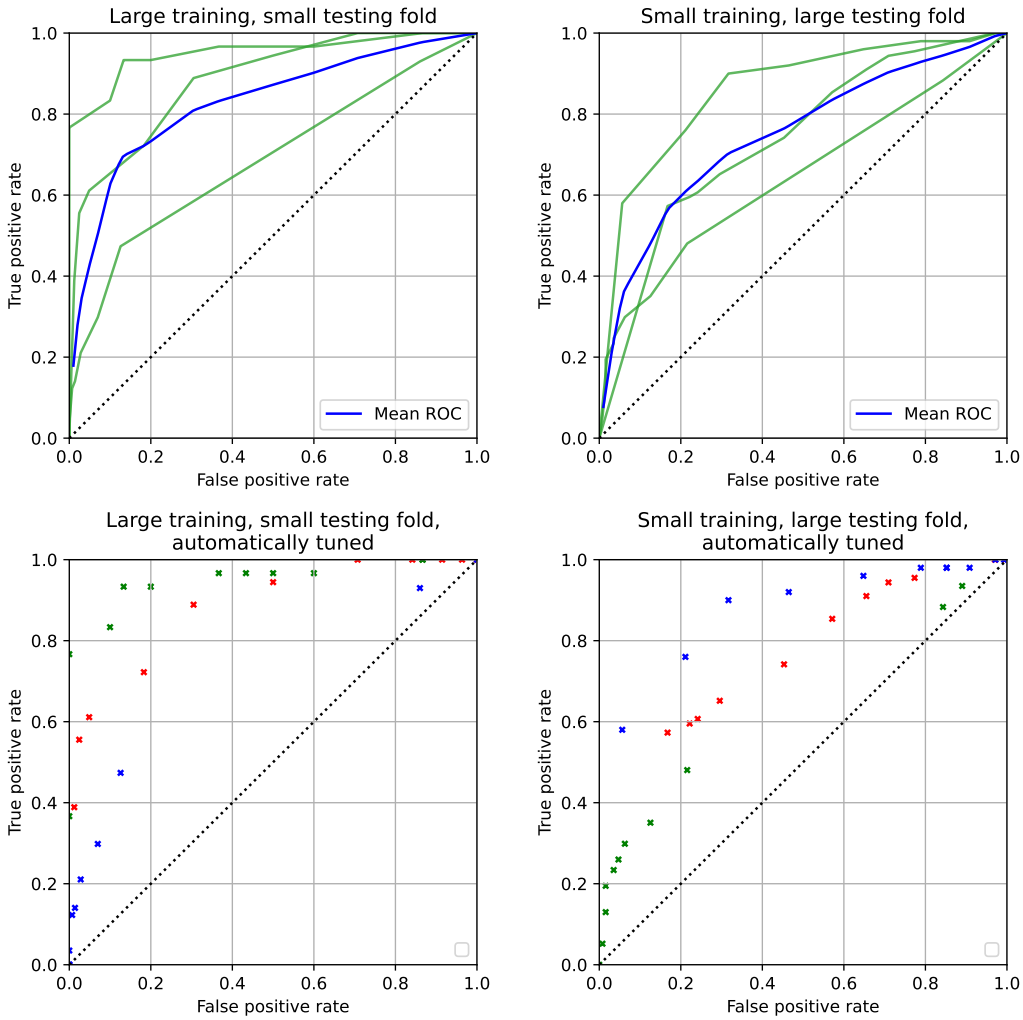

Figure 5.3. Supervised 3-σ detector ROC curves generated by day-based block CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold, the scattered charts in the bottom are the result of autotuning the model to a certain TPR value. The charts in the left figures are result of classical day-based block crossvalidation, the right figures are result of "inverse" day-based block CV.

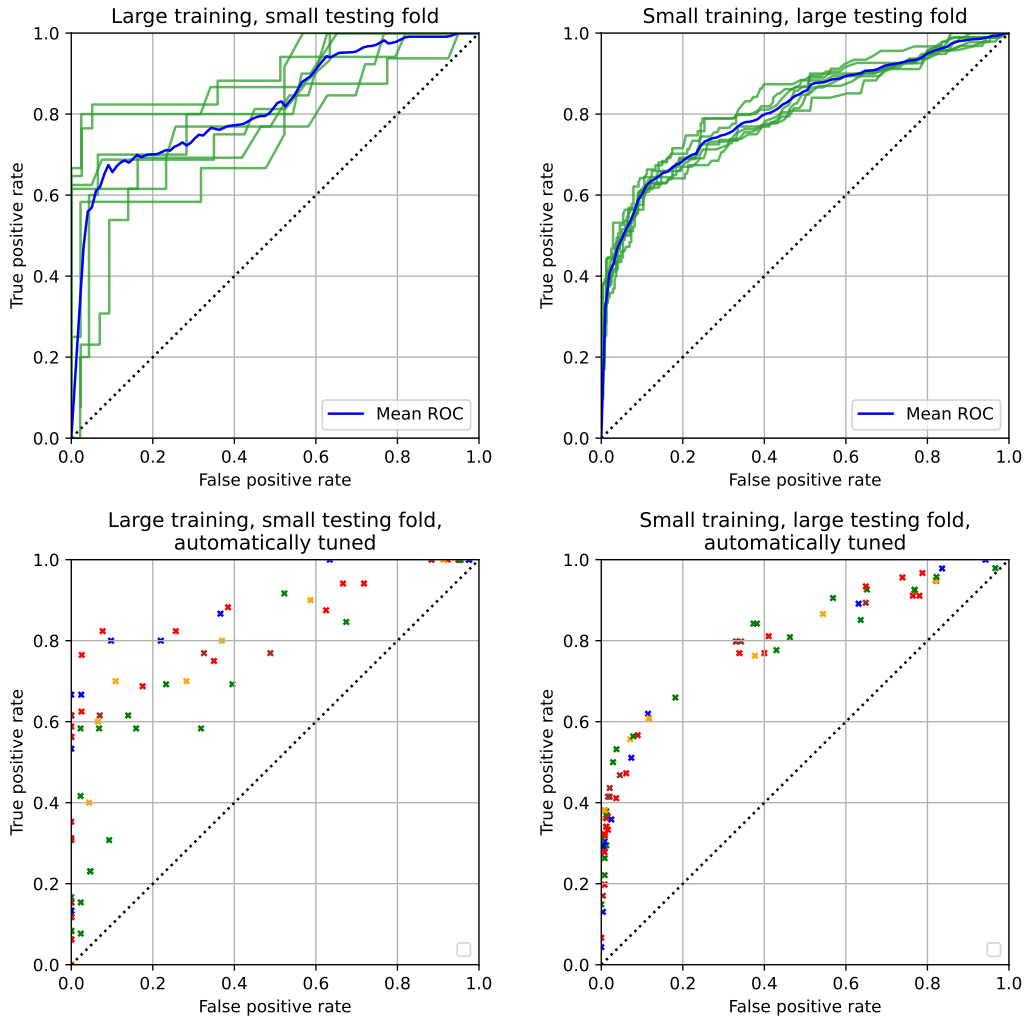

Figure 5.4. Supervised 3-σ detector ROC curves resulting from shuffled CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold, the scattered charts in the bottom are the result of autotuning the model to a certain TPR value. The charts in the left figures are result of shuffled 7-fold CV, the right subfigures are result of 7-fold inverse CV.

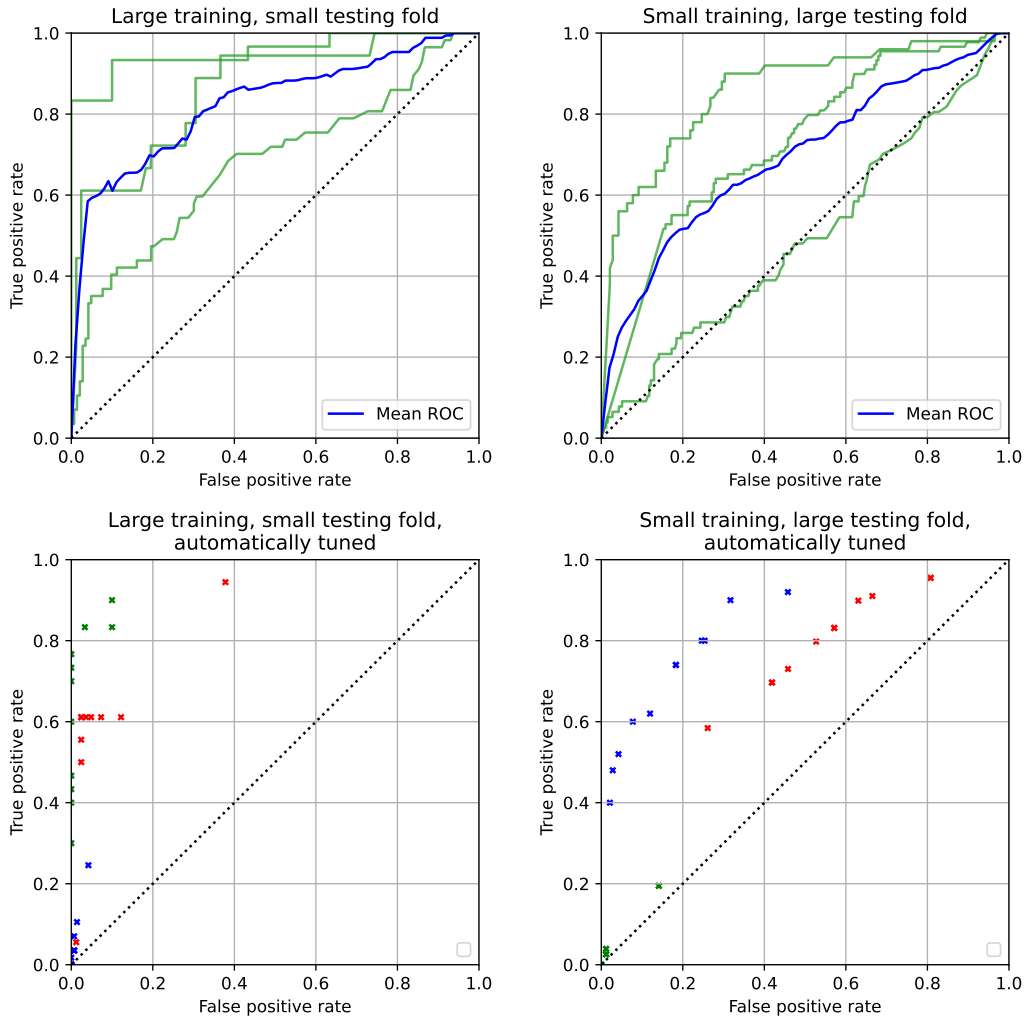

Figure 5.5. Unsupervised 3-σ detector ROC curves generated by day-based block CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold, the scattered charts in the bottom are the result of autotuning the model to a certain TPR value. The charts in the left figures are result of classical day-based block crossvalidation, the right figures are result of "inverse" day-based block CV.

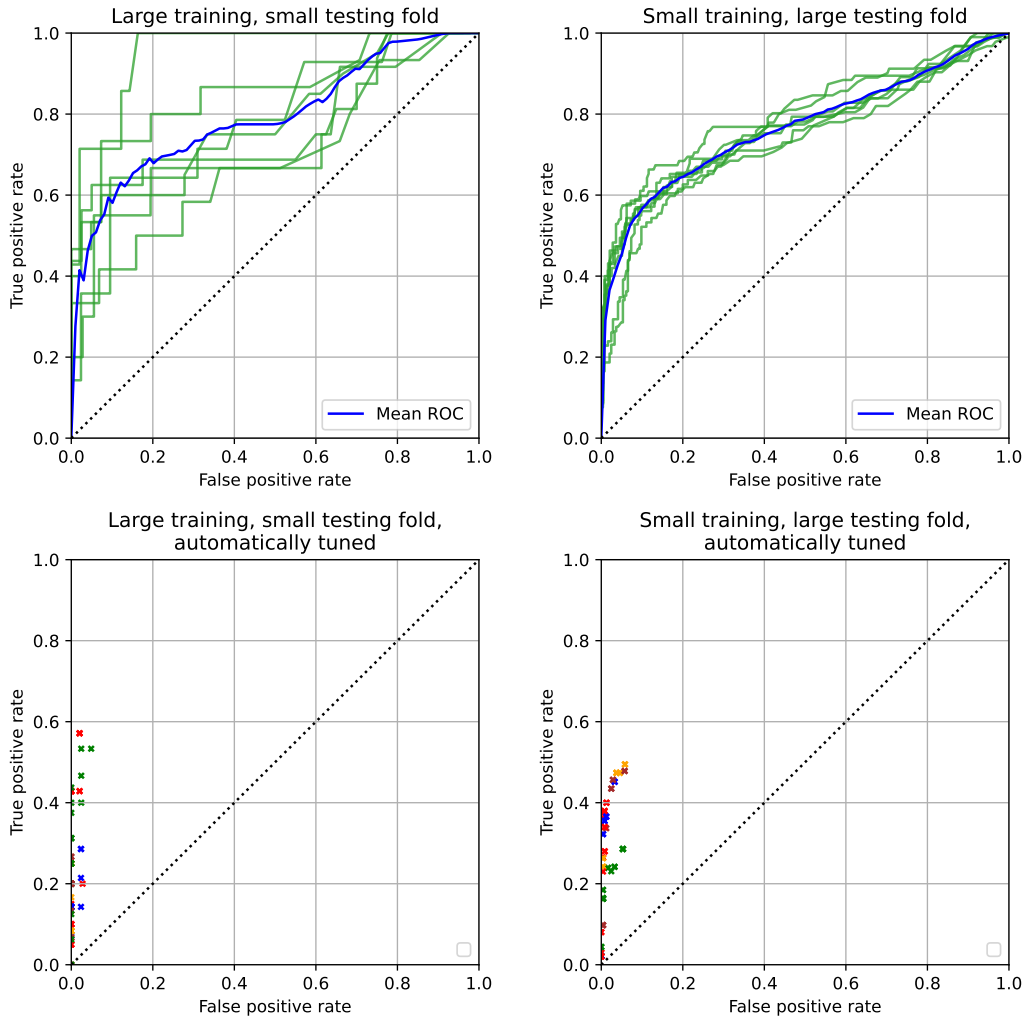

Figure 5.6. Unsupervised 3-σ detector ROC curves resulting from shuffled CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold, the scattered charts in the bottom are the result of autotuning the model to a certain TPR value. The charts in the left figures are result of shuffled 7-fold CV, the right subfigures are result of 7-fold inverse CV.

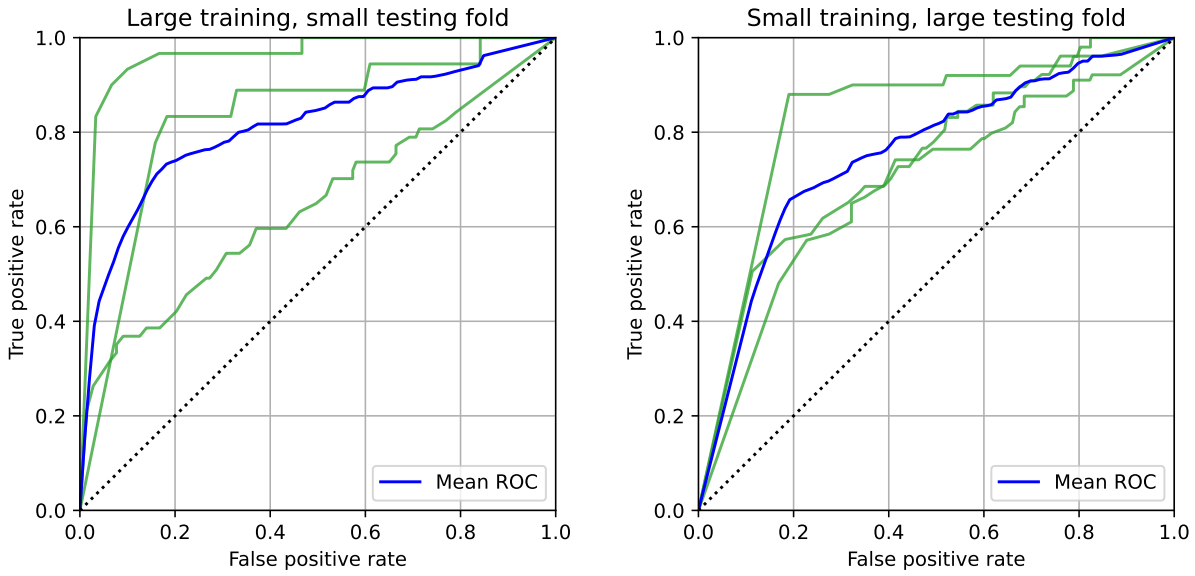

Figure 5.7. Supervised feature detector ROC curves resulting from day-based block CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold. The charts in the left figure is a result of classical day-based block CV, the right figure is a result of "inverse" day-based block CV.

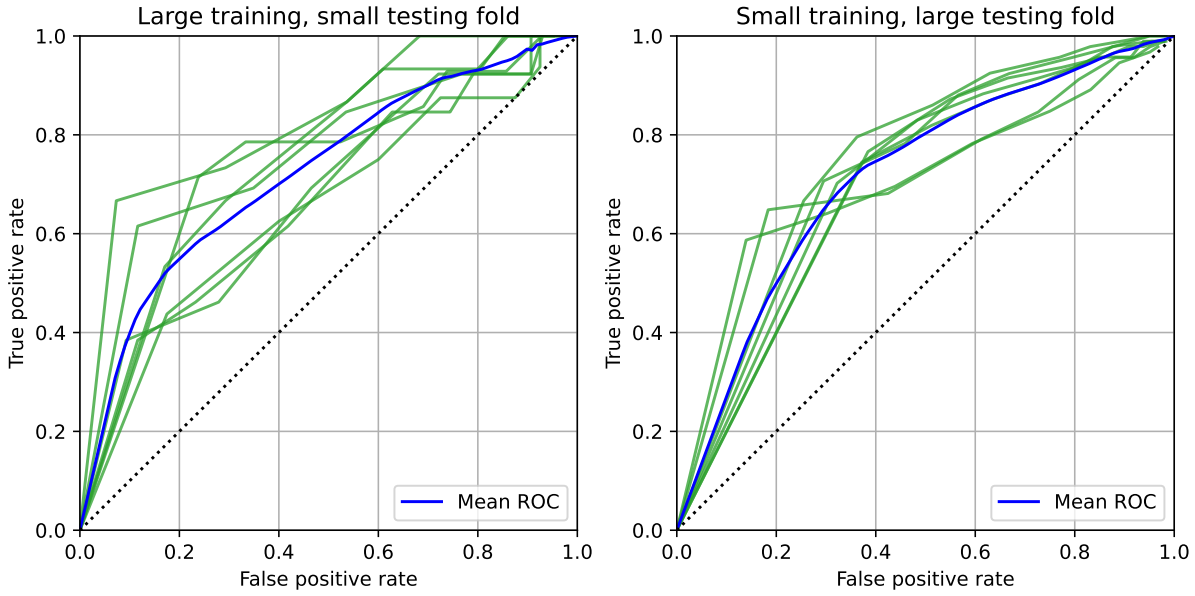

Figure 5.8. Supervised feature detector ROC curves resulting from shuffled CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold. The left subfigure is a result of shuffled 7-fold CV, the right subfigure is a result of 7-fold inverse CV.

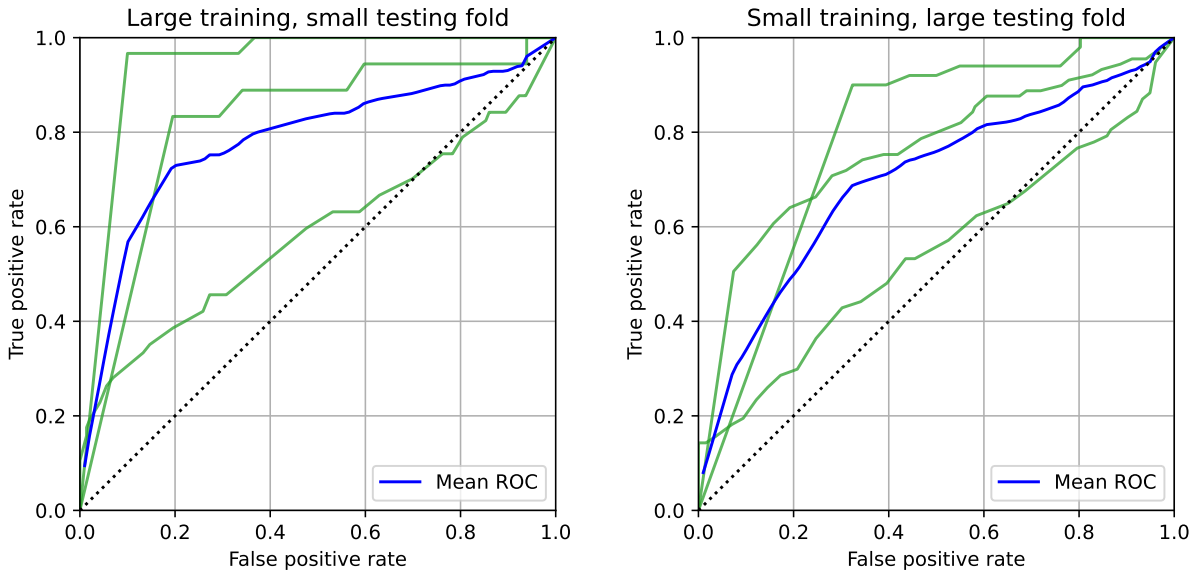

Figure 5.9. Unsupervised feature detector ROC curves resulting from day-based block CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold. The left subfigure is a result of classical day-based block crossvalidation, the right figure is a result of "inverse" day-based block CV.

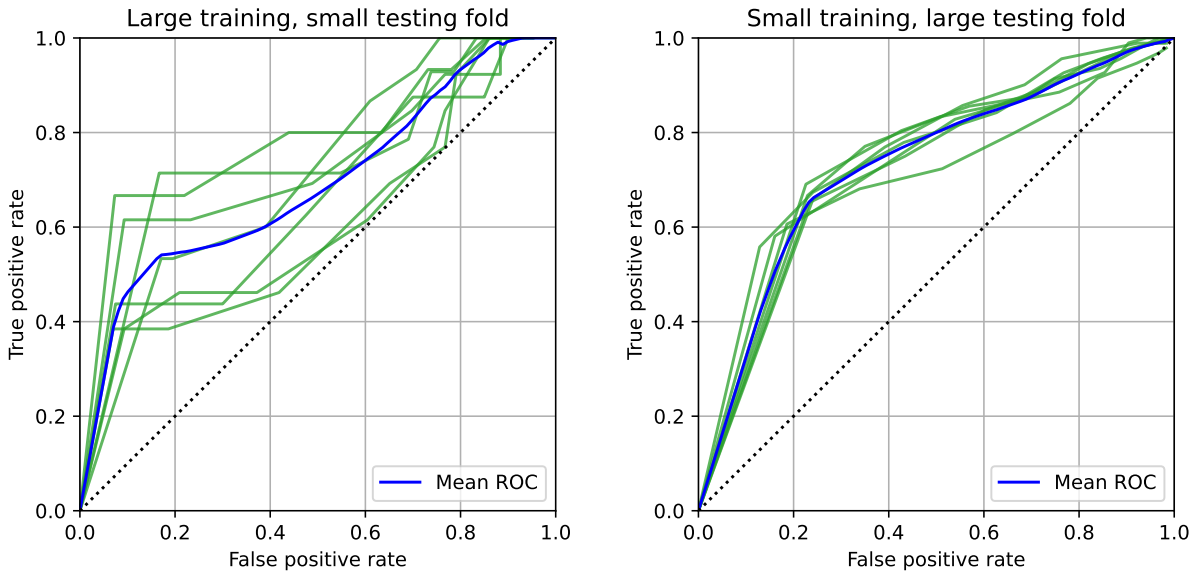

Figure 5.10. Unsupervised feature detector ROC curves resulting from shuffled CV. The green curves are the ROCs of each fold. The blue curve is the mean of these ROC curves. The curves are the result of evaluating based on all possible parameters of the threshold. The left subfigure is a result of shuffled 7-fold CV, the right subfigure is a result of 7-fold inverse CV.

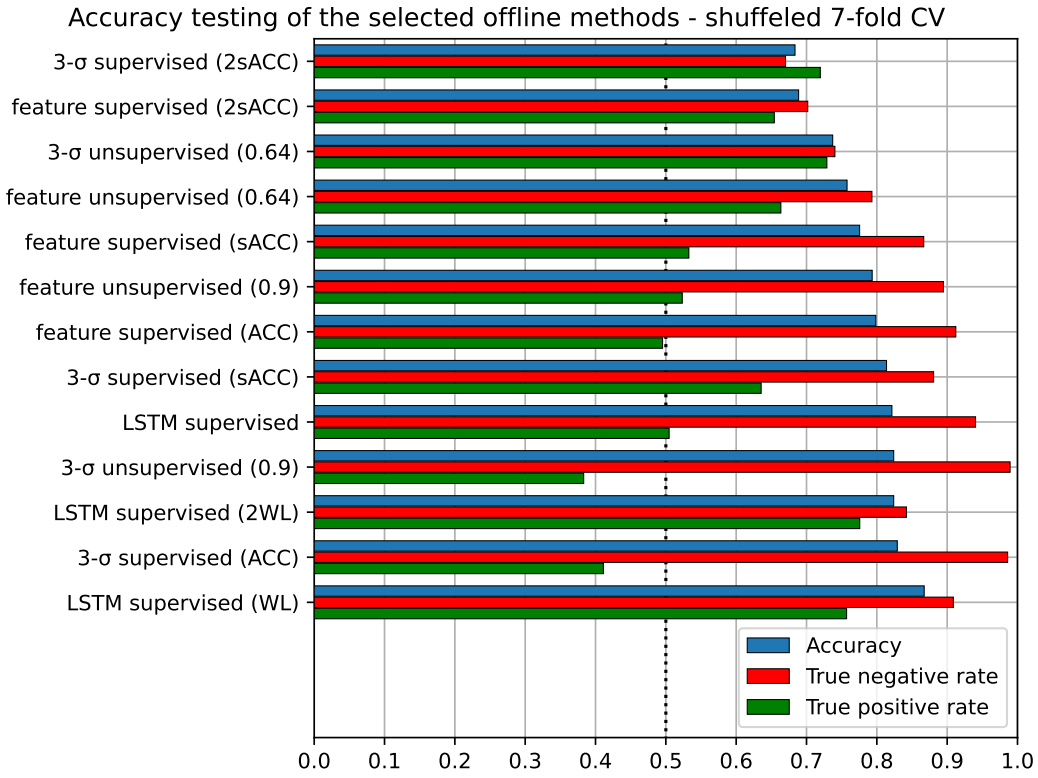

Figure 5.11. Accuracy testing of selected offline methods. The tuning of each method is in parentheses (ACC - general accuracy, sACC - smart skewed accuracy, 2sACC - the classification of the positive samples is twice as important as the negative samples, WL - weighted loss, 2WL - weighted loss where the classification of the positive samples is twice as important as the negative samples). Accuracy, TPR and TNR are measured. The results are computed on the sum of shuffled 7-fold CV.

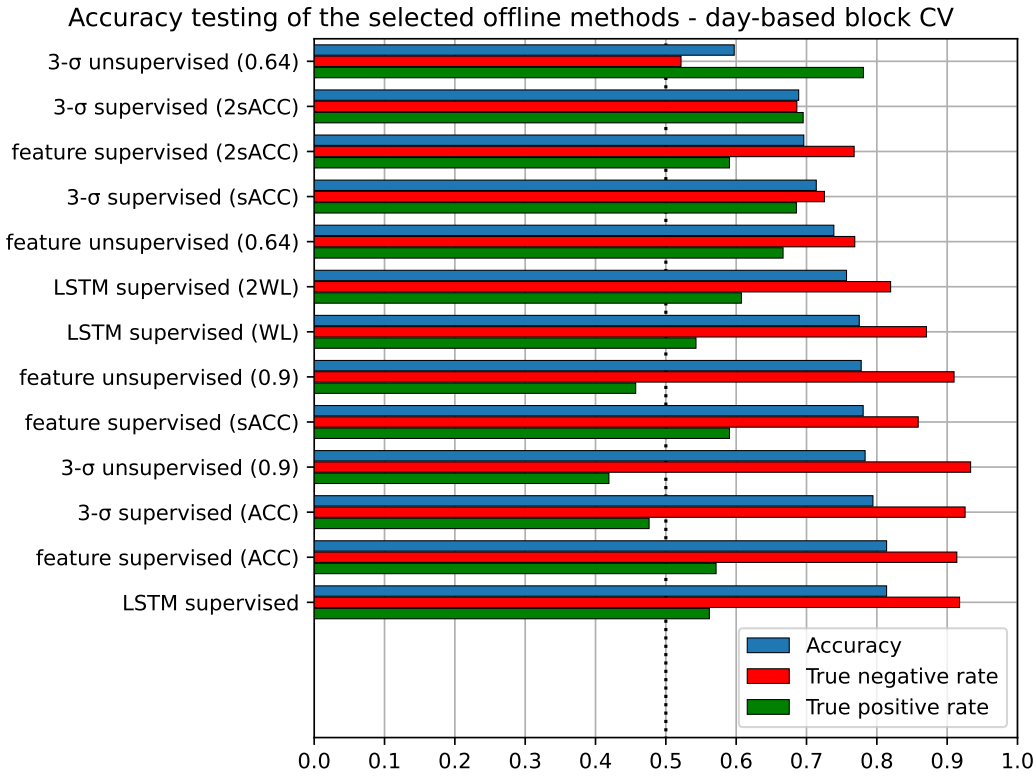

Figure 5.12. Accuracy testing of selected offline methods. The tuning of each method is in parentheses (ACC - general accuracy, sACC - smart skewed accuracy, 2sACC - the classification of the positive samples is twice as important as the negative samples, WL - weighted loss, 2WL - weighted loss where the classification of the positive samples is twice as important as the negative samples). Accuracy, TPR and TNR are measured. The results are computed on the sum of day-based block CV.

5.2 Online methods evaluation

The same two types of crossvalidation were used to test the methods - the shuffled 7-fold crossvalidation and the day-based block crossvalidation. The models were trained on a training set and the signals of the testing set were evaluated by the models. To avoid evaluating signals too short, since the length of a full signal is around a thousand timestamps, the shortest signal part was three hundred timestamps long. From each signal in the testing dataset, ten "new" testing signals were made. The length of these ten parts was evenly spaced from the three hundred up to the length of the whole full signal. Since no labels for timestamps were available, the evaluation of classification had to be done accordingly. The labels for the whole signals are available and on this, we built our evaluation system.

5.2.1 Evaluating the outcome

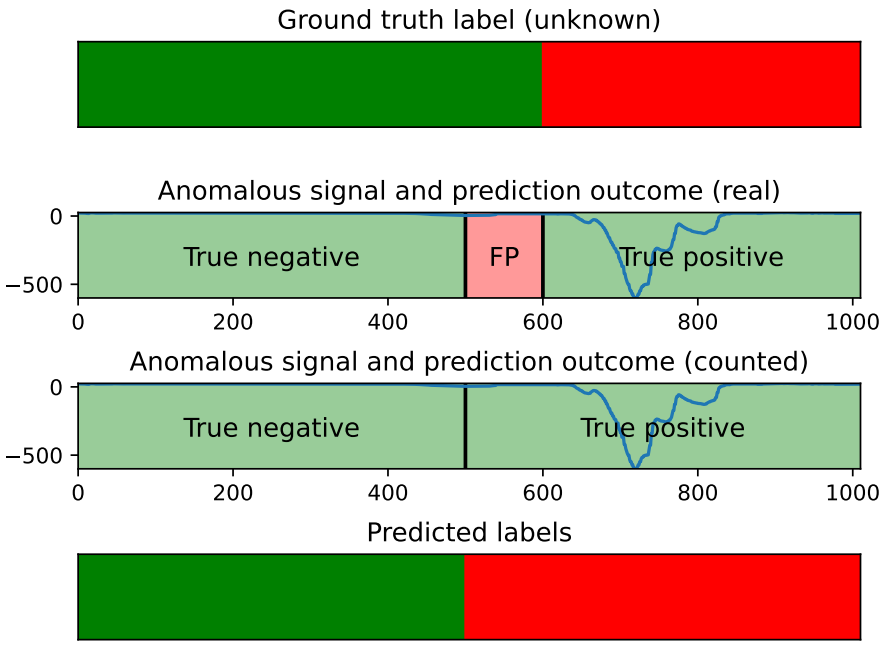

Figure 5.13. Example of the assessment of the online detector, which accidentally detects an anomaly when the signal was still normal (before the `anomaly inception`). The "Ground truth label (unknown)" subplot represents the unknown ground truth label. The subplot titled "Predicted labels" shows the sample prediction of the model. The "Anomalous signal and prediction outcome (real)" subplot is the real evaluation of the prediction, if the ground truth labels were available. The "Anomalous signal and prediction outcome (counted)" subplot represents the prediction evaluation mechanism used in this thesis.

The assessment of the proposed data-stream-driven methods is not as simple as with offline methods, since the labels for timestamps are not available in the dataset. Therefore, there is no way to find out when exactly the anomaly happened in the signal. The anomaly within a signal happens at a certain point and after that the signal is anomalous, however before the moment of "anomaly inception" the signal is normal. The online detection problem described above is not an issue if a non-anomalous signal is being inspected. Such signal should be predicted as normal in any particular moment of the process progress and the outcome of the test can either

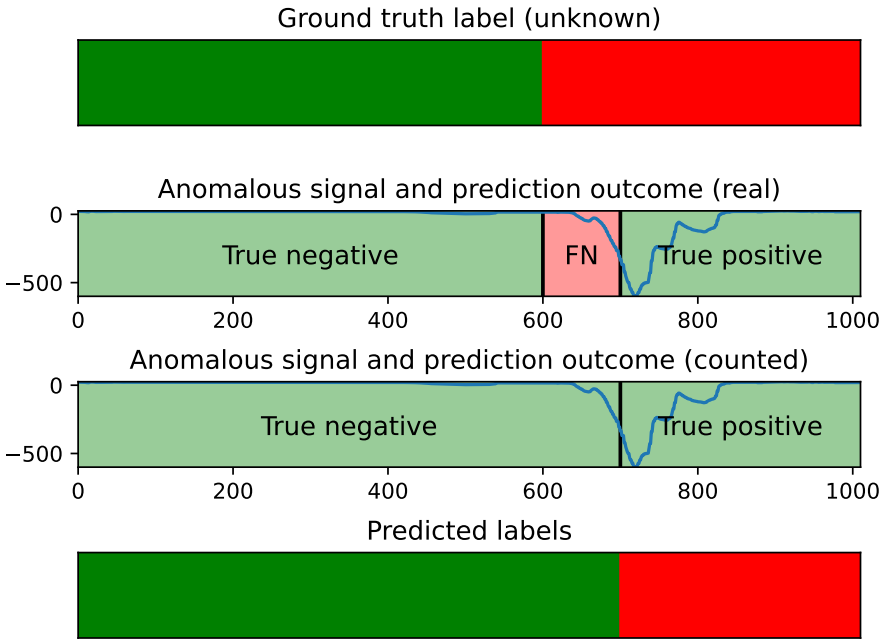

Figure 5.14. Example of the assessment of the online detector, which finds the positive detection late, after the `anomaly inception` already happened. The "Ground truth label (unknown)" subplot represents the unknown ground truth label. The subplot titled "Predicted labels" shows the sample prediction of the model. The "Anomalous signal and prediction outcome (real)" subplot is the real evaluation of the prediction, if the ground truth labels were available. The "Anomalous signal and prediction outcome (counted)" subplot represents the prediction evaluation mechanism used in this thesis.

be `true negative` or `false positive`. Any positive detection appearance is clearly `false positive`. With anomalous samples the evaluation is not as trivial. There was no way of finding out with absolute certainty, when the "anomaly inception" began. To overcome this lack of ground truth knowledge, we built the evaluation on basis of consistency of the predictor's decision. It is based on assumption that a signal that was once classified as anomalous cannot become normal. To evaluate this the parts of the signal inspected before the first positive detection were counted as `true negative`. Similarly, the first sample that was labeled as anomalous was counted as `true positive` and the ground truth of all samples coming after is considered positive.

This creates two inaccuracies in this system of measurement. `False positive` outcome cannot be detected in anomalous signal, because it treats all positive outcomes as true positives (visualized in figure 5.13). The second problem is that it cannot detect the FN samples in anomalous signals if the anomaly is detected later than the "anomaly inception", because all signal parts until the first anomalous samples are counted as TN (visualized in figure 5.14). True negatives after a false anomaly detections are also undetectable, since all the samples after the first detected anomaly are deemed positive (visualized in figure 5.15) Furthermore, if all parts of a tested anomalous signal were labeled as non-anomalous, the whole signal (10 samples) were subtracted from the TN counter and were added to FN. This creates more inaccuracies, mainly the pessimistic bias for TPR of the method because once again the dataset is not labeled on timestamp basis.

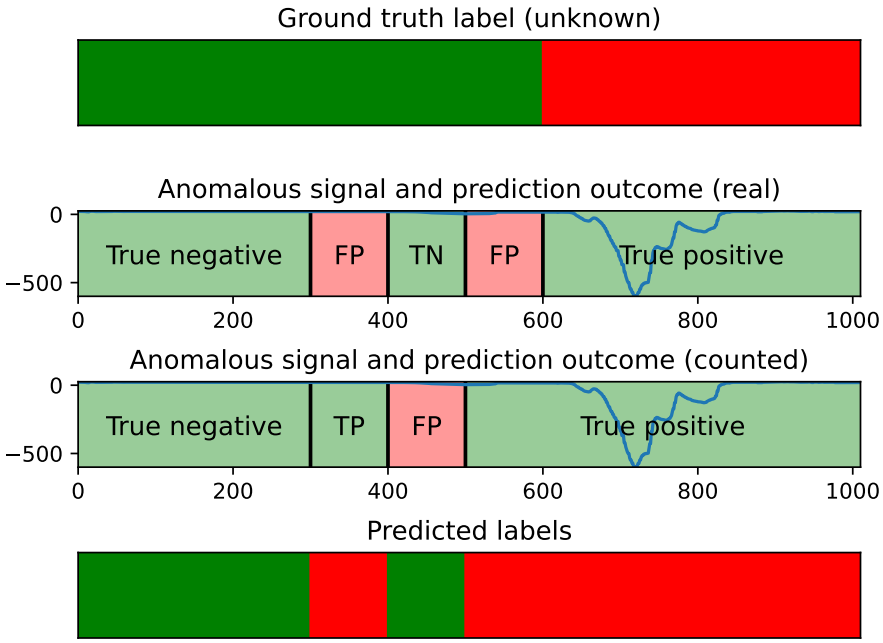

Figure 5.15. Example of the assessment of the online detector, when the predictions are inconsistent. The "Ground truth label (unknown)" subplot represents the unknown ground truth label. The subplot titled "Predicted labels" shows the sample prediction of the model. The "Anomalous signal and prediction outcome (real)" subplot is the real evaluation of the prediction, if the ground truth labels were available. The "Anomalous signal and prediction outcome (counted)" subplot represents the prediction evaluation mechanism used in this thesis.

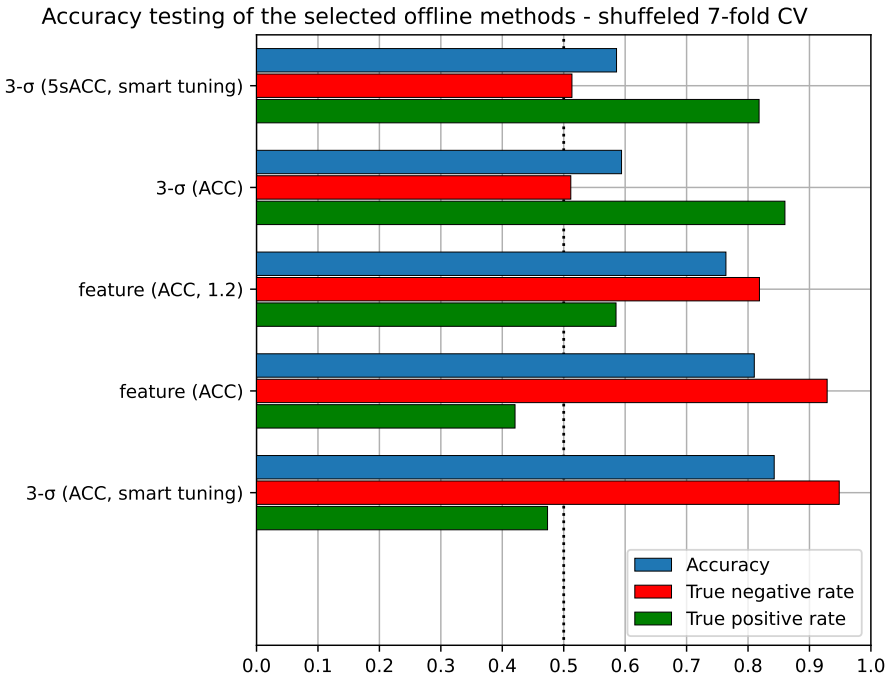

Figure 5.16. Accuracy testing of selected data-stream-driven methods. The tuning of each method is in parentheses (ACC - general accuracy, 5sACC - the classification of the positive samples is five times as important as the negative samples, smart tuning - training specific to the data stream, without smart tuning - evaluation based on the length of the segment). The number in parentheses for feature method is a temporal multiplication coefficient (an equivalent to sACC for data-stream-driven feature method). Accuracy, TPR and TNR are measured. The results are computed on the sum of shuffeled 7-fold CV.

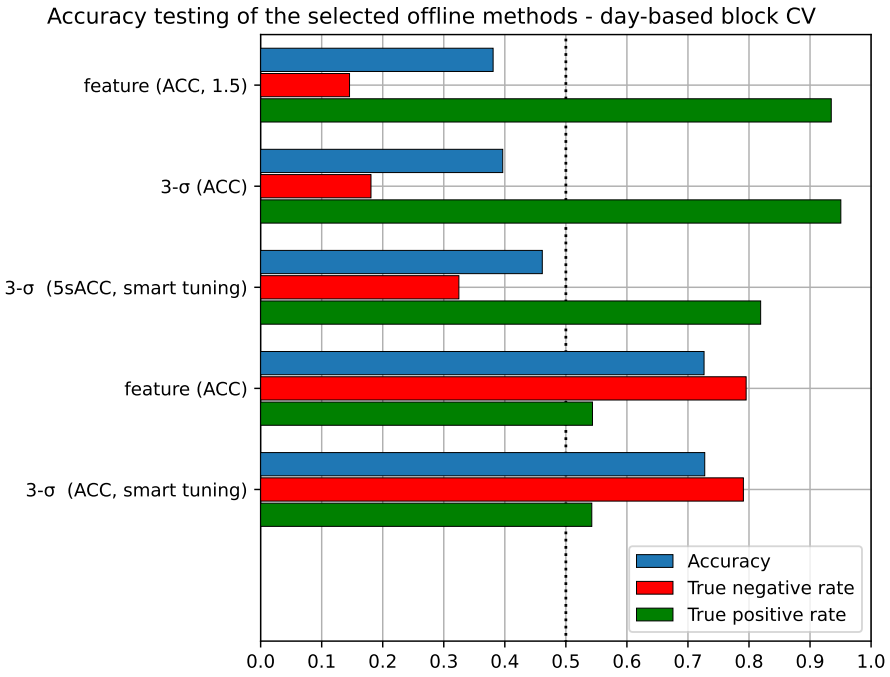

Figure 5.17. Accuracy testing of selected data-stream-driven methods. The tuning of each method is in parentheses (ACC - general accuracy, 5sACC - the classification of the positive samples is five times as important as the negative samples, smart tuning - training specific to the data stream, without smart tuning - evaluation based on the length of the segment). The number in parentheses for feature method is a temporal multiplication coefficient (an equivalent to sACC for data-stream-driven feature method). Accuracy, TPR and TNR are measured. The results are computed on the sum of day-based block CV.

5.3 Analysis of the crossvalidation results, discussion

In this section, results listed in the figures 5.5-5.17 will be discussed. As in the chapters before, at first we will examine the results of testing the offline methods, and then the online methods.

5.3.1 Offline methods testing result analysis

Before we start comparing the proposed offline methods to one another, we should compare all of the methods to the solution used before developing those methods. As mentioned in chapter 2, the solution used prior to our efforts in the CIIRC Testbed for industry 4.0 lab was CNN-based analysis of the pictures of the end product (assembled wheels) taken by a camera within the robot's operating space. The disadvantage of this method is that for each sample a picture is taken from just a single angle, so the detector can only detect the visual defects on one side of the tire. Furthermore, since this method only relies on visual representation of the result of the process, it is unable to detect the fault process based on the wrong stiffness of the tire or a defect on parts not seen on such picture. Nevertheless, given all these limitations, this method still achieves quite an impressive performance which can be seen in the figure 5.18. We did not include its performance in the figures above, because it was measured in a different way, than the crossvalidation based measurement in the figures 5.11 and 5.12. As mentioned in chapter 2 this method was available to us in the way that we could measure its outcome on the measured processes, but we had access to neither the weights of the model, nor the training loop of this network. In addition, no information about the training dataset was provided. The performance of the network was therefore measured during the gathering data phase of this thesis. Unfortunately, we do not have human-annotated data for all the measured data, but just for the last two days of the measurement (14.03.2024, 18.04.2024). The network's prediction to the outcome of the process was noted and compared to the human annotated labels. Measuring the performance directly resulted in the figure 5.18 (numeric values in appendix B in table B.2).

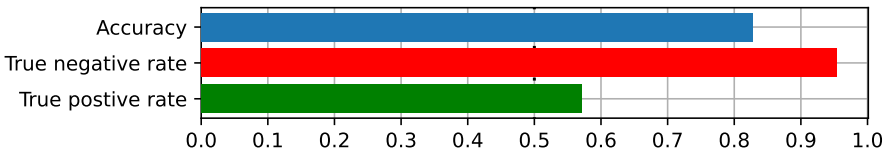

Figure 5.18. Performance of the original CNN-based visual method, that was in use in the Testbed for industry 4.0 lab in CIIRC

When comparing the result with the results of the presented methods, it can be noted, that it reached the performance of the best methods evaluated by the day-based block CV, but the estimation created by this type of crossvalidation is pessimistically biased, and less representative for performance of the method trained on the whole dataset. Therefore, we will look at the performance of the methods estimated by the shuffled 7-fold CV. Regarding the accuracy it is slightly above average, and it holds a really impressive TNR score. In fact when compared to the tested configurations of our methods (figure 5.11), the TNR of the original method would be the third best when directly compared with the results of the CV, but the TPR of the two methods which have a higher TPR is much lower. Therefore, it is dependent on the priority whether

a high TPR, TNR, or type of compromise is requested. From the general standpoint, the best performing method is in our opinion the LSTM-based detector trained with a weighted loss. The accuracy of this detector is the highest of all the methods tested, its TNR is over 90% and its TPR is the second highest of all tested. LSTM-based method came first with respect to accuracy also in day-based block CV testing (in this case without weighted loss), which is surprising since the expectation was that with fewer data, the neural network will perform worse.

It is no surprise, that the supervised methods outperformed the unsupervised ones, but with right estimation of success ratio in unsupervised methods, the result is not as big as one might expect. In the accuracy metric the best performing unsupervised method is for both cases of crossvalidation the 3-σ method with expected success ratio of 90%. However, when we take a closer look on the results, it must be noted that this is because of the considerable discrepancy in the number of successful and anomalous processes within the used dataset, and it comes at a cost of low TPR. The same unsupervised method of estimated success ratio of 64% scored a lower accuracy, on the other hand both TPR and TNR have reasonable values. And in this lies the greatest strength of those proposed methods. All of them can be self tuned to an ideal value based on the desired criterion. In the case of figures 5.11 and 5.12, the criterions were accuracy and *skewed accuracy*, and in the case of ROC curves the criterion was a minimal desired TPR value. The methods will self-optimize based on those criterions, but if human intervention is necessary, the more "low level" parameters such as decision thresholds can also be changed directly.

Another surprising thing is that the LSTM methods with weighted loss outperformed the LSTM without the usage of weighted loss. This can probably be explained by the classifier-like architecture of the network, which allowed model to learn both of the outcomes better and not predict all the samples as a non-anomalous signal. Other possible explanation is that the learning rate used was too small, and the network benefited from greater loss (and therefore greater training step) for the anomalous class.

As was already discussed, it is not easy to rank the methods in general, because the ranking depends on the requirements on the outcome (certain TPR, certain TNR, certain accuracy...). In the terms of accuracy and "skewed accuracy", it can be said that the best method in general is the LSTM, but it is dependent on the size of the dataset. The statistical methods show similar results, but out of the methods with the specific tuning we tested, the 3-σ method might perform slightly better.

From the ROC analysis we conclude that the 3-σ method generally outperforms the feature method on both CV types, and in both supervised and unsupervised way of the training. This difference in performance is only marginal in the range of 0-5%. The another interesting case from the ROC analysis is that the shuffled CV tends to have worse mean AUC results than the day-based block CV, even though when we compare the accuracy performance of these two CV measurements, the results are the other way. This is probably because the way of the measuring the mean AUC. It was computed as a real mean of AUCs of the curves generated from the folds. The TPR, TNR and accuracy in was computed in figures 5.11, 5.12 and table B.2 were computed in a way of summing the TP, TN, FP and FN from each measurement and therefore measuring these across all folds, not from averaging the values from each fold. Since in this type of CV the differences in the number of samples in each fold are great for the three days, these two measurements will come out differently. This can be also seen when we take a look at figure 5.5 (inverse crossvalidation subfigure), where one of the curves is in

line with the dashed line. This is caused by the fact that the model was trained on the small dataset (samples from 14.03.2024) of size 60 and tested on the rest of the dataset.

5.3.2 Data-stream-driven methods testing result analysis

Evaluation of the online methods is not as straightforward and is described in section 5.2.1. Given these facts, the values in figures 5.16 and 5.17, evaluate more the ability of the classifier not to change the prediction during the evaluation of the same signal. The pessimistic bias of the TPR has to also be acknowledged, since when the anomalous signal is predicted as non-anomalous during all the stages of the testing process, all the measurements are counted as FN. When evaluating the anomalous signal and anomaly is spotted, the number of TP samples is usually less than the number of signal snippets parts tested. Therefore, if for example the average anomaly is spotted on the fifth of the ten total measurements, and in some cases it is never spotted, for each correctly spotted anomalous signal, which has 5 TP samples, there is a signal that has 10 FN samples. This way the TPR is pessimisticaly biased.

Also because of this type of measuring the data, a five times skewed accuracy was used instead of twice skewed accuracy for offline methods, because the twice skewed accuracy gave almost the same result as normal accuracy.

When we put the two tested methods toe to toe, there again is no clear winner. It depends on the requirements of our model. For example interesting choice could be a feature method tuned on accuracy with multiplier 1-1.2, because on both types of examined CV this method holds impressive TNR, while keeping relatively high TPR when compared to other tested methods. The 3-σ method is not much worse though.

5.4 Experiments on the real process

On 20th of May 2024, the selected methods were evaluated in real time in the real process. An interface for these methods created by Vojtěch Hanzlík [1] was used for these measurements. It is important to measure the "real" performance of the methods on real data and compare it to the estimations based on the performed crossvalidations. The problem with the measurement from that day was once again the quality of the tires. The tested tires are designed to be equipped on the rim only once. Thus, not in the way we do it in the measurements, where every time the assembly is performed, the tires are removed from the rims to do the assembly again. Unfortunately this time we really did not have any spare undamaged tires, so the measurement was executed with the ones we possessed.

Due to the high demand for equipment in the lab, we were only able to test a portion of the methods. We aimed our efforts at testing the online methods. In total, 40 signals were evaluated by all methods. The results were surprising and slightly inconsistent with the results in the crossvalidations. The problem within this measurement was that the robot was not calibrated in this particular day and the rate of failure was unusually high when compared to the previous measurements. It reached 42.5% even without using the artificial anomalies. The damage to tires probably also played a significant role, which may have gone undetected in some cases, so the real fail rate may have been even higher.

The best performing method we developed based on the measurements from this day is the 3-σ method, which reached the exactly same performance in all three metrics as the visual CNN-based method, even though it failed on different samples. The biggest surprise of this measurement was the underperformance of the LSTM-based method,

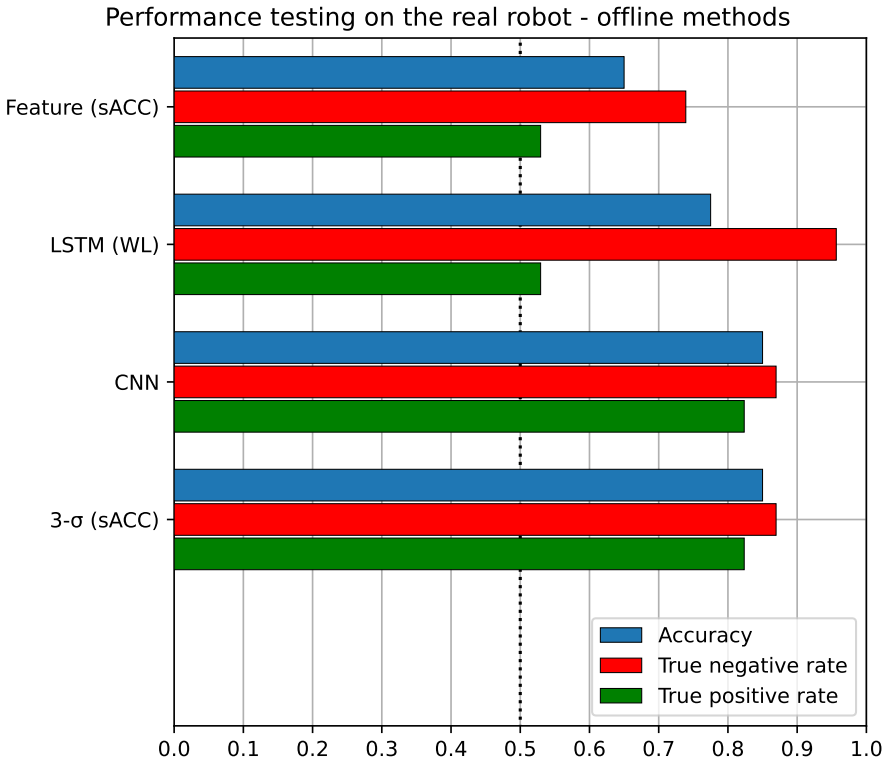

Figure 5.19. Comparison of performance of different methods on the same dataset in the Testbed for industry 4.0 lab in CIIRC.

in comparison to the crossvalidation testing. The method was unable to detect a lot of faults, however its true negative rate is the highest of all the methods. The worst performing method was the feature method. The results can be seen in figure 5.19.

Chapter 6
Conclusion

In this thesis, we developed three method architectures for detecting anomalies from force-torque signals from the sensor on the end effector of the Delta robot to detect anomalies in the assembly process. These methods, inspired by algorithms used for anomaly detection, were used for this problem, but are universally applicable to other time series classification/anomaly detection tasks. The capability of operating with data streams and real-time online prediction were implemented for both of the developed statistical methods. A great emphasis was given to comparing the methods with selected tuning from all designed method architectures to one another in a standardized testing process, which aimed to be as fair as possible. In the end we tested the performance of the methods using the interface designed by Vojtěch Hanzlík in [1] on the real Delta robot assembly process in the Testbed for industry 4.0 lab.

The result of this thorough testing process found that the methods reach and sometimes outperform the solution used before in the Testbed for industry 4.0 lab (CNN, which makes predictions from the photos of the outcome). Furthermore, our methods are capable of detecting anomalies undetectable by the previous solution.

6.1 Future work

In this thesis we evaluated only a few possibilities of approaching the problem. For example the chapter 5 only contains experiments of the basic developed methods, but the special capabilities of these methods is not tested for instance, the continual learning. We plan to publish an article based upon the solution from this and [1] thesis. In this paper, the complete results of testing will be published. With the recommended range of around thirty pages for the bachelor's thesis these results would extend this by another ten to twenty pages. Therefore, only the basic variants of the implemented methods were tested.

The python library we developed as a part of this thesis needs a proper documentation, so other parties can use it as well. We plan to make a GitHub pages with the tutorials, explanation and of usage of the developed methods. Some parts of the problems we solve within the methods could be implemented with faster, more efficient algorithms. For example the search for optimal thresholds in statistical methods is done nearly in brute force way, and could be done much more efficiently with basic search algorithms. We plan to release the optimised version with the paper.

One of the directions of future work could be the solution based on classifier ensemble methods. Combining the outcome of our methods and the outcome of the visual CNN-based method could outperform both of these separate approaches.

During the work on this thesis, new articles about interesting methods usable for our thesis were found. To point out an interesting example, the solution published in [36] could very well substitute the DBA barycenter, or add another usable feature in the feature method. Another addition is that LSTM-based method should be able to operate on data streams, for example like in [37] and [38].

We also want to develop an autoencoder NN, which based on the research seems to produce good results. The deep learning based solutions, which seem to take over this field from statistical methods should also be compared. For example trying to solve our problem through fully convolutional or transformer neural network architecture. These approaches were not developed, since our deep-learning efforts started later during the semester, when we acquired a large enough dataset. This alone would be also an interesting motive for an article.

As we can see the limits for possible improvements are sky-high, and we should probably get back to work.

References

[1] Vojtěch Hanzlík. *Edge AI Integration for Anomaly Detection in Assembly using Delta Robot*. 2024.

[2] Pavol Tanuska, Lukas Spendla, Michal Kebisek, Rastislav Duris, and Maximilian Stremy. Smart anomaly detection and prediction for assembly process maintenance in compliance with industry 4.0. *Sensors*. 2021, 21 (7), 2376.

[3] Mattia Carletti, Chiara Masiero, Alessandro Beghi, and Gian Antonio Susto. A deep learning approach for anomaly detection with industrial time series data: a refrigerators manufacturing case study. *Procedia Manufacturing*. 2019, 38 233–240.

[4] A Goncharov, A Savelev, N Krinitsyn, and S Mikhalevich. *Automated anomalies detection in the work of industrial robots*. In: *IOP Conference Series: Materials Science and Engineering*. 2021. 012095.

[5] Santosh Thoduka, Nico Hochgeschwender, Juergen Gall, and Paul G Ploger. A Multimodal Handover Failure Detection Dataset and Baselines. *arXiv preprint arXiv:2402.18319*. 2024.

[6] Kyu Min Park, Younghyo Park, Sangwoong Yoon, and Frank C. Park. Collision Detection for Robot Manipulators Using Unsupervised Anomaly Detection Algorithms. *IEEE/ASME Transactions on Mechatronics*. 2022, 27 (5), 2841-2851. DOI 10.1109/TMECH.2021.3119057.

[7] Wang Li, Yong Han, Jianhua Wu, and Zhenhua Xiong. Collision Detection of Robots Based on a Force/Torque Sensor at the Bedplate. *IEEE/ASME Transactions on Mechatronics*. 2020, 25 (5), 2565-2573. DOI 10.1109/TMECH.2020.2995904.

[8] Tobias Biegel, Nicolas Jourdan, Carlos Hernandez, Amir Cviko, and Joachim Metternich. Deep learning for multivariate statistical in-process control in discrete manufacturing: A case study in a sheet metal forming process. *Procedia CIRP*. 2022, 107 422–427.

[9] Jianbo Yu, and Yue Zhang. Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review. *Neural Computing and Applications*. 2023, 35 (1), 211–252.

[10] Tianyuan Lu, Lei Wang, and Xiaoyong Zhao. Review of anomaly detection algorithms for data streams. *Applied Sciences*. 2023, 13 (10), 6353.

[11] Matej Kloska, Gabriela Grmanova, and Viera Rozinajova. Expert enhanced dynamic time warping based anomaly detection. *Expert Systems with Applications*. 2023, 225 120030. DOI https://doi.org/10.1016/j.eswa.2023.120030.

[12] D. Situnayake, and J. Plunkett. *AI at the Edge*. O'Reilly Media, 2023. ISBN 9781098120177.
`https://books.google.cz/books?id=l6imEAAAQBAJ`.

[13] Chip Huyen. *Designing Machine Learning Systems: An Iterative Process for Production-ready Applications*. O'Reilly Media, Incorporated, 2022.

ISBN 9781098107963.
https://books.google.cz/books?id=YISIzwEACAAJ.

[14] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*. 2022, 135 364–381.

[15] Serhii Voronov. *Device for hand guiding of an industrial robot*. 2022. https://dspace.cvut.cz/handle/10467/101233.

[16] Product Information Force/torque sensor FT Mini58. SCHUNK. 2023.

[17] Douglas C Montgomery. *Statistical Quality Control*. 7 ed. Nashville, TN: John Wiley & Sons, 2012. ISBN 978-1-118-14681-1.

[18] Tony Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*. 2009, 4 (11-5), 41–42.

[19] François Petitjean, Alain Ketterlin, and Pierre Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*. 2011, 44 678-693. DOI 10.1016/j.patcog.2010.09.013.

[20] Hiroaki Sakoe, and Seibi Chiba. *A Dynamic Programming Approach to Continuous Speech Recognition*. In: *Proceedings of the Seventh International Congress on Acoustics*. 1971. 65-69.
https://api.semanticscholar.org/CorpusID:107516844.

[21] H. Sakoe, and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1978, 26 (1), 43-49. DOI 10.1109/TASSP.1978.1163055.

[22] Romain Tavenard. *An introduction to Dynamic Time Warping*. https://rtavenar.github.io/blog/dtw.html. 2021.

[23] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I. Webb. Elastic similarity and distance measures for multivariate time series. *Knowledge and Information Systems*. 2023, 65 (6), 2665-2698. DOI 10.1007/s10115-023-01835-4.

[24] Richard G Brereton. The chi squared and multinormal distributions. *Journal of Chemometrics*. 2015, 29 (1).

[25] James MacQueen, and others . *Some methods for classification and analysis of multivariate observations*. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1967. 281–297.

[26] Ville Hautamaki, Pekka Nykanen, and Pasi Franti. *Time-series clustering by approximate prototypes*. In: *2008 19th International conference on pattern recognition*. 2008. 1–4.

[27] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN 9780071154673.
https://books.google.cz/books?id=EoYBngEACAAJ.

[28] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*. 2009, 8 (7), 579–588.

[29] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*. 2012, 2 (417), 1.

[30] Alexander Rehmer, and Andreas Kroll. On the vanishing and exploding gradient problem in Gated Recurrent Units. *IFAC-PapersOnLine*. 2020, 53 (2), 1243–1248.

[31] Sepp Hochreiter, and Jurgen Schmidhuber. Long short-term memory. *Neural computation*. 1997, 9 (8), 1735–1780.

[32] Josef Tkadlec. *Diferenciální a integrální počet funkcí jedné proměnné*. Czech Technical University in Prague, 2011. ISBN 978-80-01-04792-7. English title translation: *"Differential and Integral Calculus of single variable functions"*.

[33] Diederik P Kingma, and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014.

[34] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*. 2006, 27 (8), 861–874.

[35] Cyril Goutte, and Eric Gaussier. *A probabilistic interpretation of precision, recall and F-score, with implication for evaluation*. In: *European conference on information retrieval*. 2005. 345–359.

[36] Yutao Liu, Yong-An Zhang, Ming Zeng, and Jie Zhao. A novel shape-based averaging algorithm for time series. *Engineering Applications of Artificial Intelligence*. 2023, 126 107098.

[37] Sucheta Chauhan, and Lovekesh Vig. *Anomaly detection in ECG time signals via deep long short-term memory networks*. In: *2015 IEEE international conference on data science and advanced analytics (DSAA)*. 2015. 1–7.

[38] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, and others . *Long Short Term Memory Networks for Anomaly Detection in Time Series*. In: *Esann*. 2015. 89.

Appendix A
List of abbreviations

ACC	Accuracy
AI	Artificial inteligence
ANN	Artificial Neural Network
Aruco	Augmented reality university of Cordoba
AUC	Area under curve
CE	Cross entropy loss
CIIRC	Czech institute of informatics, robotics and cybernetics
CNN	Convolutional neural network
CTU	Czech technical university in Prague
CV	Crossvalidation
DDTW	Dependent dynamic time warping
DTW	Dynamic time warping
e. g.	Exempli gratia
FN	False negative
FP	False positive
FPR	False positive rate
HITL	Human in the loop
LSTM	Long-short term memory
ML	Machine learning
MLP	Multilayer perceptron
NN	Neural network
OPC	Open platform communication
OPCua	Open platform communication unified architecture
ROC	Reciever operating characteristic
SVM	Support vector machine
TN	True negative
TNR	True negative rate
TP	True positive
TPR	True positive rate
TS	Time series
VEGP	Vanishing-exploding gradient problem

Appendix B
Offline methods comparison

Method	Classifier	CV type	Train set size	min AUC	max AUC	mean AUC
3-σ	unsupervised	shuffeled	big	0.67613	0.91399	0.77638
3-σ	unsupervised	shuffeled	small	0.73780	0.79403	0.76124
3-σ	unsupervised	days	big	0.67470	0.92111	0.81875
3-σ	unsupervised	days	small	0.49391	0.85134	0.69075
3-σ	supervised	shuffeled	big	0.72541	0.83635	0.77836
3-σ	supervised	shuffeled	small	0.71119	0.75778	0.74025
3-σ	supervised	days	big	0.68642	0.95111	0.84022
3-σ	supervised	days	small	0.64596	0.85937	0.74737
feature	unsupervised	shuffeled	big	0.56171	0.73740	0.63094
feature	unsupervised	shuffeled	small	0.67716	0.74261	0.71507
feature	unsupervised	days	big	0.58551	0.94000	0.78075
feature	unsupervised	days	small	0.54472	0.78803	0.69783
feature	supervised	shuffeled	big	0.57344	0.76585	0.67500
feature	supervised	shuffeled	small	0.67278	0.73041	0.69197
feature	supervised	days	big	0.64452	0.96000	0.81172
feature	supervised	days	small	0.71429	0.83852	0.75582

Table B.1. ROC AUC analysis comparisson of the selected offline methods

Method	Classifier	CV type	TPR	TNR	mean ACC
3-σ	unsupervised (0.9)	shuffeled	0.38318	0.98947	0.82398
3-σ	unsupervised (0.64)	shuffeled	0.72897	0.74035	0.73724
3-σ	unsupervised (0.9)	days	0.41905	0.93333	0.78333
3-σ	unsupervised (0.64)	days	0.78095	0.52157	0.59722
3-σ	supervised (ACC)	shuffeled	0.41121	0.98596	0.82908
3-σ	supervised (sACC)	shuffeled	0.63551	0.88070	0.81378
3-σ	supervised (2sACC)	shuffeled	0.71963	0.67018	0.68367
3-σ	supervised (ACC)	days	0.47619	0.92549	0.79444
3-σ	supervised (sACC)	days	0.68571	0.72549	0.71389
3-σ	supervised (2sACC)	days	0.69524	0.68627	0.68889
feature	unsupervised (0.9)	shuffeled	0.52336	0.89474	0.79337
feature	unsupervised (0.64)	shuffeled	0.66355	0.79298	0.75765
feature	unsupervised (0.9)	days	0.45714	0.90980	0.77778
feature	unsupervised (0.64)	days	0.66667	0.76863	0.73889
feature	supervised (ACC)	shuffeled	0.49533	0.91228	0.79847
feature	supervised (sACC)	shuffeled	0.53271	0.86667	0.77551
feature	supervised (2sACC)	shuffeled	0.65421	0.70175	0.68878
feature	supervised (ACC)	days	0.57143	0.91373	0.81389
feature	supervised (sACC)	days	0.59048	0.85882	0.78056
feature	supervised (2sACC)	days	0.59048	0.76774	0.69615
LSTM	supervised	shuffeled	0.50467	0.94035	0.82143
LSTM	supervised (WL)	shuffeled	0.75701	0.90877	0.86735
LSTM	supervised (2WL)	shuffeled	0.77570	0.84211	0.82398
LSTM	supervised	days	0.56190	0.91765	0.81389
LSTM	supervised (WL)	days	0.54286	0.87059	0.77500
LSTM	supervised (2WL)	days	0.60748	0.81961	0.75691
Original*	?	other**	0.57143	0.95348	0.82812

Table B.2. Comparisson of the accuracies of the offline methods based on different parameters and CV.

*"Original" refers to the CNN-based method of evaluating the process based on the picture of the outcome.

**"Other" refers to the process of obtaining these values described in section 5.3.1.

Appendix C
Online methods comparison

Method	Setting	CV type	TPR	TNR	mean ACC
3-σ	ACC	shuffeled	0.85181	0.51006	0.59184
3-σ	ACC (smart tuning)	shuffeled	0.46886	0.94830	0.84031
3-σ	5sACC (smart tuning)	shuffeled	0.80506	0.51127	0.58236
3-σ	ACC	days	0.94279	0.17886	0.39472
3-σ	ACC (smart tuning)	days	0.53433	0.78973	0.72361
3-σ	5sACC (smart tuning)	days	0.80138	0.31901	0.45528
feature	ACC	shuffeled	0.42077	0.92879	0.81020
feature	ACC (1.2)	shuffeled	0.58515	0.81858	0.76403
feature	ACC	days	0.54361	0.79533	0.72639
feature	ACC (1.5)	days	0.93476	0.14563	0.38083

Table C.3. Comparisson of the accuracies of the selected online methods based on different parameters and CV.