Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Håvard Løkensgard & Erlend Johann Skarpnes

AnoMove

Anomaly Detection in Infant Movement for Predicting Cerebral Palsy

Master's thesis in Computer Science
Supervisor: Professor Heri Ramampiaro
June 2020

NTNU
Norwegian University of
Science and Technology

Håvard Løkensgard & Erlend Johann Skarpnes

AnoMove

Anomaly Detection in Infant Movement for Predicting Cerebral Palsy

Master's thesis in Computer Science
Supervisor: Professor Heri Ramampiaro
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Cerebral palsy is a permanent motor dysfunction with no existing cure. Although there are no established tests to detect its presence, early diagnosis and treatment can greatly improve the chances of decelerating its symptoms. Children at risk of having cerebral palsy are today clinically observed over several years before receiving a diagnosis. An earlier diagnosis is based on an analysis of the idle movement of infants, but it is time consuming and requires specially trained paediatricians. A system based on computer vision may aid in this analysis, alleviating the need of scarce specialists, and with better accuracy.

To address this, we want to apply anomaly detection models to analyse the movement of infants. Earlier studies have found the position of joints based on video recordings and created an annotated dataset of healthy and impaired infants. However, there exists no method for applying anomaly detection on this type of data. Our solution to this is the AnoMove method. AnoMove uses angles between joints to represents the movements of infants and transform them into frequencies using Fourier transformation. This is followed by a dimensionality reduction using PLS-DA. The result from this processing is used as the input for XGBOD, an outlier detection model. AnoMove uses the scores from XGBOD to predict abnormal movement. In addition to this, AnoMove can visualise the results from the preprocessing and anomaly prediction.

By using AnoMove we could classify infant with promising results. Anomaly detection seems to be a well-suited approach for the problem of finding cerebral palsy in infants.

Sammendrag

Cerebral parese er en permanent motorisk dysfunksjon uten en eksisterende kur. Selv om det ikke er noen etablerte tester for å oppdage dens tilstedeværelse, kan tidlig diagnose og behandling i stor grad forbedre sjansene for å redusere symptomene. Barn med risiko for å få cerebral parese blir i dag klinisk observert over flere år før de får en diagnose. En tidligere diagnose er basert på en analyse av inaktiv bevegelse hos spedbarn, men den er tidkrevende og krever spesialutdannede barneleger. Et system basert på datasyn kan hjelpe i denne analysen ved å redusere behovet for spesialister og ha en bedre nøyaktighet.

For å adressere dette, ønsker vi å bruke anomali-deteksjonsmodeller for å analysere bevegelsen til spedbarn. Tidligere studier har funnet leddens plassering basert på videoopptak og skapt et annotert datasett med friske og syke spedbarn. Det eksisterer derimot ingen metode for å anvende anomalideteksjon på denne typen data. Vår løsning på dette er AnoMove-metoden. AnoMove bruker vinkler mellom ledd for å representere småbarns bevegelser og transformerer dem til frekvenser ved bruk av Fourier-transformasjon. Dette blir fulgt av en dimensjonalitetsreduksjon ved bruk av PLS-DA. Resultatet fra denne behandlingen blir brukt som inndata for XGBOD, en detekteringsmodell for anomalier. AnoMove bruker resultat fra XGBOD for å finne unormale bevegelser. I tillegg til dette kan AnoMove visualisere resultatene fra data-prosesseringen og anomali-prediksjonen.

Ved å bruke AnoMove kunne vi klassifisere spedbarn med lovende resultater. Anomali-deteksjon ser ut til å være en godt egnet tilnærming for problemet med å finne cerebral parese hos spedbarn.

Acknowledgements

We would like to express our sincere gratitude to both our supervisor, Professor Heri Ramampiaro, and our resources in InMotion, Associate Professor Espen Alexander F. Ihlen and PhD Research Fellow Daniel Groos.

Heri has shown interest in our work from the very first day and has given us helpful advice and motivation along in our research. He has inspired us to do our best, and we do not think our report would have gotten to the quality it is at now if it were not for him. Espen and Daniel have given us helpful guidance and have both shown great interest and confidence in our work.

We will also thank family and friends that have assisted us in the process of reading our thesis and giving us continuous feedback. Without their expertise and help the thesis would not be of the quality that we are now proud to present.

Outline

Introduction

The first chapter will give the reader an introduction to the problem at hand, the main goal, and research questions. It will also highlight work done in the specialisation project.

Theoretical Background

Chapter 2 will give the reader an introduction into the theoretical knowledge that is used in this thesis.

Previous Work

This chapter will present other research that are done before this thesis and that has given motivation and developed the necessary technology and methods for completing this thesis.

Method

Chapter 4 gives the reader an introduction to the details of the dataset before it in detail explains the different parts of the model: preprocessing, anomaly detection, post processing and visualisation.

Results

The results chapter presents the results from the entire method. It also includes results that made the baseline of important choices throughout this thesis.

Discussion

In this chapter the authors will give their interpretation of the results and discuss choices that were made in the method. It will highlight different aspects of the method that could be explored further.

Conclusion

Chapter 7 concludes our findings in the research done for this thesis.

Contents

List of Figures

List of Tables

1. Introduction

In this chapter the reader will be introduced to the background and motivation for this study. The chapter will provide basic information about cerebral palsy and explain why our study is important. Section 1.2 gives the reader an introduction to the InMotion project. In Section 1.3 we describe the problem and research questions for this thesis. The subsequent section describes the scope of the thesis. The chapter ends with a description of preliminary work done by the authors.

This thesis is a continuation of the specialisation project done by the authors in the autumn of 2019. The following sections in this chapter are gathered from this project: Section 1.1 and 1.2.

1.1. Cerebral Palsy

Cerebral palsy (CP) refers to a group of neurological disorders that appear in infancy or early childhood and permanently affect body movement and muscle coordination. CP is caused by damage or abnormalities inside the developing brain that disrupt the brain's ability to control movement and maintain posture and balance. [2]

About 2.1 children per 1000 live births are diagnosed with CP. [3] There are several risk factors that increase the likelihood of having CP, such as premature birth or complications during pregnancy.

Infants born with less than 28 weeks of gestation or with a birth weight under 1000g are in the high-risk group. Between 10-20% of infants in this group get CP. [4]

Traditionally, diagnosing CP in infants is a tedious process. Trained experts can diagnose CP in infants as early as three months after term, but it is time-consuming, and not many doctors are trained in this technique. This leads to a high cost for diagnosing infants. Typically, CP is diagnosed between an age of 1 and 5 years, depending on the severity of the disease.

Children have a higher brain plasticity the younger they are. Treatments for CP is more effective the more plastic the brain is, meaning that treatment should start as soon as possible. To be able to achieve this, new tools are made to facilitate early diagnosis of CP. The tools created should strive to be as non-invasive, easy, and accurate as possible,

to minimise the impact of the procedure on the infants and their parents.

1.2. InMotion

In recent years, experts from St. Olav hospital and researchers from NTNU have worked on a project called InMotion. The project is based on analysing symptoms of CP in infant movement from a video that can be captured in the comfort of the infant's home.

So far, the project has worked on extracting pose information from video, representing the infants by the position of its limbs. There is ongoing research on describing the symptoms from these videos, but currently no solution is in place.

Until now, there are no one involved in the InMotion project that has treated the problem as an anomaly detection problem. The method described in this thesis is seen as an alternative to the deep-learning methods currently being tested in InMotion.

1.3. Problem Description

As of today, there exists no acceptable method for analysing the videos in InMotion, nor are there any methods for anomaly detection in general human movement to the authors knowledge. This report will lay the groundwork for such a method. Our goal for this study is to further investigate the possibility of using anomaly detection models to classify infant movements. In order to achieve this, we must also try to find new ways to represent the movement of infants based on the videos in the InMotion project.

Research Questions

Our research goal for this thesis is: How can anomaly detection models be used to find abnormal patterns in infant movement caused by cerebral palsy?

To be able to answer the question above, we will work around the following research questions.

RQ1: How can infant movement be represented with minimal feature loss and a minimal number of dimensions?

RQ2: To what degree can basic anomaly detection models predict cerebral palsy in infants?

RQ3: What is the state-of-the-art in outlier detection?

RQ4: How do state-of-the-art outlier detection models perform when compared to basic anomaly detection models?

1.4. Scope

In this thesis, the focus will be on how to preprocess infant movement for use in traditional outlier detection models. No new outlier detectors will be researched, but we will evaluate the performance of existing off-the-shelf algorithms. To choose the correct outlier algorithm for our dataset we will run an extensive search through models and parameters and choose those that perform the best. These will be used as base estimators in an ensemble of outliers, which will perform as our predictor.

The deliverables in this Master will be a pipeline with preprocessing, a predictor consisting of an outlier ensemble, and a visualisation tool to visualise the result of the predictor. This visualisation tool is both to aid researchers in understanding the results of the model but could also be used by medical professionals in their analysis of patients.

1.5. Preliminary Work

The authors worked on a specialisation project that lead up to this thesis. The research questions that were answered in that specialisation project were:

RQ1: *How can infant movement be represented in few dimensions?*

RQ2: *To what degree can changes in angles as a signal be used to represent infant movement?*

RQ3: *To what degree can Fourier transformation be used to generate data that can be used in standard anomaly detection methods?*

2. Theoretical Background

This chapter will give a thorough introduction to the theory behind the proposed method presented in this thesis. The theory can be categorised into the different sections of a rather generic pipeline within data mining and machine learning, containing data preprocessing, model generation and training, and evaluation.

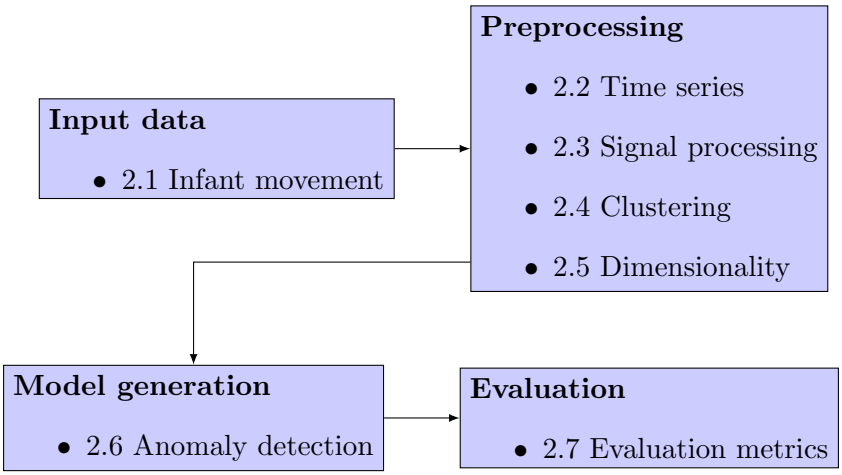


Figure 2.1.: A generic data mining pipeline with the corresponding sections in this chapter.

We start off this chapter with an introduction to some of the medical background behind this research. Section 2.1 gives the reader an introduction to different terms and definitions that are used to describe infant movement. It also defines the differences between normal and abnormal movement.

Following that, we cover the different aspects of our data and the required methods for preprocessing it. Section 2.2 explains what time series are and different types of time series. Digital Signal Processing (Section 2.3) introduces the reader to different terms, definitions and equations that are used in digital signal processing. Section 2.4 explains clustering and different methods to measure distance within clusters. Dimensionality (Section 2.5) explains different methods to reduce the dimensionality of data.

Finally, we consider the theory behind the model we must build to carry out our predictions. Section 2.6 explains anomaly detection and presents the reader to different

algorithms that are commonly used in anomaly detection. The last section explains different evaluation methods that are commonly used in computer science and when evaluating medical results.

This work is a continuation of the specialisation project done by the authors in the autumn of 2019. The following sections in this chapter are gathered from that project: Section 2.1, 2.2, 2.3, 2.4 (without the part on Minkowski distance), 2.6, 2.6.1 (until One-Class Support Vector Machine), 2.6.2 and 2.7 (until 2.7.1).

2.1. Infant Movement Analysis

Movement analysis is used for describing a subject's movement and explain the relevant information surrounding the movement. This is relevant in physical examination and when diagnosing humans with diseases which affects movement. Infant movement analysis is a subset of movement analysis.

2.1.1. General Movement Assessment

General movement assessment is a method for diagnosing cerebral palsy in infants. Different movement patterns are described in Table 2.1. They are used to classify movements as normal or abnormal, and can give an indication whether or not infants have cerebral palsy. The method used when analysing general movements is called *Gestalt perception* and is a proven method for this task [5]. According to Gestalt psychology, the whole is different from the sum of its parts [6], meaning that one can perceive patterns in entire sequences, but not necessarily in each of its parts.

Definition 1 *General movement (GM)[5]*
General movement are on-going movements involving all body parts and appears particularly suitable for assessment.

Normal GMs	
From 9 to 49 weeks postmenstrual age (PMA)	
Fetal and preterm general movements (GMs)	Similar to writhing movements (see below), but wider and jerkier, especially in the lower limbs.
Writhing movements (WMs)	Variable amplitude, slow-to-moderate speed, typically ellipsoid limb trajectories lying close to the sagittal plane with superimposed rotations. Mostly expressed around term age (40 weeks PMA).
From 46 up to 64 weeks PMA	
Fidgety movements (FMs)	Smaller than WMs, moderate average speed with variable acceleration in all directions, migrating through all body parts as an on-going flow of movement. Continual in the awake infant, except during fussing, crying and focused attention. Peak of expression around 3 months post-term (52 weeks PMA).
Abnormal GMs	
Preterm and WMs period	
Poor repertoire (PR)	Monotonous sequences, few movement components, repetitive and not so complex as in normal WMs. Fluency may be reduced too, but is usually more spared than complexity and variability.
Cramped-synchronized (CS)	No complexity, no fluency, no variability: all limb and trunk muscles contract and relax almost simultaneously.
Chaotic (Ch)	Large amplitude, high jerk and chaotic order without any fluency or smoothness. Rare, often evolving into CS.
Hypokinesia	No or very few GMs are detectable during several hours (infrequent pattern, mostly seen in the first days after the onset of a severe hypoxic-ischaemic encephalopathy).
FMs period GMs	
Absent FMs ($FM-$)	FMs are never observed in the whole period
Abnormal FMs (AF)	Fidgety-like movements, but amplitude, average speed and jerkiness are exaggerated

Table 2.1.: Description of different general movement patterns. Table from [5].

2.1.2. Fidgety Movements

Infants between 10-15 weeks have fidgety movements, which is spontaneous movements not initiated by external stimuli. Fidgety movements are characterised as a continuous stream of tiny and elegant movements in all directions, and are present in the neck, trunk, and limbs [7].

A lack of fidgety movements has been shown to indicate cerebral palsy. Doctors trained in Gestalt perception are able to analyse fidgety movements, but without an objective measuring system, physicians working alone risk drifting away from observation standards.

2.2. Time Series

A method of representing movement is as a time series. Times series can be used for multiple other problems like detecting trends, monitoring streaming data, and in different scientific measurements. In this section we will present usage of time series and common problematics.

Definition 2 *Time series [8]*
Let $k \in \mathbb{N}$, $T \subseteq \mathbb{R}$. A set of indexed elements of $\mathbb{R}^k$, $\{x_t | x_t \in \mathbb{R}^k, t \in T\}$ is called an observed time series.

A time series is a series of elements that are dependent on time. Each element x is recorded at a time step t. Time series analysis can be used in stock marked, weather forecast, speech recognition, earthquake prediction and many other areas [9]. One example of a time series can be seen in Figure 2.2.

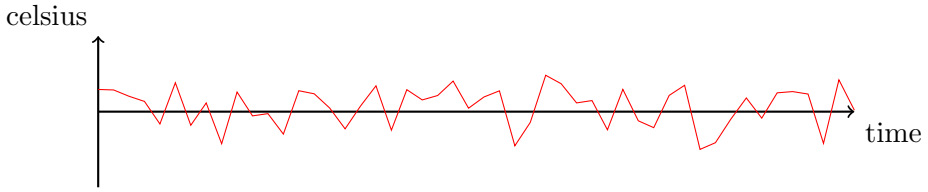


Figure 2.2.: Example of a time series. The red line is an example of temperature in November. Time is represented on the x-axis and the temperature on the y-axis.

Types of Time Series

Time series can be divided into two groups: univariable and multivariable data. Univariable has only one feature changing over time. An example of that can be temperature changing throughout the day. Multivariable on the other hand deals with multiple features. These features can all be dependent on time, but they can also be dependent on each other.

In time series there are mainly two different types of problems, namely forecasting problems and classification problems. Forecasting problems uses historical data to predict future values. Weather forecasting is a good example of a forecasting problem using multivariable time series, as it considers several factors such as temperature, humidity and wind patterns. Classification problems takes the input data and classifies it into a given number of predefined classes. One example of a classification problem for time series can be detecting heart disease based on the heartbeat.

By applying statistical analysis to time series, it is possible to detect features like trends, seasonality, outliers, long-term cycles, constant variance and abrupt changes. Detecting these features are important as they may impact the outcome of the prediction or classification. Statistical analysis can also be used to evaluate the best representation of the data, and what type of analysis it is possible to perform depending on the different features that are found in the statistical analysis.

2.3. Digital Signal Processing

Digital signal processing is essential in many applications, such as audio, sonar, digital image processing and control systems. It takes a signal and processes it into another domain. The result is a possibility to extract more information from the signal and apply different methods and operation to the signal for enhancing the different features.

Definition 3 *Signals [10]*
A signal describes how some physical quantity varies over time and/or space.

The simplest signal is a sinus curve illustrated by the blue curve in Figure 2.3. Sound waves, radio transmissions and television broadcasts are all examples of signals.

Definition 4 *Signal processing [10]*
Manipulating a signal to change its characteristics or extract information from it.

In signal processing there are typically three different classes of problems. The first class is the problem with *eliminating noise* from a signal. In Figure 2.3 there are two curves,

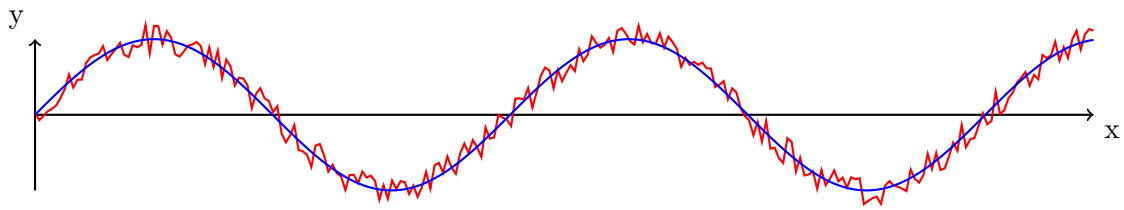


Figure 2.3.: The blue curve represents $f(x) = \sin(x)$ and the red curve represents $f(x)$ with noise.

both of them are $\sin(x)$, but one of them has added noise. The goal is then to eliminate the noise from the red curve so that the new curve matches $f(x)$. The second problem is *correcting distortion* in signals. The final class of problems is *extracting an indirect quantity from measured signals*. This problem is further explained in Example 1.

Example 1 *Indirect information*
A radar transmits a signal to an aeroplane. The distance to an aeroplane is given by the time it takes the signal to go back and forth to the plane. Another parameter that can be calculated is the velocity, which is given by the Doppler shift in the signal. The distance and velocity are calculated not from the signal itself, but rather by extracting the information embedded in the signal.

Moving Average

Definition 5 *Moving average*
A succession of averages derived from successive segments typically of constant size and overlapping of a series of values.

A moving average can be used to smoothen out sudden peaks in a time series signal, making it easier to compare different time series. The simplest form of moving average is the simple moving average (SMA). The average is taken from a window set either before the given value or on either side of the given value. By choosing a window before the given value, the value will be determined by historical data. This is often used in financial applications. By choosing a window on either side of the given value, the value is determined by its nearest neighbours. This is typically done as preprocessing in data science.

Equation 2.1 describes a simple moving average with historical data and Equation 2.2

describes moving average with nearest neighbours.

$$p_{SM} = \frac{p_M + p_{M-1} + ... + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i} \tag{2.1}$$

$$p_{SM_{nearest}} = \frac{p_{M+n/2} + p_{M+n/2-1} + ... + p_{M-n/2}}{n} = \frac{1}{n}\sum_{i=-n/2}^{n/2} p_{M-i} \tag{2.2}$$

Theorem 1 *Nyquist theorem [11]*
A signal with bandwidth B can be completely reconstructed if $2B$ samples per second are used. The theorem introduces the Equation 2.3, where R_{max} is the maximum data rate and M is the discrete level of signal.

Definition 6 *Aliasing [10]*
Aliasing occurs when the rate of sampling is too low to capture the true nature of a signal.

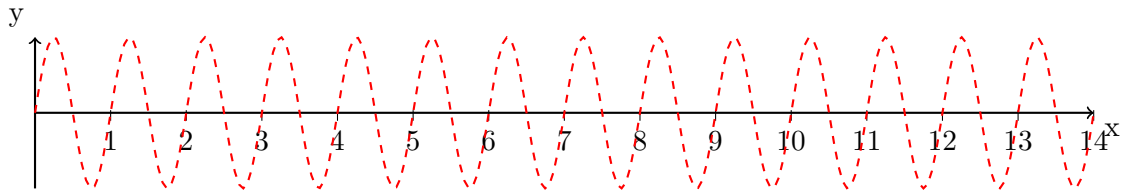
$$R_{max} = 2B \log_2 M \tag{2.3}$$

Aliasing is a common problem when sampling signals and occurs when the sampling rate is lower than the Nyquist frequency. Aliasing can be seen in Figure 2.4, where we have a sinus signal that is sampled. Figure 2.4a shows the original signal, with a frequency of 1. If we try to sample it with a sampling rate of 1, we get the signal in 2.4b, and not the original signal.

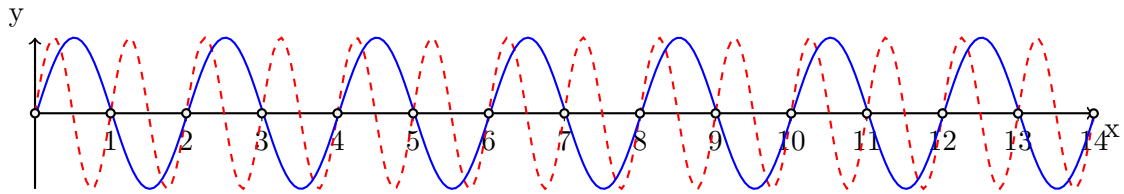
Rather than finding the right sampling rate through experiments, Theorem 2.3 gives the correct sampling: $\frac{1}{2}$. Sampling with this rate gives the points represented by circles in Figure 2.4c and results in $f(x) = f(x)_{sampled}$. One important feature of the Nyquist Theorem is that by applying the sampling rate given by the theorem the result will be the simplest function that fits every sample. This function obtained from fitting a function through every point $f(x)_{sampled}$ is the correct representation of the signal.

Continuous Fourier Transform

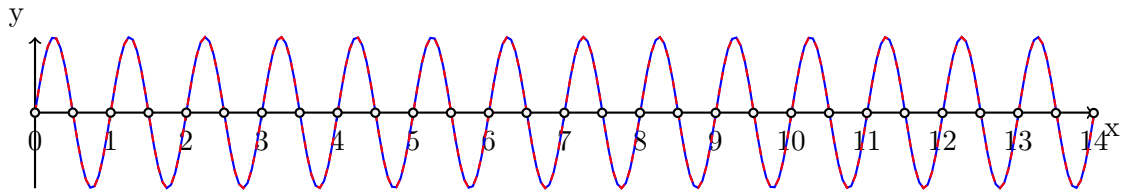
When working with signal analysis there are several different tools for describing or transforming the signal into another domain, one of these being the continuous Fourier transform. A Fourier transformation decomposes a signal into its constituent frequencies. This makes it possible to analyse the parts that make up a signal instead of the signal itself.



(a) The red curve represents $f(x) = \sin(2\pi x)$



(b) The simplest function that intersect every point given with a sample rate of 1 is the sinus function $f(x) = sin(\pi x)$.



(c) Sampling with the rate of $\frac{1}{2}$ given by the Theorem 2.3 gives $f(x)_{sampled} = f(x)$. This is clear when plotting both functions.

Figure 2.4.: Aliasing in a signal.

An example of a signal transformed by continuous Fourier transform can be seen in Figure 2.5. The continuous Fourier transform is defined in Equation 2.4 and the inverse Fourier transform in Equation 2.5

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-2\pi i f t} dt \tag{2.4}$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{2\pi i f t} df \tag{2.5}$$

Discrete Fourier Transform

When working with time series we often have a sampled signal instead of a continuous one. The Discrete Fourier Transform (DFT) can be used when the data is discrete values as it iterates over every point and apply the Fourier transform to each point. This is showed by the Equation 2.6, and the inverse Fourier transformation is defined in Equation 2.7. $H(k)$ will be a function that is a combination of sinusoids and that lies in the frequency domain.[12]

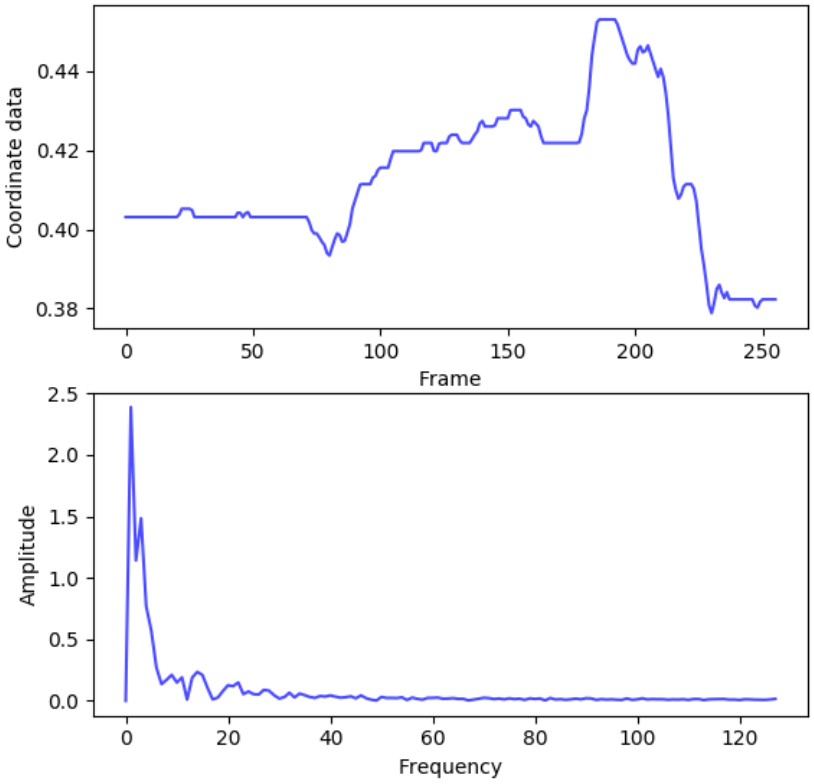


Figure 2.5.: A Fourier transformed signal. The original signal is at the top and comes from a x-coordinate of a limb in an infant in movement. The frequency data is at the bottom.

$$H(k) = \frac{1}{N} \sum_{n=0}^{N-1} h(k) e^{-2\pi i \frac{kn}{N}} \tag{2.6}$$

$$h(n) = \frac{1}{N} \sum_{n=0}^{N-1} H(n) e^{2\pi i \frac{kn}{N}} \tag{2.7}$$

Fast Fourier Transform (FFT)

Due to the high computational cost of calculating the discrete Fourier transform given by the Equation 2.6, the fast Fourier transform algorithm was created. The algorithm is based on the Cooley-Tukey [13] method, where it was proposed how manipulation of the window size and using recursion could simplify the problem from having a $\mathcal{O}(n^2)$ runtime to a $\mathcal{O}(n \log n)$ runtime.

The algorithm works by using the Danielson-Lanczos lemma [14]. It says that a DFT in the size of a power of two can be rewritten as the sum of the DFT of the odd and even indexed entries of the original signal. Since the size is a power of two, this can be done recursively until the size of the DFT is 1, making it trivial to calculate. The Fast Fourier Transform algorithm can be seen in Algorithm 1.

Algorithm 1: The Fast Fourier Transform algorithm

Data: Vector $\vec{x}$ of 2^n samples
Result: Discrete Fourier Transform of $\vec{x}$
Set N to length of vector $\vec{x}$
if *N is 1* **then**
    return $\vec{x}$
else
    Set $\vec{x}_{even}$ to FFT of the even indices of $\vec{x}$
    Set $\vec{x}_{odd}$ to FFT of the odd indices of $\vec{x}$
    forall $k = 0$ **to** $N/2 - 1$ **do**
        Set t to $\vec{x}_{odd,k} * e^{-2\pi ik/N}$
        Set DFT_k to $\vec{x}_{even,k} + t$
        Set $DFT_{k+N/2}$ to $\vec{x}_{even,k} - t$
    end
    return DFT
end

2.4. Clustering

Clustering can be used to group together similar data points. There are multiple ways to cluster data, with common methods of data clustering being centroid-based clustering and density-based clustering. Centroid-based clustering revolves around clustering a data point to the nearest cluster prototype point. These prototype points are created iteratively based on the dataset and does not have to be in the dataset itself. Density-based clustering finds areas in the data with a similar density between the data points and assigns clusters to the areas. An example of clusters can be seen in Figure 2.6.

Definition 7 *Cluster Analysis [15]*
Cluster analysis divides data into groups (clusters) that are meaningful, useful, or both.

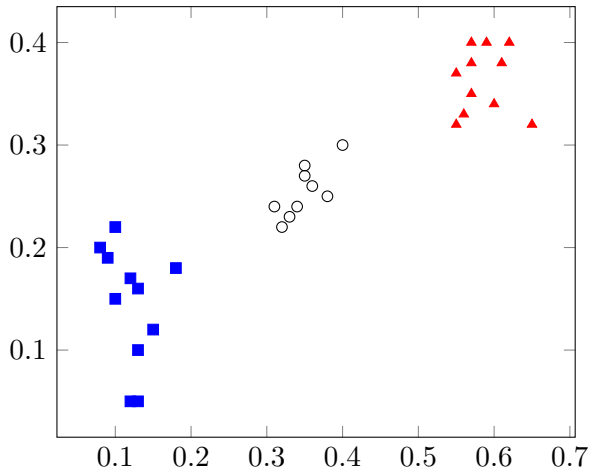


Figure 2.6.: Example of three clusters in a two dimensional space.

2.4.1. Distance Measures

When clustering datasets, two points must be directly comparable by measures such as geometric distance, density or other methods. As the data available for clustering can model any values, different measurements can be implemented. Three common choices for distance are *manhattan distance*, *euclidean distance*, and *cosine similarity*.

Manhattan Distance

Manhattan distance is the sum of the differences in all dimensions between two data points. It is named after the distance a cab would have to drive in Manhattan by following the grid-pattern in the streets. It is also known as *taxicab* distance with the same origin, but also *rectilinear distance*, L^1 *distance*, L_1 *distance*, *snake distance*, and more. Manhattan distance is the simplest form of measurement between two data points. The equation for Manhattan distance can be seen in Equation 2.8 and an illustration in Figure 2.7.

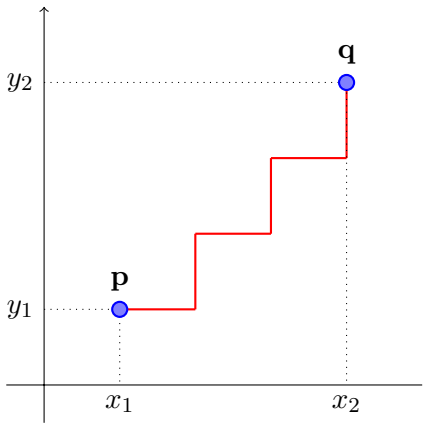


Figure 2.7.: Illustration of Manhattan distance

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.8}$$

Euclidean Distance

Euclidean distance, also known as the straight-line distance, is the shortest path between two points in a Euclidean space. The equation for an n-dimensional Euclidean distance can be seen in 2.9 and an illustration in Figure 2.8.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.9}$$

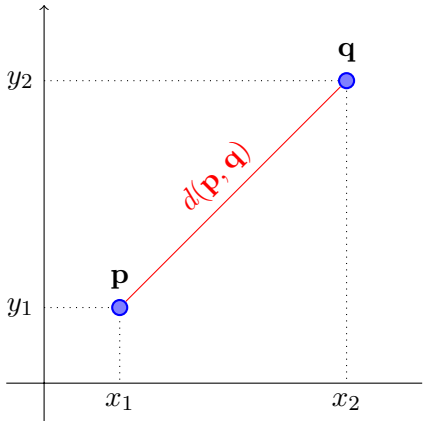


Figure 2.8.: Illustration of Euclidean distance

Minkowski Distance

Minkowski distance is a generalised version of Manhattan and Euclidean distance. It is defined as follows:

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathcal{R}^n$. The *Minkowski distance* $d(\mathbf{x}, \mathbf{y})$ of order p between the two point is defined in Equation 2.10

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) = \{\sum_{i=1}^{n} |x_i - y_i|^p\}^{\frac{1}{p}} \tag{2.10}$$

Minkowski distance has two special cases: When $p = 1$, the Minkowski distance is the same as the Manhattan distance, and for $p = 2$ it is the same as the Euclidean distance.

Cosine Similarity

The cosine similarity is a similarity measure where the cosine of an angle between two vectors created by the data points are the measure of similarity. The measure normalises the amplitude of the dimensions in the data points, making only the angle between the data points matter. Cosine similarity is often used in document similarity, as it is effective on large sparse vectors. The equation a n-dimensional cosine similarity can be seen in 2.11 and an illustration in Figure 2.9.

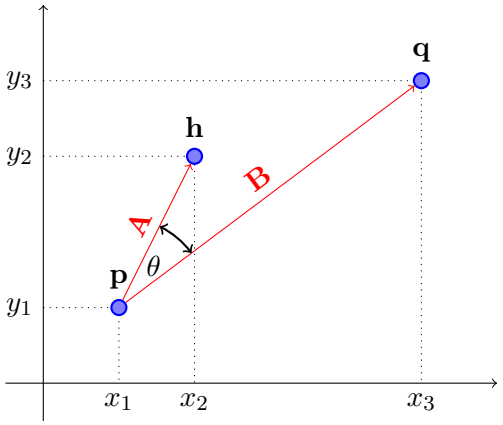


Figure 2.9.: Illustration of Cosine similarity

$$\text{Similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.11}$$

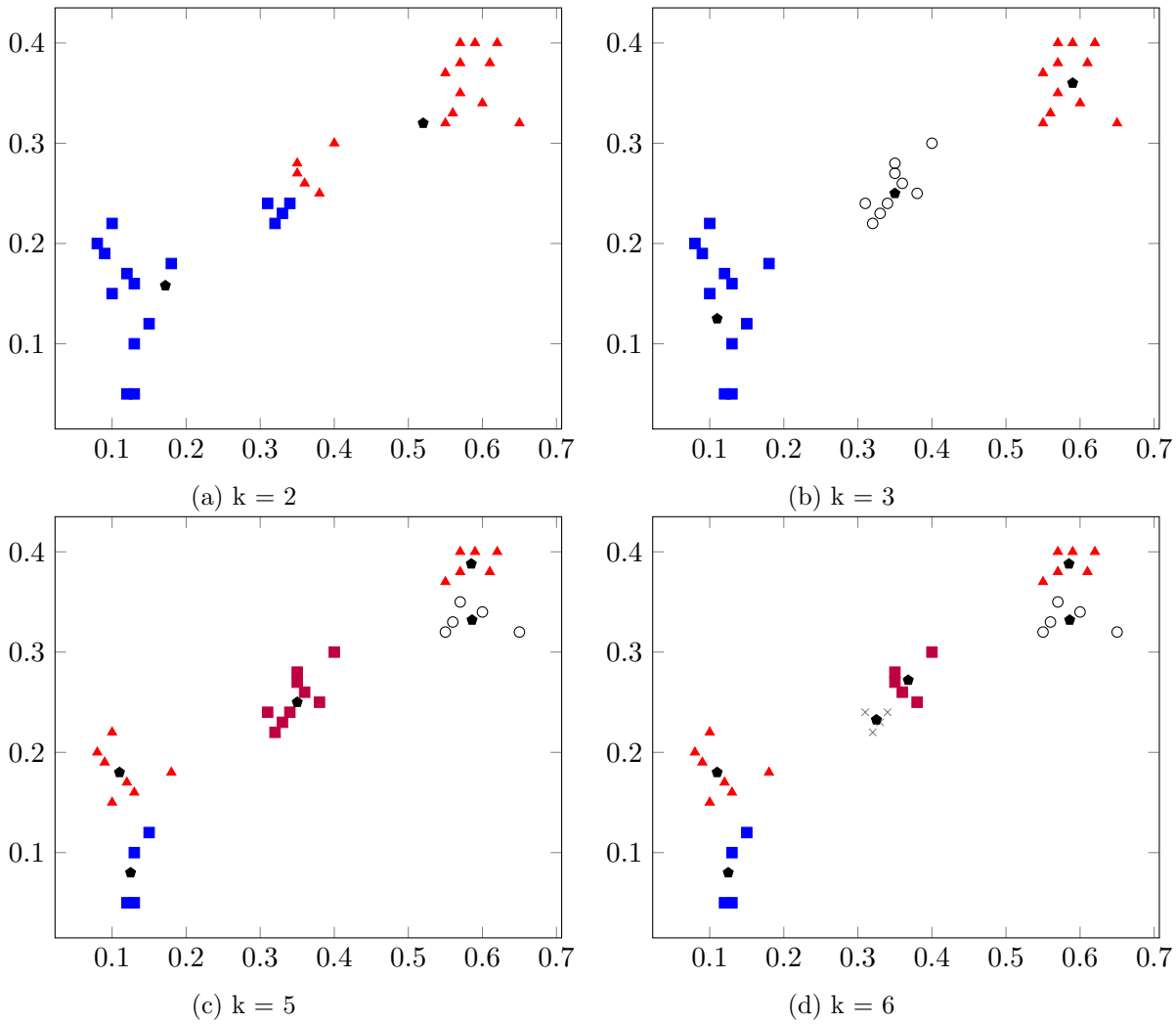
2.4.2. K-means Clustering

K-means clustering is a popular method of centroid-based clustering. It is based on dividing a dataset into $k \in \mathbb{N}$ different clusters, with k being a tuneable parameter. [15]

The algorithm starts by initialising k points within the data set. These prototype points can either be chosen randomly between the data points, or new points can be created within the range of the data. All data points are assigned to their nearest prototype point, creating the clusters.

These clusters are then improved by iteratively recalculating the prototype points and reassigning data points to their nearest cluster. The recalculation is done by moving the prototype points to the mean value of the points currently assigned to it. All data points are then again assigned to their closest prototype point. This is repeated until the prototype points recalculation yields the same results as last iteration. The algorithm can be seen in Algorithm 2.

In Figure 2.10, K-means with different k-values can be seen on the same dataset. K-values of 3, 5, and 6 all seem to fit the dataset, meaning there is no absolute correct k-value without additional information.



(a) k = 2 (b) k = 3

(c) k = 5 (d) k = 6

Figure 2.10.: Example of how k-means clusters a dataset with different k values. The centroid for each cluster is a black pentagon.

Algorithm 2: K-means clustering

Select K points as initial centroids
while *Centroids do not change* **do**
    Form K clusters by assigning each point to its closest centroid
    Recompute the centroid of each cluster

2.5. Dimensionality

The *curse of dimensionality*, first introduced by Bellman [16], is a common problem that often arises when dealing with data in high-dimensional spaces. One of the main problems is that when the dimensionality increases the volume also increases. This leads to the distance between each datapoint becoming so large that the data becomes increasingly sparse. There exists a lot of different techniques to mitigate this problem. Some of those methods are the principal component analysis (PCA) and partial least squares discriminate analysis (PSL-DA)

2.5.1. Principal Component Analysis

The definition of principal component analysis is given by Alexander N. Gorban [17] and is cited in Definition 8. The goal of PCA is to reduce the number of variables in the dataset while containing as much information as possible. To achieve this, PCA finds the variables that correlates the most and combine them to form a smaller set of variables that can describe the data.

Definition 8 *PCA is a data analysis technique that relies on a simple transformation of recorded observation, stored in a vector* $\mathbf{z} \in \mathcal{R}^N$*, to produce statistically independent score variables, stored in* $\mathbf{t} \in \mathcal{R}^N, n \leq N$*, see Equation 2.12.*

$$\mathbf{t} = \mathbf{P}^T\mathbf{z} \tag{2.12}$$

Here, P is a transformation matrix, constructed from orthonormal column vectors. [17]

$\mathbf{x_1}$	$\mathbf{x_2}$	...	$\mathbf{x_n}$
2	1.9	...	–
9	1.1	...	–
8	0.9	...	–
1	0.5	...	–
1	1.7	...	–
0	0	...	–
5	2	...	–
3	1	...	–
4	1.1	...	–
$avg = 3.67$	$avg = 1.13$	...	–

Table 2.2.: Original datapoints before normalisation.

$\mathbf{x_1}$	$\mathbf{x_2}$	...	$\mathbf{x_n}$
-1.67	0.77	...	–
5.33	-0.03	...	–
4.33	-0.23	...	–
-2.67	-0.63	...	–
-2.67	0.57	...	–
-3.36	-1.13	...	–
1.33	0.87	...	–
-0.67	-0.87	...	–
-0.33	-0.97	...	–
$avg = 0$	$avg = 0$	...	0

Table 2.3.: Datapoints after normalising them.

PCA has four main steps:

1. Normalize the data by subtracting the mean from each point. An example of this can be seen in Table 2.2 and Table 2.3

2. Calculate the covariance matrix for the data with Equation 2.13.

$$COV_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n-1} \tag{2.13}$$

3. Calculate the eigen values and eigen vectors for the covariance matrix. Then multiply the original data with the eigenvectors.

4. From the scores from the above step it is possible to find the variables that correlates the most. Drop the original datapoint and use the transformed data (eigen vectors) to find new patterns that shows the correlation between points.

2.5.2. Partial Least Squares Discriminant Analysis

Partial least squares discriminant analysis (PLS-DA) [18] has many different methods and usages. In this thesis, a short introduction to the classification technique will be given to the reader. This is a technique to determine what group a new sample has the highest likelihood to belong to. Other techniques that are related to PLS-DA will not be covered in this thesis.

PLS-DA tries to decide which of two groups a sample belongs to. It can be regarded as a linear two-class classifier. Figure 2.11 shows an example of this, one straight line that divides the data into two groups. The figure illustrates this by using two dimensions. When there is more than two dimensions PLS-DA represents the variables in hyperplanes in multidimensional space. PLS-DA is a continuation of the partial least squares regression (PLS-R) and is therefore build on the basics from PLS-R. That is, building a regression model between a data matrix (X) that represents a set of analytical measurements and a vector (C) that represents the labels for each sample. This is illustrated in Figure 2.12.

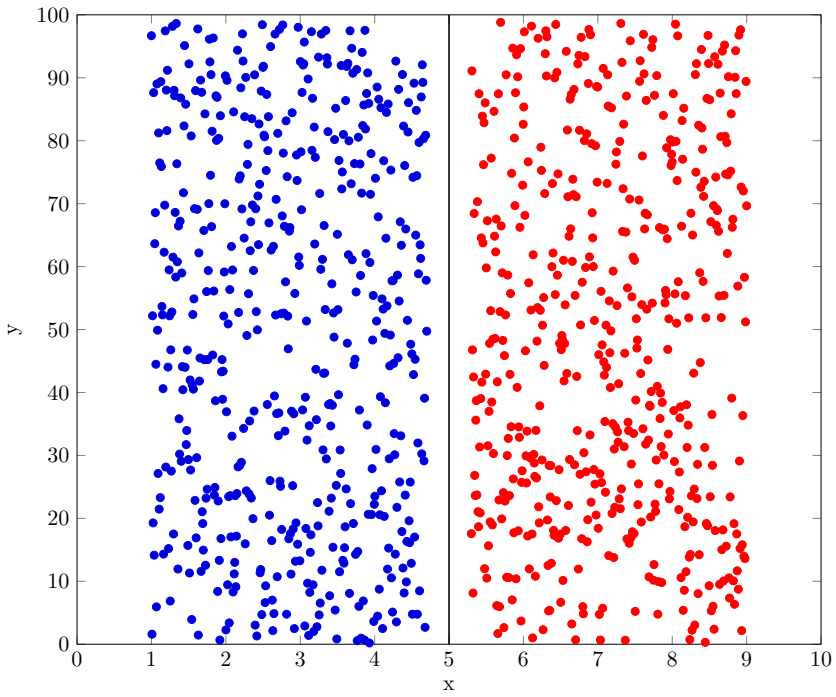


Figure 2.11.: Example on a linear two-class classifier.

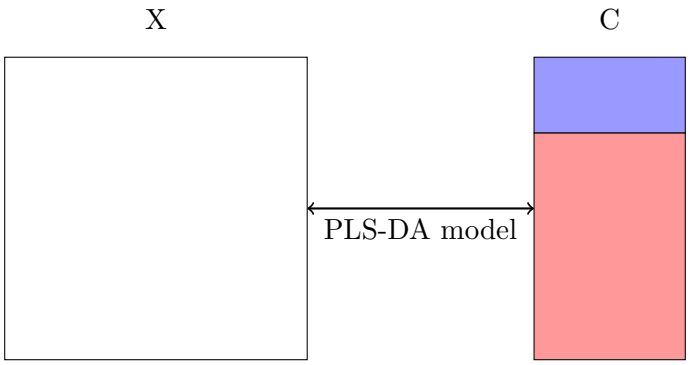


Figure 2.12.: PLS-DA regression

When the model is built it is possible to use it to predict data in the original data and future data points that arises.

PLS-DA is used for sharpening the separation between different groups in a dataset. This is done by applying the above method and maximise the separation among classes in the dataset and to gather information about which variables contains the necessary information.

2.6. Anomaly Detection

Anomaly detection consist of finding data that is different from some defined normal behaviour. It has many different applications and can be used in a lot of domains, such as malware detection, fraudulent email, credit card fraud, investments, healthcare diagnosis, and patient monitoring. An example of a point anomaly can be seen in Figure 2.13.

Definition 9 *Anomaly [19]*
Anomalies or outliers are substantial variations from the norm.

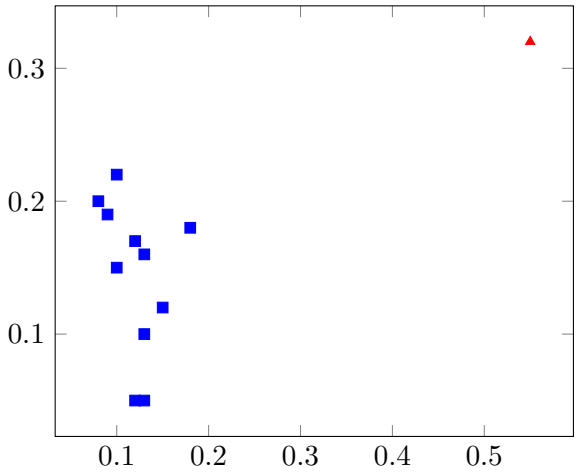


Figure 2.13.: Example of point anomaly. The red triangle is a point anomaly. It is significantly far away from the rest of the points. The blue points can be viewed as a cluster and the red triangle is therefore an anomaly.

One important problem in anomaly detection is differentiating between noise and anomalies. This is an important and challenging task, because in some cases anomalies and noise can be interpreted as the same thing. Take Figure 2.14 and 2.15, in the first figure the point a_1 and a_2 are anomalies in a given time series, but they can also be viewed as noise. If we treat it as noise, we end up with the data represented in Figure 2.15, which will give a completely different result then the data from Figure 2.14. Because of this it is important to choose an approach that can handle noise and differentiate between noise and anomalies.

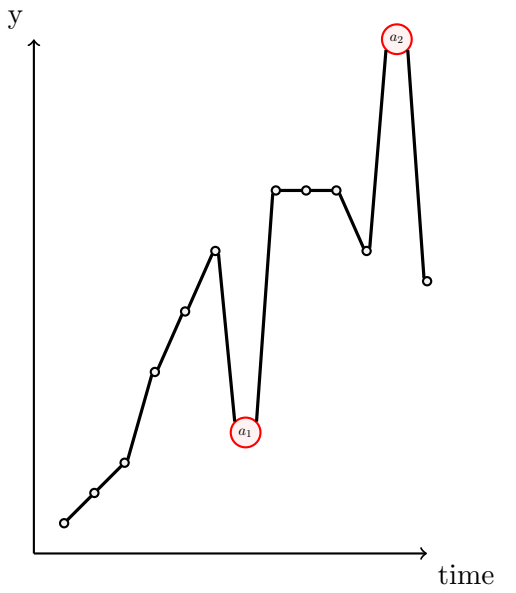


Figure 2.14.: Data with anomalies (a_1 & a_2).

Figure 2.15.: Removed the anomalies from Figure 2.14

In anomaly detection there are mainly three approaches: *distance-based*, *density-based* and *rank-based*. [19]

- *Distance-based*: Points that are farther from others are considered more anomalous. The points (a_1 & a_2) with red circles in Figure 2.13 can be an example of distance-based anomaly.
- *Density-based*: Points that are in relatively low density regions are considered more anomalous. The two blue squares in the bottom in Figure 2.13 can be seen as a cluster with high density. The red triangle can also be a cluster with only one element, or an anomaly.
- *Rank based*: The most anomalous points are those whose nearest neighbours have other points as nearest neighbours. See Example 2.

Example 2 *Rank based anomaly detection*
Imagine a social network where the people rank their friends from closest friends to acquaintances. You want to cluster this network to find circles of friends. To do this you analyse how the people rank one another as friend. Bob has four highly ranked friends, and all of Bobs friends has Bob and each other as their highest ranked friends. This makes Bob and his friends a cluster. However, if none of Bobs friends has him as one of their closest friends, it makes Bob an outlier.

When trying to choose an approach it is important to analyse the nature of the data. The nature of the data can be classified into three groups: *supervised*, *semi-supervised* or *unsupervised*.

Data that are supervised have known labels for a set of training data. This training data defines the comparison and distance for the rest of the dataset. Unsupervised data have no labels, so when calculating the distance or the comparison it needs to take the whole dataset into account. The final group is semi-supervised, where there are some known and some unknown labels. This is often the case in anomaly detection problems, where you have the labels for normal data, but no labels for abnormal data.

2.6.1. Anomaly Detection Algorithms

There has been done a lot of research in the anomaly detection domain and there are numerous different algorithms for detection anomalies. Some of the best known algorithms are *k-NN* and *Local outlier factor*. These algorithms have different strengths and weaknesses and gives good results in detecting anomalies.

k-NN

The k-NN algorithm determines outliers based on the distance to the neighbours of the data points. A point is labeled as an outlier if the distance from its neighbours is more than the rest of the set. The algorithm can either use only the distance to its kth neighbour $d_k(p)$ [20], or the sum of the distances to the k nearest neighbours $\sum_{i=1}^{k} d_i(p)$. [21]

One weakness of the k-NN algorithm is that the user must specify the number of the parameter k, which is the number of nearest neighbours. The output of the algorithm is a ranked list of anomalies, and a parameter n extracts the top n ranked results as outliers.

Local Outlier Factor

The local outlier factor (LOF) is an algorithm to find anomalies based on the local deviation of a data point with respect to its k nearest neighbours. [22] A point is an anomaly if the LOF is large, and the LOF is calculated with the following steps:

1. Calculate the distance to the kth nearest neighbour $d_k(p)$
2. Let the set of the k nearest neighbour be denoted by all points which are closer than the kth nearest neighbour. $N_k(p) = q \in D - p : d(p, q) <= d_k(p)$

3. Define the reachability distance of a point q from p as $d_{reach}(p,q) = max(d_k(p), d(p,q))$

4. Define the local reachability density lrd as the reciprocal of average reachability distance.
$$lrd_k(p) = (\frac{\sum_{q \in N(p)} d_{reach}(p,q)}{|N_k(p)|})^{-1}$$

5. The local outlier factor is then determined by comparing the lrd of all points in $N_k(p)$.
$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd(o)}{lrd(p)}}{|N_k(p)|}$$

A LOF of around 1 indicates a normal datapoint, while a LOF larger 1 indicates anomalies.

One-Class Support Vector Machine

One-Class support vector machine (OCSVM) [23] is a version of support vector machines that can be used for outlier detection. It has a methodology containing three steps:

1. In the first step, map the sampled data into a higher dimensional feature space.

2. After the data is mapped, they are enclosed in a geometric figure. To accomplish this, the different parameters of the figure needs to be determined. These parameters are calculated by solving a linear optimisation problem.

3. The last step is to find the global outliers. This is done by each node broadcasting its parameters to other nodes. In the end one node will receive parameters from all other nodes. This is known as the central node. The central node merges the node and computes the global parameter for the data. The merged parameters are then broadcasted from the central node to every other node in the network. Each node classifies themselves as an outlier or not, based on their distance from the central node.

Angle-Based Outlier Detection

Angle-based outlier detection (ABOD) [24] tries to solve the problem with a high number of dimensions. When mining high-dimensional data the distance between points becomes less important. ABOD uses instead the angles between points to compare them with each other. In Figure 2.16 a simple data set is illustrated and shows the main idea of ABOD. Points within a cluster have a high variance in their angle between other points in the same cluster, while outliers will have a small variance in angle to all points. The

outlier angle γ has a much lower angle than the rest of the angles within the cluster and is thereby an outlier.

The main advantage of ABOD is that it is parameter-free, however this increases the computational cost, which is $O(n^3)$.

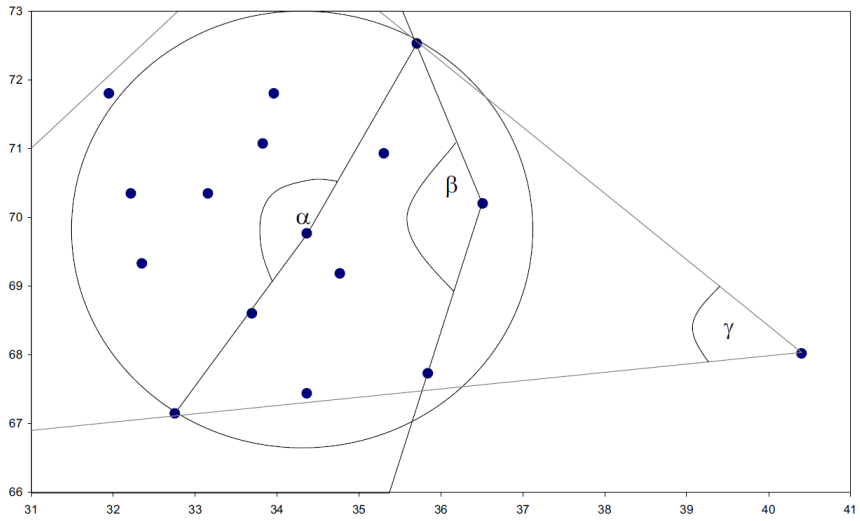


Figure 2.16.: Example of ABOD detecting an outlier.

Histogram-Based Outlier Scoring

Histogram-based outlier scoring (HBOS) [25] makes a univariate histogram for each dimension in the dataset. For numerical values two different methods are evaluated. (1) Static bin-width histograms or, (2) dynamic bin-width histograms. The first method is based on the standard histogram technique using k equal width bins. The frequency of samples that are seen in the same bin are used to estimate the density for that bin. The second method starts by sorting the values and grouping $\frac{N}{k}$ values into a single bin. k is the number of bins and N is the total number of instances.

Each dimension in the dataset have now an individual histogram where the height of each bin represents the density of data. The next step is to normalise the histogram in such a way that the maximum height is 1.0. Then the HBOS over every instance p is calculated using Equation 2.14.

$$HBOS(p) = \sum_{i=0}^{d} log(\frac{1}{hist(p)}) \tag{2.14}$$

Cluster-Based Local Outlier Factor

Cluster-Based Local Outlier Factor (CBLOF) [26] is a measure of both the size of the cluster that a given point belongs to and the distance between that point and its cluster. CBLOF is an outlier factor that gives a degree of a record's deviation. CBLOF starts with a clustering algorithm that partitions the dataset into clusters. It then divides these clusters into small and large clusters, and for every point the outlier factor is calculated. Small clusters are treated as outliers relative to the large clusters. These two steps, clustering and calculating the outlier factor, only need one scan of the dataset each. This gives CBLOF a linear complexity, so it handles large dataset well.

2.6.2. Ensemble Models

A technique that is getting traction within anomaly detection is ensemble models. When using ensemble models, a system based on the scores of multiple other outlier detection algorithms is created. There are two different architectures of ensemble models, *sequential ensembles* and *independent ensembles*.

In an *independent ensemble* you calculate the anomaly score from many different anomaly algorithms like *k-NN*, *LOF*, *OCSVM*, and others. You can also run the same algorithms with different parameters or subsets of the data. The output of the algorithms is then normalised and combined to one score. A *sequential ensemble* does the anomaly algorithms in sequence, making the output of one algorithm the input of another. The final output is then the score of the last anomaly algorithm, with no need to combine outputs.

Another distinction between ensemble models is *model-centred* and *data-centred* ensembles. A model-centred ensemble focuses on running different algorithms or parameters of algorithms on the same dataset. Data-centred ensembles runs the same algorithm on different dimensions or subsets of the dataset. Ensemble models can either be model-centred, data-centred, or a combination of both.

2.7. Evaluation Metrics

When evaluating a method, it is important to have some standardised measures. *Sensitivity* and *specificity* are two statistical measures that are used when evaluating binary classification. Sensitivity is given in Equation 2.15, and it gives a measure on the proportion of positives that are correctly classified. One example can be a measure on the percentage of how many people that have a given disease and have been classified with the disease. Specificity is given in Equation 2.16 and is a measure on how many negatives that are correctly classified. True positive, false positive, false negative and true negative are given in Table 2.4.

		True condition	
		Positive	**Negative**
Predicted condition	**Positive**	True positive	False positive
	Negative	False negative	True negative

Table 2.4.: Table of true positive, false positive, false negative and true negative.

$$Sensitivity = \frac{True\ positives}{True\ positives + False\ negatives} \tag{2.15}$$

$$Specificity = \frac{True\ negatives}{True\ negatives + False\ positives} \tag{2.16}$$

2.7.1. Receiver Operating Characteristic

Receiver operating characteristic (ROC) is a common way to analyse results in clinical research. It is used to describe the performance of a given method. A ROC curve is created by plotting the true positive rate (sensitivity) as a function of the false positive rate (1-specificity). [27]

ROC analysis is mostly used when the task is to differentiate two mutual exclusive conditions, like whether a disorder is present or absent in an infant.

ROC is also used as a tool to choose the optimal method and to discard methods that are less optimal. When the ROC curve is above the diagonal (the black line in Figure 2.17) it is an indicator that the method is able to separate the classes and not only guessing the predictions.

2.7.2. Area Under Curve

The area under the curve (AUC) is a measure for how good a method is to classify data correctly. An example of a AUC can be seen in Figure 2.18. The higher the AUC score is, the better the method can predict a 0 as a 0 and a 1 as a 1. The AUC score is calculated by taking the integral of the ROC curve, also known as the area under the ROC curve.

AUC scores goes from 0 to 1. A good method has a score near 1, which implies that the method has a good measure for separability. A bad method will have an AUC score closer to 0.5 and will perform badly at classifying data correctly. If the score is 0 it will predict negative classes as a positive class and vice versa perfectly. When the score is 0.5 it means that the method does not have any form of class separation and results are more about luck than anything else.

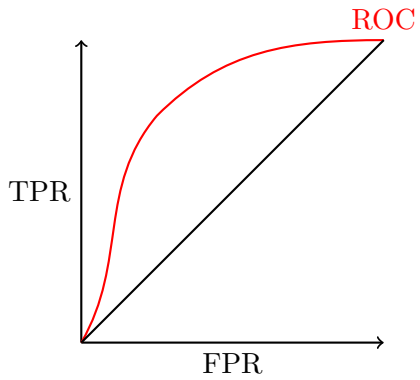


Figure 2.17.: Example of a ROC

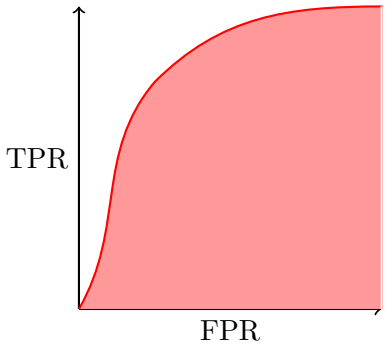


Figure 2.18.: Example of a ROC-AUC

3. Previous Work

In this chapter the reader will be introduced to previous work that has been done in related fields. The following topics will be covered in this chapter: earlier work on infant movement analysis, previous work done by the InMotion project, anomaly detection in human movement, anomaly detection in the frequency domain, classification of infant movement using ensemble learning, and state of the art ensemble models.

This thesis is a continuation of the specialisation project done by the authors in the autumn of 2019. The following sections in this chapter are gathered from that project: Section 3.1, 3.2 (without State of the Art), 3.3, 3.4 and 3.5.

3.1. Early Work on Video Based Infant Movement Analysis

In 2009, Adde et al. predicted CP by measuring motion in a video through comparing pixels on a frame by frame basis. [28] They analysed quantity of motion by observing what ratio of pixels changed between each frame. This metric, as well as the velocity and acceleration of motion, was used to predict CP.

The results from Adde et al. indicates that the features engineered in their paper are too simple to correctly model the movement patterns of an infant. With a target sensitivity of 81.5, the quantity of motion metric was only able to score a specificity of 44.4. This low level of specificity means that more than half of healthy infants gets diagnosed as impaired. Since 90% of infants in the risk group is healthy, a large number of infants get a false diagnosis with this test.

New methods in video tracking facilitates the possibility of engineering a more accurate model without adding too much complexity.

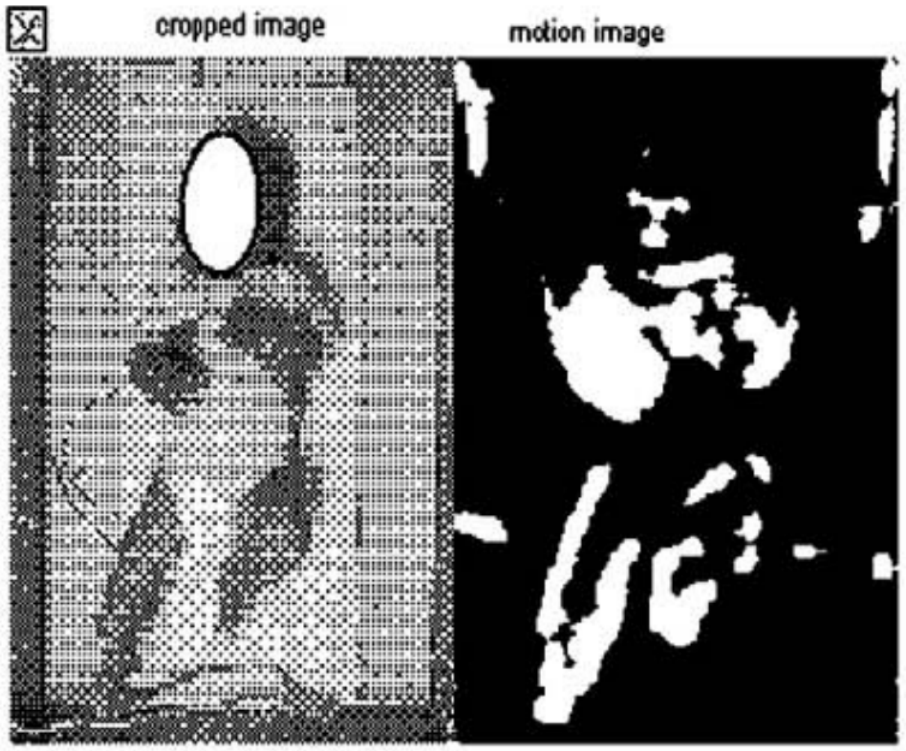


Figure 3.1.: Image representation of motion analysis by Adde et al. To the left is the cropped input image to the algorithm, and to the right is the changed pixels between two frames in the video. These pixels are used to calculate a quantity of motion, used to predict CP. Image from [28].

3.2. Automated General Movement Assessment Analysis in InMotion

In the recent years, multiple students at NTNU have worked on automating the process of general movement assessment analysis as their thesis.

In 2018, Groos and Aurlien used machine learning to extract skeleton data from videos of infants. [29] The infant was tracked in 7 different points, with an accuracy of about 1 cm. The results from this master is the CIMA dataset, which opened possibilities for other methods of classification than image analysis.

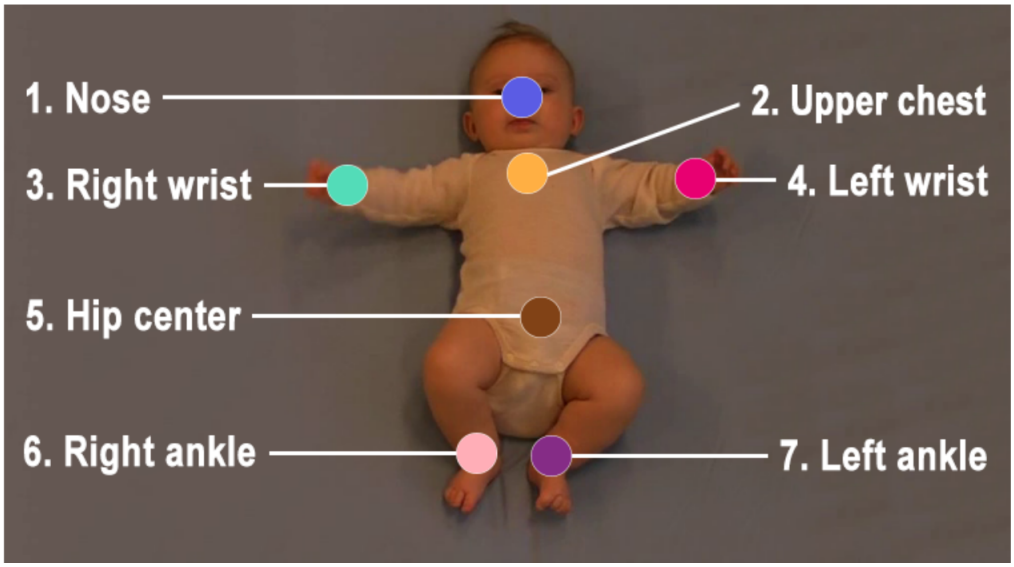


Figure 3.2.: Tracking points in CIMA. Image from [29].

In 2019, Wiik and Theisen tried to solve the prediction problem by training a convolutional neural network on the dataset generated by Groos and Aurlien. [30] They represented the data in 10 seconds intervals by tracking the movement of the points and representing them with lines of unique colours. The solution experienced a large number of false negatives, with some configurations failing to predict a single infant. Wiik and Theisen only had a dataset of 378 videos and discussed the possibility of the results coming from the network not being able to generalise the data on such a small dataset. They also discussed the difficulties of classification on an unbalanced dataset.

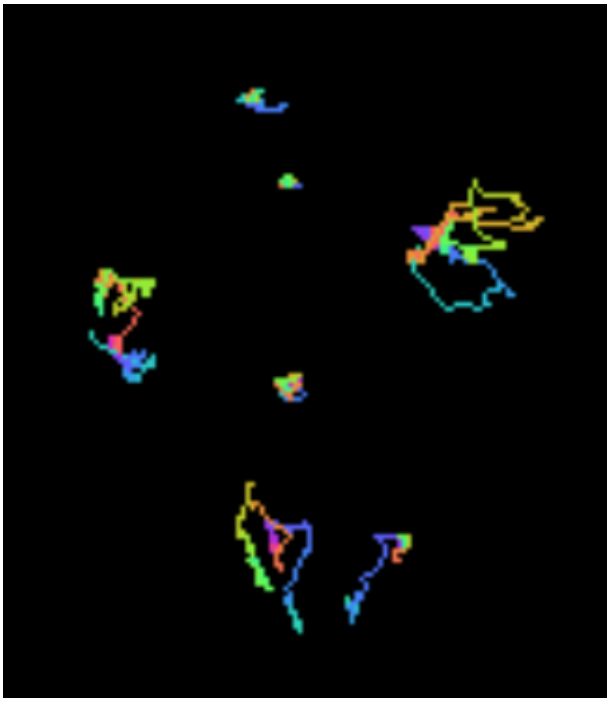


Figure 3.3.: Data representation in the method of Wiik and Theisen. Each line is 10 seconds of movement from one tracking point. Noise is cleaned by using clustering. Colour spectres are used to signify time in the movement. Image from [30].

State of the Art

In 2020 Espen A.F Ihlen et al. [31] presented a novel machine-learning model, the Computer-based Infant Movement Assessment (CIMA) model for predicting CP based on video recordings. The CIMA model works by finding a percentage of how much movement of an infant is predicted to be CP-related.

Espen et al. converted the x- and y-coordinates from the tracking data into time-frequency by using multivariate empirical mode decomposition (MEMD) and Hilbert Huang transformation. They divided the time-frequency data into period of 5 second each with no overlap. Each of the time windows were labelled with CP or non-CP according to the infants CP status. From this data they used partial least square regression with a backward feature selection to select the features. Then Espen et al. used a linear discriminative analysis to classify each of selected features as features that are commonly found in infants with CP. The CIMA model gives each 5 second window a label of 0 or 1 corresponding to the presence of CP. The final classification uses a threshold of 50% to decide the absence or presence of CP movement for an infant. This implies that over

50% of the 5 seconds windows for the infant need to be classified as CP to predict the presence of CP in an infant.

Espen et al. got a sensitivity of 92.7% and a specificity of 81.6% on their CIMA model. The ROC graph from the model can be seen in Figure 3.4. To the authors knowledge this is the state-of-the-art model for predicting CP in infants.

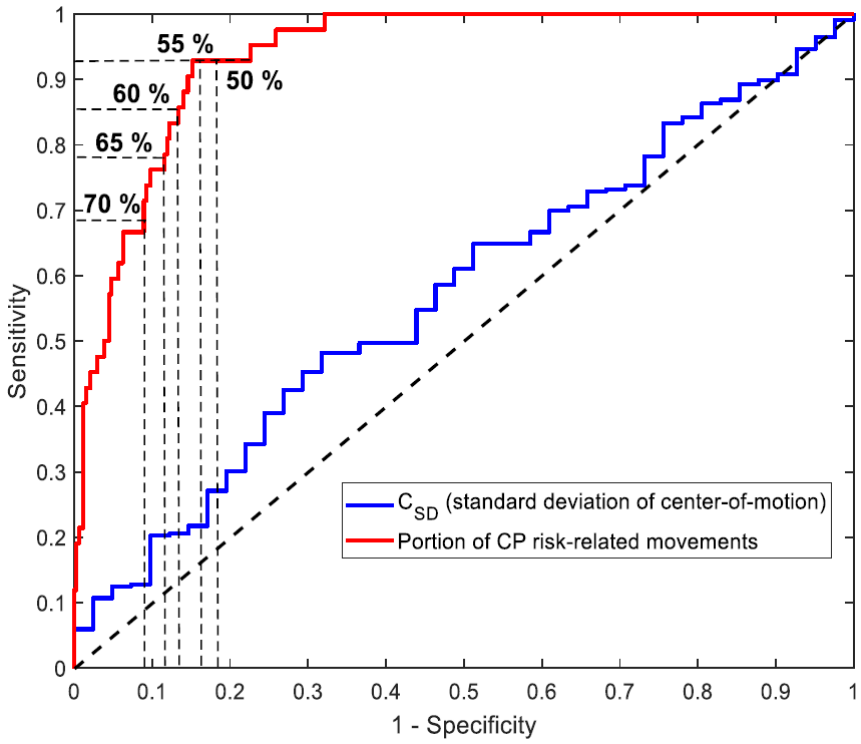


Figure 3.4.: The ROC graph from the CIMA model. Taken from [31].

3.3. Anomaly Detection in Human Movement

Not much research has been done on anomaly detection in human movement for medical diagnosis. In 2016, Ngyuen et al. used a hidden Markov model on 3D pose data collected from a Kinect to recognise abnormal gait. [32] They prepared the data by selecting angles in joints which described human posture. These postures were collected by clustering all angles at given times into clusters. The clusters were marked with a unique id, and a hidden Markov model was trained on the transition between the clusters. On a dataset containing healthy individuals and individuals with either Parkinson or stroke, the system was able to diagnose them from their gait with a specificity of 0.8 and sensitivity of 0.88.

This exact method cannot be used in the task of detecting anomalies in infant movement. It assumes that the movement happens in cycles, which is the case for gaits but not for fidgety movement in infants.

3.4. Anomaly Detection of Building Systems using Frequency Domains

In 2012, Wrinch et al. used frequency domain analysis to find anomalies in power usage of buildings with smart energy meters. [33] They used travelling time windows for frequency domain analysis with different window sizes. By doing this they could look at frequencies for energy usage in windows from 4 hours to 1 week.

They found that with different window sizes they could extract different features. The smaller windows were able to capture the lesser power usage while workers were out for lunch, while windows of larger length were able to capture the daily cycles of power usage. The anomalies in the frequency domains were found by experts, so no automated system was proposed.

The results from Wrinch et al. indicated that it is possible to use the same method of feature engineering by using different window sizes of the Fourier transform to capture different characteristics of infant movement.

3.5. Video-Based Early Cerebral Palsy Prediction using Motion Segmentation

In 2014 Hodjat Rahmati et al. [1] used a motion segmentation method for extracting motion data from videos of infants. They based their framework on [34] which is divided into three parts, dense trajectories, graph-based segmentation algorithm and a tracking algorithm.

The dense trajectory algorithm is used to track the entire body of the infants. Segmentation is then applied to distinguish the different body parts from each other. The different trajectories are divided into groups belonging to different body parts. This split is done with the use of a graph-cut optimisation algorithm. The output of this step is a label for each segment. The last step tracks the segmentation and outputs a single trajectory for each segment.

Classification was done using a support vector machine classifier and the result can be seen in Table 3.1. The results from this research shows a low sensitivity of 50%, but with a high specificity of 95%. Hodjat Rahmati et al. conclude that it is possible to make prediction of CP sub-types by analysing different body parts separately.

Data set	Sensitivity	Specificity	Accuracy
Motion segmentation	50%	95%	87%

Table 3.1.: Classification results for motion-segmentation data set from [1].

3.6. Image-Assisted Discrimination Method for Neurodevelopmental Disorders

In 2019 Xiaohui Dai et al. [35] used ensemble learning classification to classify motion patterns in infant's limbs. They used a Kernel Correlation Filter to track the movement of the limbs, and obtained the motion trajectory for each limb for the x- and y-axis. Xiaohiu Dai et al. found that difference between normal and abnormal infants was present in the y-axis, while no discriminative features was found in the x-axis.

Xiaohui Dai et al. used the discrete wavelet transform to transform the signal and keep the frequency and time information. Since the waveform has high and low-frequency parts they used low- and high-pass filters to differentiate them.

To get more information about the captured signal, Xiaohui Dai et al. looked at the power spectrum for the signal. The power spectrum was calculated by taking the square of the amplitude of the original signal.

For the ensemble model, they used a two-layer stacking model. For the first layer they used Support Vector Machine (SVM), Random forest (RF) and Adaboost. The second layer used XGBoost. To lessen the possibility of overfitting the training data, they used 4-fold for calculating the results for each model, taking the average of the 4-folds as the input for the next layer.

By combining the results from the wavelet and power spectrum they obtained a sensitivity of $(95.0 \pm 5.0)\%$, a specificity of $(91.7 \pm 6.3)\%$ and an accuracy of $(93.3 \pm 5.7)\%$.

3.7. LSCP: Locally Selective Combination in Parallel Outlier Ensembles

In 2019 Yue Zhao et al. [36] proposed a new framework for an unsupervised outlier ensemble, named LSCP. LSCP consist of four phases: base detector generation, pseudo ground truth generation, local region definition and model selection and combination.

In the first phase the algorithm generates a pool of base detectors. These base detectors are initialised with a range of different hyperparameters. The base detectors are trained on the training data before the results from each of the detectors are collected in a matrix and then normalised.

Since the ground truth for unsupervised outlier detection don't exists, LCSP generates a pseudo ground truth. LCSP uses Equation 3.1 and the training data to generate the pseudo ground truth for the detector selection. In Equation 3.1 ϕ represents the two different aggregation that is used, namely average (LSCP_A) and maximum (LSCP_M).

$$target = \phi(O(X_{train})) \in \mathcal{R}^{n \times 1} \tag{3.1}$$

The local region ψ is defined in Equation 3.2. The process is divided into three steps. The first step is to generate t random groups of $[\frac{d}{2}, d]$ features to construct a feature space. The second step is to find the k nearest object in each group using the Euclidean distance. The final step is to take all object that appear more than $\frac{t}{2}$ times and add them to $kNN_{ens}^{(j)}$, which defines the local region.

$$\psi_j = \{x_i | x_i \in X_{train}, x_i \in kNN_{ens}^{(j)}\} \tag{3.2}$$

Based on the pseudo ground truth LSCP uses the Pearson correlation to choose the best detectors. The detectors with the highest similarity are regarded as the best local detector. When the detectors are chosen LSCP has two methods for combining the results, namely maximum of average (LCSP_MOA) and average of maximum (LCSP_AOM).

The LCSP_MOA method is when the pseudo ground truth is generated by taking the average and LCSP takes the maximum of their prediction as the outlier score. Inverting this gives LCSP_AOM when the pseudo ground truth is generated by using maximum and then compute the average of the subsets.

The experiments done by Yue Zhao et al. showed that the LCSP_AOM achieves the highest score on 13 of the 20 datasets they benchmarked the model on. This combined with a runtime of $O(nd + nlog(n) + s)$ where n is test instance, d is dimensionality and s is the base detectors, makes LCSP a interesting choice for an ensemble model to test out.

3.8. XGBOD: Improving Supervised Outlier Detection

In 2019 Yue Zhao et al. [37] presented a new semi-supervised ensemble algorithm for outlier detection called Extreme gradient boosting outlier detection (XGBOD). The proposed framework is divided into three parts, unsupervised representation learning, transformed outlier score selection and prediction with the gradient boosting method XGBoost. For information on XGBoost, the reader can look at the work done by Chen and Guestrin [38].

In the first part the outlier scoring function is defined as a mapping function. Each scoring function gives an output in form of a real-value vector. This vector describes the degree of anomaly and is called a transformed outlier score (TOS). The outlier scoring function can be any scoring function that are used in unsupervised outlier detection. TOS are used as new features to change the original feature space, combining the scores from different scoring functions to create an outlier scoring matrix.

When the outlier scoring matrix is constructed, it can be combined with the original features from the dataset. Not every TOS in the matrix will contribute to the prediction in a positive way, so three methods for choosing TOS are presented: random selection, accurate selection, and balanced selection. When the TOS are selected the refined feature space is created by concatenating the original features with the TOS.

After the new refined feature space is created it is used as input in an XGBoost classifier to create the final output.

The testing that Yue Zhao et al. did shows that the XGBOD achieves improved results compared to other methods. XGBOD gets a higher score on predictions than other unsupervised models. It has good scalability because of how it handles the TOS. This, combined with it being a complete framework for unsupervised outlier detection with supervised machine learning, makes XGBOD a promising method for the future.

4. Method

In this chapter the reader will be introduced to the AnoMove method. The chapter goes through the data pipeline, starting with the raw input data and ending with a prediction. The AnoMove method is divided into three main parts that can be seen in Figure 4.1.

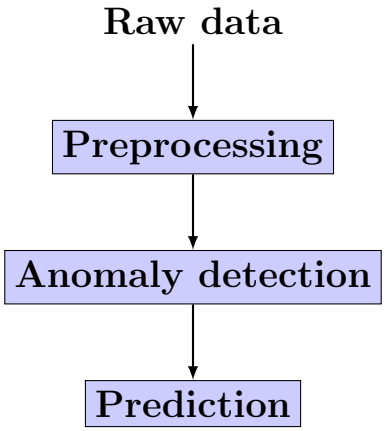


Figure 4.1.: A simplified overview of the three main building blocks for the AnoMove method.

- **Preprocessing:** Preprocess the data and make it prepared for the anomaly detection.
- **Anomaly detection:** Trains the chosen anomaly detection model.
- **Prediction:** Makes the prediction for each infant.

The first section introduces the larger building blocks and the data flow for the AnoMove method. Section 4.2 explains how the data is gathered and how we split the data into different sets. Preprocessing (Section 4.3) goes through every stage of the preprocessing and explains the different operations. Section 4.4 presents the anomaly detection models. Visualisations (Section 4.6) presents a tool developed to visualise the data and result of the analysis. The last Section (4.7) experiments on the accuracy of the preprocessing.

4.1. Overview of the AnoMove Method

It is important to understand the flow of the data and how the building blocks of the proposed method are connected. The rest of this chapter explains the AnoMove method. Each section explains one of three building blocks of the AnoMove method: preprocessing, anomaly detection, and prediction that can be seen in Figure 4.2. In addition to the three building blocks we have made a visualisation tool, which can also be seen in the figure. Each operation inside each block is further explained within their corresponding sections.

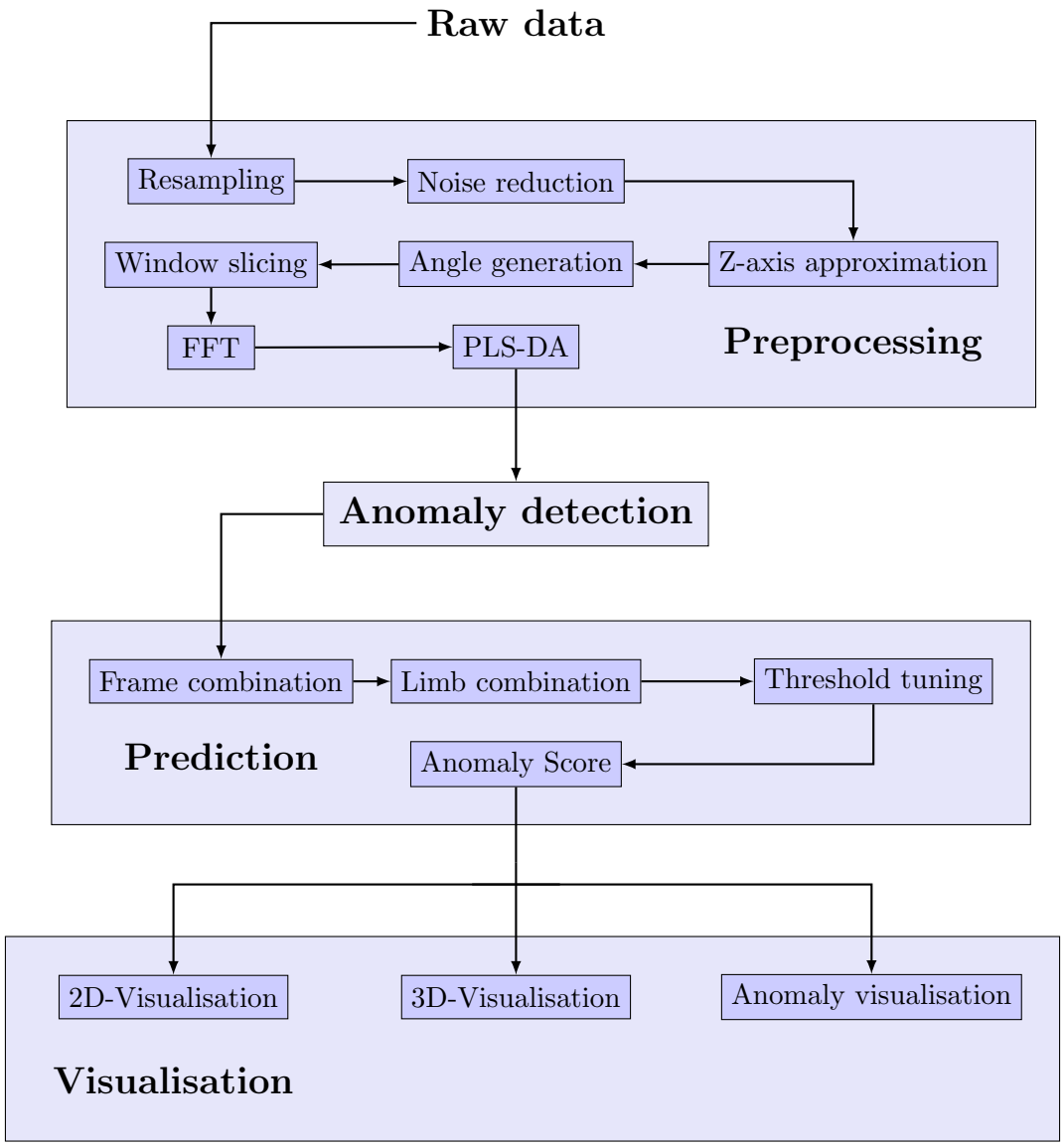


Figure 4.2.: Data flow through the three main building blocks of the AnoMove method. The visualisation is not a part of the prediction, but is a tool used to visualise both the data and results if needed.

4.2. Raw Data

The input data used in this method is a result of previous work by Groos, where he tracked 19 points in videos of infants using EfficientPose [39]. The points build a skeletal overlay over the infant in the video and a visualisation of the overlay can be seen in Table 4.1 and Figure 4.3. The points of interest are captured in x- and y-coordinates ranging from 0 to 1.

The process of video capturing infants is not standardised, and the varying distances from the infant to the camera will cause variations in the coordinates, as well as the relative size of the infants. This makes it hard to directly compare infants. We propose some means to solve these problems in our method. One of the ways is the use of angles. Using angles to represent the movement of infants helps normalise the data in the spatial dimension.

The first step in our method is to separate 20% of the data for use in a test set. The test set has the same distribution of healthy and impaired infants as the rest of the data.

Training Data

The training data is used for training our anomaly detection model and tune the parameters in the preprocessing step. This data is further split into 5 folds using k-fold cross validation during training.

Test Data

The test data is separated from the training data and has the same distribution. We use the test data to make sure that our method is generalisable.

Node	Body part
1	Top of head
2	Right ear
3	Nose
4	Left ear
5	Chin
6	Right shoulder
7	Thorax
8	Left shoulder
9	Right elbow
10	Left elbow
11	Right hand
12	Left hand
13	Right hip
14	Pelvis
15	Left hip
16	Right knee
17	Left knee
18	Right foot
19	Left foot

Table 4.1.: The tracked points in the CIMA dataset.

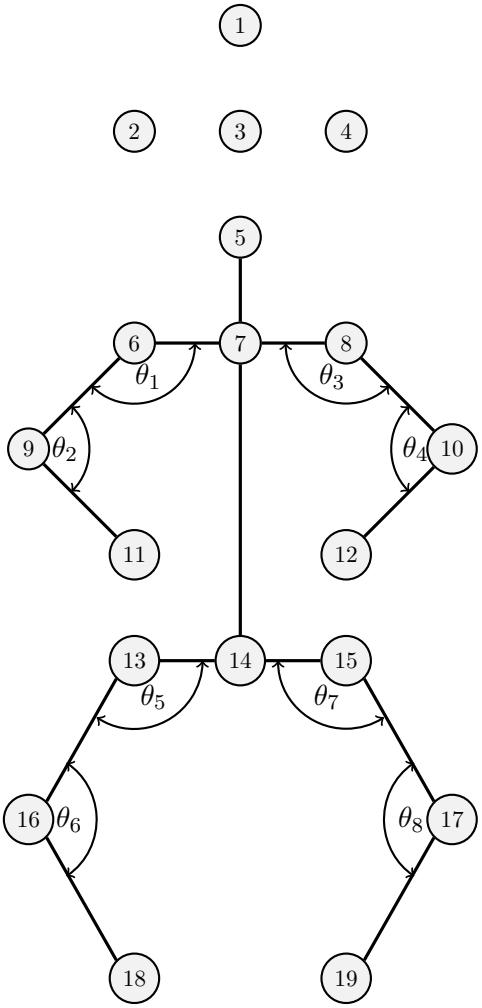


Figure 4.3.: Skeleton from the CIMA dataset with 19 tracked points

Angle	Description
θ_1	Angle between upper chest (7), right shoulder (6) right elbow (9)
θ_2	Angle between right shoulder (6), right elbow (9) and right hand (11)
θ_3	Angle between upper chest (7), left shoulder (8) left elbow (10)
θ_4	Angle between left shoulder (8), left elbow (10) and left hand (12)
θ_5	Angle between center hip (14), right hip (13) and right knee (16)
θ_6	Angle between right hip (13), right knee (16) and right foot (18)
θ_7	Angle between center hip (14), left hip (15) and left knee (17)
θ_8	Angle between left hip (15), left knee (17) and left foot (19)

Table 4.2.: Description of each angle in Figure 4.3. The numbers in the parenthesis represents the node number in the figure.

4.3. Preprocessing

The raw data will be represented as time-series data where we map the change of angles in joints in the skeleton. There are multiple ways to choose which angles to use in the analysis. We can either select all angles and risk using redundant data but ensure we do not prune away features, choose which angles to use in the analysis based on expert assessment, or an automated feature extraction technique. We chose to use all angles that represents limbs of the infant, seen in Table 4.2.

4.3.1. Resampling

The videos in the CIMA dataset were captured without a common standard for frame rates. This led to the videos being captured at 24, 29 and 30 frames per second. To facilitate direct comparison between infants, the videos had to be resampled to a standardised frame rate. The frame rate chosen is 30.

The resampling is done by a linear interpolation. It calculates the timing of the original frames using the frame rate and inserts the frame timings that must be interpolated for achieving the target frame rate. It then applies the interpolation using built in functions from the Pandas library [40] and discards the frame timings that are not a part of the target frame rate.

4.3.2. Noise Reduction

Two measures are made to ensure that the data does not contain noise that can inhibit the performance of our method.

The first problem stems from the inaccuracy in tracking specific points in the infant. It is almost impossible to correctly track points on large body parts (e.g. the thorax) with pixel perfect accuracy. This leads to our data containing noise, where the points vibrate randomly around the point from frame to frame.

To smooth out this movement we use a simple moving average (SMA), which sets each value equal to the average of the preceding and succeeding values. The window-size of the moving average is set to be 3.

The second problem comes from moments where the tracker fails to correctly track a point for a single frame, which is deemed acceptable by the creators of the tracker. Such an error would however greatly affect the Fourier transformation, so we use a peak detector from the signal-processing package in SciPy [41] to detect such errors and set them to the average of the preceding and succeeding value.

A numerical experiment has been conducted on our noise reduction and processing, and is further explained in Section 4.7.

4.3.3. Z-axis Approximation

Since the source of the data is in three dimensions while our data is in two, just calculating the angles in joints can introduce a large amount of misrepresentation.

To demonstrate the problem, consider the following example.

Example 3 *Imagine an infant laying on its back, being filmed from above. If you want to analyse the angle generated by his elbow, you can represent it as the angle between two vectors, one being from his elbow to his shoulder, and the other from his elbow to his wrist. In a scenario where the arm is laying close to the ground, the length of the vectors will equal the length of the limbs of the infant. A small movement in the wrist will be translated to a small change in the angle of the elbow.*

However, imagine the same infant, but this time with his elbow and shoulder close to the ground, but his wrist high in the air. From above, the vector between his wrist and elbow now has a much shorter length than the length of the limb. It also causes the same small movement in the wrist to now translate to a large change in the angle we are tracking.

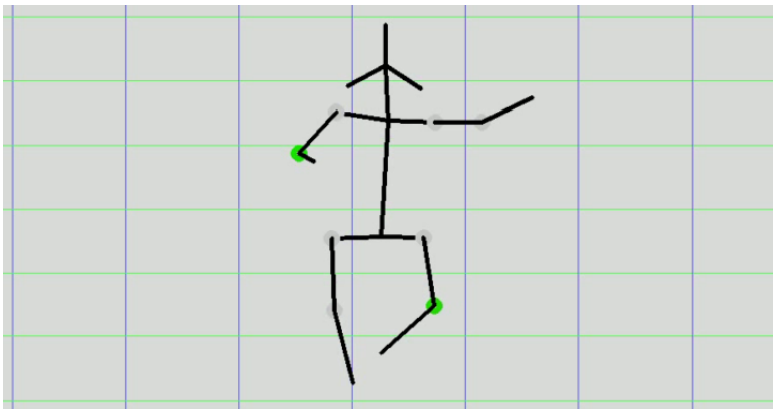


Figure 4.4.: Infant laying on its back. The left hand is stretched out along the floor, so a small movement in the wrist in the y axis will lead to a small change in the left elbow angle. However, the right hand has the wrist stretched toward the camera, so a small movement in the y axis in the right wrist will lead to a large change in the right elbow angle. Screen capture taken from the AnoMove visualisation, with colours representing healthy movements.

To decrease the impact of the problem mentioned above, we have approximated the

z-value of some of the tracking points in the infant. Our method assumes that the infant lies on its back, with a stationary camera perpendicular to the surface, removing perspective issues caused by some limbs being closer to the camera than others.

With these assumptions, it is possible to approximate the third dimension by using the difference of vector-lengths in the data. The length of the limbs of infants is constant throughout the video, and all difference must come from movement in the unseen dimension. The vector length in three dimensions is given by Equation 4.1.

$$|v| = \sqrt{\delta x^2 + \delta y^2 + \delta z^2} \tag{4.1}$$

We find $|v|$ for every limb by finding the maximum length of the vector in two dimensions. This should occur when $z = 0$. This $|v|$ is constant throughout the video, and δx and δy can be measured directly.

$$\delta z = \pm\sqrt{|v|^2 - \delta x^2 - \delta y^2} \tag{4.2}$$

This means that we can calculate δz, i.e. the change in the third spatial dimension with Equation 4.2. As long as we only analyse one limb at the time, the sign of the δz does not matter, as only the differences from frame to frame affects the Fourier transform.

By using the thorax as an absolute zero, we can cascade this calculation for every limb we expect to move under the assumptions. The tree that the cascading forms can be seen in Figure 4.5.

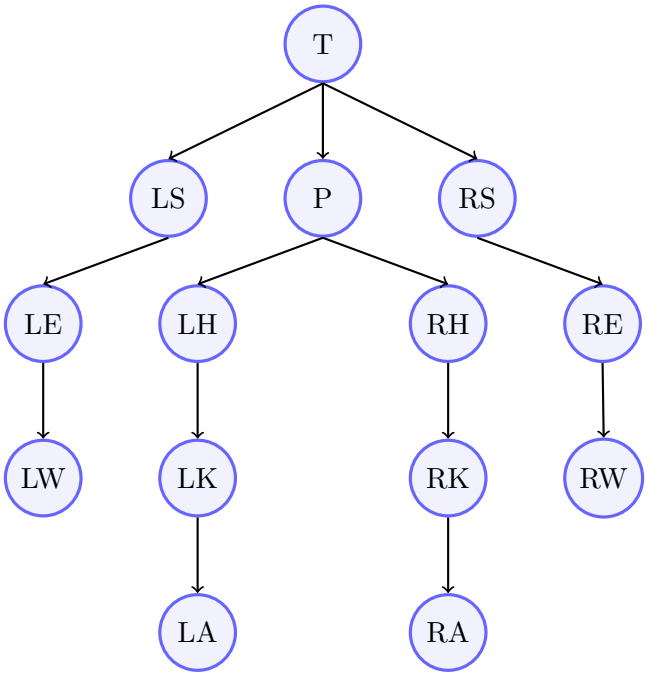


Figure 4.5.: Infant represented as a tree. The thorax acts as the root of the tree, and the reference point for z differences. The difference in z value is calculated for every edge in the tree. The names that corresponds to the nodes can be seen in Table 4.3

Node name	Abbreviation
Thorax	T
Left shoulder	LS
Left elbow	LE
Left hip	LH
Left wrist	LW
Left knee	LK
Left ankle	LA

Node name	Abbreviation
Pelvis	P
Right shoulder	RS
Right hip	RH
Right elbow	RE
Right knee	RK
Right wrist	RW
Right ankle	RA

Table 4.3.: The corresponding node name and their short abbreviation.

The result of this approximation reduces the misrepresentation of the data introduced by the missing dimension. However, it is unknown what happens when the assumptions that are made are broken, for example if an infant lies on its side instead of on its back.

4.3.4. Angle Generation

By representing the infant as a set of angles, it should be easier to directly compare infants, as spatial differences in the frame is ignored. The angles chosen will be the input for the Fourier transformation and later the anomaly detection model.

The angles are generated by calculating the inner angle between the two three-dimensional vectors that connects to the joint using Equation 4.3. This calculation is done for every angle in every frame.

$$\cos\alpha = \frac{\bar{a} \cdot \bar{b}}{|\bar{a}||\bar{b}|} \tag{4.3}$$

4.3.5. Window Slicing

To conserve the temporal data of infant movement, we choose to split the data into different window sizes in time before applying the Fourier transformation. By choosing different window sizes, we hope to capture features both in short bursts of movement, as well as trends over longer time frames.

Fourier transformation can be sensitive to where we choose to slice the movement. The same movement pattern with slightly different start- and end-times can result in very different frequency data. To improve this, we have the option to have overlapping window slices to increase the number of window slices and reduce the chance that we miss features by having bad start-frames. The amount of overlap will greatly increase the time it takes to process the data and train the models. Having a window overlap of two will also multiply the amount of data by two. The overlap is created by starting a new window every $\frac{\text{window size}}{\text{overlap}}$ frames. Window overlaps of 1, 2, and 4 have been tested.

4.3.6. Fourier Transformation

After the angle generation, the data is transformed using Fast Fourier Transformation, giving windows of frequency domains for each joint's movements in short intervals. The chosen window sizes are varying powers of two, as the algorithm performs best on data sizes of powers of two. The candidate window sizes can be seen in Table 4.4.

After the data is sliced into time windows, the mean value is reduced to zero. This is done to remove the zero-frequency of the Fourier-transform, so we can compare different frequency data more easily.

Frames	Seconds
128	4.26
256	8.53
512	17.07
1024	34.14

Table 4.4.: Candidate window sizes. The Fast Fourier Transformation (FFT) algorithm has the best performance when the input size is a power of two.

4.3.7. PLS-DA

When running the Fourier transform, the output of the transform has a granularity equal to the framerate of which the data was captured. This means that the granularity for our dataset is 1/30hz. The amount of data we can extract from each window is given by the Nyquist theorem (Equation 2.3). This equals half the window size of the Fourier transform. This gives the input data for the anomaly models dimensions between 63 and 511 depending on the window size, discarding the 0 frequency.

Following the *curse of dimensionality*, many models suffer from overfitting when the number of dimensions become large. Many models are also computationally dependent on the number of dimensions of its input data, making it beneficial to keep the number of dimensions minimal.

To reduce the number of dimensions we used partial least squares discriminant analysis (PLS-DA), which helps us construct new dimensions designed to maximise the difference between the healthy and impaired. We tested different numbers of dimensions to keep, ranging from 3 to 10.

4.3.8. Preprocessing Parameter Search

In many of the above-mentioned operations, different parameters will lead to vastly different results, without us knowing which would be most effective at capturing the features that differentiates healthy from impaired. To study the effects of the parameters, we chose to do an exhaustive search on the parameters to find the best combination.

This was done by doing a grid search on the parameters with values we found relevant. The values had to be chosen carefully and kept to a minimum, as each added value increased the amount of combinations multiplicatively.

Important parameters to tune include the amount of movement we require in a time window for it to be evaluated. If the threshold is too high, insufficient data will be

extracted from an infant to do meaningful evaluation. However, if the threshold is too low, the results from the predictor can be drowned out by data where no significant movement is present.

Another important parameter is the number of dimensions that are extracted from the frequency data. The more frequencies we extract, the more of the original features that might be present will persist. However, the performance of the outlier algorithms might suffer both in performance and results with a too high number of dimensions.

We tested the parameters on simple outlier algorithms, to focus on the difference in the preprocessing, as well as to reduce the computation time. There could be a chance that more advanced methods compensate for lack in preprocessing with internal procedures, which would suppress the results from the preprocessing and lead the advanced models to overfit.

We tested every permutation of the parameters and their values in Table 4.5. During the parameter search we used a stratified k-fold with 5 folds to cross validate our results. The result from this search can be found in Section 5.1.

Parameter	**Tested values**
Minimal movement	0.25, 0.5, 0.75, 1.0
Moving average (SMA)	3, 5
Window overlap	1, 2, 4
PLS-DA	3, 5, 10

Table 4.5.: Parameters and test values for the preprocessing step.

4.4. Anomaly Detection

The number of anomaly algorithms is vast, and they all have different pros and cons. Some of the different algorithms that were tested are used as base estimators for the ensemble model.

4.4.1. Basic Anomaly Algorithms

For the basic anomaly detection algorithms, we tested the following:

- Angle-based outlier detection (ABOD)
- Cluster-Based Local Outlier Factor (CBLOF)
- Histogram-based outlier scoring (HBOS)
- k-NN
- Local outlier factor (LOF)
- Isolation forest (IForest)
- One-Class support vector machine (OCSVM)

All these algorithms are implemented in a Python library called pyod [42]. Since all these algorithms are unsupervised, they function by finding some subset of the data that differs from the norm, called the contamination ratio. About 10% of the infants in our data is impaired, but since not necessarily all impaired infants' movement differ from the norm, we set the contamination ratio to 5%.

These models were tested both with different parameters for the models as well as all the parameters from the preprocessing search using stratified k-fold and the results for each of them can be seen in Chapter 5. For more information about the different algorithms, see Section 2.6.1.

4.4.2. Ensemble Algorithms

Several different ensemble models were tested on the dataset. All ensembles have been taken from the combo library [43], written by the same person behind pyod.

Aggregating Ensemble Models

For some different global ensemble models, we used the SimpleClassifierAggregator from the combo library. We used this to aggregate scores from the base estimators mentioned in Section 4.4.1. The methods used for aggregating results were maximisation and median. The aggregation happened across all the models, and each model used all the data. This was tested as the naivest ensemble for our method.

LSCP

The first advanced ensemble we tested was Locally Selective Combination (LSCP). It works by separating the data into regions, and finding the best base estimators for that region. We expected this model could be effective for our data, as the paper by Xiaohui Dai et al. [35] proved that the movement by impaired infants could be found in specific regions of the data when processed with a wavelet analysis.

XGBOD

The second advanced ensemble we tested was the Extreme Gradient Boosting Outlier Detection (XGBOD) algorithm. It is a supervised outlier ensemble model based on the popular gradient boosting library XGBoost. It combines the original features with outlier scores from base estimators as an input to the gradient boosting. The base estimators are chosen to maintain both diversity and accuracy.

These ensemble models were tested using the same method as the basic anomaly detectors. However, since the ensemble methods have a much higher run time than their basic counterparts, we chose to run a subset of the preprocessing parameters in the ensemble testing. The chosen parameters can be found in Table 4.6.

Parameter	Tested values
Minimal movement	0.5, 0.75
Moving average (SMA)	3
Window overlap	1, 2, 4
PLS-DA	3, 5, 10

Table 4.6.: Parameters and test values for the ensemble model testing.

4.5. Prediction

As the data of infants are processed by the anomaly models, each frame and joint are processed separately. This is useful for visualisation purposes, but to be able to make a prediction for an entire infant as well as evaluate the method, the scores should be combined to a single score for each infant. Several different approaches for combining the scores has been tested.

We wanted this scoring method to be naive, since using a trained model could easily lead to an overfitted model, as the dataset is rather limited.

The anomaly model we used returns the probability of a single joint in a single frame coming from abnormal movement. Frames without enough movement gets a score of zero and is ignored in further processing. To combine the outlier scores to a single score, we tried different approaches to combining the scores. One thing we tried was the combinations of taking the average and maximum of the different set of scores.

4.5.1. Frame Combination

First, we had to combine each score the model gave for each frame. Since we used overlapping windows of frequencies as well as different window sizes, each frame could have from 0 to overlap $*$ window sizes scores depending on how much was filtered away because of low movement. We combined this using either an average or maximum of the scores.

We also tested different combinations of window sizes to see which performed the best.

4.5.2. Limb Combination

We then combined every frame for each body part we evaluated, giving a score for each body part. This was also done by taking either the average or maximum of every frame.

As well as creating one score through averages and maximums, we tried to calculate how many of the frames in the video went over a certain anomaly threshold value. This ratio was the score for each limb.

4.5.3. Threshold Tuning

When using thresholds ratios to combine scores across the video frames, the outlier score set as the threshold will have a large effect on the final ratio score. To find the outlier

score that serves best as a threshold value, we tested every value between 0 and 1 with increment of 0.01.

4.5.4. Anomaly Score

The final anomaly score for an infant is created using either the maximum or the average of the limb scores.

A cross-validation parameter search is done is this thesis to compare the impact of the different methods, and to find the combination of parameters that gives the best results. Every permutation of the values in Table 4.7 are tested to find the combination that gives the best prediction.

The result from the parameter search can be found in Section 5.3.

Window	Window size	Angle	Body part	Threshold
Mean, Max	128, 256, 512, 1024	Mean, max, threshold	Mean, Max	0 - 1

Table 4.7.: Every permutation of the values was tested to find the best solution for the prediction.

4.6. Visualisations

To make the system usable for clinical evaluations, a tool for visualising the result is necessary. By design, our method gives individual scores for each joint it processes in each frame. By visualising this, a doctor would be able to use the system as an initial screening, and from there inspect the movement where the model found anomalies. This makes the system a tool for doctors, instead of a replacement for their analysis.

As well as having a use for doctors, visualisations are also a helpful tool for debugging and understanding the method.

4.6.1. 2D Visualisation

The 2D visualisation is done by making a skeleton overlay of the data. This represents the infant the same way that the raw data from the video does, with only the tracking points visualised. By colouring the different joints in a range from green to red, we can visualise how abnormal the movement in a joint and frame is. The visualisations can then be stitched to a video, giving a real time visualisation of the infant with our predictions.

4.6.2. 3D Visualisation

Since we approximate the z-values of the tracking points from the video, a visualisation of that can be helpful to understand where the approximation might fail.

We do the visualisation using an isomorphic projection of the data. This allows us to set the viewpoint so that the z-values are visible. The chosen viewpoint was rotated 15 degrees around the y-axis up from surface level, and 45 degrees anti-clockwise around the z-axis.

As the 2D visualisation, the frames of the 3D visualisation are stitched together to a video, so that it can be viewed in real time.

A snapshot from the visualisation can be seen in Figure 4.6. Here an infant has been evaluated by the method, and the visualisation video has been generated.

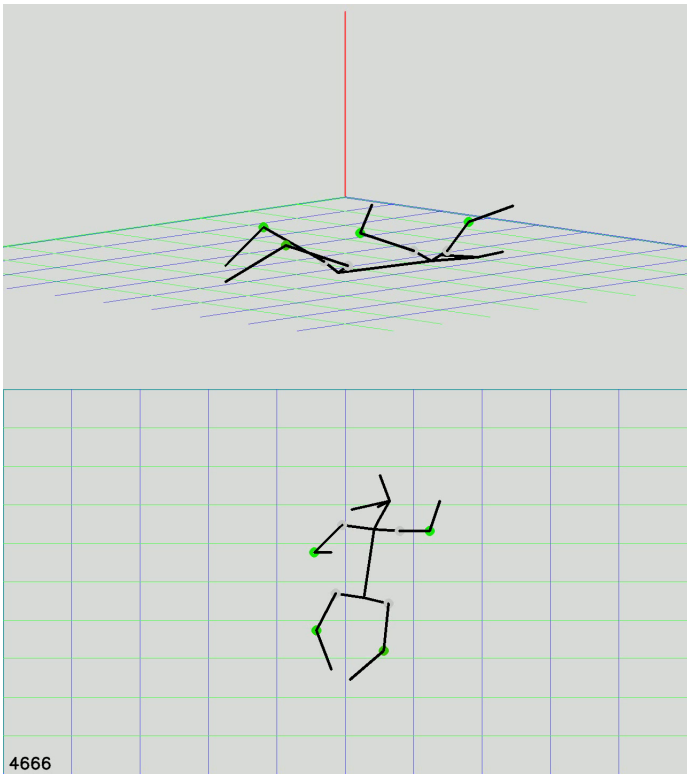


Figure 4.6.: Snapshot from the visualisation of predicted infants. The green dots on its joints represents movement that the method finds healthy. The bottom representation is a one-to-one representation of the two-dimensional input data, while the top representation is an isomorphic projection of the approximated three-dimensional data.

4.7. Ad-Hoc Error Analysis

To evaluate how noise in the input data affects the preprocessing of the infants, we did a numerical experiment with artificial noise added to the raw data. In this experiment we picked one infant that served as the ground truth, as if the tracking on that infant was perfect. We then added a Gaussian distribution on top of the data to serve as artificial noise. The noise was added to both the x and y axis independently.

We chose our value for the Gaussian distribution based on the work by Groos et al. [39] in making EfficientPose, the network that is used to generate our data. Analysis of their tracking method found that about 90% of tracked points was within 50% of the vertical head size, and about 35% was within 10%. In the infant we used as ground truth the vertical head size was 0.12, giving 0.012 as the 10% threshold. Since about 68% of data fall within one standard deviation in a Gaussian distribution, we found the 20% threshold to be a conservative value to use as the standard deviation, so the chosen value was 0.024.

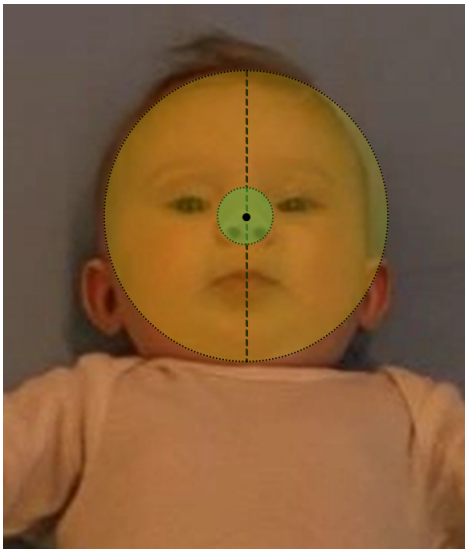


Figure 4.7.: Key point accuracy illustrated on an infant. The chosen standard deviation for our analysis is double the diameter of the green circle. Image courtesy of Daniel Groos.

We created 100 infants with artificial noise to normalise the randomness. This was done since the z-axis approximation uses the max length of the vectors it approximates the z value for, and this is calculated only once per infant. We then isolated the noise in both the x- and z-axis, as well as the angles. This was done by comparing the results from the ground-truth infant with the results from the same infant with added noise. The results from our analysis can be found in Section 5.5.

5. Results

This chapter will present the results from each step done in the proposed method. There are four steps that have results, namely preprocessing, anomaly detection, prediction and the error analysis. In the preprocessing section, the results from the parameter search will be presented. Section 5.2 shows the results from the anomaly detection part. Each base model and ensemble model will be presented. Section 5.3 presents the results from the prediction step. The next section (Testing) presents the results from the testing of the AnoMove method. The last section presents the results from the error analysis done on in the preprocessing step. All the results are sorted by area under curve score. The full tables with all parameters can be seen in Appendix A, Results.

5.1. Preprocessing

The results from the preprocessing are from tuning the different parameters. There are four parameters that have been tuned, *minimal movement*, *moving average*, *window overlap* and *PLS-DA components*. The tuning was executed by using the base estimators to create a baseline for the performance. The different values for the parameters that have been tested are shown in Table 5.1. In Table 5.2 are the parameters for the base estimators that were used. The results presented in this section are from the highlighted step in Figure 5.1.

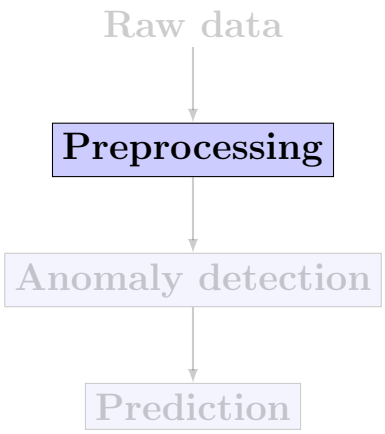


Figure 5.1.: A simple overview of the proposed method with the preprocessing step highlighted.

Parameter	Tested values	Chosen value
Minimal movement	0.25, 0.5, 0.75, 1.0	0.5
Moving average (SMA)	3, 5	3
Window overlap	1, 2, 4	4
PLS-DA	3, 5, 10	3

Table 5.1.: The four parameters that have been tuned in the preprocessing step, with their corresponding tested and chosen values.

Model	Parameter
ABOD	Neighbors = 3, 5, 10, 20
CBLOF	Clusters = 8, 10, 12, 14, 16, 18, 20
HBOS	Bins = 3, 5, 7, 9, 12, 15, 20, 25, 30, 50
IForest	Estimators = 10, 20, 50, 70, 100, 150, 200, 250
KNN	Neighbors = 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
LOF	Neighbors = 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
OCSVM	nu = 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99

Table 5.2.: The different base models with their corresponding parameters that were used when tuning the parameters in Table 5.1 and their values.

5.1.1. Moving Average

The two results from the moving average tuning are shown in Table 5.3. The SMA tuning was done in the noise reduction step, highlighted in Figure 5.2. Figure 5.3 illustrates the effect of SMA on a real subset of our data.

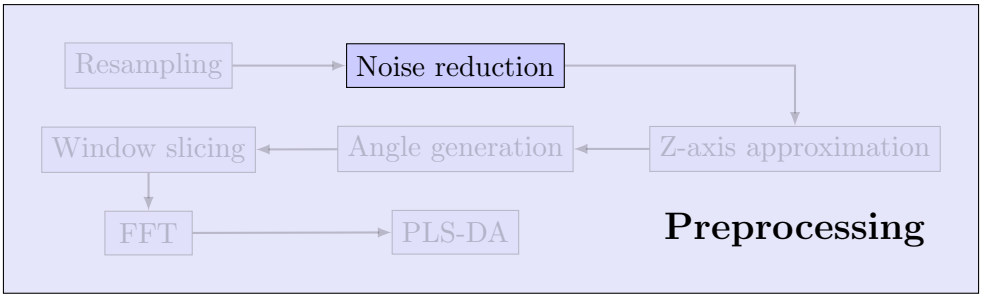


Figure 5.2.: Data flow through the preprocessing step with noise reduction step highlighted.

Model	SMA	Sens	Spec	AUC
LOF	3	0.120	0.954	**0.634**
LOF	5	0.122	0.955	0.622

Table 5.3.: The best models when sorted by AUC for SMA.

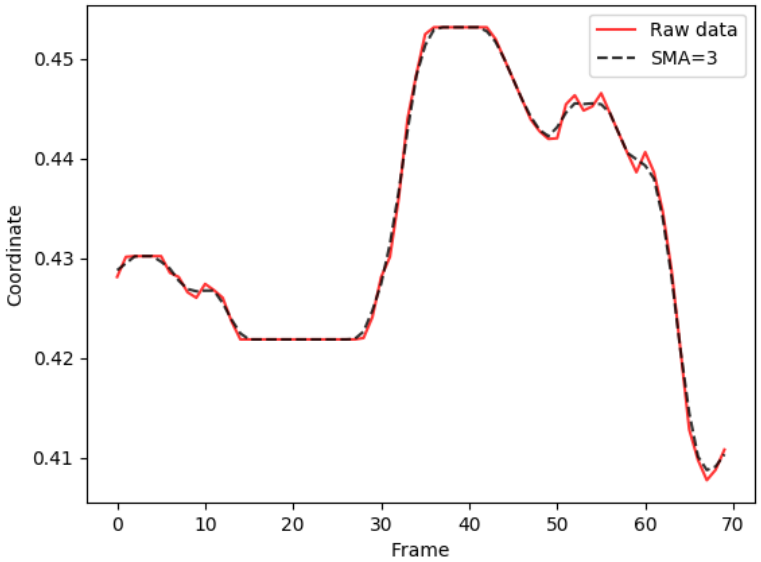


Figure 5.3.: Before and after the SMA operation for the x values for the right wrist of a randomly selected infant. The smoothing can be seen when the raw data oscillates after a sudden change, or randomly while still.

5.1.2. Minimal Movement

Four values were tested for the minimal movement parameter. The results from these tests can be seen in Table 5.4. When using minimal movement equal to 1 the testing crashed, a discussion about why this happen can be found in Chapter 6. The minimal movement operation was executed in the window slicing step, highlighted in Figure 5.4.

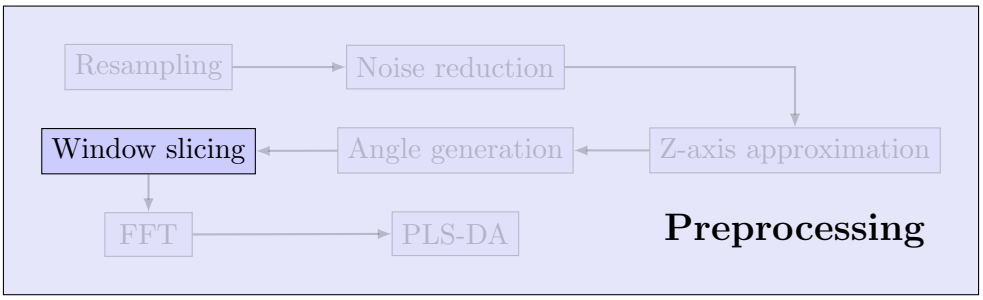


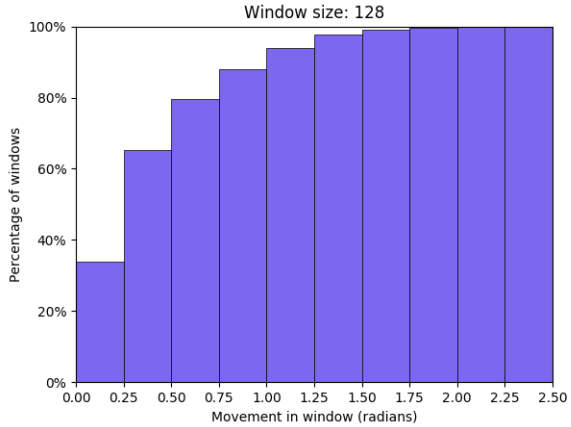
Figure 5.4.: Data flow through the preprocessing step with the step for minimal movement highlighted.

Model	Movement	Sens	Spec	AUC
LOF	0.25	0.120	0.954	**0.634**
LOF	0.50	0.115	0.955	0.629
LOF	0.75	0.115	0.957	0.617

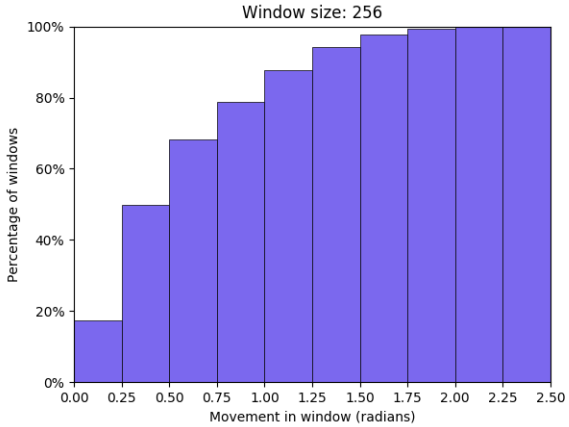
Table 5.4.: The best models when sorted by AUC for minimal movement.

Window Sizes

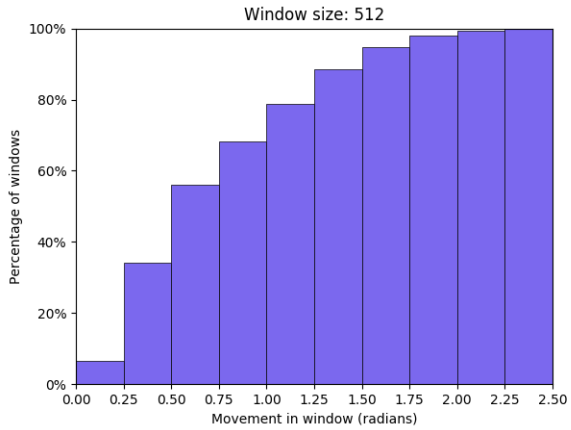
Figure 5.5 show the movement within each window size. It shows the percentage of movement done per 0.25 radians. The graph is cumulative.



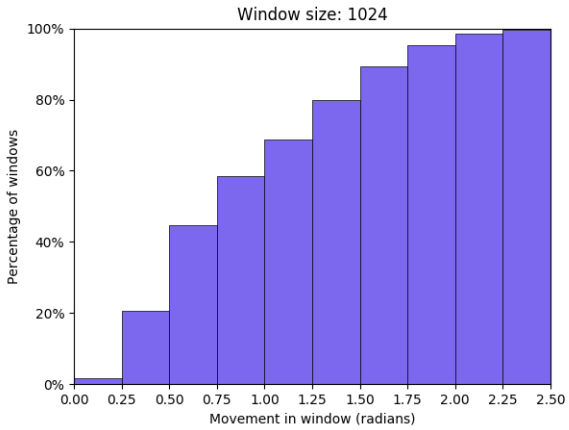
(a) Percentage of movement for window size 128.



(b) Percentage of movement for window size 256.



(c) Percentage of movement for window size 512.



(d) Percentage of movement for window size 1024.

Figure 5.5.: Cumulative percentage of how much movement is in the different windows in radians.

5.1.3. Window Overlap

Three different values were tested for the window overlap parameter. The results from these can be seen in Table 5.5. The window overlap operation was done in the window slicing step, highlighted in Figure 5.6.

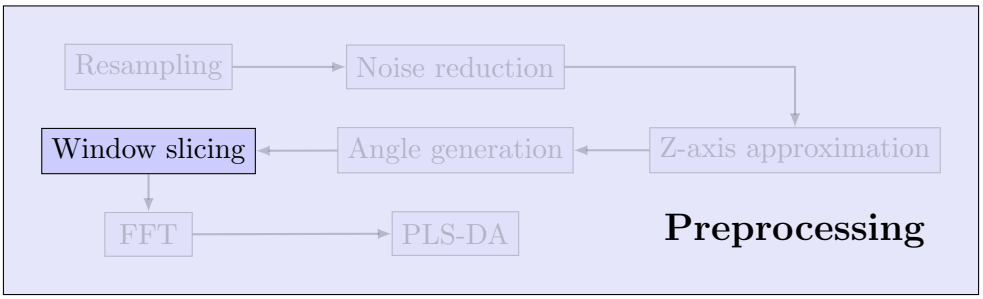


Figure 5.6.: Data flow through the preprocessing step with the window overlap step highlighted.

Model	WO	Sens	Spec	AUC
OCSVM	1	0.281	0.807	0.607
LOF	2	0.126	0.953	0.627
LOF	4	0.120	0.954	**0.634**

Table 5.5.: The best models when sorted by AUC for window overlap. WO = window overlap.

5.1.4. PLS-DA

Three values were tested for the PLS-DA parameter. The results from these values can be seen in the Table 5.6. The PLS-DA operation was done in the PLS-DA step, highlighted in Figure 5.7.

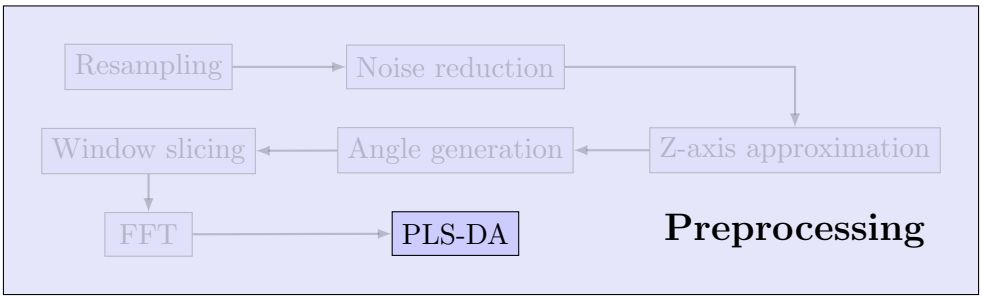


Figure 5.7.: Data flow through the preprocessing step with the PLS-DA step highlighted.

Model	PLS-DA	Sens	Spec	AUC
HBOS	3	0.057	0.970	0.606
LOF	5	0.128	0.954	0.622
LOF	10	0.120	0.954	**0.634**

Table 5.6.: The best models when sorted by AUC for PLS-DA.

5.2. Anomaly Detection

The best result for each model from the model testing step can be seen in Table 5.7. After the results from the base models, the results from the ensemble models are presented. The results presented in this section are from the anomaly detection step, highlighted in Figure 5.8. It is an intermediate step, and the results are used for choosing which detector to use in the prediction step. The 5 best results for each model with parameter can be found in Appendix A.

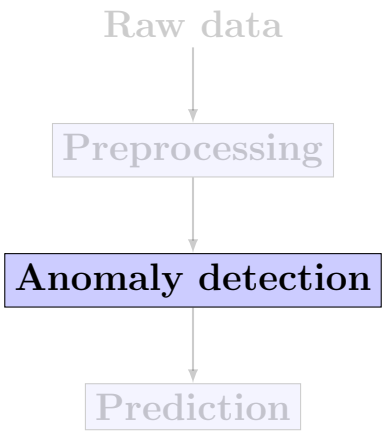


Figure 5.8.: A simple overview of the proposed method with the anomaly detection step highlighted.

5.2.1. Model Search

The best results from each model can be found in Table 5.7.

Model	Sens	Spec	$\mathbf{AUC}_{mean}$	$\mathbf{AUC}_{std}$
ABOD	0.073	0.966	0.579	0.031
CBLOF	0.086	0.955	0.580	0.040
HBOS	0.057	0.970	0.606	0.049
IForest	0.047	0.954	0.593	0.038
KNN	0.030	0.950	0.574	0.085
LOF	0.120	0.954	**0.634**	0.020
OCSVM	0.281	0.807	0.607	0.069

Table 5.7.: Results from the different base models. Sorted on AUC.

5.2.2. Novelty Based Anomaly Detection

The novelty test was done only using LOF and with the same parameters as the base model search. The parameters can be seen in Table 5.9 and the result from the novelty test can be seen in Table 5.8.

Parameter	Movement	SMA	PLS-DA	Sens	Spec	AUC
Neighbors: 40	0.75	3	3	0.203	0.943	0.573
Neighbors: 60	0.75	3	3	0.198	0.946	0.572
Neighbors: 80	0.75	3	3	0.192	0.951	0.571

Table 5.8.: The 3 best results when treating the problem as a novelty problem.

5.2.3. Ensemble

We tested three different ensemble models, Simple Classifier Aggregator, XGBOD and LSCP. Their base estimators and their parameters can be seen in Table 5.9. Every permutation of the parameters was made, and the ensembles used all the models as base estimators. The results from the ensemble testing can be seen in this section.

Model	Parameters
KNN	Neighbors = 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 Method = largest, mean
LOF	Neighbors = 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
OCSVM	nu = 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99
IForest	Estimators = 10, 20, 50, 70, 100, 150, 200, 250

Table 5.9.: Base estimator parameters. For models with more than one set of parameters, every permutation of parameters is made.

Parameters

The results from the parameter tuning from the ensemble models can be seen in Table 5.10, Table 5.11, and Table 5.12. SMA = 5 and minimal movement = 0.25 and 1 were removed before this testing.

Model	Movement	Sens	Spec	$\mathbf{AUC}_{mean}$	$\mathbf{AUC}_{std}$
XGBOD	0.5	0.014	0.994	0.678	0.035
XGBOD	0.75	0.171	0.957	**0.700**	0.032

Table 5.10.: The best ensembles models when sorted by AUC for minimal movement.

Model	PLS-DA	Sens	Spec	$\mathbf{AUC}_{mean}$	$\mathbf{AUC}_{std}$
XGBOD	3	0.171	0.957	**0.700**	0.032
XGBOD	5	0.029	0.991	0.671	0.020
XGBOD	10	0.028	0.99	0.666	0.032

Table 5.11.: The best ensembles models when sorted by AUC for PLS-DA.

Model	WO	Sens	Spec	$\mathbf{AUC}_{mean}$	$\mathbf{AUC}_{std}$
XGBOD	1	0.078	0.984	0.681	0.100
XGBOD	2	0.171	0.957	**0.700**	0.032
XGBOD	4	0.106	0.972	0.699	0.070

Table 5.12.: The best ensembles models when sorted by AUC for window overlap. WO = Window overlap.

Ensemble Models

The best results from the four different ensemble models can be seen in Table 5.13.

Model	Sens	Spec	$\mathbf{AUC}_{mean}$	$\mathbf{AUC}_{std}$
SCA_{max}	0.147	0.907	0.573	0.064
SCA_{avg}	0.130	0.906	0.586	0.063
XGBOD	**0.171**	**0.957**	**0.700**	**0.027**
LSCP	0.153	0.880	0.583	0.028

Table 5.13.: Results from the ensemble model testing. XGBOD performed significantly better than the rest of the models.

5.3. Prediction

The results presented in this section are from the prediction step, which can be seen in Figure 5.9. In the prediction step we combined the results from the ensemble models for the different window sizes, angles, and body part. Every combination of window sizes were tested to find the combination that yielded the best result. The results show how the different models performed. This section can be regarded as the main result we score AnoMove with.

Table 5.14 presents the best score gotten for every ensemble model we tested, and Figure 5.10 presents the ROC curve for the training data for the best performing model. Figure 5.11 shows the different AUC scores for the different threshold values for the models that performed the best using the threshold method for combining scores.

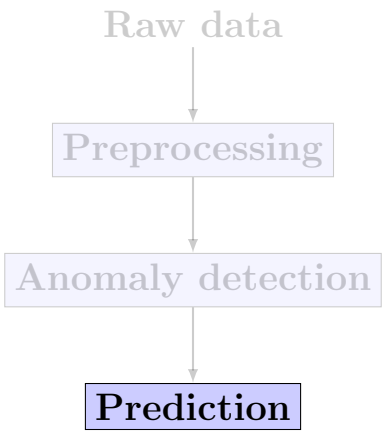


Figure 5.9.: Dataflow with prediction highlighted.

The column names are short for Window Method (WM), Window Size (WS), Angle Method (AM), Body Part Method (BPM) and Threshold (T).

Model	WM	WS	AM	BPM	T	AUC_{mean}	AUC_{std}
XGBOD	**max**	**[128]**	**threshold**	**mean**	**0.17**	**0.812**	**0.072**
LSCP	max	[256, 512]	threshold	max	0.97	0.654	0.069
SCA_{mean}	max	[128]	mean	mean	-	0.722	0.142
SCA_{max}	mean	[128, 256]	threshold	mean	0.91	0.703	0.123

Table 5.14.: Best scoring prediction parameter for each of the models tested. SCA_{mean} is the only method that does not perform best with the threshold method of combining scores across the video.

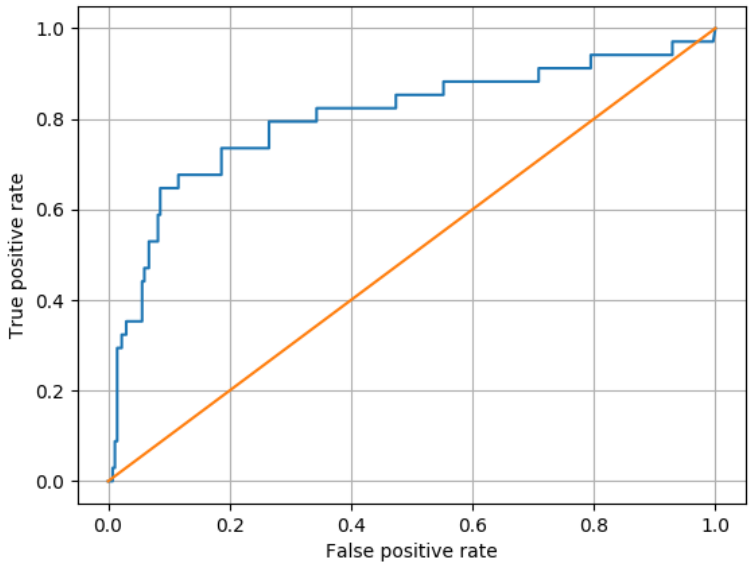


Figure 5.10.: ROC curve for the training set of the best performing model in Table 5.14. The curve is made by collecting all test runs from the different folds, so that each infants has been tested on once.

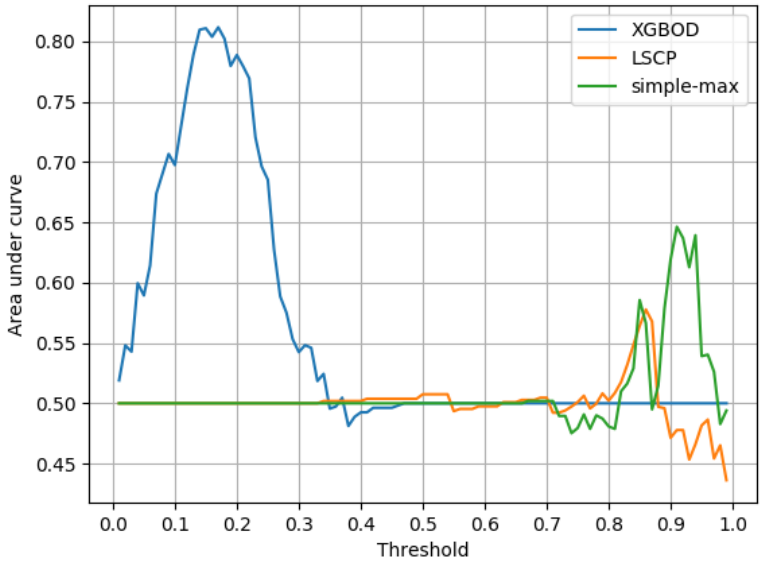


Figure 5.11.: Area under curve score for different threshold values on the models that scored best using the threshold method. We can see that the best performing model, XGBOD, is much less sensitive than the others, and has a very early peak at around 0.15 as the threshold. The other two methods have a more sensible sensitivity, peaking at around 0.85 and 0.9 respectively. X axis has threshold values, and y axis has the AUC score.

5.4. Testing

The results from the testing of the AnoMove method can be seen in Figure 5.12. It shows the ROC curve for the test set. Only the best performing model was tested, which is the same model used in Figure 5.10 and shown in Table 5.14.

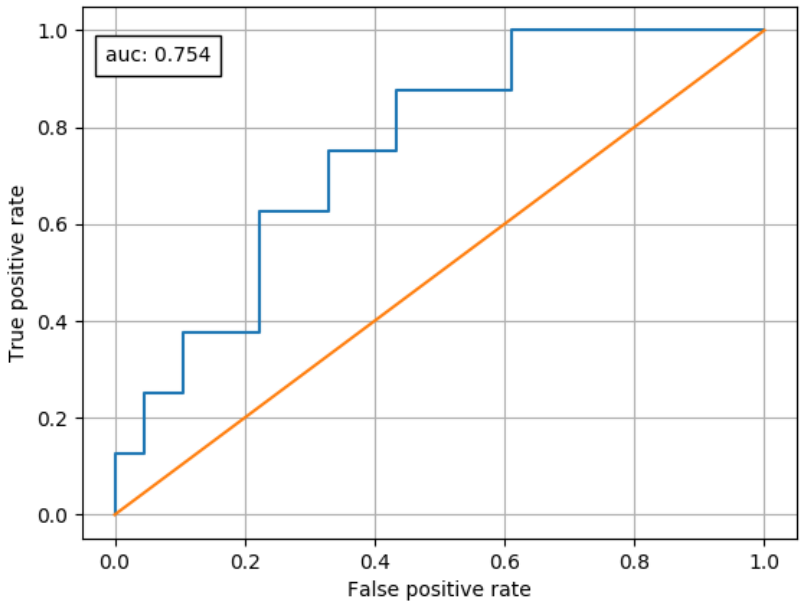
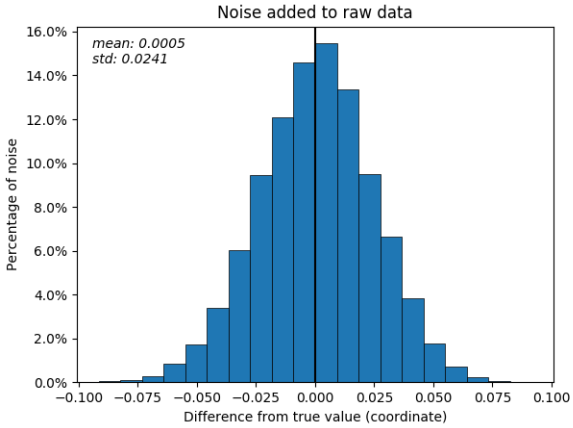


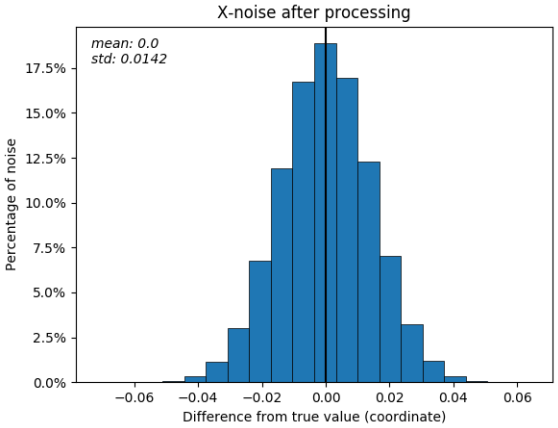
Figure 5.12.: ROC curve for the AnoMove method for the test set.

5.5. Ad-Hoc Error Analysis

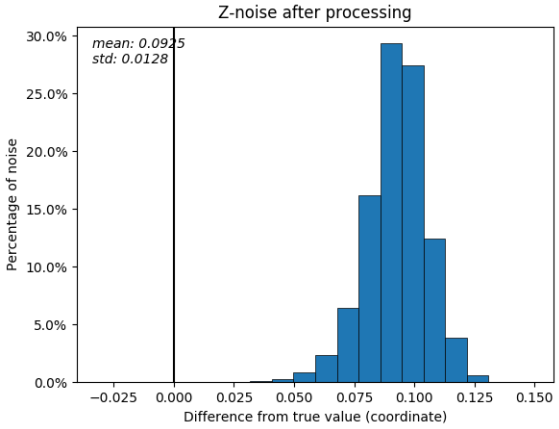
In this section, the results from the error propagation analysis from Section 4.7 are presented.



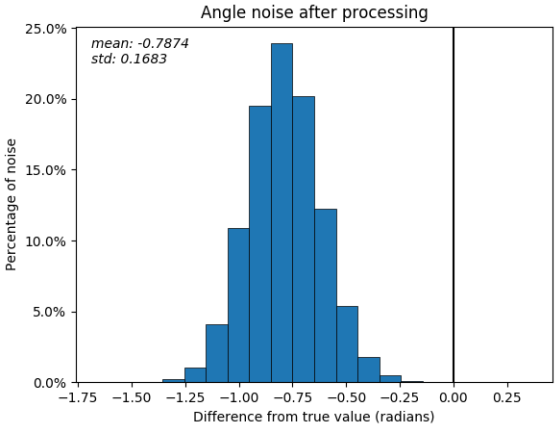
(a) The artificial input noise added to the coordinate data in an infant.



(b) The resulting noise in the x-coordinates after preprocessing.



(c) The resulting noise in the z-coordinates after preprocessing.



(d) The resulting noise in the angle data after preprocessing.

Figure 5.13.: Propagation of error throughout the preprocessing. Zero is marked with a vertical line.

6. Discussion

In this chapter we will discuss the results and choices for each step made in this report. The first section examine the different parameters in the preprocessing stage and the different models we implemented. Section 6.2 will bring the reader through our thoughts around the different models and why we wanted to test each of them. It will also contain a discussion about the results from each model and why or why not they performed as expected. The subsequent section will go through the results from each calculation in the score aggregation and why the different steps are needed and how different values can affect the results. Section 6.4 discuss the implications of noisy labels in our data. Error Propagation (Section 6.5) contains an discussion on how the uncertainty in our data propagates through the preprocessing stage. Section 6.6 presents and discuss the different limitations of our study. After that the research questions are discussed and answered. The last section will give a summary of elements of our study that can be further researched.

6.1. Preprocessing

The main focus in the preprocessing stage were to find the optimal values for the parameters and how we could transform the data so it represented the movement for each infant. Each stage in the preprocessing had a lot of different aspects that we needed to take into consideration when creating the AnoMove method.

6.1.1. Parameters

For each chosen parameter a thorough analysis was conducted to try to find the best value. Each parameter is looked at in isolation and evaluated with the AUC score and the standard deviation for the given parameter. When choosing the final values for the parameters we also looked at which combination of values had the lowest standard deviation while having a high AUC score.

Simple Moving Average

The implementation of a simple moving average (SMA) was successful in reducing the effect of noise in the raw data. As seen in Figure 5.3 the black line has less spikes than the red. This implies that the noise has been reduced. The parameter we could tune in SMA were how many frames we would use in the smoothing. Normal values to use are 3 or 5 points when applying SMA, so both were tested.

The results from the SMA tuning are shown in Table 5.3. Based on the AUC value, an SMA window of 3 performed slightly better than one of 5, but there is no significant difference. When there is no large difference between the results of the two parameters, the smaller one is preferred. This is to minimise the possibility for removing features from the data as well as also keeping it as simple as possible. Based on these arguments we choose a SMA window size equal 3 for further testing.

Minimal Movement

When pruning low movement, we wanted to remove the time slots where the infant is laying still. From the graphs in Figure 5.5 it is possible to conduct an analysis of the impact of this parameter. As the graph shows, removing everything that is less than 1 radians will reduce the amount of data with around 90 % for window size 128, 80% for window size 256, 75% for window size 512 and 60% for window size 1024. This will affect the results later in the pipeline, as with less data it will become harder to create a model that is generalised and not sample specific. It is also possible that we could lose some features by reducing the amount of data. On the other hand, to the authors knowledge there are no features in an infant that is laying still, and not a defined threshold for how much movement is necessary. If we do not prune the movement the chances of getting a lot of data that has no information is present. This will make it harder to detect features that are representative for abnormal movement.

We could see a clear correlation between the minimal movement parameter and the run time for the models. In early testing, using a high minimal movement parameter seemed to give better results. However, using a minimal movement of 1 crashed the training of many models, as they required more neighbours in their calculation than there were data available. We chose to use minimal movement 0.5 and 0.75 in further testing.

We had a minor bug in the AUC calculation in our early testing. It punished models that were less sensitive, which favoured the larger minimal movement limits. When the bug was fixed the higher minimal movement parameters on the basic models were no longer significantly better, and even a bit worse. The difference in scores is so small that we do not think the choice of only testing 0.5 and 0.75 on ensemble models can greatly affects the results. All scores for minimal movement tuning can be found in Table A.8 in Appendix A.2.

This parameter is all about balancing the data size against the data relevance. If we remove too much data, we are left with data that has a lot of movement, but too little to use it for analysis. If we leave to much data, we increase the possibility that the features are lost in the data. It will also increase the run time for the model drastically.

Window Overlap

Window overlapping gives us more relevant data and increases the accuracy of captured time-dependent movement. In theory, having a window overlap equal to the size of the window would be ideal. In that scenario, every possible window would be captured, and it would be easy to compare two infants with similar movement. This includes the assumption that small movements already have been removed. However, this method would increase both the amount of generated data and the run time required to train the model, as well as risk overfitting the model. This is important to address if a high value for the window overlap parameter is chosen.

From the results we can see that window overlap of 4 performs better than the other two. A window overlap of 2 and 4 have a significant better result than no window overlap. This is supporting our assumption of having a higher value for window overlap would yield more relevant data for the models. The results from the window overlap parameter tuning can be seen in Table 5.5.

PLS-DA

We started out with using PCA to reduce the number of dimensions in the data. The problem we had with PCA was that it attempts to construct dimensions that best explains the variance in the dataset. This gives dimensions that are good at separating each point in the dataset, but do not necessarily help with separating the healthy and impaired. From the results we got when analysing PCA we can represent the data with a variance of 95% with 10 PCA components, but the results in the AUC score were not adequate. We also tried pooling together the frequencies into different bandwidths using either maximum or average values to represent their bandwidth before applying PCA. This was done with the assumptions that it would make the results more generalised. However, the results were not significantly better.

To mitigate the problem of only explaining the variance in the data and to try to amplify the features that separates the classes, we implemented PLS-DA. In contrast to PCA, PLS-DA tries to find variables that best explains the difference between the classes of the data. It suited our goal for the dimension reduction better.

When we implemented PLS-DA, we found that we got significant higher scores than we did with PCA. The problem with PLS-DA is that it is a machine learning algorithm,

and with that has the problem associated with machine learning. The biggest problem we faced was overfitting the model so it would only perform well with our data, but not be generalised enough for other samples. From analysing the results from the different folds, it seems unlikely that PLS-DA overfits on our data. But it is important to take this factor into consideration, especially with the low amount of data we have from impaired infants.

From the PLS-DA tuning we found that using 10 PLS-DA components performed the best on the basic models, with an AUC of 0.634. Both 5 and 10 components are significantly better than 3. More dimensions mean a higher possibility of capturing features in the data. Again, the choice is about data quality and data size. Choosing between the different values for the PLS-DA in this stage of the process is not possible, based on the analysis of the results. We choose to use all three values in further testing of the different models and ensembles. The results from the PLS-DA parameter tuning on base estimators can be seen in Table 5.6.

6.1.2. Data Generation

In the preprocessing step we had four operations that generated and transformed data, namely z-approximation, angle generation, Fourier transformation and window slicing.

Z-Approximation

The implementation of the z-approximation solves the problem with the data being in only two dimensions. But it also introduces a new problem. When infants are laying on their side the assumptions that are used when calculating the z-approximation are broken. This means that if an infant is laying on their side the entire video the z-approximation will create corrupt data that will decrease the performance of the model.

We did a random sample test on the input videos for our analysis and did not find many infants on their side. The total number of infants that lay on their side in our data is unknown, but based on our analysis this number is not high enough to greatly affect the performance of our method. However, it is important to ensure that this assumption is sustained when increasing the data with new videos, or to find a solution to bypass the assumption.

Angle Generation

When using angles instead of coordinates the distance from the camera to the infant is no longer a problem since the length of each limb has no effect on the angle in the joint. The problem with resolution and centring of the infant is also removed. Angles gives a

much higher generalisability than using x- and y- coordinates, since we can now directly compare each infant with every other infant. In addition, there is a dimensionality reduction when going from the multiple sets of coordinates to a single angle for each joint.

From the results we can see that it is easiest for the models to classify the movements from the knee. This can be because there is a lot of movement in the knees, and the movement is quite large. The other angles, elbow, hip, and shoulder performed almost identical. From the results it is also clear that the LOF model is one of the best performing basic model. In comparison to OCSVM it has a higher specificity and a lower sensitivity. Using both in an ensemble could increase the performance. The results from the different angles can be seen in Table A.13 in Appendix A.

Window Slicing

The biggest problem when transforming the changes in angles to frequencies is that we do not know when the movement starts, and therefore we do not know where to split the data before the transformation. So, when splitting up the different windows to feed the Fourier transformation, we do not have any assurance that we are capturing the entire movement or some part of it. We solved this problem by dividing the data into different slices of 128, 256, 512 and 1024. The idea is that with different window sizes we could capture different single movements and movement patterns.

From the window size results we found that window size of 1024 performed the best, with the three other window sizes almost identical with each other. When analysing the results for the window sizes we noticed that a lot of different models performed well. This is in contrast from the previous results we have analysed, were LOF performed the best. The results from the window sizes can be seen in Table A.12.

Fourier Transformation

After the z-approximation and angle generation the data is still a time series, and for our analysis the time component is not important. To avoid dealing with the different challenges time series has we implemented the Fourier transformation. With the Fourier transformation we could keep the time aspect without needing to deal with a time series. It transforms the time series data to frequencies, so the time data is embedded in the frequencies.

6.2. Anomaly Detection

There are several different models for anomaly detection and choosing the correct one is crucial. Every model has pros and cons, so it is important to find the model that fits our data. The models we have tried are KNN, LOF, CBLOF, HBOS, ABOD, OCSVM and IForest.

The main reason for choosing the models are that they are well known and have proved to be reliable. They are easy to implement and within our scope for this thesis. We wanted a diverse category of models because we did not know what would work when we started. The use and comparison of different models facilitates further work by establishing a baseline for the performance, as well as indicating what kind of features are found in the data.

From the model testing we found that LOF performed best with an AUC of 0.634, and it also has the lowest standard deviation of the seven aforementioned models. After testing the seven models we wanted to reduce the number of models we needed to test in the ensemble step. ABOD and CBLOF performed significantly worse and had a larger run time complexity than the rest of the models and were excluded from further testing. HBOS performed very well in the model testing stage with an AUC = 0.606, but only in a small subset of the preprocessing parameters. Based on this we removed HBOS from further testing. KNN also performed significantly worse than the best models, however when we looked at the overall performance with the chosen values for the different parameters, we found that KNN performed well and therefore we continued the testing with KNN. The rest of the models has high values for the AUC and a low standard deviation. That left us with four models that we continued testing in different ensemble models, namely: IForest, KNN, LOF, OCSVM. They all showed promising results and are a diverse collection of models that should capture important features in the data. The results from the different models can be seen in Tables 5.7 and in Appendix A.1.

Novelty Detection

It is important to discuss whether the separation of movements is an outlier detection problem or a novelty detection problem. The main reason to think that this is a novelty detection problem is the fact that we know the labels for each infant, meaning that it is possible to construct a dataset containing only healthy movement, and labelling everything that is a novelty as an impaired movement.

From the novelty testing we got an AUC of 0.573. Since we only tested the novelty detection for LOF we compared it to the result from LOF for the base models. LOF got an AUC of 0.634 and a standard deviation of 0.020 when using outlier detection. The reason we only tested LOF were that it was recommended as a novelty model by SciPy.

Based on this we concluded with not doing any more testing in the novelty detection area.

6.2.1. Base Model Parameters

Each model has several different parameters that can be tuned to improve the performance of that model. We started out with the goal of using standard anomaly detection models with default parameters. Therefore, we have only tested each model with the default parameters and a few other parameters. The main reason for this is time constraint and the that the default parameters has showed promising results from other researches. We have also based a lot of our work on the work by Zhao, the creator of the pyod library. He has done a lot research on this topic and is using the default parameters for the models.

6.2.2. Ensemble Detection

We treated the ensemble models as a black box. The main reason is that the building and tuning of an ensemble model is outside of our scope as well as being a rather new research field. Secondly, Zhao has done a lot of research on this topic and developed different ensemble models for outlier detection. His research is well documented and suited for our problem.

The results from our testing on ensemble models indicates that XGBOD greatly outperforms the rest. They were all run with the same base estimators, and primarily with their default parameters. The only exception was the *local region size* parameter of LSCP, which was increased from the default of 30 to 150. This parameter is dependent on the total amount of data in the model and was therefore increased.

The library we used for the ensemble models is under development, so we came across some undocumented behaviour that may have affected LSCP. The Pearson score calculation crashed due to it having less than two data points. However, the training did not halt, so we assume that the error was handled by the model. The errors may indicate that the model is not suited for our data.

XGBOD performs well almost regardless of preprocessing parameters. The AUC scores range from about 0.65 to 0.7, which is almost within a standard deviation of each other. Given these results, we chose a set of parameters for the preprocessing to use in the prediction. The results from the parameters can be found in Appendix A.6.

Isolated, 0.75 radians minimal movement has the best performing results, but the results from 0.5 radians was just a standard deviation away with most other parameter combinations. We decided on a minimal movement of 0.5 radians. A limit of 0.5 radians will

prevent us from removing too much data, and it will not set the criteria for movements too high when applying the method to future infants.

We ended up using 3 for our moving average based on the results from the base estimators. We did not test other values on the ensemble models as the parameter is solely used to remove noise, and it is preferred to use the smallest value possible.

For window overlap we ended up choosing the largest value we tested, 4. The difference between 2 and 4 window overlap was small, but with the increased data from a 4 window overlap, the model is more accurate. Using a large window overlap in combination of removing windows with a minimal movement requirement should also ensure that we find all relevant windows of movement while discarding the ones without the features we want.

The differences in scores when comparing PLS components were negligible. We decided on using the lowest number of components we tested following Occam's razor, by trying to keep it as simple as possible. A low number of dimensions should equal a simple input. The final values for the parameters in the preprocessing step in AnoMove can be seen in Table 6.1.

Parameter	**Value**
Minimal movement	0.50
SMA	3
Window overlap	4
PLS-DA	3

Table 6.1.: The final values for the preprocessing in AnoMove.

6.3. Prediction

We wanted to keep the aggregation of the outlier scores as simple as possible. The idea was to combine the score for each angle, window and body part to one score that will indicate if the infant has CP or not. When combining these scores, we looked at three possible methods: mean, max and threshold. We expected these three methods to enhance different aspects of the propagation of our outlier scores to a final score.

As expected from the ensemble model testing, XGBOD was the top performing model when predicting infants. Using a threshold for combining the scores across the frames of the video ended up being the best method, with both a larger mean AUC score as well as a lower standard deviation between runs. This was accompanied by taking the max score within the overlapping frames in overlapping windows and taking the mean score of the limbs we evaluated. The best scoring method had only one window size, so the window aggregation only aggregated overlapping frames from the same window size. All top scoring results for XGBOD and the other models can be found in Appendix A.10.

A reason for threshold being the best method for aggregating scores across the video might be its resilience to variable length of the videos. When using mean, the score could converge to a set value range with increasing video length regardless of label, and using max, the value would likely increase with increasing video length. However, we must be able to compare infants without video length being a contributing factor. When using a threshold value, the value must also be set. This value depends on the sensitivity of the model used. For us, XGBOD turned out to have rather low sensitivity, so a low score of 0.17 gave the best results. The final value for the prediction step in AnoMove can be seen in Table 6.1.

Parameter	**Value**
Window method	Max
Window size	128
Angle method	Threshold = 0.17
Body part method	Mean

Table 6.2.: The final values for the prediction step in AnoMove.

We expected the performance of the test set to be within one standard deviation of the training scores. The results on the test set ended up being 0.754, which is well within the training score of 0.812 and its standard deviation of 0.072.

6.4. Noisy Labels

Since the number of infants that are labelled with CP are so low, it is important to understand the implication of noisy labels. Noisy labels can for example occur if an infant is wrongly diagnosed with CP at an early stage which later gets corrected. If the label is not updated in the data, the model is trained on wrong assumptions. Another possibility is human error. A single bit flip from 0 to 1 in the labelling process would be an expected human error but could have large effects on the models we train.

We had a case of noisy labels in our project. We were given two different metadata files with the data, one with medical data and one with technical data, and both with labels for the outcome of the diagnosis. Since we were not concerned with medical data other than the diagnosis outcome, we moved forward with the technical metadata. However, when we ran the test set, which contained in total 8 infants with CP, one infant stood out. Our model gave it a score very close to 0, indicating that the model found no indication of any impairment. An examination of the medical metadata confirmed that the infant was indeed healthy, and we assume that it was a human error that led to its' label as impaired in the technical metadata.

Under a further investigation, we found that there were two cases where an infant had switched from healthy to impaired between the two metadata files. We fixed the label of the infant in the test data but did not fix the label of the infant in the training data. This would have required a rerun of the entire parameter search, which would be a significant time investment. We must expect that noisy labels can exists and design our models around it. Noisy labels can be a reason for high standard deviation between runs, as infants correctly labelled as healthy by the models will still get a bad score if the infant is registered as impaired in the metadata.

6.5. Error Propagation

In this section the results from Section 5.5 Ad-Hoc Error Analysis will be discussed. As seen in Figure 5.13a, the noise added to the data had a mean value equal to 0 and a standard deviation equal to 0.024, and was based on a Gaussian distribution.

After the preprocessing, the amount of noise in the same x-value is almost halved, with the mean still being 0 and the deviation equal to 0.014. The operations the noise has been subject to are resampling and the simple moving average. It is doubtful that the resampling had any effect on the noise, but the simple moving average should be effective in removing Gaussian noise.

The amount of deviation did not grow when the z-axis was approximated, but the mean was offset substantially from 0, equalling 0.09 after the calculations. This is probably because of the vector length that the approximation is based on, which takes the longest value throughout the video. This way the largest noise will propagate from this point onward.

The effect of the large offset on the noise is also apparent in the angle-data. Here, the mean of the radian is offset -0.79 from the reference data. However, the standard deviation is 0.17 radians. In our transformation on the data, the absolute values of the data are insignificant, with only the differences from frame to frame affecting the Fourier transformation. This makes the standard deviation of the angle data the only factor affecting our model performance. In this experiment, the noise is about a factor of 7 larger in radians then the noise in raw coordinate values.

In real world performance, the noise is less likely to take the form of a Gaussian distribution on a frame to frame basis, as the model is likely to settle on the same erroneous value when the point is not in movement. This can be seen in Figure 5.3, where the raw data settles on the same value for long periods of time.

6.6. Limitations

There are mainly three different limitations in our thesis. The time allotted, the data available, and the computer resources we had available.

- **Time**
  Since this is a project for a master thesis, it had to be done in the time slot of one semester. Because of this we had to choose selectively which preprocessing, models, and parameters to explore.

- **Data**
  CIMA might be the largest set of data of infant movement with labels in the world, but it is still only about 370 infants, with about 12% of them being impaired. When you split up the data into test sets and validation folds, you quickly end up with a single digit of impaired infants in each set. This paired with the possibility of noisy labels makes it hard to guarantee the significance of our results.

- **Computer resources**
  As our data is considered both personal information according to GDPR and confidential according to InMotion, we had to sign a data delivery agreement where we could not make the data available on the internet or send it over the internet. Because of this, our supervisor did not find it feasible to use the clusters available to students on NTNU. All our code is ran on home computers, where the most powerful had a AMD Ryzen 3900X 12 core CPU and 32 gigabyte of RAM.

6.7. Research Questions

From the result and discussion, we can conclude with the following for the research questions in this thesis.

RQ1: How can infant movement be represented with minimal feature loss and a minimal number of dimensions?

  – With the AnoMove method the information is preserved throughout the preprocessing with every step being reversible. We also achieve a dimensionality of three dimensions per time frame window and body part angle using PLS-DA. PLS-DA also ensures that features that has the highest separation between healthy and impaired are enhanced.

RQ2: To what degree can basic anomaly detection models predict cerebral palsy in infants?

  – Basic anomaly detection models did not perform as well as we hoped. We

conclude that basic anomaly detection models are too simple to detect the correct features in the data.

RQ3: What is the state-of-the-art in outlier detection?

- At the authors knowledge the state-of-the-art in outlier detection are outlier ensemble models, possibly enhanced by boosting methods. Tested models in this thesis was XGBOD and LSCP, where XGBOD performed the best.

RQ4: How do state-of-the-art outlier detection models perform when compared to basic anomaly detection models?

- The ensemble models proved to be more able to correctly identify the features that separate healthy and impaired infants.

6.8. Further Work

We think that the most interesting aspect of our method to improve upon could be the window slicing. We have implemented a naive approach, where the start and end times of a window is chosen only based on a set window size. However, if a program was able to find patterns in the data and slice the windows based on them, one would be able to directly compare equal movements between healthy and impaired infants. This could greatly improve the input data quality to the machine learning models, making it possible to train great models regardless of the small amount of available data.

7. Conclusion

Our research goal was: *How can anomaly detection models be used to find abnormal patterns in infant movement caused by cerebral palsy?*.

By representing movement with angles and transforming the signal with Fourier transform we could represent the movement using frequencies. Using frequencies, we could capture the movement done in each angle isolated in time without using time-series. This allowed us to extract features that later could be analysed. The anomaly scores were found in isolation on each angle and time step, and then aggregated to get a final prediction.

We found that given the correct preprocessing, anomaly detection in movement is possible. The best anomaly detection algorithm we tested, XGBOD, got an area under curve score of 0.754 on the test set, which is a high score given the low amount of impaired infants in the data. An important element with the AnoMove method is the possibility of detecting and visualising when the method is classifying abnormal movements.

We conclude that using anomaly detection methods is a feasible alternative to current machine learning and deep learning alternatives. Since anomaly detection algorithms are designed to perform well on unbalanced data classification, it is well suited to detect cerebral palsy in infants.

References

[1] H. Rahmati, O. M. Aamo, Ø. Stavdahl, R. Dragon, and L. Adde, "Video-based early cerebral palsy prediction using motion segmentation," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014, pp. 3779–3783.

[2] "Cerebral palsy: Hope through research." [Online]. Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Hope-Through-Research/Cerebral-Palsy-Hope-Through-Research#3104_2

[3] M. Oskoui, F. Coutinho, J. Dykeman, N. Jetté, and T. Pringsheim, "An update on the prevalence of cerebral palsy: a systematic review and meta-analysis," *Developmental Medicine & Child Neurology*, vol. 55, no. 6, pp. 509–519, 2013.

[4] M. Bracewell and N. Marlow, "Patterns of motor disability in very preterm children," *Mental retardation and developmental disabilities research reviews*, vol. 8, no. 4, pp. 241–248, 2002.

[5] G. Cioni, V. Belmonti, and C. Einspieler, *Early diagnosis and prognosis in cerebral palsy.* Elsevier Ltd., 9 2013, pp. 177–187.

[6] K. Koffka, *Principle of Gestalt Psychology.* Routledge, 1935.

[7] L. Adde, M. Rygg, K. Lossius, G. K. Øberg, and R. Støen, "General movement assessment: predicting cerebral palsy in clinical practise," *Early human development*, vol. 83, no. 1, pp. 13–18, 2007.

[8] J. Beran, *Mathematical Foundations of Time Series Analysis: A Concise Introduction.* Cham: Springer International Publishing, 2017.

[9] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, ser. Springer Texts in Statistics,. New York, NY: Springer New York, 2002.

[10] J. J. Tan, Lizhe, *Digital Signal Processing - Fundamentals and Applications (3rd Edition).* Elsevier, 2019. [Online]. Available: https://app.knovel.com/hotlink/toc/id:kpDSPFAE02/digital-signal-processing/digital-signal-processing

[11] W. K. Chen, *The Electrical Engineering Handbook.* Elsevier Science, 2004.

[12] M. Mastro, "Fourier transform," in *Financial Derivative and Energy Market Valuation*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013, pp. 397–457.

[13] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[14] G. C. Danielson and C. Lanczos, "Some improvements in practical fourier analysis and their application to x-ray scattering from liquids," *Journal of the Franklin Institute*, vol. 233, no. 5, pp. 435–452, 1942.

[15] P.-N. Tan, M. Steinbach, and V. Kumar, *Intro to Data Mining*. P. Ed Australia, 2005.

[16] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[17] A. N. Gorban, B. Kégl, D. C. Wunsch, A. Y. Zinovyev *et al.*, *Principal manifolds for data visualization and dimension reduction*. Springer, 2008, vol. 58.

[18] R. G. Brereton and G. R. Lloyd, "Partial least squares discriminant analysis: taking the magic away," *Journal of Chemometrics*, vol. 28, no. 4, pp. 213–225, 2014.

[19] K. G. Mehrotra, *Anomaly Detection Principles and Algorithms*, 1st ed., ser. Terrorism, Security, and Computation. Cham: Springer International Publishing : Imprint: Springer, 2017.

[20] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 427–438.

[21] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 15–27.

[22] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[23] N. Shahid, I. H. Naqvi, and S. B. Qaisar, "One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments," *Artificial Intelligence Review*, vol. 43, no. 4, pp. 515–563, 2015.

[24] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 444–452.

[25] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsu-

pervised anomaly detection algorithm," in *35th German Conference on Artificial Intelligence September 24-27, 2012 Saarbrucken, Germany*. Citeseer, 2012, p. 59.

[26] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recogn. Lett.*, vol. 24, no. 9–10, p. 1641–1650, Jun. 2003. [Online]. Available: https://doi.org/10.1016/S0167-8655(03)00003-5

[27] D. Mossman, "Three-way rocs," *Medical Decision Making*, vol. 19, no. 1, pp. 78–89, 1999, pMID: 9917023. [Online]. Available: https://doi.org/10.1177/0272989X9901900110

[28] L. Adde, J. L. Helbostad, A. R. Jensenius, G. Taraldsen, and R. Støen, "Using computer-based video analysis in the study of fidgety movements," *Early human development*, vol. 85, no. 9, pp. 541–547, 2009.

[29] K. Groos, Daniel & Aurlien, "Infant Body Part Tracking in Videos Using Deep Learning," Master's thesis, NTNU, 2018.

[30] M. Ling Wiik, Marie & Theisen, "Time series movement data represented as 2d image: Prediction of cp with pretrained autoencoder," Master's thesis, NTNU, 2019.

[31] E. A. Ihlen, R. Støen, L. Boswell, R.-A. de Regnier, T. Fjørtoft, D. Gaebler-Spira, C. Labori, M. C. Loennecken, M. E. Msall, U. I. Möinichen *et al.*, "Machine learning of infant spontaneous movements for the early prediction of cerebral palsy: A multi-site cohort study," *Journal of Clinical Medicine*, vol. 9, no. 1, p. 5, 2020.

[32] T.-N. Nguyen, H.-H. Huynh, and J. Meunier, "Skeleton-based abnormal gait detection," *Sensors*, vol. 16, no. 11, p. 1792, 2016.

[33] M. Wrinch, T. H. El-Fouly, and S. Wong, "Anomaly detection of building systems using energy demand frequency domain analysis," in *2012 IEEE Power and Energy Society General Meeting*. IEEE, 2012, pp. 1–6.

[34] H. Rahmati, R. Dragon, O. M. Aamo, L. V. Gool, and L. Adde, "Motion segmentation with weak labeling priors," in *GCPR*, 2014.

[35] X. Dai, S. Wang, H. Li, H. Yue, and J. Min, "Image-assisted discrimination method for neurodevelopmental disorders in infants based on multi-feature fusion and ensemble learning," in *Brain Informatics*, P. Liang, V. Goel, and C. Shan, Eds. Cham: Springer International Publishing, 2019, pp. 105–114.

[36] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li, *LSCP: Locally Selective Combination in Parallel Outlier Ensembles*. SIAM, 2019, pp. 585–593. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975673.66

[37] Y. Zhao and M. K. Hryniewicki, "Xgbod: improving supervised outlier detection with unsupervised representation learning," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[39] D. Groos, H. Ramampiaro, and E. Ihlen, "Efficientpose: Scalable single-person pose estimation," *arXiv preprint arXiv:2004.12186*, 2020.

[40] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[41] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[42] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html

[43] Y. Zhao, X. Wang, C. Cheng, and X. Ding, "Combining machine learning models and scores using combo library," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York, USA, Feb 2020.

A. Results

A.1. Base Models

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Neighbors: 3	0.75	3	3	1	512	shoulder	0.073	0.966	0.579	0.031
Neighbors: 20	0.75	5	3	1	128	shoulder	0.044	0.946	0.572	0.086
Neighbors: 10	0.75	5	3	1	128	shoulder	0.044	0.956	0.571	0.085
Neighbors: 5	0.75	5	3	1	128	shoulder	0.052	0.956	0.57	0.085
Neighbors: 20	0.5	3	3	4	1024	hip	0.059	0.949	0.567	0.035

Table A.1.: The five best results from ABOD, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Clusters: 18	0.25	5	3	1	512	shoulder	0.086	0.955	0.58	0.04
Clusters: 20	0.75	5	5	1	128	shoulder	0.044	0.946	0.577	0.025
Clusters: 16	0.25	5	3	1	512	shoulder	0.096	0.953	0.576	0.044
Clusters: 8	0.75	5	5	1	128	shoulder	0.044	0.942	0.575	0.068
Clusters: 10	0.25	3	3	2	512	shoulder	0.11	0.954	0.572	0.048

Table A.2.: The five best results from CBLOF, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Bins: 3	0.5	3	3	1	1024	shoulder	0.057	0.97	0.606	0.049
Bins: 5	0.75	3	3	1	1024	hip	0.048	0.962	0.588	0.074
Bins: 5	0.75	5	5	1	128	shoulder	0.03	0.966	0.586	0.037
Bins: 12	0.5	3	3	1	1024	hip	0.054	0.956	0.585	0.046
Bins: 9	0.75	5	3	1	128	shoulder	0.052	0.962	0.585	0.051

Table A.3.: The five best results from HBOS, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Estimators: 150	0.5	3	3	1	1024	hip	0.047	0.954	0.593	0.038
Estimators: 100	0.5	3	3	1	1024	hip	0.051	0.953	0.593	0.037
Estimators: 70	0.5	3	3	2	1024	hip	0.035	0.95	0.592	0.051
Estimators: 50	0.5	3	3	1	1024	hip	0.044	0.954	0.592	0.037
Estimators: 70	0.5	3	3	1	1024	hip	0.047	0.952	0.592	0.039

Table A.4.: The five best results from IForest, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Neighbors: 10	0.75	5	3	1	128	shoulder	0.03	0.95	0.574	0.085
Neighbors: 3	0.75	5	3	1	128	shoulder	0.037	0.954	0.572	0.083
Neighbors: 70	0.75	5	3	1	128	shoulder	0.015	0.946	0.571	0.066
Neighbors: 30	0.25	5	3	4	1024	shoulder	0.107	0.957	0.57	0.063
Neighbors: 20	0.5	3	3	4	1024	hip	0.048	0.948	0.57	0.034

Table A.5.: The five best results from KNN, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Neighbors: 50	0.25	3	10	4	1024	knee	0.12	0.954	0.634	0.019
Neighbors: 60	0.25	3	10	4	1024	knee	0.121	0.955	0.634	0.019
Neighbors: 40	0.25	3	10	4	1024	knee	0.127	0.954	0.634	0.02
Neighbors: 70	0.25	3	10	4	1024	knee	0.126	0.955	0.633	0.019
Neighbors: 30	0.25	3	10	4	1024	knee	0.131	0.953	0.632	0.02

Table A.6.: The five best results from LOF, sorted on AUC

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Nu: 0.01	0.75	5	5	1	128	shoulder	0.281	0.807	0.607	0.069
Nu: 0.1	0.75	5	5	1	128	shoulder	0.281	0.807	0.604	0.069
Nu: 0.2	0.75	5	5	1	128	shoulder	0.104	0.891	0.598	0.064
Nu: 0.3	0.75	5	5	1	128	shoulder	0.081	0.927	0.581	0.05
Nu: 0.6	0.75	5	3	1	128	shoulder	0.044	0.944	0.581	0.072

Table A.7.: The five best results from OCSVM, sorted on AUC

A.2. Base Model Parameters

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
LOF	**0.25**	3	10	4	1024	knee	0.12	0.954	0.634	0.019
LOF	**0.25**	3	10	4	1024	knee	0.121	0.955	0.634	0.019
LOF	**0.25**	3	10	4	1024	knee	0.127	0.954	0.634	0.02
LOF	**0.25**	3	10	4	1024	knee	0.126	0.955	0.633	0.019
LOF	**0.25**	3	10	4	1024	knee	0.131	0.953	0.632	0.02
LOF	**0.5**	3	10	4	1024	knee	0.115	0.955	0.629	0.016
LOF	**0.5**	3	10	4	1024	knee	0.113	0.954	0.628	0.016
LOF	**0.5**	3	10	4	1024	knee	0.114	0.955	0.628	0.015
LOF	**0.5**	3	10	4	1024	knee	0.115	0.956	0.627	0.015
LOF	**0.5**	3	10	4	1024	knee	0.117	0.953	0.627	0.019
LOF	**0.75**	5	10	4	1024	knee	0.115	0.957	0.617	0.032
LOF	**0.75**	5	10	4	1024	knee	0.119	0.956	0.617	0.033
LOF	**0.75**	5	10	4	1024	knee	0.115	0.956	0.617	0.033
LOF	**0.75**	5	10	4	1024	knee	0.115	0.958	0.616	0.032
LOF	**0.75**	5	10	4	1024	knee	0.117	0.955	0.616	0.035

Table A.8.: The five best results from each minimal movement value from the base models. First sorted on minimal movement then on AUC.

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
LOF	0.25	**3**	10	4	1024	knee	0.12	0.954	0.634	0.019
LOF	0.25	**3**	10	4	1024	knee	0.121	0.955	0.634	0.019
LOF	0.25	**3**	10	4	1024	knee	0.127	0.954	0.634	0.02
LOF	0.25	**3**	10	4	1024	knee	0.126	0.955	0.633	0.019
LOF	0.25	**3**	10	4	1024	knee	0.131	0.953	0.632	0.02
LOF	0.5	**5**	5	4	1024	knee	0.122	0.955	0.622	0.041
LOF	0.5	**5**	5	4	1024	knee	0.129	0.956	0.621	0.042
LOF	0.5	**5**	10	4	1024	knee	0.117	0.956	0.62	0.029
LOF	0.5	**5**	5	4	1024	knee	0.124	0.957	0.62	0.04
LOF	0.5	**5**	5	4	1024	knee	0.124	0.956	0.619	0.038

Table A.9.: The five best results from each SMA value from the base models. First sorted on SMA then on AUC.

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
HBOS	0.5	3	**3**	1	1024	shoulder	0.057	0.97	0.606	0.049
LOF	0.75	5	**3**	2	512	hip	0.038	0.962	0.598	0.101
LOF	0.75	5	**3**	4	1024	elbow	0.12	0.955	0.596	0.063
LOF	0.75	3	**3**	1	1024	hip	0.036	0.956	0.596	0.05
LOF	0.75	5	**3**	4	1024	elbow	0.12	0.954	0.595	0.064
LOF	0.25	3	**5**	4	1024	knee	0.128	0.954	0.622	0.024
LOF	0.5	5	**5**	4	1024	knee	0.122	0.955	0.622	0.041
LOF	0.25	3	**5**	4	1024	knee	0.129	0.953	0.622	0.025
LOF	0.25	3	**5**	4	1024	knee	0.13	0.951	0.622	0.024
LOF	0.25	3	**5**	4	1024	knee	0.129	0.95	0.621	0.024
LOF	0.25	3	**10**	4	1024	knee	0.12	0.954	0.634	0.019
LOF	0.25	3	**10**	4	1024	knee	0.121	0.955	0.634	0.019
LOF	0.25	3	**10**	4	1024	knee	0.127	0.954	0.634	0.02
LOF	0.25	3	**10**	4	1024	knee	0.126	0.955	0.633	0.019
LOF	0.25	3	**10**	4	1024	knee	0.131	0.953	0.632	0.02

Table A.10.: The five best results from each PLS-DA value from the base models. First sorted on PLS-DA then on AUC.

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
OCSVM	0.75	5	5	**1**	128	shoulder	0.281	0.807	0.607	0.069
HBOS	0.5	3	3	**1**	1024	shoulder	0.057	0.97	0.606	0.049
OCSVM	0.75	5	5	**1**	128	shoulder	0.281	0.807	0.604	0.069
LOF	0.5	5	5	**1**	1024	knee	0.102	0.956	0.604	0.067
LOF	0.5	5	5	**1**	1024	knee	0.102	0.955	0.6	0.068
LOF	0.25	3	10	**2**	1024	knee	0.126	0.953	0.627	0.026
LOF	0.25	3	10	**2**	1024	knee	0.116	0.954	0.625	0.027
LOF	0.25	3	10	**2**	1024	knee	0.116	0.956	0.624	0.025
LOF	0.25	3	10	**2**	1024	knee	0.114	0.957	0.622	0.024
LOF	0.25	3	10	**2**	1024	knee	0.111	0.957	0.62	0.023
LOF	0.25	3	10	**4**	1024	knee	0.12	0.954	0.634	0.019
LOF	0.25	3	10	**4**	1024	knee	0.121	0.955	0.634	0.019
LOF	0.25	3	10	**4**	1024	knee	0.127	0.954	0.634	0.02
LOF	0.25	3	10	**4**	1024	knee	0.126	0.955	0.633	0.019
LOF	0.25	3	10	**4**	1024	knee	0.131	0.953	0.632	0.02

Table A.11.: The five best results from each window overlap value from the base models. First sorted on window overlap then on AUC.

A.3. Base Models Window Size

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
OCSVM	0.75	5	5	1	**128**	shoulder	0.281	0.807	0.607	0.069
OCSVM	0.75	5	5	1	**128**	shoulder	0.281	0.807	0.604	0.069
OCSVM	0.75	5	5	1	**128**	shoulder	0.104	0.891	0.598	0.064
IFOREST	0.75	5	3	1	**128**	shoulder	0.044	0.952	0.586	0.097
HBOS	0.75	5	5	1	**128**	shoulder	0.03	0.966	0.586	0.037
LOF	0.75	5	3	1	**256**	hip	0.113	0.955	0.592	0.058
IFOREST	0.75	3	3	2	**256**	shoulder	0.056	0.953	0.581	0.061
IFOREST	0.75	3	3	2	**256**	shoulder	0.059	0.954	0.579	0.062
IFOREST	0.75	3	3	2	**256**	shoulder	0.056	0.961	0.578	0.067
IFOREST	0.75	3	3	2	**256**	shoulder	0.059	0.957	0.577	0.063
LOF	0.75	5	3	2	**512**	hip	0.038	0.962	0.598	0.101
LOF	0.75	5	3	2	**512**	hip	0.031	0.962	0.594	0.117
LOF	0.25	5	10	2	**512**	knee	0.093	0.952	0.594	0.033
LOF	0.25	5	10	2	**512**	knee	0.092	0.952	0.593	0.034
LOF	0.25	5	10	2	**512**	knee	0.097	0.951	0.592	0.029
LOF	0.25	3	10	4	**1024**	knee	0.12	0.954	0.634	0.019
LOF	0.25	3	10	4	**1024**	knee	0.121	0.955	0.634	0.019
LOF	0.25	3	10	4	**1024**	knee	0.127	0.954	0.634	0.02
LOF	0.25	3	10	4	**1024**	knee	0.126	0.955	0.633	0.019
LOF	0.25	3	10	4	**1024**	knee	0.131	0.953	0.632	0.02

Table A.12.: The five best results from each window size. First sorted on window size then on AUC.

A.4. Base Models Angles

The results for which model that performance the best for each angle can be seen in Table A.13.

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
LOF	0.75	5	3	4	1024	**elbow**	0.12	0.955	0.596	0.063
LOF	0.75	5	3	4	1024	**elbow**	0.12	0.954	0.595	0.064
LOF	0.75	5	3	4	1024	**elbow**	0.119	0.955	0.595	0.062
LOF	0.75	5	3	4	1024	**elbow**	0.121	0.955	0.595	0.065
LOF	0.75	5	3	4	1024	**elbow**	0.118	0.954	0.595	0.062
LOF	0.75	5	3	2	512	**hip**	0.038	0.962	0.598	0.101
LOF	0.75	3	5	1	1024	**hip**	0.012	0.962	0.596	0.068
LOF	0.75	3	3	1	1024	**hip**	0.036	0.956	0.596	0.05
LOF	0.75	3	3	1	1024	**hip**	0.048	0.958	0.595	0.045
LOF	0.75	3	5	1	1024	**hip**	0.012	0.963	0.595	0.062
LOF	0.25	3	10	4	1024	**knee**	0.12	0.954	0.634	0.019
LOF	0.25	3	10	4	1024	**knee**	0.121	0.955	0.634	0.019
LOF	0.25	3	10	4	1024	**knee**	0.127	0.954	0.634	0.02
LOF	0.25	3	10	4	1024	**knee**	0.126	0.955	0.633	0.019
LOF	0.25	3	10	4	1024	**knee**	0.131	0.953	0.632	0.02
OCSVM	0.75	5	5	1	128	**shoulder**	0.281	0.807	0.607	0.069
HBOS	0.5	3	3	1	1024	**shoulder**	0.057	0.97	0.606	0.049
OCSVM	0.75	5	5	1	128	**shoulder**	0.281	0.807	0.604	0.069
OCSVM	0.75	5	5	1	128	**shoulder**	0.104	0.891	0.598	0.064
IFOREST	0.75	5	3	1	128	**shoulder**	0.044	0.952	0.586	0.097

Table A.13.: The five best results from each angle from the base models. First sorted on angle then on AUC.

A.5. Novelty

Parameter	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
Neighbors: 40	0.75	3	3	1	512	shoulder	0.203	0.943	0.573	0.053
Neighbors: 60	0.75	3	3	1	512	shoulder	0.198	0.946	0.572	0.046
Neighbors: 80	0.75	3	3	1	512	shoulder	0.192	0.951	0.571	0.043
Neighbors: 30	0.75	3	5	1	512	shoulder	0.203	0.938	0.571	0.039
Neighbors: 70	0.75	3	3	1	512	shoulder	0.192	0.95	0.571	0.042

Table A.14.: The five best results from LOF novelty, sorted on AUC

A.6. Ensemble Models

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
SDA_{avg}	0.5	3	3	4	1024	hip	0.13	0.906	0.586	0.063
SDA_{avg}	0.5	3	3	2	1024	hip	0.129	0.903	0.582	0.068
SDA_{avg}	0.5	3	3	1	1024	hip	0.128	0.898	0.575	0.092
SDA_{avg}	0.5	3	3	2	512	hip	0.105	0.901	0.57	0.044
SDA_{avg}	0.75	3	3	2	128	shoulder	0.16	0.891	0.568	0.044
SDA_{max}	0.75	3	3	2	1024	shoulder	0.147	0.907	0.573	0.064
SDA_{max}	0.5	3	3	2	1024	hip	0.127	0.906	0.571	0.058
SDA_{max}	0.5	3	3	4	1024	hip	0.12	0.911	0.567	0.046
SDA_{max}	0.75	3	3	2	128	shoulder	0.136	0.901	0.564	0.044
SDA_{max}	0.5	3	10	4	1024	knee	0.132	0.913	0.56	0.046
LSCP	0.5	3	3	4	1024	hip	0.153	0.88	0.583	0.063
LSCP	0.5	3	3	2	1024	hip	0.152	0.884	0.58	0.071
LSCP	0.5	3	3	1	1024	hip	0.142	0.884	0.576	0.092
LSCP	0.75	3	3	2	1024	shoulder	0.181	0.89	0.569	0.059
LSCP	0.75	3	3	2	256	shoulder	0.145	0.892	0.567	0.065
XGBOD	0.75	3	3	2	1024	shoulder	0.171	0.957	0.7	0.032
XGBOD	0.75	3	3	4	1024	shoulder	0.106	0.972	0.699	0.07
XGBOD	0.75	3	3	1	512	hip	0.078	0.984	0.681	0.1
XGBOD	0.75	3	3	4	1024	knee	0.015	0.994	0.68	0.041
XGBOD	0.5	3	3	4	1024	knee	0.014	0.994	0.678	0.035

Table A.15.: The five best results from each ensemble model. First sorted on ensemble model then on AUC.

A.7. Ensemble Models Parameters

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
XGBOD	**0.5**	3	3	4	1024	knee	0.014	0.994	0.678	0.035
XGBOD	**0.5**	3	3	2	1024	shoulder	0.055	0.991	0.672	0.024
XGBOD	**0.5**	3	3	4	1024	shoulder	0.041	0.99	0.668	0.034
XGBOD	**0.5**	3	5	4	1024	knee	0.023	0.99	0.666	0.025
XGBOD	**0.5**	3	10	4	1024	knee	0.024	0.991	0.663	0.03
XGBOD	**0.75**	3	3	2	1024	shoulder	0.171	0.957	0.7	0.032
XGBOD	**0.75**	3	3	4	1024	shoulder	0.106	0.972	0.699	0.07
XGBOD	**0.75**	3	3	1	512	hip	0.078	0.984	0.681	0.1
XGBOD	**0.75**	3	3	4	1024	knee	0.015	0.994	0.68	0.041
XGBOD	**0.75**	3	5	4	1024	knee	0.029	0.991	0.671	0.02

Table A.16.: The five best results from each minimal movement value. First sorted on minimal movement then on AUC.

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
XGBOD	0.75	3	**3**	2	1024	shoulder	0.171	0.957	0.7	0.032
XGBOD	0.75	3	**3**	4	1024	shoulder	0.106	0.972	0.699	0.07
XGBOD	0.75	3	**3**	1	512	hip	0.078	0.984	0.681	0.1
XGBOD	0.75	3	**3**	4	1024	knee	0.015	0.994	0.68	0.041
XGBOD	0.5	3	**3**	4	1024	knee	0.014	0.994	0.678	0.035
XGBOD	0.75	3	**5**	4	1024	knee	0.029	0.991	0.671	0.02
XGBOD	0.5	3	**5**	4	1024	knee	0.023	0.99	0.666	0.025
XGBOD	0.75	3	**5**	4	512	knee	0.005	0.998	0.662	0.018
XGBOD	0.5	3	**5**	4	512	knee	0.004	0.998	0.659	0.01
XGBOD	0.75	3	**5**	2	1024	shoulder	0.15	0.94	0.658	0.055
XGBOD	0.75	3	**10**	4	1024	knee	0.028	0.99	0.666	0.032
XGBOD	0.75	3	**10**	2	512	knee	0.007	0.997	0.665	0.017
XGBOD	0.75	3	**10**	4	512	knee	0.005	0.998	0.664	0.018
XGBOD	0.5	3	**10**	4	1024	knee	0.024	0.991	0.663	0.03
XGBOD	0.5	3	**10**	4	512	knee	0.005	0.998	0.659	0.013

Table A.17.: The five best results from each PLS-DA value. First sorted on PLS-DA then on AUC.

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
XGBOD	0.75	3	3	**1**	512	hip	0.078	0.984	0.681	0.1
XGBOD	0.75	3	3	**1**	1024	shoulder	0.23	0.934	0.663	0.033
XGBOD	0.5	3	3	**1**	1024	shoulder	0.067	0.991	0.658	0.051
XGBOD	0.75	3	3	**1**	1024	knee	0.018	0.991	0.657	0.055
XGBOD	0.75	3	3	**1**	512	elbow	0.015	0.996	0.654	0.042
XGBOD	0.75	3	3	**2**	1024	shoulder	0.171	0.957	0.7	0.032
XGBOD	0.5	3	3	**2**	1024	shoulder	0.055	0.991	0.672	0.024
XGBOD	0.75	3	10	**2**	512	knee	0.007	0.997	0.665	0.017
XGBOD	0.75	3	3	**2**	1024	knee	0.023	0.992	0.664	0.056
XGBOD	0.75	3	3	**2**	1024	hip	0.094	0.98	0.661	0.084
XGBOD	0.75	3	3	**4**	1024	shoulder	0.106	0.972	0.699	0.07
XGBOD	0.75	3	3	**4**	1024	knee	0.015	0.994	0.68	0.041
XGBOD	0.5	3	3	**4**	1024	knee	0.014	0.994	0.678	0.035
XGBOD	0.75	3	5	**4**	1024	knee	0.029	0.991	0.671	0.02
XGBOD	0.5	3	3	**4**	1024	shoulder	0.041	0.99	0.668	0.034

Table A.18.: The five best results from each window overlap value. First sorted on window overlap then on AUC.

A.8. Ensemble Models Window Size

The 5 best results for each window size can be seen in Table A.19.

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
XGBOD	0.75	3	3	4	**128**	elbow	0.0	1.0	0.632	0.033
XGBOD	0.75	3	10	4	**128**	elbow	0.0	1.0	0.631	0.03
XGBOD	0.75	3	5	4	**128**	elbow	0.0	1.0	0.63	0.031
XGBOD	0.75	3	3	2	**128**	elbow	0.0	1.0	0.626	0.039
XGBOD	0.75	3	10	2	**128**	elbow	0.0	1.0	0.625	0.035
XGBOD	0.5	3	3	2	**256**	shoulder	0.038	0.992	0.648	0.042
XGBOD	0.75	3	5	4	**256**	knee	0.002	0.999	0.648	0.025
XGBOD	0.75	3	10	4	**256**	knee	0.001	0.999	0.648	0.022
XGBOD	0.75	3	10	4	**256**	elbow	0.003	0.999	0.646	0.031
XGBOD	0.75	3	3	4	**256**	elbow	0.003	0.999	0.646	0.027
XGBOD	0.75	3	3	1	**512**	hip	0.078	0.984	0.681	0.1
XGBOD	0.75	3	3	4	**512**	shoulder	0.123	0.966	0.666	0.09
XGBOD	0.75	3	10	2	**512**	knee	0.007	0.997	0.665	0.017
XGBOD	0.75	3	10	4	**512**	knee	0.005	0.998	0.664	0.018
XGBOD	0.75	3	5	4	**512**	knee	0.005	0.998	0.662	0.018
XGBOD	0.75	3	3	2	**1024**	shoulder	0.171	0.957	0.7	0.032
XGBOD	0.75	3	3	4	**1024**	shoulder	0.106	0.972	0.699	0.07
XGBOD	0.75	3	3	4	**1024**	knee	0.015	0.994	0.68	0.041
XGBOD	0.5	3	3	4	**1024**	knee	0.014	0.994	0.678	0.035
XGBOD	0.5	3	3	2	**1024**	shoulder	0.055	0.991	0.672	0.024

Table A.19.: The five best results from each window size. First sorted on window size then on AUC.

A.9. Ensemble Models Angles

The results from the different angles for the ensemble models can be seen in Table A.20.

Model	MM	SMA	PLS	WO	WS	Angle	Sens	Spec	AUC_{mean}	AUC_{std}
XGBOD	0.75	3	3	4	1024	**elbow**	0.057	0.994	0.66	0.029
XGBOD	0.75	3	3	4	512	**elbow**	0.022	0.997	0.657	0.025
XGBOD	0.5	3	3	4	1024	**elbow**	0.039	0.995	0.655	0.034
XGBOD	0.75	3	3	1	512	**elbow**	0.015	0.996	0.654	0.042
XGBOD	0.75	3	3	1	1024	**elbow**	0.062	0.991	0.654	0.034
XGBOD	0.75	3	3	4	1024	**knee**	0.015	0.994	0.68	0.041
XGBOD	0.5	3	3	4	1024	**knee**	0.014	0.994	0.678	0.035
XGBOD	0.75	3	5	4	1024	**knee**	0.029	0.991	0.671	0.02
XGBOD	0.75	3	10	4	1024	**knee**	0.028	0.99	0.666	0.032
XGBOD	0.5	3	5	4	1024	**knee**	0.023	0.99	0.666	0.025
XGBOD	0.75	3	3	1	512	**hip**	0.078	0.984	0.681	0.1
XGBOD	0.75	3	3	2	1024	**hip**	0.094	0.98	0.661	0.084
XGBOD	0.5	3	3	2	512	**hip**	0.001	0.998	0.659	0.048
XGBOD	0.75	3	3	4	1024	**hip**	0.062	0.982	0.655	0.065
XGBOD	0.5	3	3	4	512	**hip**	0.002	0.999	0.652	0.047
XGBOD	0.75	3	3	2	1024	**shoulder**	0.171	0.957	0.7	0.032
XGBOD	0.75	3	3	4	1024	**shoulder**	0.106	0.972	0.699	0.07
XGBOD	0.5	3	3	2	1024	**shoulder**	0.055	0.991	0.672	0.024
XGBOD	0.5	3	3	4	1024	**shoulder**	0.041	0.99	0.668	0.034
XGBOD	0.75	3	3	4	512	**shoulder**	0.123	0.966	0.666	0.09

Table A.20.: The five best results from each angle. First sorted on angle then on AUC.

A.10. Prediction

WM	WS	AM	BPM	T	AUC_{mean}	AUC_{std}
max	[128]	threshold	mean	0.17	0.812	0.072
max	[128]	threshold	mean	0.15	0.811	0.081
max	[128]	threshold	mean	0.14	0.81	0.096
mean	[256]	mean	max	-1.0	0.807	0.104
mean	[128, 256]	mean	max	-1.0	0.806	0.103
max	[128, 256]	threshold	mean	0.19	0.805	0.123
mean	[256]	threshold	max	0.14	0.805	0.109
max	[128]	threshold	mean	0.16	0.804	0.071
max	[128]	threshold	mean	0.18	0.802	0.085
max	[128, 256]	threshold	mean	0.2	0.801	0.129

Table A.21.: Prediction results using XGBOD model.

WM	WS	AM	BPM	T	AUC_{mean}	AUC_{std}
max	[256, 512]	threshold	max	0.97	0.654	0.069
max	[128, 512]	threshold	max	0.97	0.647	0.081
max	[128, 256, 512]	threshold	max	0.97	0.644	0.077
max	[512]	threshold	max	0.97	0.63	0.108
mean	[512, 1024]	threshold	max	0.97	0.629	0.145
mean	[1024]	threshold	max	0.97	0.629	0.108
mean	[128, 256, 512, 1024]	threshold	max	0.96	0.622	0.114
mean	[128, 256, 512, 1024]	threshold	max	0.97	0.608	0.139
mean	[256, 512, 1024]	threshold	max	0.96	0.607	0.126
mean	[128, 512, 1024]	threshold	max	0.97	0.607	0.165

Table A.22.: Prediction results using LSCP model.

WM	WS	AM	BPM	T	AUC_{mean}	AUC_{std}
mean	[128, 256]	threshold	mean	0.91	0.703	0.123
mean	[128, 256]	threshold	mean	0.9	0.687	0.069
mean	[128, 256]	threshold	mean	0.92	0.684	0.158
mean	[128]	threshold	mean	0.9	0.658	0.217
mean	[128, 256, 512]	threshold	mean	0.91	0.651	0.051
mean	[128, 256, 512]	threshold	mean	0.92	0.651	0.027
max	[128, 256]	threshold	mean	0.91	0.646	0.073
max	[256, 512, 1024]	threshold	max	0.96	0.646	0.071
mean	[256]	threshold	mean	0.91	0.645	0.096
mean	[128, 256]	threshold	mean	0.89	0.644	0.088

Table A.23.: Prediction results using Simple classifier aggregator with max.

WM	WS	AM	BPM	T	AUC_{mean}	AUC_{std}
max	[128]	mean	mean	-1.0	0.722	0.142
mean	[128, 256]	threshold	mean	0.82	0.717	0.048
max	[128, 256]	threshold	mean	0.99	0.707	0.139
mean	[256]	threshold	mean	0.82	0.699	0.043
mean	[128, 512]	threshold	mean	0.91	0.691	0.165
mean	[128, 256]	threshold	mean	0.86	0.688	0.086
mean	[128, 256]	threshold	mean	0.83	0.686	0.086
mean	[256]	threshold	mean	0.83	0.685	0.057
max	[128, 256]	threshold	mean	0.98	0.684	0.129
mean	[128, 512, 1024]	threshold	mean	0.88	0.684	0.169

Table A.24.: Prediction results using Simple classifier aggregator with mean.

Håvard Løkensgard & Erlend J. Skarpnes

AnoMove

NTNU
Norwegian University of
Science and Technology