Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Marie Ling Wiik
Mathilde Theisen

Time Series Movement Data represented as 2D image: Prediction of CP with Pretrained autoencoder

Masteroppgave i MTDT
Veileder: Heri Ramampiaro
Juni 2019

NTNU
Norwegian University of
Science and Technology

Abstract

We look at how time series coordinate movement data can be represented as images which have the ability to be understood by both a non-technological clinician and a deep learning architecture, more specific an autoencoder. Autoencoders are known as good feature extractors, a beneficial quality in order to recognise fidgety movement patterns made by infants. Fidgety movements are age-specific movements, and absence of these are strongly related to cerebral palsy. Movement data is previously extracted using a version of Human Pose Estimation called CIMA-Pose on the original video recordings of the infants.

The vision of this project is to make a contribution for the InMotion project at St. Olav's University Hospital, which aims to develop an automatic computer-based CP predictor. Our main contributions focus on highlighting the relevant data features as 2D images and implementing an autoencoder for cerebral palsy classification.

Findings made during the process include evaluation and insight about the proposed data representation together with a discussion and recommendations for future work. Autoencoder related findings cannot exclude the autoencoder as a feature extractor but indicates that more research is required.

This master thesis concludes that the proposed data representation is not ideal, before recommending future work for the InMotion project, in order for them to develop an automatic CP prediction model that can give an early and reliable CP diagnosis.

Sammendrag

Denne masteroppgaven kartlegger hvordan tidsseriedata i form av koordinater kan representeres som bilder som både kan forstås av ikke-teknologiske klinikere og av en dyplæringsarkitektur, mer spesifikt en autoencoder. Autoenkodere er kjent for å være gode til å oppfatte egenskaper ved inputdata, en egenskap som kommer godt med når fidgety bevegelser fra spedbarn skal gjenkjennes. Fidgety bevegelser er aldersbestemte bevegelsesmønstre, og fravær av disse er en sterk indikasjon for CP. Bevegelsesdataen er generert ved hjelp av en CIMA-Pose tracker som ble brukt på de originale videoene av spedbarna.

Visjonen til dette prosjektet er å bidra til at InMotion prosjektet ved St. Olavs Hospital i Trondheim klarer å utvikle en automatisk datamaskinbasert CP prediksjon. Vårt hovedbidrag fokuserer på å belyse de relevante dataegenskapene som 2D bilder, og implementere en autoencoder for CP klassifisering.

Funn gjort i løpet av prosessen inkluderer evaluering og innsikt om den foreslåtte datarepresentasjonen, samt en diskusjon og forslag til videre arbeid. Erfaringer knyttet til dyplæringsmetoden indikerer at andre maskinlæringsalgoritmer bør undersøkes nærmere.

Masteroppgaven konkluderer med at den foreslåtte datarepresentasjonsmetoden ikke er ideell for en autoencoder, før InMotion prosjektet får anbefalinger for videre arbeid i prosessen med å utvikle en automatisk CP prediksjon slik at en tidlig og pålitelig CP diagnose kan gis.

Preface

The following master thesis is written in collaboration between Mathilde Theisen and Marie Ling Wiik and concludes a five years Master of Science degree in Computer Science at the Norwegian University of Science and Technology.

This master thesis is written as a part of a collaboration initiative between St. Olavs University Hospital and the Norwegian University of Science and Technology. The main supervisor of the project has been Heri Ramampiaro, Associate Professor at the Department of Computer Science. Our co-supervisors, Espen Alexander F. Ihlen, Associate Professor, and Daniel Groos, PhD Research Fellow, are both from the Department of Neuromedicine and Movement Science.

Acknowledgement

First of all we would like to express our special thanks of gratitude to our main supervisor, Associate Professor Heri Ramampiaro, for giving us the opportunity to work on this exciting and meaningful project, and giving valuable guidance, insight and supportive encouragement.

We are also thankful to our co-supervisors, Associate Professor Espen Alexander F. Ihlen and PhD Research Fellow Daniel Groos, who provided expertise that greatly assisted the research and made good suggestions for our work. We would also like to thank Researcher Lars Adde for valuable feedback and great insight on cerebral palsy.

Two very important persons, who have cheered during hard times, made brownies when the workload felt too heavy and contributed in this thesis are our boyfriends, Johannes Andersen and Ole Ravna. Thank you for the pretty figures and the recreation of old versions of the data representation. We love you to the Moon and back.

Last but not the least, we would like to thank our friends and families for supporting us throughout writing this thesis and our lives in general.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Cerebral Palsy (CP) is the most common physical disability in childhood [4]. An article from 2012 describes CP as *an umbrella term for conditions that are characterised by a non-progressive, but not unchanging, motor impairment related to brain injury early in development* [5].

Due to technological advances and enhanced medical care the survival rate for preterm infants with an extremely low birth weight have improved over the last decades. These infants are at high risk for brain damage [6]. 2% of live-born infants are afflicted by CP, and for preterm infants this risk increases. The risk of CP for infants born in week 22 and 23 is 22%, and for infants born in week 24 this number is 17% [7].

Diagnosing CP is challenging. In Europe there exists training courses for clinicians to learn the art of recognising specific movement patterns that characterises a normal infant. An early prediction of CP can be given if a clinician observes a high-risk infant of 2-4 months for 15 minutes. This observation can be done at the bedside as well as remotely. Abnormal or absent of the mentioned movement patterns is a strong indication of CP [5][6][8]. The number of clinicians with the knowledge to decide if an infant most likely has CP is limited. In order to spread the knowledge about early CP prediction and make it more available, a larger research project was initiated at St. Olav's University Hospital in Trondheim, Norway. The project, InMotion, aims to develop an automatic computer-based system to do the same prediction as the one that have successfully completed formal training in movement assessment. It focuses on an early and more precise

prediction of CP. Researchers in the InMotion project have been working over 15 years with assessing the possibility of early CP detection. During these years, they have collected a large dataset of infant videos from hospitals around the world.

An automatic method of predicting CP is advantageous in several ways. The prediction can help clinicians to determine an early diagnosis which increases the chances of more efficient treatment. The earlier a patient can begin treatment the better since the brain is more plastic when the children are around two years and younger. A plastic brain can adapt to change easier and can learn to compensate for the damaged parts. Earlier diagnosis is also especially important to the parents, who's fear of the consequences from the infant's brain injury can lead to unnecessary stress and worry over a longer period of time, as well as depression. Receiving a diagnosis also streamlines appropriate funding and social support. In 2013 the American Academy of Cerebral Palsy and Development Disability observed that children with CP received care dependent on where they lived and who they saw, rather than by their condition [9].

An automatic method of predicting CP means no need for specially trained clinicians and expensive equipment, which in turn results in a more time efficient and less expensive diagnosis. This again results in a method that that is available to even more people around the world.

1.2 Problem Statement

1.2.1 Problem Specification

The main goal of the InMotion team is to have an automatic computer-based classification/prediction system that can with the same or better accuracy as clinicians predict if an infant likely has CP or not. They have collected video data over many years, and they have proven that fidgety movements are indications of movement development so clinicians can, with decent accuracy, tell manually if it is likely that the infant has CP. Now the challenge is to see if a computer can be able to do the same. Last year the project achieved a computer vision algorithm where seven body parts of the infants movements was tracked by using deep learning, extracting only the position of these body parts. As the next step of this pipeline we will attempt to build upon this and test if the extracted data can be used as input to a different deep learning algorithm to make a prediction of CP. Our main challenge is that a deep learning algorithm is not always intuitive and often is a black box, meaning that it is not trivial to understand exactly what the algorithm learns and which biases affects the resulting prediction. This means we need to

find a suitable model as well as a data representation to represent the movements in a meaningful way.

1.2.2 Goals and Research Questions

When we first met this master thesis its description was as follows:

1. To identify the cerebral palsy-related movement of infant body segments (legs, arms, trunk and head) from the database of video recordings using computer vision and machine learning algorithms.
2. Make this procedure time effective, feasible and available for researchers within the medical field without any technical or computer vision expertise.

This was a vague description with a wide scope, and in order to make it more manageable we wrote our own task description:

In this master thesis we want to continue the work of the InMotion project, and therefore implement a data representation that uses the already extracted data. A first step of making this CP diagnosis process less black box is to generate a data representation that is understandable for humans, as well as for the machine learning algorithms. The basis for the choice of representation was made in the Autumn of 2018, in our previous work [10].

In order to validate the data representation we want to make and train a Deep Learning algorithm, more specifically an autoencoder, that can extract the CP-related movements of infant body segments perform, and further make an accurate prediction of CP. The autoencoder will be trained and tested on images generated with our proposed data representation.

This description summarises what this Master thesis aim to achieve. More specifically the goals are addressed in the following research question and subquestions:

RQ: *Can a deep learning classification/prediction system recognise fidgety movement patterns represented as human-understandable images, with data extracted from video recordings, sufficiently to give an accurate prediction of Cerebral Palsy in an infant?*

RQ1: *Can time series data of infant movements be represented as a 2D image in such a way that a deep learning algorithm can recognise healthy movement patterns, and, in addition, be understandable for the human eye?*

RQ2: *Can an autoencoder used as a feature extractor learn to extract the infant fidgety movements from the above data representation, in such manner that it becomes a reliable prediction that can compete with toady's General movement assessment?*

1.2.3 Scope

As part of a much larger system we are not making a complete system but focus more on a proof-of-concept approach. This means we will not be concerned with commercialisation issues or the practicality of making a fully functional system including performance of the complete system. We are limited by time and resources in terms of our own methods performance and cannot test all possible alternatives even if we would like to. This thesis will solely look at

1. data representation in a human understandable manner
2. deep learning methods, specifically convolutional neural networks

and will omit statistical methods or other machine learning methods as they will be further explored by others within the research team. Among them, fellow student Marie S. Kristiansen from Department of Engineering Cybernetics study the use of statistical methods and how they can solve the same task in more details and we do not wish to overlap her work.

1.3 Research Method

This project started with a literature study to gather information about the state-of-the-art method related to the problem, but it evolved fast into a wide scoped study in order to examine what's done within the field of classification of movement tracking. Based on an evaluation of these methods we propose a 2D data representation, in which we believe to be as human friendly as it can be. In the literature study different deep learning methods were also investigated, and in co-operation with the InMotion research group at St. Olav's, we hypothesised that an autoencoder could be used as a feature extractor. Before the network were implemented, the extracted data are cleaned and represented as images. The experiments are carried out, and results are compared to two baseline models, before the project ends with a discussion of the findings, and a conclusion are extracted.

1.4 Context

This project is part of the larger research project, InMotion, at St. Olav's University Hospital in Trondheim, Norway. InMotion is a cooperation between the Department of Clinical and Molecular Medicine, and the Department of Neuromedicine at St. Olavs Hospital, and the AI-LAB at the Norwegian University of Science and Technology (NTNU), and consists of both clinicians and researchers. Our project aims to help in the process of reaching the goal of an automatic method of predicting CP without the need of specially trained clinicians and expensive equipment, so that the method becomes available to even more people around the world.

1.5 Contributions

This master thesis provide valuable insight for the InMotion project in several ways. First, it evaluates the CIMA-Pose tracker's quality of the tracked subjects. This information can contribute in the process of customising the datasets used for training, validation and testing in further experiments, as well as a guideline for which subjects that need an extra follow-up in the process of generate a good data representation. Another valuable learning this project provides is an evaluation about one type of human understandable data representation, the process of getting there, and suggestions for previous representations. It also provides some initial testing in using deep learning where the project earlier has primarily used statistical methods. The use of an autoencoder as feature extractor will be another step in attempting to find a good representation of relevant patterns. This information is valuable in the future work of InMotion, especially for the next year's master students that will continue the process of developing an automatic computer-based classification/prediction system.

1.6 Thesis Outline

We inform the reader that parts of this master thesis is taken form or adapted from from our own previous work in autumn 2018 [10]. This is especially true for chapter 2 and chapter 3.

This thesis is structured as follows:

- **Chapter 1 Introduction** introduces the project, its background and its goals.
- **Chapter 2 Theoretical Background** presents some theoretical background and the main concepts that this project is built upon.
- **Chapter 3 Previous Work** gives an overview of previous work both within the InMotion project, as well as some recent literature and publications.
- **Chapter 4 Method** contains the approach of the work methods, as well as the work done in this project.
- **Chapter 5 Results** presents the findings from the training of deep learning methods.
- **Chapter 6 Discussion** gives a discussion on the results, experiences and the choices made for this project.
- **Chapter 7 Conclusion and Future Work** concludes this report and provides insight into what needs to be done in future work.

Chapter 2

Theoretical Background

2.1 CP and Infant Spontaneous Movements

2.1.1 Cerebral Palsy

According to Rosenbraum et al *Cerebral palsy describes a group of permanent disorders of the development of movement and posture, causing activity limitation, that are attributed to nonprogressive disturbances that occurred in the developing fetal or infant brain. The motor disorders of cerebral palsy are often accompanied by disturbances of sensation, perception, cognition, communication, and behaviour, by epilepsy, and by secondary musculoskeletal problems* [11].

The human brain is complex, and a damage to it will never have the same outcome. As a result, every child with CP is different, and each child's outcome will be unique. A common CP classifier system is the Gross Motor Classification System, which categorise CP into 5 levels of severity [9]:

I: independently ambulates
II: independently ambulates with limitations
III: ambulates with walking aids
IV: independently mobilises with powered mobility
V: dependent for all mobility

2.1.2 General Movements

From early fetal life and until the end of the first half year of life the human nervous system generates a variety of age-specific, spontaneous movement patterns. One set of these movement patterns are called general movements (GM).

Normal general movements are at any age characterised by [12]:

- a gradual onset of the movement
- a waxing and waning of movement intensity
- movement of the whole body; each body part moving with variation in duration and timing of the movements
- fluency
- complex movement trajectories

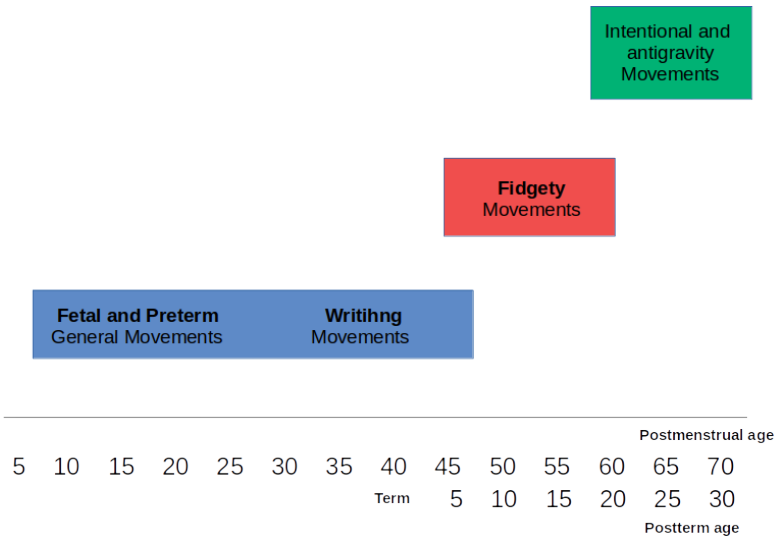


Figure 2.1: Developmental course of general movements

The first movement patterns that starts to emerge begins in the fetus at 9 to 12 weeks post-menstrual age. This is illustrated in the blue box in figure 2.1. The movements include twitches, stretches, isolated limb movements and breathing movements. These spontaneous movement patterns continue after birth, independent of when birth occurs [13].

Fidgety Movements

At 6 to 9 weeks post term age, GMs with a writhing character gradually disappear, and fidgety GMs gradually emerge. These rhythmical, small, circular movements are in all directions, of moderate speed, and are generated continually when the infant is awake. They only stop during fussing and crying, or when the baby is distracted by its surroundings. Fidgety movements are present until intentional and antigravity movements starts to emerge at the end of the first half a year of life [13].

2.1.3 General Movement Assessment

Gestalt perception [12] is a method that clinicians use to evaluate and observe the complexity of GMs. In order to identify fidgety movements, they look at the overall movement repertoire of the infant body. Clinicians with the right knowledge and required training are able to recognise if an infant does not show these fidgety movements.

Abnormal quality or absence of fidgety movements indicates that there is an abnormal development in the brain, and is believed to have a strong association with CP [14]. Prechtl's assessment of general movements can therefore be used as a prognostic tool to identify infants with neurodevelopmental disabilities [6] [8].

In a study from 2013, the absence of fidgety movements was utilised to predict CP on 326 children. The study got a sensitivity of 98% and a specificity of 91% [5]. In medical diagnosis, sensitivity is the ability to correctly identify those with CP, the true positives, as shown in equation 2.1. Specificity is the ability to correctly identify those without CP, the true negatives, as shown in equation 2.2.

$$Sensitivity = \frac{True\ positives}{True\ positives + False\ negatives} \tag{2.1}$$

$$Specificity = \frac{True\ negatives}{True\ negatives + False\ positives} \tag{2.2}$$

Although general movement assessment (GMA) is efficient for recognising fidgety movements, it has its limitations. The need for a trained clinician to observe the infant is impossible to fulfil in most parts of the world. Also, the technique is

qualitative which means that the result depends on the clinician's interpretation of the movement patterns.

Other factors that make cerebral palsy hard to diagnose is that CP can be hard to differentiate from other progressive conditions. Also, children develop with various speed. A normal child learns to walk between 9 and 18 month old, and with normal development it learns to talk when it is 8 month to 3 years old.

2.2 Time Series Analysis and Representation

A time series is simply defined as a sequence of observations x_t, each recorded at time-step t, which means that a lot of sequential data can be interpreted as time series. Time series are often used for predictions and simulations based on historical data, like weather forecasting, trends in the stock market and looking at patterns over time [15]. This project will focus on the latter, attempting to look at movement patterns over a given period of time and see if the patterns can be categorised to give a probable CP diagnosis in the future.

Time series are often divided into two groups, univariate or multivariate data, based on the number of features present. The univariate data only concerns one feature at the time. Examples are house pricing, temperature, sensor signals etc. Multivariate data look at more than one feature and these may have correlations and important relations between them. Our data is an example of multivariate data with each body part being one feature.

When working with time series it is common to distinguish between classification problems or forecasting problems. Forecasting uses historical recordings to predict the future, like weather forecasting or predicting trends in the stock market. A classification problem, on the other hand, tries to fit observed data into a finite number of categories. One example is to look at sign language movements and try to determine what word or letter that specific movement represents. Likewise, we will attempt to make a classification based on detected movement patterns where the categories will be if the model thinks the infant has CP or not. We use both the terms *prediction* and *classification* for our model as we think of our classification into CP or $NotCP$ as a prediction of CP diagnosis. It is primarily on the basis of classification that we did our research and we think of this as closer to a time series classification problem rather than forecasting.

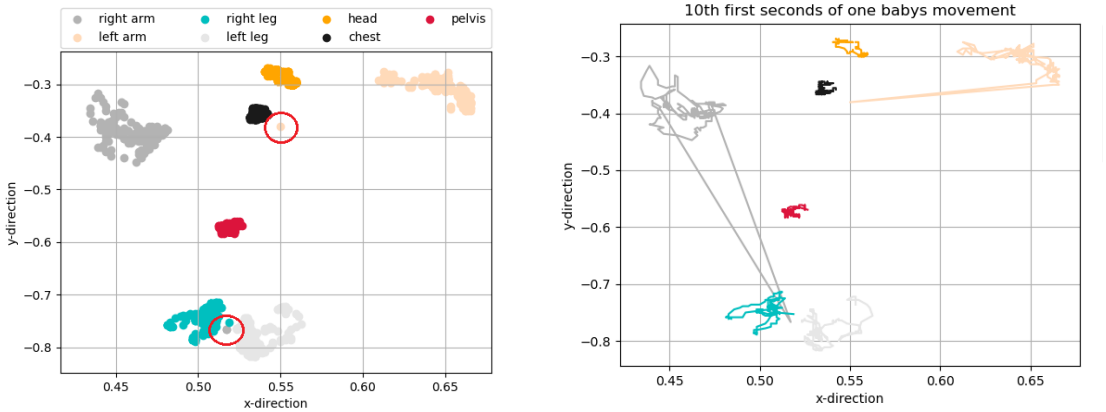
It is also common to do statistical analysis and/or preprocessing of time series data to look for trends, seasonal components, sudden changes or outliers. This is especially important with prediction models where such changes and patterns in

the data can highly influence the outcome of the prediction. A statistical analysis can also be used to determine what type of representation could fit the data the most but the representation could also dictate which analysis is possible to perform depending on what information one seeks to obtain. While we limit our study of such methods for this project to outlier detection on an image representation, we are aware that it could be useful or necessary to expand on such analyses in future work.

2.3 Outlier Detection

Outliers are observations that are considered to be unusually far from the bulk of data.

The dataset of videos collected by the InMotion project, have been ran through a tracker, which returns coordinates of motion per time. This coordinate dataset includes errors from the tracking phase, as illustrated in figure 2.2. Sending images with outliers into a deep learning network will disturb the training phase, and will at worst confuse the network to learn connections between outliers and CP. The outliers therefore have to be handled.



(a) Outliers marked with red circles

(b) Outliers shown in an image with lines that reflect the child's movement

Figure 2.2: Movement of a child, with errors from the CIMA-tracker, illustrated in two ways.

There are many statistical tests that are designed to detect outliers. In this thesis standard score will be applied and evaluated, as well as some clustering algorithms.

2.3.1 Statistical Methods

Statistical inference consists of those methods by which one makes inferences or generalisations about a population. In classical methods of estimation a parameter, inferences are based strictly on information obtained from a random sample selected from the population. Populations are collections of all individuals or individual items of a particular type [16].

Standard Score

The standard score of an observation is a metric that indicates how many standard deviations a data point is from the sample's mean, assuming a Gaussian distribution [16]. In other words, how many standard deviations is a point outside the mean. The standard score is calculated as

$$z = \frac{x - \mu}{\sigma} \tag{2.3}$$

where μ is the population mean (2.4) and σ is the standard deviation of the population (2.5), both illustrated in figure 2.3:

$$\mu = \frac{\sum_{i=1}^{n} x}{n} \tag{2.4}$$

$$\sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n}} \tag{2.5}$$

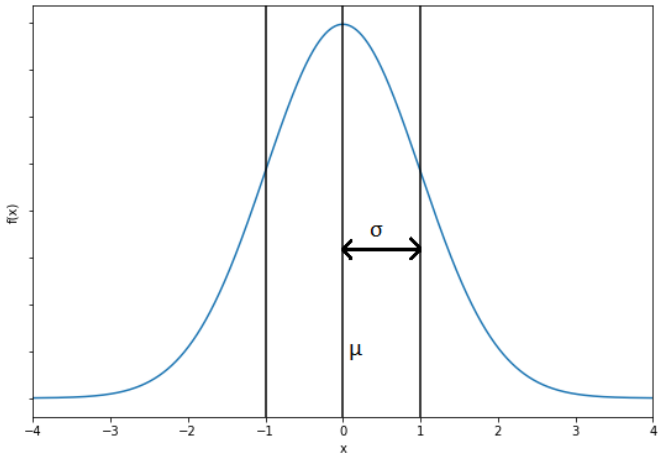


Figure 2.3: Normal Gaussian Curve

Standard score can be used as a outlier filter. Every motion coordinates/ data points that are further away from the cluster mean than a given threshold can be labelled as noise/outlier.

Other mathematical and statistical methods

There are a lot of other statistical and mathematical methods that in some way could be utilised to detect and fix outliers. An other member of the InMotion project is looking into this, and therefore this master thesis does not have more of these methods in focus.

2.3.2 Cluster Analysis

An other way of detecting outliers is cluster analysis. Cluster analysis divides data into groups (clusters) that are meaningful, useful or both, and have played an important role in fields like pattern recognition, statistics, and machine learning. Clustering, an entire collections of clusters, can be used both for understanding and for utility. The first one plays an important role in how humans analyse and describe the world, while the latter can describe cluster analysis as the study of techniques for finding the most representative cluster prototypes, that is, data objects that are representative of the other objects in the cluster [1].

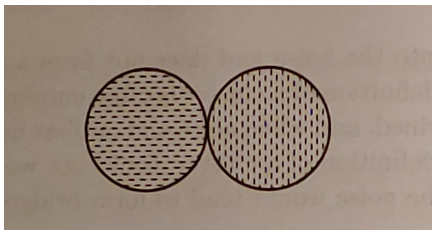
When grouping data, the goal is that the objects within a group be similar to one another and different from the objects in other groups. The greater the similarity

within a group and the greater the difference between groups, the better or more distinct the clustering [1].

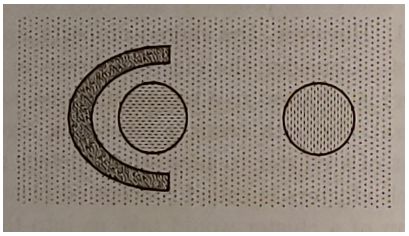
Clustering aims to find useful clusters. There are several types of clustering, and different notions of a cluster, and the following two boxes aims to give a short introduction of some of them.

A *partial clustering* does not assigns every object to a cluster, whereas a *complete clustering* does. Partial clustering is beneficial when an object does not belong to a well-defined group, which often is the case for noise and outliers.

An other type of clustering focuses on whether or not the clustering is nested. A *partitional clustering* divides the data points into non-overlapping clusters, whereas a *hierarchical clustering* allows clusters to have subclusters.



(a) Center-based clusters. Each point in a cluster is closer to the center of its cluster than to any other cluster.



(b) Density-based clusters. Regions of high density are separated by regions of low density.

Figure 2.4: Different types of clusters. Illustrations borrowed from [1].

The term *prototype-based* or *center-based* cluster is employed where each object is closer to the prototype that defines the cluster than to the prototype of any other cluster. The prototype is in this case often the mean of the points in the cluster. A cluster that is defined by it's similar density is called *density-based* (figure 2.4b, and the term is used when noise and outliers are present [1].

In the following sections some clustering techniques are presented.

K-means

K-means is a prototype based clustering technique, and one of the oldest and most widely used clustering algorithms. The basic algorithm starts with choosing K initial centroids, where K is the user-specified numbers of clusters. Every point is then assigned to the closest centroid. Every collection of points assigned to the same centroid is a cluster. Based on the assigned points, each centroid for each

cluster is then updated, and the assignment and updating are repeated until no points changes clusters. K-means is described in algorithm 1 [1].

Algorithm 1 Basic K-means algorithm

1: Select K points as inital centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

Step 3 in the algorithm states that each point should be assigned to its closest centroid. There are many similarity measures that can be used, but Euclidean distance is commonly used if the data contains points in Euclidean space.

Even though K-means is a well known clustering algorithm it does not handle noise and outliers. DBSCAN, SNN density-based clustering, Chameleon and CURE are other algorithms that deal with noise and outliers.

DBSCAN

DBSCAN is a density-based clustering algorithm that locates regions of high density. Points in low-density regions are classified as noise. Here, a center-based approach are presented for defining density. Thus, density is estimated for a particular point in the dataset by counting the number of points within a specified radius, *Eps*, of that point (including itself) [1].

In DBSCAN all data points are classified as either a *core point*, a *border point*, or a *noise point*:

- **Core points** describes points in the interior of a dense region, that is points that have $MinPts$ number of neighbours within a given neighbourhood, *Eps*.
- **Border points** are not core points, but are in the neighbourhood of a core point.
- **Noise points** are points in a sparsely occupied region, and are neither a core point or a border point.

With these three types of points defined, the DBSCAN algorithm works as follows. First, all points are labelled. Then all core points within *Eps* distance of another are assigned to the same cluster, and border points are putted in the same cluster as its core point. Noisy points are, as described in algorithm 2, eliminated.

Algorithm 2 DBSCAN algorithm

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

Since all noisy point are grouped together with the same label, they can be handled instead of elimination.

SNN density-based clustering

The SNN algorithm is also a density-based clustering algorithm. For high-dimension data, Euclidean distance is meaningless, so DBSCAN with its centre-based view will get in troubles.

Instead of using Euclidean distance a similarity measure like Shared Nearest Neighbour Similarity (SNN Similarity) described in algorithm 3, can be used. SNN is based on the principle *if two points are similar to many of the same points, then they are similar to one another, even if a direct measurement of similarity does not indicate this* [1]. This addresses the problem of finding clusters of varying density. Since SNN similarity takes the context of an object into account, it also addresses the problem where two objects are relatively close, but belongs to different classes.

Algorithm 3 Computing shared nearest neighbour similarity

1: Find the k-nearest neighbours of all points.
2: **if** two points, $\mathbf{x}$ and $\mathbf{y}$ are *not* among the k-nearest neighbours of each other **then**
3: similarity($\mathbf{x}$,$\mathbf{y}$) $\leftarrow 0$
4: **else**
5: similarity($\mathbf{x}$,$\mathbf{y}$) $\leftarrow$ number of shared neighbours
6: **end if**

SNN density-based clustering is suited for data sets with a wide density variation. Points in regions with low and high density will have high SNN similarity, while points between clusters will tend to have low SNN similarity [1]. The algorithm utilises the advantages from a similarity measure, and combines it with DBSCAN, as described in algorithm 4.

Algorithm 4 SNN density-based clustering

1: Compute the SNN similarity graph.
2: Apply DBSCAN with user-specified parameters for Eps and $MinPts$.

After computing the SNN-similarity graph, DBSCAN labels all data points as their respective clusters or as outliers. The outliers can then be handled before the images are sent into the network.

2.4 Colour Image Processing

Using colours in image processing often simplifies object identification and extraction.

2.4.1 Colour Fundamentals

Colours that humans perceive are determined by the nature of the light reflected from objects. Light can be characterised by its intensity. Achromatic light, light that is void of colour, has intensity as its only attribute. Intensity level (or grey level) refers to a scalar measure that ranges from black, to greys, and to white, and is therefore often used to denote achromatic light. Chromatic (colour) light spans in the range of 400 to 700 nm in the electromagnetic energy spectrum [17], as illustrated in figure 2.5.

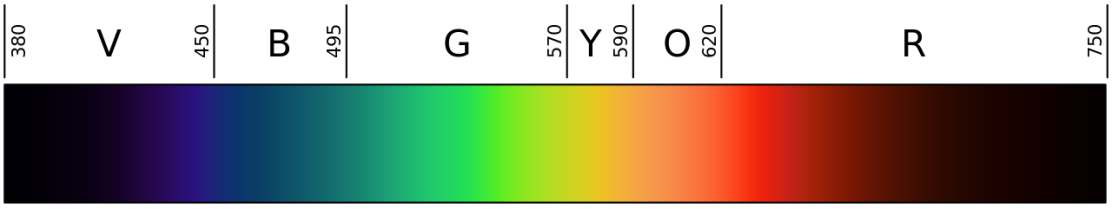


Figure 2.5: Linear visible spectrum, figure borrowed from [2].

Chromatic light are mainly described by three quantities: brightness, luminance and radiance. Radiance and luminance are respectively the total amount of energy flowing from the source, and a measure of the amount of energy that an observer perceives from the source. Brightness is a subjective descriptor, and includes the achromatic notion of intensity [17].

In the human eyes, cones are responsible for colour vision. The cones are divided into three principal categories, each sensitive to either red, green or blue light. Because of this, humans see colours as combinations of the primary colours: red (R), green (G) and blue (B). When added together, primary colours can produce secondary light: magenta, cyan and yellow [17].

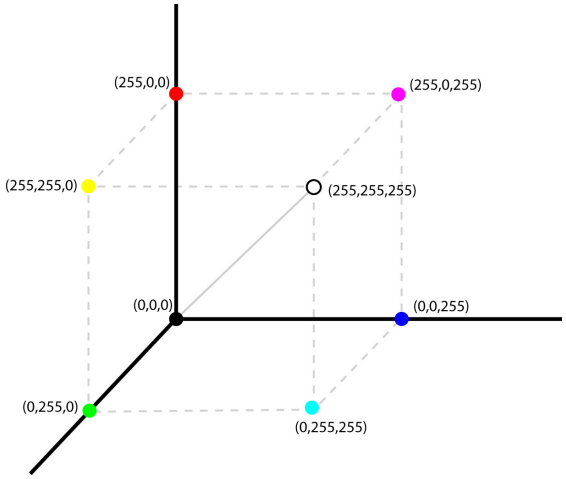
2.4.2 The RGB Colour Model

A colour model is a specification of

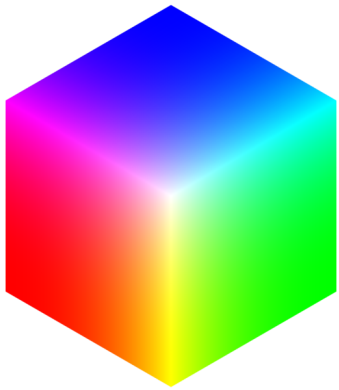
1. a coordinate system, and
2. a subspace within that system, so that each colour can be represented as a single point within that subspace

and has the purpose of specifying colours in some standard way [17].

One of the most common colour model is the RGB model, where each colour is described as a combination of red, green and blue. Figure 2.6a illustrates the RGB model, which is based on a Cartesian coordinate system. Black is at origin, and white is at the corner farthest from black. The greyscale is the line joining black and white. Colours in the RGB model are defined by vectors extending from the origin, and the values of red, green and blue are in the range [0,1] [17]. In digital images, the colour values are scaled, and R, G and B are represented in the range [0,255].



(a) A schematic RGB colour cube. Illustration borrowed from [18].



(b) RGB colour cube. Illustration borrowed from [19].

Figure 2.6: The RGB colour cube.

2.4.3 Pseudo Processing

One area of colour image processing is pseudo processing, which focuses on assigning colours to greyscale intensities. It is mainly used for human visualisation and interpretation of greyscale events in an image, since humans can discern thousands of colour shades, but less than two dozen shades of grey [17].

Colour Intensity

As mentioned in section 2.4.1 colour intensity is a measure that ranges from black, to greys and then to white. In the RGB colour cube, see figure 2.6a, intensity is illustrated by the line joining black and white. From this it can be derived that all colours on a cross-section normal to the greyscale line will have the same intensity [17].

2.5 Deep Learning

Within the field of artificial intelligence, there are different approaches as to how we can make a machine "intelligent". One approach that has gained a lot of attention over the last decades is machine learning. The goal of machine learning is to

have the algorithm itself extract knowledge from raw data and recognise patterns without having them predefined by us humans. With the increased availability in data and computational power, machine learning algorithms have gotten a major boost and proven to achieve good results on many applications. Among the machine learning methods, deep learning and deep neural networks have gotten increasingly more popular as they have shown new state-of-the-art performance in a very short amount of time. They have also proven to be good at more complex learning tasks. Deep learning has been applied to an increasing set of different problems, and different fields of research are trying out these methods on a wide range of datasets and applications [20].

The term "deep" is referring to the many layers utilised in the latest architectures and was adopted by researches after a breakthrough in 2006 where Hinton [21], and then later in 2007 Bengio and Ranzato [22][23], showed that it was possible to train deep multi-layered networks using a strategy called greedy layer-wise pre-training. There is no formal definition as to how deep or how many layers a network has to have to be called a deep network. The network is learning features in a hierarchical manner from simple to more abstract or complex concepts, and in order to do that one needs multiple layers which correspond to deeper networks, hence the name of deep learning.

2.5.1 Artificial Neural Networks

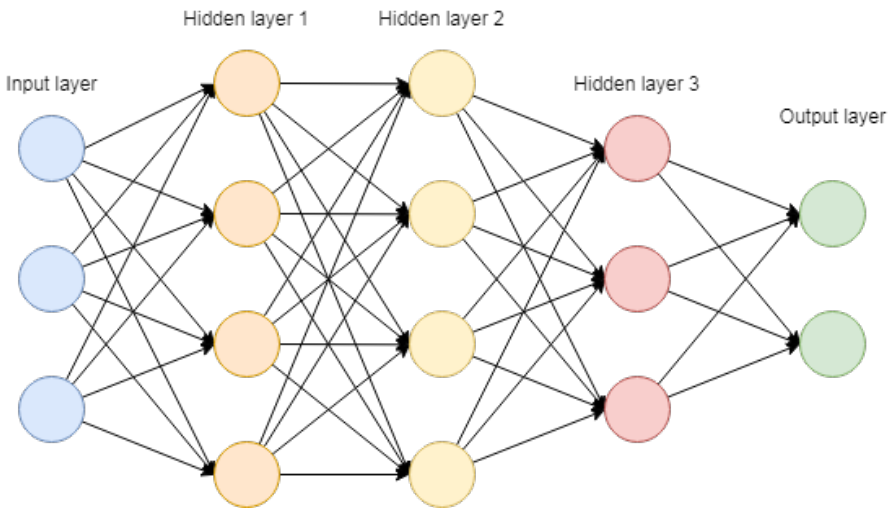


Figure 2.7: Example of a simple fully connected network with three hidden layers.

The basic component for a neural network is the concept of artificial neurons as computational units in a graph structure. As shown in Figure 2.7 a regular network can be described as layers of neurons connected in a chain from the input layer, through one or more hidden layers and to the output layer. Each neuron takes each incoming input value and multiplies it with an individual weight value, sums the results of the multiplications and feeds them through an activation function which outputs a single value based on that. The weights in the network are what enables the network to learn by updating these during training. The activation function is often added to provide some non-linearity as well as forcing the output to be within a given range that is useful for further computation. In-depth details of common activation functions can be found in Goodfellow's *Deep Learning* [20, 6.3].

The field of deep learning can be roughly divided into some main architectures. These architectures have traditionally been developed for specific areas, although researchers now study the possibility of using them interchangeably or in combination with each other. Most networks are feed-forward networks, meaning that the data only flows from the input through the network to the output with no loops. It can be compared to a directed acyclic graph (DAG). The most general architecture is the extension of a regular, fully connected neural network with many layers. Unfortunately, the more layers a network have the more computationally expensive it is to train, and it does not always suit the type of data that is provided. Therefore, different architectural structures have been developed to fit different types of problems. For sequential data that needs to maintain a sequential or temporal ordering, recurrent neural networks were developed. Recurrent neural networks (RNN) do have loops, meaning that the output is fed back as the input to the same neuron. For image processing, convolutional layers was introduced not only to reduce the feature space but these networks worked really well with features and feature learning. This is similar to how concepts are learned in the brain, starting from simple to more complex as the depth of the network increases. Based on our literature review done last autumn we chose to not look further into RNNs although they are in general the first choice for time series. Instead we looked at a way to represent the time series i such a way that a convolutional neural network (CNN) could be easily applied. The argument for this was that a visual representation corresponded better with the original video data and would also be more intuitive to clinicians and those without insight into technical details. More about our choice of data representation will be described in section 4.2.

Convolutional Neural Network (CNN)

A regular multi-layer neural network, as explained above, is fully connected or dense, meaning that output from all neurons is input to every neuron in the next layer. A disadvantage of this is that these networks don't scale very well. In images for example, the higher the resolution, the larger the number of features (and weights) become. The dimensions of the feature space become so huge it becomes disproportionate with the data available and the network is unable to learn properly. In addition, time and resources often put a constraint on how large a fully connected network can be before it is too costly and practically impossible to run. Figure 2.8 shows an example of what would be considered a relatively small network today because it uses convolution. Had the same input picture of 128x128 pixels been used in a regular fully connected structure the number of weights between the two first layers would be 37748736 given the second layer had half the number of input nodes. With convolution the number of parameters between the two layers can be reduced to only 131072 if the numbers of kernels/filters is 8 and the second layer is a convolution layer, greatly improving memory usage.

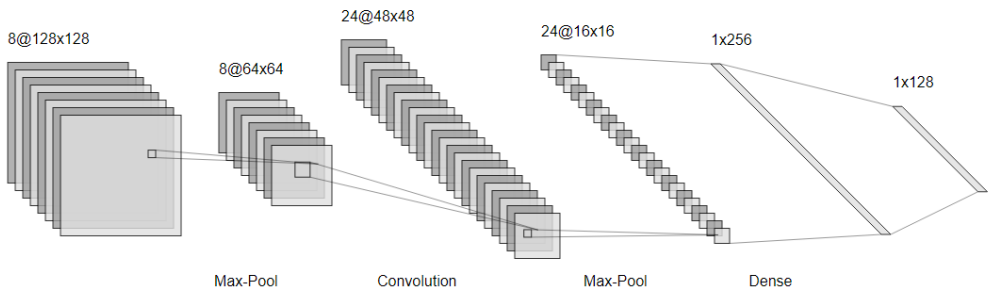


Figure 2.8: Example of an arbitrary convolutional network. Courtesy of the visualisation tool by Alexander LeNail [3]

The idea of convolutional networks is largely biologically inspired and one of the first networks was introduced as early as 1980 when Fukushima [24] presented his *Neocognitron* based on studies about the visual cortex and visual fields by Hubel and Wiesel [25]. It was further popularised by LeCun [26] during the 1990's and introduced many of the elements that modern convolutional neural networks are built upon, like using gradient descent learning and backpropagation. This allowed for bigger and deeper networks to be trained on larger datasets, leading to the breakthrough in 2012 when AlexNet won the IILSVRC challenge [27]. In addition to the biological inspiration from the visual system in the brain, Goodfellow [20]

lists three main attributes of using convolution that makes convolutional networks work so well with images: sparse interactions, parameter sharing and equivariant representation.

To ensure sparsity a convolution layer uses the concept of a perceptive field where each neuron only looks at a *smaller* part of the layer underneath. A filter, also called kernel, is applied to the perceptive field which maps the underlying subset of the input layer to a feature map based on the filter applied. An example can be seen in Figure 2.9 where I is the input layer with the current perceptive field marked in light green on the input (blue), and K is the filter marked in green, a so called kernel. Marked in darker green is the resulting computation in the output feature map (red) between the data in the perceptive field and the filter. Multiple filters can be applied to the same input and they generate multiple feature maps. This can reduce the number of features and extract higher level concepts than the previous layer.

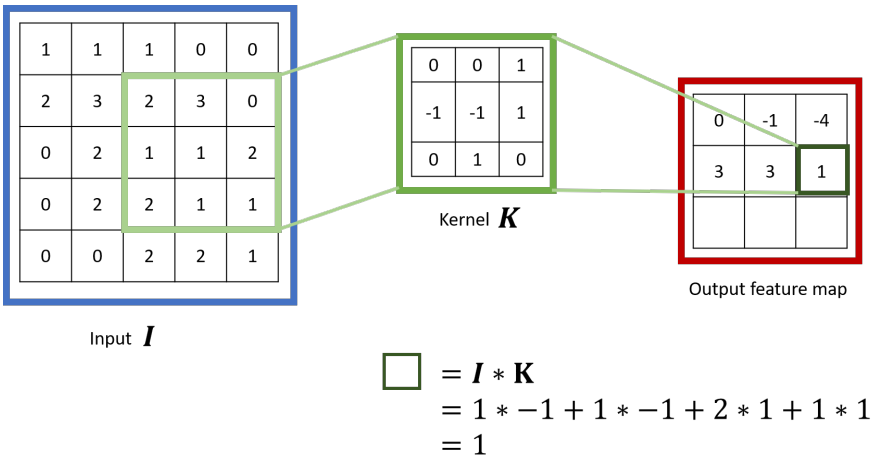


Figure 2.9: Illustration of simple convolution where I is the input to the layer, K is the kernel and the resulting computation of $I * K$ results in one specific feature map for this kernel.

Convolutional neural networks also uses the concept of parameter sharing which reduces the number of trainable parameters that need to be updated. This not only improves computational cost but adds a layer of generalisation that one does not get from a "regular" layer. Using image as an example again, it means that if the network detects a shape in one location, it is not limited to only recognising that shape in that very location. This helps the network achieve equivariance translation, though it does not solve for all transformations. Convolution layers are usually followed by a pooling layer and together they are counted as a convolution

unit. Pooling layers contribute to further reduce the number of features and the most common types of pooling are max pooling and average pooling. Pooling also helps the network achieve approximately local translation invariance which prevents the network from having big changes in the pooled outputs when there is smaller changes in the translation of input, e.g. when exact position is less important than whether an feature is present or not. Goodfellow elaborates more on all these three attributes in detail in [20, Chapter 9]

The AlexNet architecture [27] was the breakthrough for convolutional neural networks. Along with the introduction of using GPUs for computation, large image datasets, like the ImageNet database [28], and improved methods like using ReLU instead of sigmoid activation function, researchers could finally train deeper networks than before. AlexNet, as mentioned won the ILSVRC challenge in 2012, was the first to apply deep convolutional layers to solve the task of image recognition and achieved state-of-the-art results by a considerable amount. Since then, CNNs has become the number one choice for computer vision and image analysis. AlexNet has later been beaten by numerous networks like VGG [29], ResNet [30] and DenseNet [31], and researchers are still working to improve these or find new methods to solve even more problems.

2.5.2 Learning

The majority of machine learning algorithms are supervised, meaning that the dataset contains the true target value, also called label, for each example case. The algorithm can then compare the output prediction/classification of the model to the label and adjust the weights thereafter. With unsupervised learning, the target value is unknown and the algorithm needs other means to assess whether it's going in the right direction. It is also possible to have a semi-supervised algorithm where some example cases have labels and others don't. We will primarily focus on supervised learning since our data is labelled but an autoencoder, that we will attempt to use for feature extraction, is generally considered an unsupervised method as the label is just the input itself.

In order to be able to learn effectively with supervised learning, the concept of backpropagation was introduced in 1986 by Rumelhart [32]. Backpropagation allowed the calculated error between the true target value and the predicted value from the output layer to propagate back through the network in order to adjust and update the weights. The error function, or loss, is chosen based on the type of data and problem to be solved and can be calculated in various ways. We refer to [20, 6.2] for a detailed discussion of the most common loss function in use for deep learning today. Most network architectures today uses variations of

gradient descent as the main heuristic on how to update the weights. The goal is to minimise the loss or the error of the network, and by using gradient descent the weights are updated in the direction that minimises the error function with respect to each individual weight.

2.6 Feature Extraction

Feature extraction is an important factor that influences the performance of a neural network (or any other machine learning algorithm). Choosing which features are relevant to represent the desired structure in the data largely decides what type of network is suitable. The goal is also to reduce the dimension of variables under consideration. Ideally, the features need to capture both general information that is common between cases as well as good discriminating features to separate between samples from different classes in a classification task. There are many methods for doing this, like Principal Component Analysis, matrix factorisation and discriminant analysis, just to mention a few. For this project, however, we benefit from the work that was done last year by Groos and Aurlien [33] where the entire tracking method works as a feature extractor.

In addition to having already filtered data from the tracker, we want to see if the feature space can be reduced even more to give a better representation of the patterns we are looking for. A CNN is very good at extracting features and that's why they work so well with images. As discussed in section 2.5.1 the ability to have ?, parameter sharing and equivarians allows the networks learn both general and specific patterns in the input. By using different filters, multiple feature maps are generated, each containing different concepts or information that is then passed on to the next layer. With more layers, more complex concepts can be learned by combining knowledge from the previous layers. Pooling also helps reduce the sizes of these feature maps and forces the network to learn the most important features because of this reduction.

2.6.1 Autoencoders

One way to exploit this ability to reduce the feature space and ensure a limited number of features is to use an autoencoder. An autoencoder is an unsupervised method that takes in an input, transforms it into a code layer and then attempts to reconstruct the input based on the code layer[20, Chapter 14]. The code layer may be any size but is typically smaller than the input, undercomplete as shown in figure 2.10, as it then forces the network to chose a limited number of features that

represents input the best. Traditionally, autoencoders has primarily been used in dimensionality reduction and feature learning, but the concept of latent variables has made autoencoders an important part of generative modeling [20]. We will however focus on the ability of feature reduction and learning.

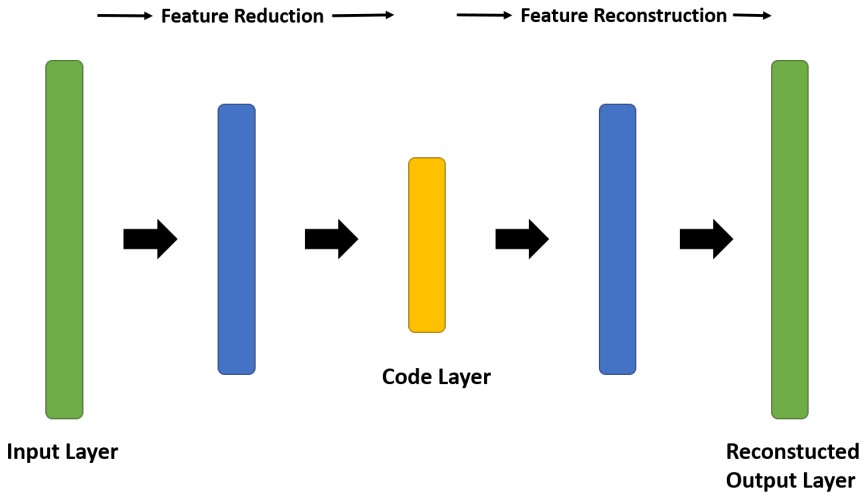


Figure 2.10: Illustation of an undercomplete autoencoder with a smaller code layer than input.

One example where this principle is used is in data compression where the code layer typically is smaller than the input forcing it to only keep those features that the network assumes represents the data best so that it can regenerate a copy as close to the original as possible. This can also be used to eliminate noise or irrelevant information, e.g. a denoising autoencoder [20, Chapter 14.5]. In our case, since the data is images, using convolution and pooling can reduce the feature space significantly by determining the size of the kernels, the number of kernels in each layer and the stride length of the kernels and the pooling layers. By reducing the image size, the autoencoder has a limited number of pixels to store the information on and when upsampling and reconstructing the image, it will preferably be close to the input. A visualisation of the code layer is shown in section 4.6.1.

In many cases, as with our own, the actual reconstruction is not the main goal. By training the network as an autoencoder we hope the code layer captures the most useful properties or patterns in the data that can then be used for further computation. Because of the great imbalance in the dataset, we hope a pretrained autoencoder on a selection of the data will extract some features that is defining of this selection. In addition, an autoencoder does not need much human engineering

of features like some statistical methods, and can be very specifically trained to the dataset. There are however some drawbacks of using autoencoders that one should keep in mind. It can become too specialised, requiring that the input data is not too diverse or sometimes it only learns as much as possible, not always as relevant as possible. Finally it adds extra training time and complexity if used with another model which means added resource usage, hyperparameter tuning and testing.

Chapter 3

Previous Work

3.1 InMotion Project

InMotion is an ongoing research project led by Lars Adde and his research group at St. Olav's University Hospital in Trondheim, Norway. As stated earlier, fidgety movements is an indication of a healthy infant, and can be classified by GMA. In order to make this more available throughout the world, the overall goal of the InMotion-project is to obtain a system that can automatically detect CP, solely based on video recording of the patients. The system should also provide meaningful insights to clinicians, and facilitate the use of intelligent computer-based solution in medical settings.

The InMotion research group have proved that video recordings can be used for qualitative and quantitative analyses of fidgety movements [34]. In order to collect video recordings, the research group have developed a set-up for the recording process that includes a mattress, video camera, camera stand and a suitcase, as shown in figure 3.1 [33]. These suitcases were distributed to different hospitals around the world, and since 2002 the research group have collected videos of both high-risk (term and preterm) and low-risk infants from various countries.

(a) Standardised camera stand

(b) Camera view

Figure 3.1: Standardised set-up for video recordings collected in the InMotion project.

In 2018 Groos and Aurlien developed a model for Computer-based Infant Movement Assessment, called CIMA-Pose [33]. The model uses the raw videos as input and extracts the movements by tracking seven body parts. The extracted movement data is a series of x- and y- coordinates for each body part. The result can then be viewed as a set of time series with the position of each body part for each time step. Their proposed architecture is illustrated in Figure 3.2 [33].

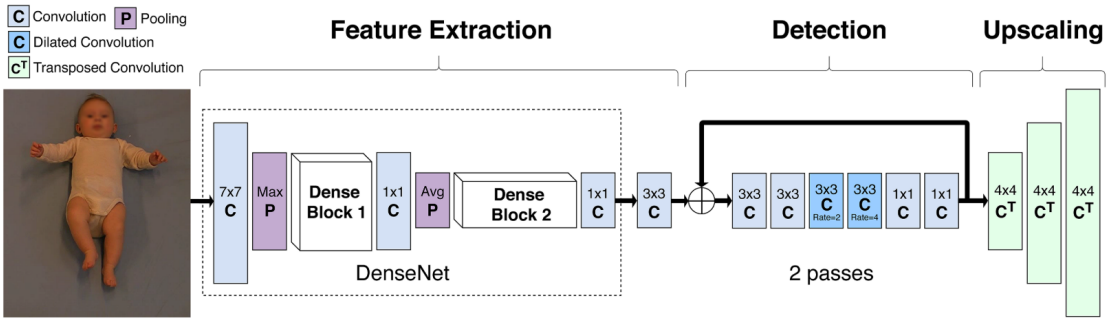


Figure 3.2: The CIMA-Pose architecture

A natural next step in the process of obtaining an automatic CP detector, is to use output from the CIMA-Pose to analyse the movement patterns and see if a method for predicting CP can be obtained. Fidgety movement recognition is of particular interest, as GMA is one of the leading methods for giving an early CP diagnosis (section 2.1).

3.2 Deep Learning on Time Series Classification

Traditionally time series analysis has been done within a lot of different fields, ranging from economy, engineering to medical application. The reason is that a lot of data that has been collected and recorded have a temporal aspect that can hold important correlations and information. Most previous research focus on either forecasting or modelling of data and is thus not relevant to our project as we aim to do classification on patterns. Additionally, time series have often been associated with statistical analysis and methods which is excluded from our study as mentioned earlier. Most methods and work presented in literature on time series classification have been tested on or compared to baseline testing on data from the *UCR Time Series Classification Archive* [35] (Updated version from 2018 [36]) which contains a large collection of different time series data. The data we have in this project does not compare well to the type of data that is in UCR, and so it is not used as a baseline for our model but it is the main baseline for time series classification methods in general.

The following two sections present a selection of some proposed methods from literature where deep learning has been utilised for time series classification. Some comments on some data representation approaches that may be of interest is included. The literature study was primarily done in our previous report and these sections are largely taken from or adapted from what was found then[10]. Only publications and articles that are of recent writing are included here, i.e. from the last five years or so, and they also has to present methods that explicitly concerns time series classification with the use of deep learning as the main architecture.

3.2.1 Methods Based on RNN or hybrid methods

Traditionally in deep learning, sequential data and time series have been strongly associated with methods that are based on RNN architectures. As mentioned in section 2.5 RNNs have looping structures in the network, feeding information of the last step to the next and retaining "memory" in a sequence. Published in 1997, Long Short-Term Memory[37] (LSTM) is still one of the most popular approaches today, and variations of LSTM combined with other methods are keeping up with the state-of-the-art on most of the baseline datasets for time series. Recently, LSTM has more often been combined with convolutional units, which give an indication that combining the strengths of the two is a promising idea. CNN finds important patterns, RNN retains the temporal dynamics and combined they might form a better network. As we do not focus on RNNs for this project we will not go into depth here but refer to our original study for more details on approaches

based on RNN and LSTM.

3.2.2 Methods Based on CNN

There is an increase of researchers in recent literature that looks at the possibility of letting the CNNs handle the temporal dynamics alone, omitting the recurrent layers entirely. Zheng et al. [38] presented in 2014 such a method, and showed that the multi-channel deep CNN achieved higher accuracy than compared methods on two specific datasets. They later published an updated version in 2016 [39]. What we found most interesting about their proposed method was the use of multiple 1D channels, one for each feature in the input. The result of these individual convolutional calculations is then combined into one classification.

Another algorithm based on convolutional networks was presented by Jin and Dong [40] in 2016. They provide an ensemble approach for bio-medical data where each individual network is a CNN combined by simple averaging of the results from each. The authors claim an ensemble approach will solve an important issue with using a simple convolutional network with bio-medical time series data. The challenge with convolution, they say, is that the convolutional operation is applied in both horizontal and vertical direction. With images, this ability to look at both directions is necessary, but with other data we might not want the network to do this.

This challenge with pure CNNs is also seen in Hajime et al. [41] even though their approach and method is quite different. In their work they attempt to represent the time series data as an image and then feed it through a regular CNN, as illustrated in Figure 3.3 from their original article. Representing the temporal aspect of data in an image, because the most successful applications of CNNs is on image data, is intriguing, yet not without problems. The main problem here is when there is multivariate data representing multiple features that may not be strongly correlated but they are to be represented together in one image, the ordering of the features does matters to the CNN, even if it does not in reality. That means the network has to see every permutation of the features, something the authors themselves point out.

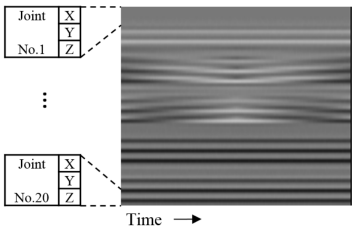


Figure 3.3: Joint coordinates X, Y, Z represented as a grey-scale image with time from left to right.

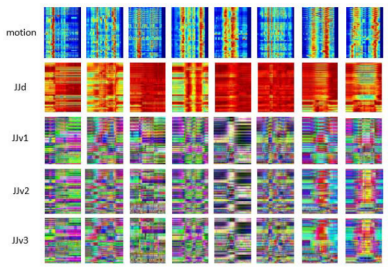


Figure 3.4: Motion, joint-joint distance and joint-joint vectors.

Rostami et al. [42] presents a method that has a similar approach as [41] where they also represent the time series as images and presents it to a CNN. Instead of 2D coordinates, as we have, they have 3D coordinates from each joint and uses this data to create three image encodings. Those encodings are joint-joint vector, joint-joint distance and motion feature. They then utilises a pre-trained AlexNet and fine-tune on their encoded images as shown in Figure 3.4. It is unclear if this method suffers the same issue as [41] due to a different representation of the time series in the images and not enough implementation details on these encodings. The similarity to our data, the image representation of the movements and the use of pre-trained network with transfer learning was what inspired parts of the method presented in this thesis.

The idea of representing the temporal dynamics of the data as an image stood out to us because we felt it could be easier understood and explained to those with no background in our field. This is an important requirement of our system if we want clinicians and other non-computer scientists to be able to use our work later. We were also encouraged by our co-supervisors on this as they too saw it as a feasible way to represent the data. The time series comes from 2D representation, and to keep it in 2D representation is visually more pleasing and more natural with the origin of the data and where it is coming from.

Chapter 4

Method

4.1 Dataset

The raw dataset that our data is extracted from is a set of 378 video recordings where each video corresponds to one infant. 42 of these did get a CP diagnosis later on. If we were to use the raw data directly, our time series would be a sequence of images and we would have an image analysis task. Unfortunately, we know that these raw images contain much more information than we need, including background noise and factors that does not contain information related to CP. Even with a standardised setup there are slight differences between videos that can cloud the information the computer system "sees". By reducing noise and unnecessary information we increase the chance that the algorithm learns from the information we want and thus learns what we want it to learn.

Therefore, we use the result from the method presented by Groos and Aurlien last year as a feature extractor to eliminate the excess information. The remaining features are the coordinates from the infant body tracking, frames per seconds in the original video recording, and the target which tells if the recorded infant had CP or not. The raw video is used as input to the tracking model and it identifies body parts frame by frame with the use of computer vision and CNNs. The tracked body parts are right and left wrist, right and left ankle, head, chest and pelvis. The tracker returns fourteen univariate time series representing the x- and y-coordinates for each body part or joint per image frame. This gives us a representation of the movement of each body part over time.

The dataset we obtained from this feature tracking was split into a separate training, validation and test set. Because of the class imbalance between CP and

$NoCP$, we tried to ensure the ratio of CP was approximately the same between the three sets. Table 4.1 show the distribution of the dataset, and table 4.2 shows the distribution of images. Images connected to one infant is only present in one of the sets. Even though the ratio of CP is approximately the same on infant level it differs more on image level. This is because the video recordings are of different length, and there will be generated a varying number of images per infant.

	Training	**Validation**	**Test**	**Total**
CP	34 (11.2%)	4 (10.8%)	4 (10.8%)	42 (10.9%)
$NoCP$	270	33	33	336
# subjects	304	37	37	378

Table 4.1: Distribution of subjects into training, validation and test set. Percentage is the prevalence of samples belonging to class CP

	Training	**Validation**	**Test**	**Total**
CP	1606 (10.0%)	162 (8.2%)	164 (8.5%)	1932 (9.7%)
$NoCP$	14468	1825	1755	18048
# images	16074	1987	1919	19980

Table 4.2: Distribution of images in the training, validation and test set. Percentage is the prevalence of samples belonging to class CP

The CIMA-tracker's performance vary from video to video. To get an overview over the accuracy of the tracked infants, a manually evaluation were done by one of us. The evaluation was done as follows:

1. For each video recording representing one infant, every 10th second of movement coordinated generated by the tracker were plotted in an image.
2. Looked through all images representing one infant and labelled the images as well tracked or bad tracked, and if none of these categories fitted – no label was given.
3. Aggregated all labels representing the same infant into one label, listed in the table below.

A summary of the result of the evaluation is shown in table 4.3, where the integers indicates the number of subjects in each category, split up in CP and $NoCP$. The whole evaluation can be found in appendix 7. Figure 4.4 shows the percentage of subjects with a CP diagnosis that are well or bad tracked, and figure 4.5 shows the distribution of bad tracked subjects on training, validating and test set.

	Well tracked	Bad tracked	Non-labelled
CP	7	9	26
$NoCP$	109	78	149

Table 4.3: Number of subjects evaluated as well tracked, bad tracked or non-labelled.

	Well tracked	Bad tracked
CP	16.7%	21.4%
$NoCP$	32.4%	23.2%

Table 4.4: Percentage of well and bad tracked subjects.

	Training	Validation	Test
CP	7 (2.3%)	1 (2.7%)	1 (2.7%)
$NoCP$	60 (19.9%)	10 (27.0%)	8 (21.6%)
# bad tracked subjects	67 (22.0%)	11 (29.7%)	9 (24.7%)

Table 4.5: Distribution of bad tracked videos.

4.2 Data Representation

4.2.1 Outlier Detection and Removal

For the purpose of detecting outliers, one statistical method and some clustering techniques were studied, and added in different combination. Each outlier detected is handled by replacing it with the mid value of the previous and next normal points. This is done independent of detecting method.

Clustering techniques should in theory work well on the given dataset, with data points mainly separated in groups for each bodypart. When working with cluster analysis following data characteristics are of interest [1]:

- **Size** As long as the data set are of small or medium size, most clustering algorithms work well. When detecting outliers in the dataset described in section 4.1, only 10 seconds are processed at a time, and the computation is done on one body part at a time. The fact that most clustering algorithms are unable to handle lager datasets no longer applies.

- **Noise and Outliers** As mentioned earlier, outliers can often degrade the performance of clustering algorithms, like it does for algorithms like K-means. Other algorithms, like DBSCAN and SNN density-based clustering can detect noisy points during the clustering process, and they will therefore be given more focus in the next sectons.
- **Sparseness** In spare datasets zeros are often of lower importance than the non-zero values, and similarity measures appropriate for asymmetric attributes are often used. The dataset served by the CIMA-tracker does have some error points, but their value are never missing, and sparseness does no longer need to be emphasised.
- **Mathematical Properties of the Data Space** Some clustering techniques use mathematical operations that only makes sense in Euclidean space. The tracked motions of each body part is given in Cartesian coordinates, which is well suited for mathematical operations like Euclidean distance.

With this characteristics in mind, the following sections describes different approaches for outlier handling in the data representation.

K-means

As described in section 2.3.2, squared error can be used to compute centroids of clusters. Outliers are therefore likely to have a huge impact on the clusters that are found, and the resulting cluster centroids may not be as representative as they otherwise would be. When working with K-means it is often recommended to detect and eliminate outliers beforehand [1].

Since the goal of this sub-process is to discover outliers, K-means is not suited for this operation.

DBSCAN

The DBSCAN algorithm described in algorithm 2 are simple, but not very efficient. As stated in section 1.2.2, this thesis wants to identify the cerebral palsy-related movement of infant body segments using computer vision and machine learning algorithms, and make this procedure time effective, feasible and available for researchers within the medical field without any technical or computer vision expertise. The time complexity for DBSCAN is given by $\mathcal{O}$(m x time to find points in the *Eps*-neighbourhood), where m is the number of points. Worst case time complexity is then $\mathcal{O}(m^2)$. In the process of detecting outliers with DBSCAN, data

from only 10 seconds are treated at time. Also, the clustering process is applied on one bodypart at time, so that points far away from its belonging cluster (outlier) is treated as a point belonging to another cluster. This results in relatively few data points, and the time complexity of the algorithm is not that bad.

DBSCAN used to detect outliers on the provided dataset have some weaknesses. First, the selection of the right values of the parameters Eps and $MinPts$ is challenging. With great variation in motion from baby to baby and from time to time, there were almost impossible to find the right combination. Second, DBSCAN do have some troubles with density if the density of clusters varies widely. Since the data points from the dataset varies from small clusters with small movement trajectories, to less dense clusters that represents greater trajectories.

All in all, DBSCAN should in theory work well on the outliers in this dataset. But, the varying density and the difficulties with setting the right parameters make it more unstable.

SNN density-based clustering

Trying to handle clusters of varying density, SSN density-based clustering were implemented. As described in section 2.3.2, SNN density-based clustering calculates the SNN similarity between all points, and then combines the results with DBSCAN. This should in theory work well, and there is great chances that outliers not detected by DBSACN alone will be detected. But, the dataset is of such vary that there was hard to find the right combination of the parameters Eps, $MinPts$, and K (k-numbers of neighbours). This limited the methods too much, and as a result the focus of outlier detection method moved to other alternatives.

Standard Score

As mentioned in chapter 2.3.1, when working with standard scores, all data points are described by their relationship to the standard deviation and mean of the population. Each point representing the position of a body part at a given timestamp get their standard score computed with equation 2.3. The absolute value of this score represents the distance between each point and the population mean.

While calculating the standard scores the system looks for data points which are too far from the mean. Points with a score greater than a given threshold, i.e. too far away from the centre of the cluster, are treated as outliers.

The standard score, standard deviation and population mean will vary from body part too body part, and from time range too time range. The size of one standard deviation for body part clusters with great trajectories will be greater than body part cluster with almost no movement. While working with a threshold that is equal to a multiple of σ, there will always be a given number of point that are fixed. For body part clusters like the pelvis this might not work well. Well tracked data points are replaced by "fixed" points.

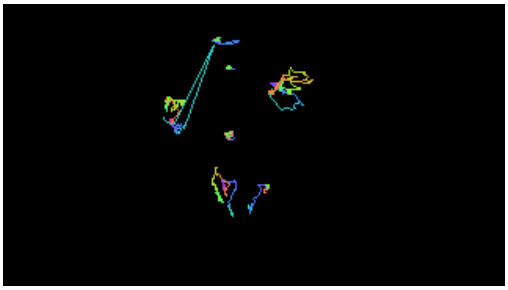
Combination of DBSCAN and length of vectors

Looking over the images generated with the outlier correction methods above, it could look like DBSCAN had a small head start on the other methods, even dough it let some huge data point errors through. Many of these non-detected outliers had in common that the Euclidean distance to the next data point were great, but close enough to be within the *Eps*. A two-step-outlier-detector were implemented, where DBSCAN was put in combination with a vector length function. First, the Euclidean distance between all data points are calculated, and all length greater than 10 x cluster mean were labelled as noisy data, and corrected. Then, DBSCAN was applied.

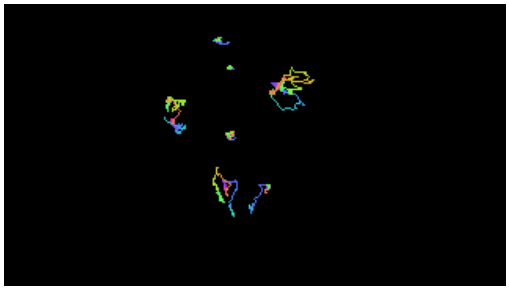
Comparing Methods

When comparing the images where standard score, DBSCAN and the two-step-outlier-detector were used for outlier detection, there were few differences to spot for the human eye. Independent of detecting method, each outlier detected was handled by replacing it with the mid value of the previous and next normal points. So there was no difference to discover in the method used for replacing bad data points.

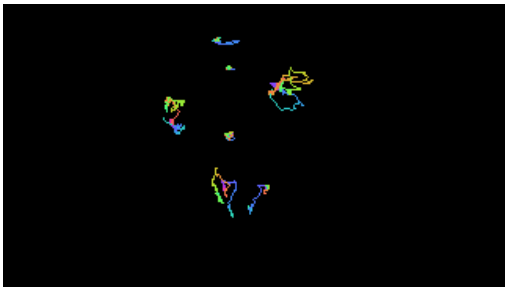
In theory both BDSCAN and standard score should work well, and both methods have their strengths and weaknesses. The method based on standard score work well on clustered data, and extreme outliers are detected and handled. But, the method will always try to fix a given number of points, independent of whether they are labelled as outliers or not. DBSCAN only correct data points labelled as outliers, but have trouble detecting small errors within a cluster. Both methods are limited by the difficulties of defining the parameters that fits every body part for every video recording. One example of the output differences between the two methods are illustrated in figure 4.1.



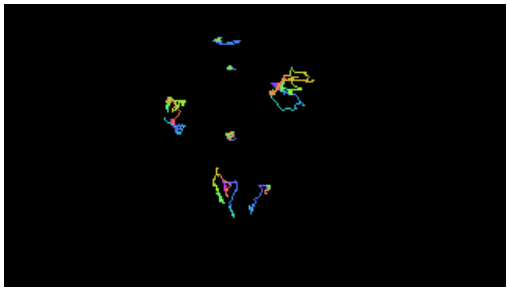
(a) Original image



(b) Outliers removed with standard score method.



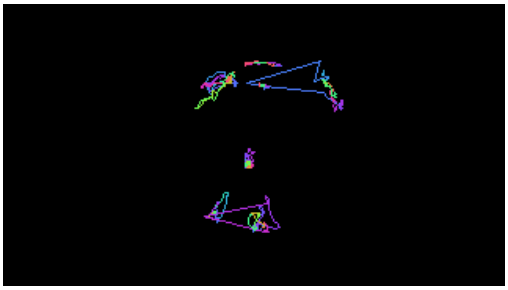
(c) Outliers removed by applying DBSCAN twice.



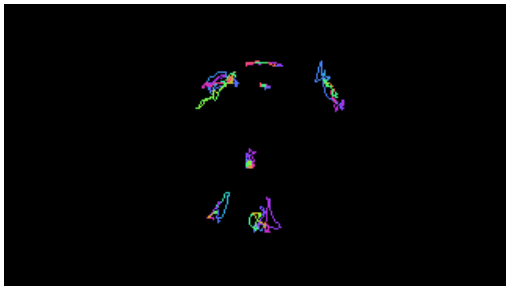
(d) Outliers removed by first applying vector length method, then DBSCAN.

Figure 4.1: Illustration of how the different outlier removal affects the data points, where outlier removal with standard score is the weakest.

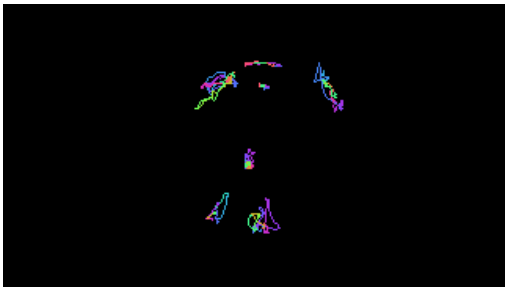
All images in figure 4.1 represents the same 10 seconds from the same subject. Image 4.1a shows the original image before any outliers are handled. In image 4.1b standard score is used as outlier filter. Notice that some data points representing the head are removed. Figure 4.1c and 4.1d looks the same, and the outlier methods applied are respectively 2 x DBSCAN (image 4.1c), and a combination of DBSCAN and length of vectors (image 4.1d).



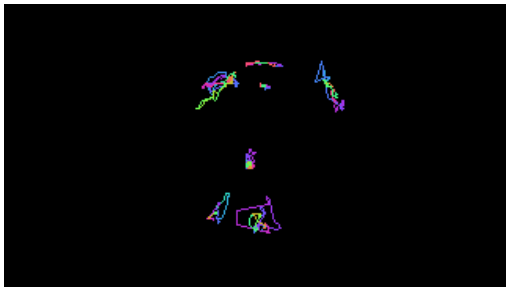
(a) Original image



(b) Outliers removed with standard score method.



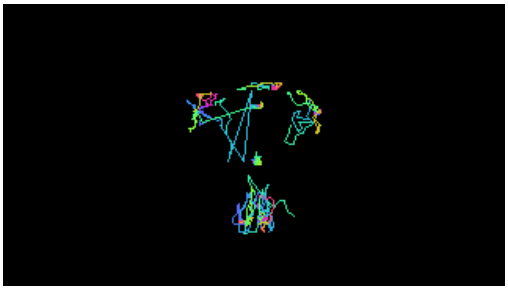
(c) Outliers removed by applying DBSCAN twice.



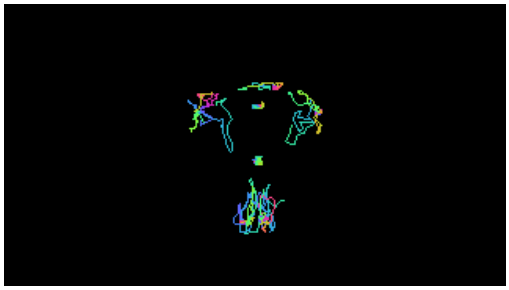
(d) Outliers removed by first applying vector length method, then DBSCAN.

Figure 4.2: Illustration of how the different outlier removal affects the data points, where outlier removal with the combination of vector length and DBSCAN is the weakest.

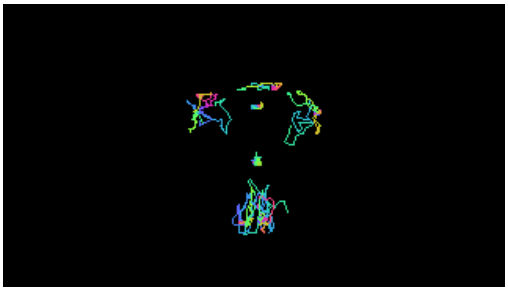
In figure 4.2 the different outlier removal methods are applied on 10 seconds of another subject. In this case standard score and DBSCAN work well, and the combination of DBSCAN and vector length is the weakest, as it cannot handle the outlier on the right leg. In figure 4.3 DBSCAN is the weakest method. The movement trajectory for the left arm are corrected too much, and for this subject standard score and combination of DBSCAN and vector length works better.



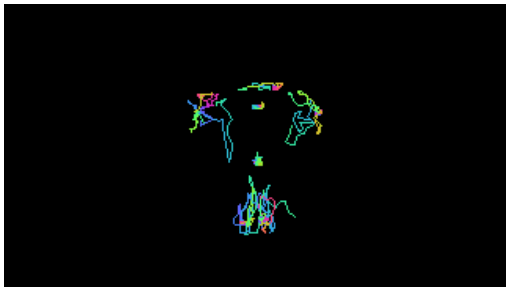
(a) Original image



(b) Outliers removed with standard score method.



(c) Outliers removed by applying DBSCAN twice.



(d) Outliers removed by first applying vector length method, then DBSCAN.

Figure 4.3: Illustration of how the different outlier removal affects the data points, where outlier removal with the combination of vector length and DBSCAN is the weakest.

While looking through all images and comparing them, it is clear that each outlier removal method has its strengths and its weaknesses. The patterns the human eyes can observe does not need to be representative for what a deep learning network sees. In order to make one's mind about which method that should be utilised for outlier detection before the images were sent into the autoencoder, a pretrained VGG16 were used as a baseline for comparing the image sets. The VGG16 were trained with binary crossentropy loss, class weights and pretrained imagenet weights. The network were trained with each image set several times, and the best results from each image set are shown in table 4.6.

	L-Rate	Sensitivity	Specificity
Original image	$1e-3$	0.75	0.727
Standard Score	$1e-3$	0.49999	0.8788
DBSCAN	$5e-3$	0.999999	0.4242
Combination	$1e-3$	0.74999	0.6061

Table 4.6: Results from training the VGG16 with binary crossentropy loss, class weights and pretrained imagenet weights. The sixteen first layers locked i.e. not updated further.

There is a big gap between the best and the worst results of the VGG16 testing, independent of outlier method used. The dataset is unevenly distributed, and this makes it easy for the VGG to get a relatively high score on either sensitivity or specificity. What impact this has on the results are further discussed in chapter 6.

There were never an image dataset that resulted in consistent better predictions. From the results presented in table 4.6 and the visible weak outlier correction illustrated in figures 4.1-4.3, we cannot tell if one outlier method works better than the others. Standard score was the first implemented method, and already tested on the autoencoder when these results were ready. Due to resource restrictions we therefore continued with the standard score dataset for further training and testing of the autoencoder.

4.2.2 Colour Model

When working with deep learning it is important to input the right features in the right format. In order to keep the time aspect of the movements in the image each timestamp is assigned a unique colour from a colour range where all colours in the range have the same colour intensity. This to prevent the model from favouring timestamps with more intense colours.

In section 2.4.3, colours with same intensity are described as colours on a cross-section normal to the greyscale line. To make sure that the model has a colour range with unique colours, colours were chosen from the arc of a circle normal to the greyscale line in the RGB colour cube. This arc is illustrated as the green dotted line in figure 4.4.

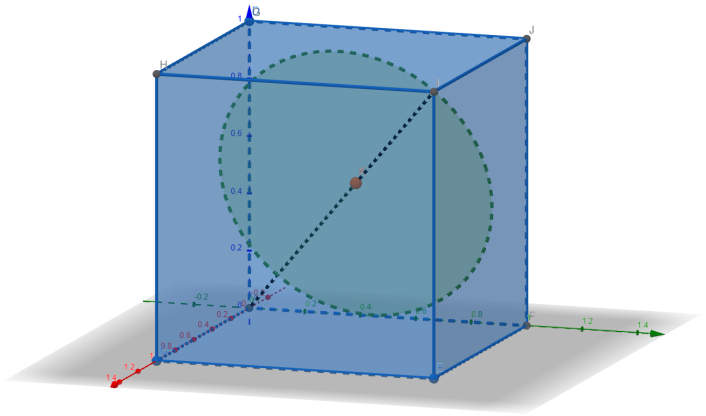


Figure 4.4: The green dotted line illustrates the arc that gives the equations for the colour range used for data representation.

The equation representing the green arc is the basis for the RBG values for a given timestamp, in which is given by:

$$\begin{aligned} R &= 0.58 - 0.35 * cos(t) - 0.2 * sin(t) \\ G &= 0.58 + 0.35 * cos(t) - 0.2 * sin(t) \\ B &= 0.58 + 0.41 * sin(t) \end{aligned} \tag{4.1}$$

where $t \in [0, 2\pi\rangle$.

4.2.3 Final Data Representation

How the data is represented before sent into a deep learning network is important. Therefore, the development of the data representation have gone through several steps and transformations, in order to make the selected representation as understandable as possible. Early in the process it was decided that each image should represent 10 seconds of infant movements. From that point the images have step wise changed, as shown in figures 4.5-4.10.

In the first version the data points were implemented as *matplotlib* scatter plots, as in figure 4.5. Each body part were assigned a unique colour, attempting to make the different movement patterns clearer to the deep learning network.

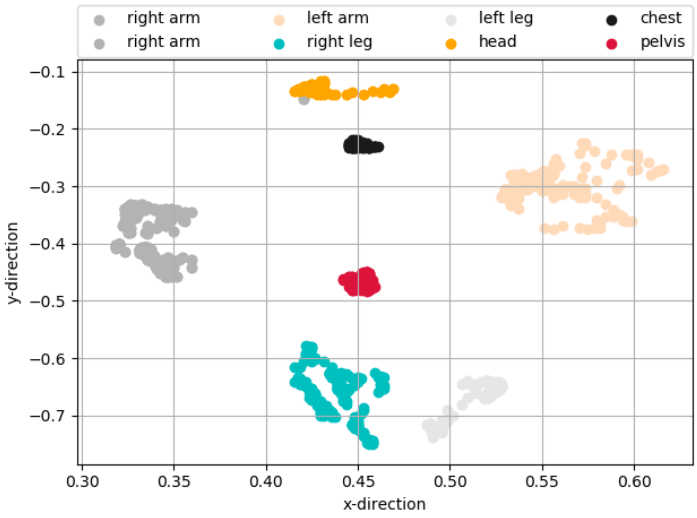


Figure 4.5: Each body part assigned a unique colour.

In the next version, each body part were assigned a unique colour range, so that each time step could get a unique colour. The goal was to visualise the time aspect, as well as distinguish the body parts. This is illustrated in figure 4.6.

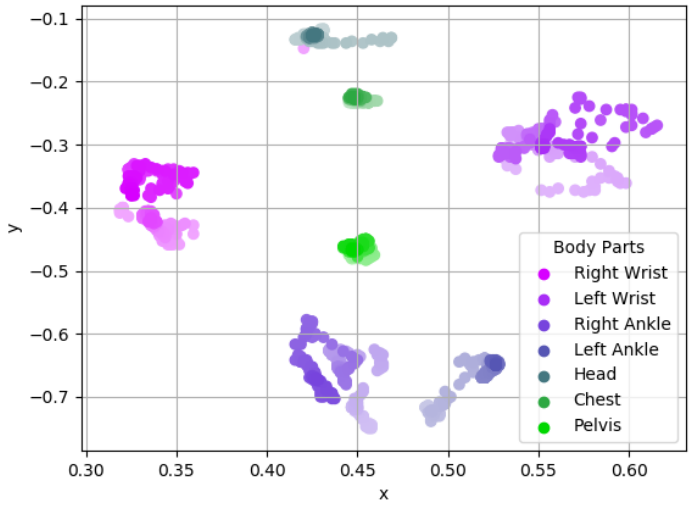


Figure 4.6: Each body part assigned a unique colour range.

Then the project discussed that the movement patterns might become clearer with

lines between the data points, so in the third version of the data representation, figure 4.7, the data points were replaced by lines.

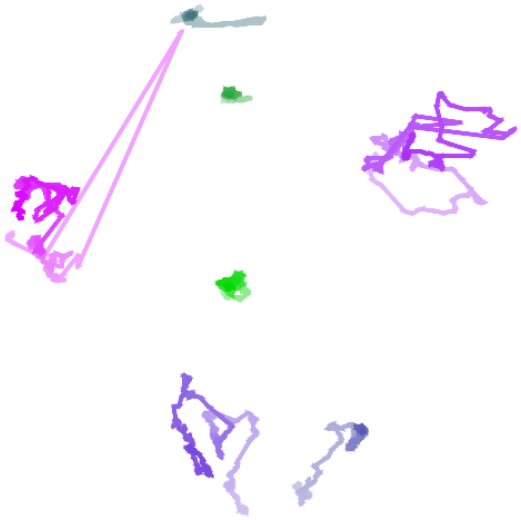


Figure 4.7: Data points replaced by lines.

I the attempt to make the images smaller *matplotlib* could not compress the images as we wanted. A *PILLOW image* on 180x320 pixels were implemented with data points as a guinea pig.

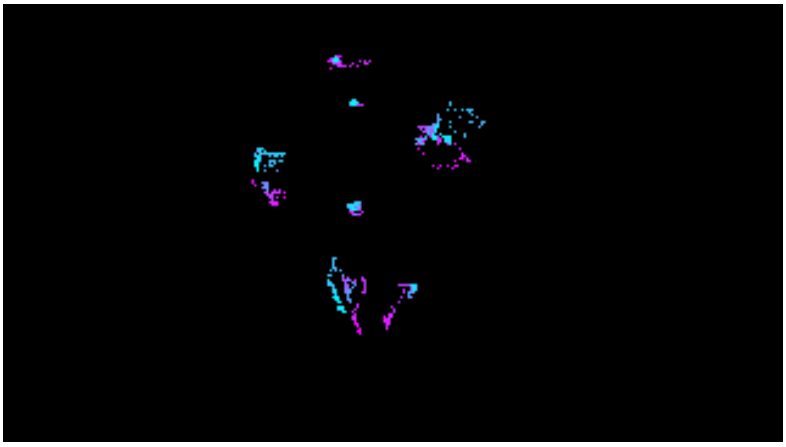


Figure 4.8: A PILLOW image on 180x320.

PILLOW images turned out to work well, and in version five (figure 4.9) lines again replaced the data points. In an attempt of making the network understand which

movements that are made at the same time, each body part were assigned the same colour range. Every colour have the same intensity, in order to be equally evaluated by the deep learning method. The colour range were generated as described in section 4.2.2, and each line representing movement at the same time stamp got the same colour.

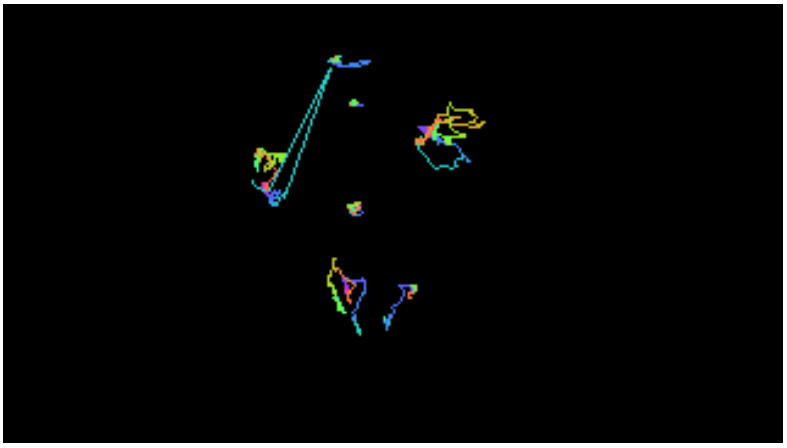


Figure 4.9: Every body part's movement represented by the same colour range.

The final version of the data representation have standard score applied as the outlier detection method, so that images sent into the network represent the original infant movement trajectories as good as possible. The final data representation is illustrated in 4.10.

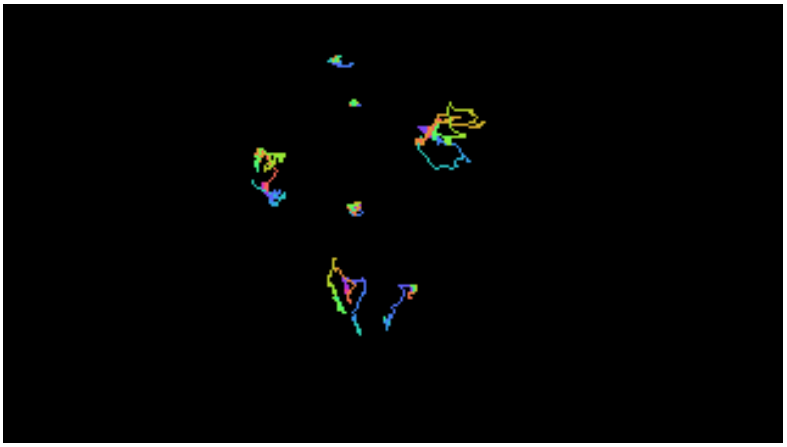


Figure 4.10: Final data representation.

4.3 Baseline Model

To get a better understanding of the usefulness of the model, and to compare its performance, it is common to choose a baseline that either solve a similar problem or has become a "standard" in literature. Because of the specific nature of our dataset and lack of methods solving similar problems to our knowledge there were no obvious choice of baseline.

4.3.1 DenseNet

As baseline for the classification model denseNet was

4.4 Prediction Model

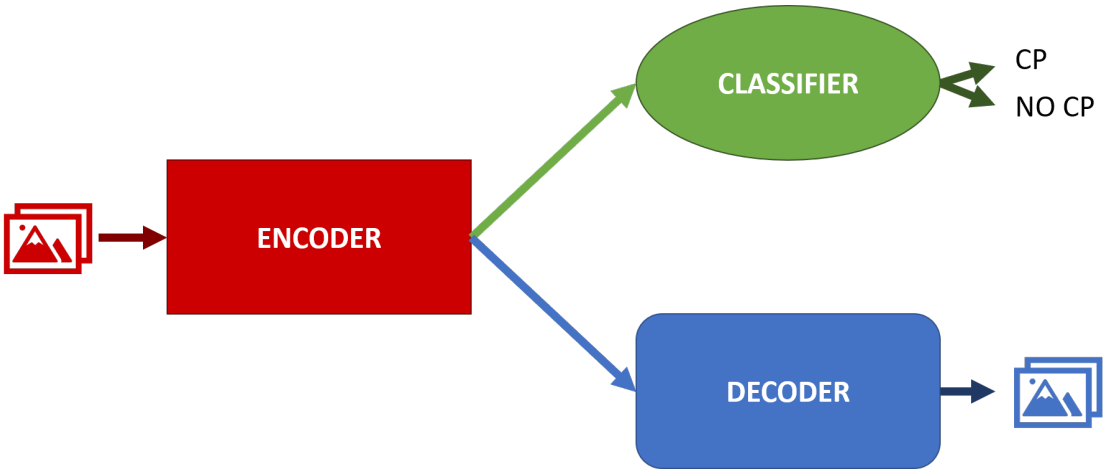


Figure 4.11: Complete network structure simplified. Decoder is used for pretraining the encoder part while the classifier makes the prediction on the probability of an infant having CP or not.

The proposed model we have tested consists of two parts. The first part is an encoder part, pretrained as part of an autoencoder. The second part is the classification itself with the purpose of assigning each sample to either of two classes: CP or $NoCP$.

4.4.1 Autoencoder Part

As described in section 2.6.1, an autoencoder is an algorithm that uses its own input as target, trying to reconstruct the input after extracting features and in most cases compressing the information into a smaller feature space. Different variants of autoencoders has been developed, among them sparse encoders, denoising encoders and variational encoders. It is also common to have stacked autoencoders. The focus in this project however will be on building a convolutional autoencoder and will not look into these variations as that would require a whole another thesis. With images, both the convolution layers and pooling layers can reduce the number of pixels that represents the input image, thus forcing the algorithm to choose those features it considers best represents the image.

Our autoencoder model is build up of four blocks of convolution layers and batch normalization with three max pooling layers distibuted between the blocks. This corresponds to the encoder part of the autoencoder, where the initial image size of 180x320 pixels is reduced to 15x40 pixels. Following is a mirrored decoder part that uses upsampling instead of pooling to produce the output image with the same size as the input. Figure 4.12 shows an overview of the autoencoders encoder structure only.

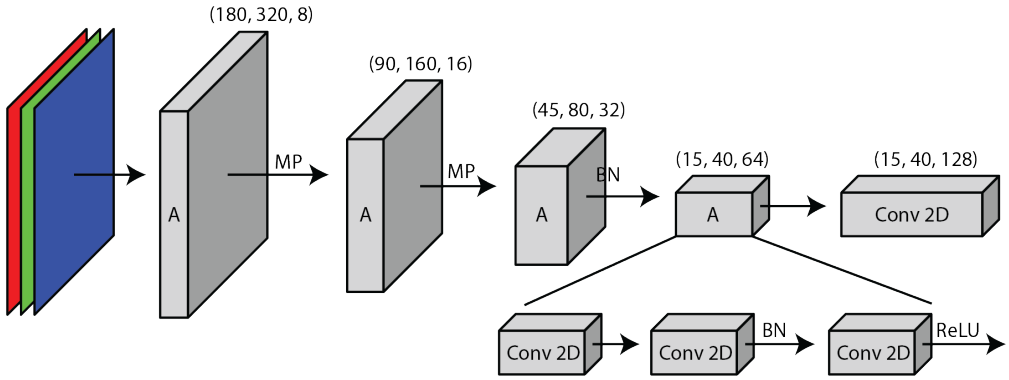


Figure 4.12: Encoder part of our autoencoder version 1. The input image is size 180x320 pixels and the output is 128 feature maps of size 15x40 pixels. Each block A has the same structure as shown below the last block, two convolution layers followed by batch normalisation (BN) and another convolution layer with ReLU activation function. Between the blocks is a max pooling layer (MP).

It is not straightforward to assess whether an autoencoder has a good or relevant

representation in the encoder based on the accuracy of the reconstruction alone. We did try to visualise the feature maps in the code layer, but it was challenging to make final conclusions about the actual gain in our particular case. An autoencoder can achieve high accuracy and almost perfect reconstruction by simply memorising if the code layer is big enough, and even though we did reduce the number of features it is not trivial to determine if those features captured the patterns that the classification needs. We mostly relied on the joint model with the classification to see if what impact the autoencoder had on the result.

4.4.2 Classification Part

We originally wanted to keep the classification part very simple, leaving most of the feature extraction to the autoencoder. The classification layers are usually a limited number of fully connected layers at the end of a bigger network structure with the purpose to take the extracted features found through the rest of the network an assign it to one of the classes or categories. Our simple version consists of two regular fully connected layer and then a final softmax layer to assign each sample to either of our two classes. To mitigate overfitting we included a dropout layer between the two dense layers.

The simple model it did not give too promising results however, partially due to the size of the feature space from the encoder, and we tried to expand the model by making it deeper and drawing inspiration from the denseNet architecture [31]. As described in section 4.3.1 denseNet consists of multiple blocks of convolutional layers with forward connections between the blocks, thus retaining more information from one block to another and from input to output. Due to memory constraints and issues with the number of pooling layers it was not trivial to simply run the predefined keras version of denseNet architecture on our proposed autoencoder, and so figure 4.13 shows our extended classification model modified to fit within these constraints. The details of block B and C is shown in figure 4.14a and 4.14b. Block B is a fairly regular convolution block with three units of batch-normalisation-¿convolution-¿ReLU following each other. Block B is followed by a max pooling layer and then a concatenation block C. Block C is made up of three consecutive passes consisting of convolution-¿batch-normalisation-¿convolution in which the resulting computation is concatenated after each pass. This creates the forward connections where the result of the previous computations are kept and forwarded into later computations as can be found in denseNet.

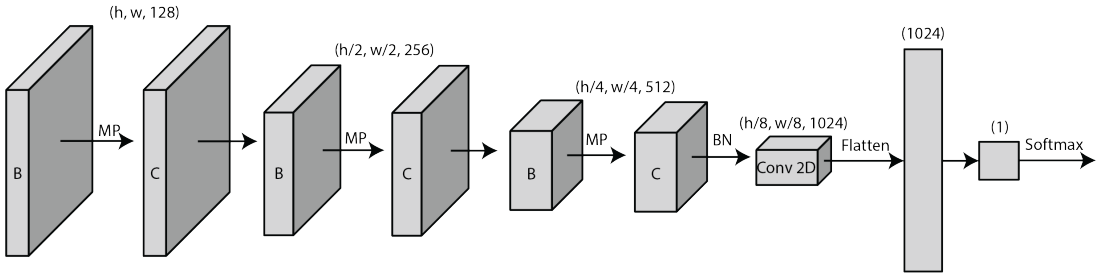
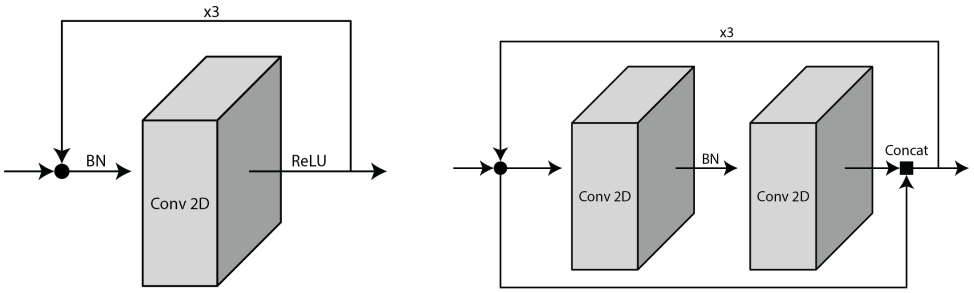


Figure 4.13: The extended classification model. Block B and C is illustrated in figures 4.14a and 4.14b. Final output is 0 or 1 corresponding to classes $NoCP$ or CP.



(a) Convolution block B

(b) Concatenation block C

Figure 4.14: Detailed view of the convolution and concatenation blocks used in extended classification model.

4.5 Training

4.5.1 Hardware, Software and Computational Resources

This project has utilised Keras[43] with Tensorflow backend[44] as the machine learning library to create and manage the neural networks. The codebase is written in Python with libraries matplotlib and Pillow used to create the image data. The code was developed on either NVIDIA Tesla P100 GPU's or NVIDIA Tesla V100 GPU's, depending on which sever or cluster was available at the time. Memory was usually limited to 16 GB per GPU, again depending on which server. There was a limit, both by Keras and by the servers that we could only use a maximum

of two GPU's at a time, and of course resources was limited by the amount of traffic by other users.

4.5.2 Pretraining on Fidgety and No-CP Datasets

As mentioned in section 2.6.1 the main purpose of including an autoencoder in this project was to pretrain it on a selection of the data in the hopes that the patterns learned would help the algorithm distinguish between the cases of CP and $NoCP$ better. In the data we got, a clinician had classified each subject with a degree of fidgety movements recognised from the videos. As the degree of fidgety movements has been shown to give good indication of whether the infant has CP or not, we hoped that by pretraining the autoencoder on those with normal degrees of fidgety movements would help the classification single out those who did not show much of these movements. It was however children with a CP diagnosis who showed these fidgety movements and children who did not show much movements but did not have a diagnosis. Thus we decided it was best to test another selection of data in which we excluded all subjects with CP altogether and only training on the majority class of $NoCP$ in the hopes that this would give a better distinction between the classes.

4.5.3 Class Weights

To accommodate for the imbalance between the classes, Keras does have a function called *class_weights* that can specify preference or importance to each class. Because only 10% of our data belonged to subjects with CP we wanted to give these samples higher importance during training so that the impact of one sample of CP would equal the impact of ten no-CP samples. Thus we set one sample of class CP to have equal importance as ten samples of class $NoCP$.

4.5.4 Callbacks

We used two of Keras built-in callback functions during training. Early stopping was used to stop the training once the validation loss reached convergence. The parameter *patience* determines how many epochs with no update the training should do before stopping. This can not only reduce training time, but also prevent overfitting by stopping when no improvements seemingly are made. Experimenting with different patience in our case did not seem to give much difference. Model checkpoint is used to save the model to disk during training. A model was saved

whenever validation loss was improved upon so that only the best model was saved. An advantage of this is that if a training run crashes for some reason, the last model can be reloaded and trained further, saving the previous work. A modified version of model checkpoint had to be applied when multiple GPU's were used as Keras's wrapper *multi_gpu_model* makes it difficult to work with the paralleled model later on.

4.5.5 Configs and Parameters

Hyperparameters is one of the most determining factors when training a neural network and to find the right combination of said parameters is often a matter of empirical testing to see what works for the specific network structure and data at hand. The number of parameters we could test in this project were limited because of the size of this project, but also because the tests we did gave a decent indication as to how well the network performed in general and changing a lot of parameters were unlikely to have noticeable impact on the results. Adam was the final optimiser used, though stochastic gradient descent was also tested. It proved to be much too slow compared to Adam, and with multiple tests on different learning rates, the results seemed to converge to the same using either. We used both mean squared error and binary crossentropy as loss function when training. As can be seen from the results in chapter 5 binary crossentropy in general gave a higher accuracy. Results will be discussed in chapter 6.

4.5.6 Data Augmentation

Data augmentation is the process of increase the size of the training set by adding extra copies of the training examples that have been modified with transformations that to not change the class [20]. This includes for example image rotation and image flipping, and is a very common technique in machine learning to not only increase the amount of training data, but also to include variation and diversity in samples. With a small dataset, as we have, utilising augmentation can be vital in making an algorithm robust and generalise better. We did not think it necessary to prioritise data augmentation in this project since we already thought that the data representation was challenging and we did not want to include more noise at an early stage in the project. Our few tests with data augmentation was done with Keras' predefined class *ImageDataGenerator* and though we do not have enough testing to give solid conclusions it suggested what we already suspected, that with data augmentation the algorithm (as it was) would be more confused and not improve results without other modifications to the model or data.

4.6 Evaluation

4.6.1 Autoencoder

The goal of the autoencoder on its own is to reconstuct the input image by up-sampling the features from the code layer. Thus this reconstruction also gives an indication of how well the autoencoder has learned features that represent the data. Examples of how the autoencoder reconstructed the images can be seen in section 5.1.1.

To see that the autoencoder does learn something inside the black box the feature maps from the code layer is included in figure 4.15. Even though it is not trivial to interpret these image fully, we see that they highlight slightly diffrent aspects of the input, which indicate that the autoencoder is learning different features and multiple features which is something we want.

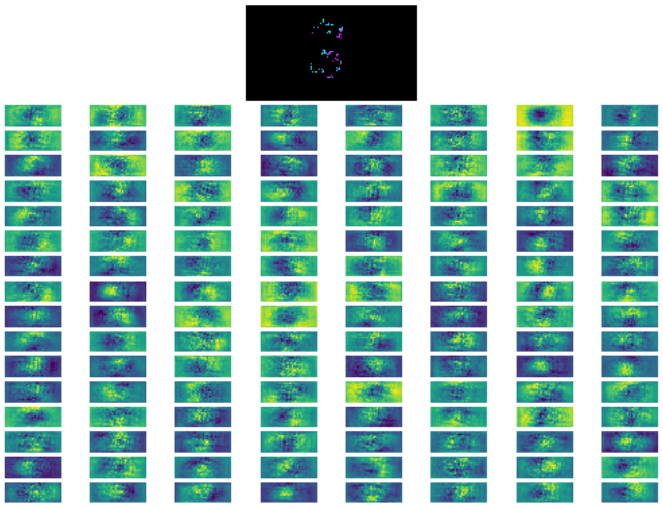


Figure 4.15: Visualisation of the feature maps in the encoder layer of autoencoder. The top image is the original input image.

4.6.2 Classification

When evaluating the performance of a classification algorithm accuracy is often the default metric, as it tells us how many samples the algorithm managed to classify correctly. However, with the type of data and problem we worked with in this project, accuracy is not always the most informative metric. In medical practice, it is often more informative to use sensitivity and specificity as described in 2.1.3 and we will use that as the main metrics beside accuracy and loss.

4.6.3 Classification Per Subject

Because each subject was split into multiple images showing multiple intervals in time, one image alone would not be representative for the complete movement pattern of the subject. It was also evident that even if the algorithm managed to guess a fair amount to have CP, it seldom got many of those correctly. Because a subject did so many different movements over the course of the time span it was recorded, we hoped a combination of the predictions on all the images belonging to the subject would give a better prediction. The final prediction was decided by majority vote, thus giving a higher level of confidence and presumably a better result.

Chapter 5

Results

This chapter is a summary of the testing results done in this project. We do want to mention that our initial testing got corrupted by copying issues of the data and is not included here. The results did however guide some of the testing choices made when re-running the tests on the corrected dataset.

5.1 Autoencoder Results

Tables 5.1 and 5.2 shows the main results from pretraining the autoencoder on both the fidgety dataset and the No-CP dataset. Table 5.3 shows test with mean squared error loss on the No-CP dataset, which evidently show less consistent results and much lower accuracy. Although not thoroughly tested enough, table 5.4 shows training the autoencoder on an augmented dataset.

	Epochs	**L-Rate**	**Loss**	**Accuracy**
1	12	$1e-3$	0.0156930308	0.988654284
2	41	$1e-4$	0.014320508	0.9884202521
3	142	$1e-5$	0.0144221634	0.9884303644

Table 5.1: Results from training autoencoder on the fidgety dataset with binary crossentropy loss.

	Epochs	L-Rate	Loss	Accuracy
1	3	$1e-1$	0.1084115927	0.9884963937
2	5	$1e-2$	0.0273047154	0.9884963937
3	14	$1e-3$	0.0151227199	0.9884351173
4	29	$1e-4$	0.0148768301	0.9883657722
5	36	$1e-5$	0.0201510189	0.9875198565
6	407	$1e-6$	0.0154333055	0.9884886924
7	1394	$1e-7$	0.0227307531	0.9883882837
8	123	$1e-8$	0.7922884027	0.476975013

Table 5.2: Results from training autoencoder on the No-CP dataset with binary crossentropy loss.

	Epochs	L-Rate	Loss	Accuracy
1	72	$1e-5$	0.0033004883	0.0755990821
2	706	$1e-6$	0.0030163218	0.1363132378
3	1198	$1e-7$	0.0580254665	0.0171052454
4	296	$1e-8$	0.143537989	0.5871587355

Table 5.3: Results from training autoencoder on the No-CP dataset with mean squared error loss.

	Epochs	L-Rate	Loss	Accuracy
Version 1	9	$1e-4$	0.0327558472	0.9884963937

Table 5.4: Results from training autoencoder with augmented data on the No-CP dataset with binary crossentropy loss.

5.1.1 Reconstructed Images

Figures 5.1, 5.2 and 5.3 shows some example of how the reconstructed images from the decoder looked like. We see that the code layer when using mean squared error results in a very different image than when using binary crossentropy.

Figure 5.1: Reconstructed image from autoencoder trained on fidgety dataset with binary crossentropy loss.

Figure 5.2: Reconstructed image from autoencoder trained on No-CP dataset with binary crossentropy loss.

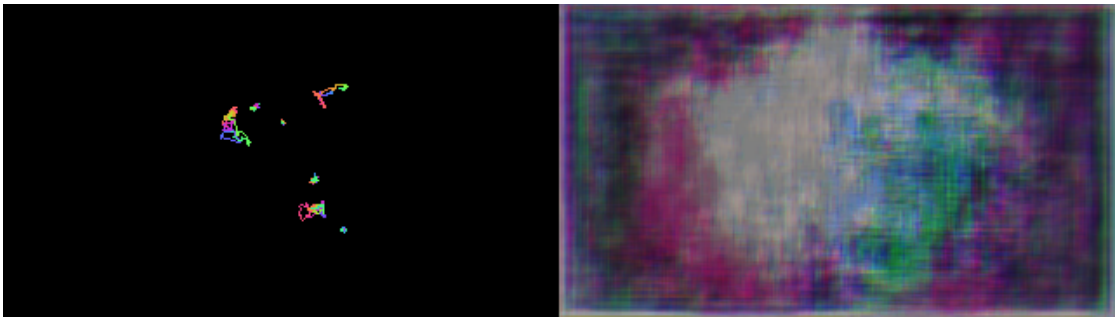


Figure 5.3: Reconstructed image from autoencoder trained on No-CP dataset with mean squared error loss.

5.2 Classification Results

Tables 5.5 and 5.6 shows the result of training the simple classification model with pretrained autoencoder on the fidgety dataset. Both mean squared error and binary crossentropy gave similar results. Using the autoencoder pretrained on the No-CP dataset yielded similar results as seen in table 5.7. The extended classification model gave more varying results and though table 5.8 shows only the best runs, most results were similar to the simple classifier.

	L-Rate	**Loss**	**Accuracy**	**Sensitivity**	**Specificity**
1	$1e-3$	1.4110682914	0.9124544029	0.00	1.00
2	$1e-4$	0.3461933402	0.8843147473	0.0714	0.9623
3	$1e-5$	0.3301850506	0.8968212606	0.0179	0.9812
4	$1e-6$	0.3150812159	0.8994267843	0.00	0.9857
5	$1e-7$	0.2839108179	0.9124544029	0.00	1.00
6	$1e-8$	0.3163377324	0.9124544029	0.00	1.00

Table 5.5: Results from training the simple classifier on the fidgety dataset with binary crossentropy loss.

	L-Rate	**Loss**	**Accuracy**	**Sensitivity**	**Specificity**
1	$1e-3$	0.0875455966	0.9124544029	0.00	1.00
2	$1e-4$	0.0875455966	0.9124544029	0.00	1.00
3	$1e-5$	0.1723569957	0.7566440855	0.2738	0.803
4	$1e-6$	0.0854184205	0.9114121934	0.00	0.9989
5	$1e-7$	0.0827173296	0.9114121934	0.00	0.9989

Table 5.6: Results from training the simple classifier on the fidgety dataset with mean squared error loss.

	L-Rate	**Loss**	**Accuracy**	**Sensitivity**	**Specificity**
1	$1e-3$	1.4110682914	0.9124544029	0.00	1.00
2	$1e-4$	0.3461933402	0.8843147473	0.0714	0.9623
3	$1e-5$	0.3301850506	0.8968212606	0.0179	0.9812
4	$1e-6$	0.3150812159	0.8994267843	0.00	0.9857
5	$1e-7$	0.2839108179	0.9124544029	0.00	1.00
6	$1e-8$	0.3163377324	0.9124544029	0.00	1.00

Table 5.7: Results from training the simple classifier on the No-CP dataset with binary crossentropy loss.

	L-Rate	Loss	Accuracy	Sensitivity	Specificity
1	$1e-3$	0.8488609509	0.8202188636	0.3333	0.8669
2	$1e-4$	2.1360534807	0.3923918706	0.8988	0.3438
3	$1e-5$	0.7217357015	0.8478374149	0.1488	0.9149

Table 5.8: Results from training the extended classifier on the No-CP dataset with binary crossentropy loss.

5.2.1 Classification Per Subject

Table 5.9 shows the result of combining the many image-based classifications of each subject to one classification from majority vote.

	Data	Accuracy	Predicted	Predicted Correctly
1	No-CP	0.89189189	0	0
2	No-CP	0.83783783	4	1
3	Fidgety	0.89189189	0	0
4	DenseNet	0.81081081	5	1

Table 5.9: Results from combining the classifications by majority vote on all images for each of the 37 subjects in the test set.

5.3 Baseline Models

Table 5.10 shows training a pretrained denseNet classification on our data. Overall, the accurcay is about the same or lower, but the sensitivity is significantly higher than most run on our model. Table 5.11 has the fifty first layers locked so that only the later layers are trained and fine-tuned further on our data. Table 5.12, 5.13 and 5.14 shows the results of testing the data representations with VGG16.

5.3.1 DenseNet121

	L-Rate	Loss	Accuracy	Sensitivity	Specificity
1	$1e-3$	0.6822078438	0.7910369981	0.2619	0.8418
2	$1e-4$	1.4609487767	0.6618030224	0.4048	0.6865
3	$1e-5$	0.5374561046	0.8269932256	0.2619	0.8812
4	$1e-6$	0.3838673248	0.8650338718	0.2143	0.9275

Table 5.10: Results from training the keras denseNet121 classifier with binary crossentropy loss and pretrained imagenet weights.

	L-Rate	Loss	Accuracy	Sensitivity	Specificity
1	$1e-4$	2.2106767329	0.4210526315	0.869	0.3781
2	$1e-4$	0.3786505868	0.9062011464	0.0119	0.992
3	$1e-4$	4.5599589914	0.2318916102	0.9821	0.1599

Table 5.11: Results from training the keras denseNet121 classifier with binary crossentropy loss and pretrained imagenet weights. The fifty first layers locked i.e. not updated further.

5.3.2 VGG16

The best results from training the VGG16 network. The non-presented results suffers for great variation in performance, and indicates that the following results should be validated critically.

	L-Rate	Optimizer	Loss function
1	$1e-3$	Adam	Categorical Crossentropy
2	$1e-3$	SGD	Categorical Crossentropy
3	$5e-3$	SGD	Categorical Crossentropy
4	$1e-3$	Adam	Binary Crossentropy
5	$1e-3$	SGD	Binary Crossentropy

Table 5.12: Parameter setup from training the VGG16 with class weights and pretrained imagenet weights. The sixteen first layers locked i.e. not updated further.

	Predicted CP	**Correct predicted CP**	**Sensitivity**	**Specificity**
1	564	76	0.4524	0.7213
2	1919	168	0.9999	0
3	1919	168	0.9999	0
4	909	113	0.6726	0.5454
5	240	36	0.2143	0.8835

Table 5.13: Results from the setup in table 5.12, with classification on image level. The total number of images labelled with CP was 168.

	Predicted CP	**Correct predicted CP**	**Sensitivity**	**Specificity**
1	7	1	0.2500	0.8181
2	37	4	0.9999	0
3	37	4	0.9999	0
4	18	3	0.7500	0.5455
5	1	1	0.2500	0.9999

Table 5.14: Results from the setup in table 5.12, with classification on subject level. The total number of subjects labelled with CP was 4.

Chapter 6

Discussion

6.1 Dataset

The dataset used in this master thesis is, as described in section 4.1 a small dataset when working with deep learning. Deep learning methods are known for their need of data. Too little training data often results in a poor approximation, and too little test data results in an optimistic model with high variance estimation. The dataset provided is also unevenly distributed, representing only 10% infants with a CP diagnosis. As a result, the model can predict every infant as healthy, and still reach an accuracy on 90% present. Sensitivity and specificity, described in section 2.1.3, will in this case provide better insight in the models performance, as they measure how well the model predicts true positives and true negatives. A model should reach a high specificity and a high sensitivity in order to be called well working. As presented in chapter 5, almost all classifications reach either a good sensitivity or a good specificity. A characteristic for the VGG16 is that a high number of predicted CP more often results in higher sensitivity and specificity. These numbers will in theory be considered as interesting, but the high number of false positives makes the prediction unreliable.

Another limitation is that the datapoints generated by the CIMA-tracker only represents 7 body parts, and not 19, in which would have been a full bio-mechanical model. How much impact this have on the model is hard to tell, but there is reasonable to assert that a deep learning network will do it better when provided more accurate information about the infants movements. When clinicians uses GMA as a tool for giving a CP diagnosis, they look at movement trajectories across the whole body (2.1.3), and the small movements are as important as the big ones. Therefore it is likely to believe that the network has greater possibilities

for detecting the right patterns, i.e. fidgety movements, when provided a dataset with more accurate information about the body movements.

The CIMA-tracker provided by Groos and Aurlien [33] has its limitation too. Even the sightliest error in the tracking will result in noisy data. These small errors are extremely hard to detect, as they look non-erroneous. As a result, the the CP prediction model is fed with data points that might not represent the infants exact movements, and there are chances that the model learns movement patterns that does not really exists.

The evaluation of the CIMA-tracker's performance indicates how well the images represents an infants movements. The evaluation was done manually and will therefore suffer from a subjective assessment. As shown in table 4.4, there are almost twice as many healthy subjects (32.4%) that are labelled *well tracked* as subjects that later on got a CP diagnosis (16.7%). For bad tracked subjects this percentage is more equal with respectively 23.2% and 21.4%. For this prediction task one of the essentials for the network is to learn recognising the fidgety movements healthy infants make (section 2.1.2). Well tracked subjects will results in more accurate information about the infants movement trajectories and the fidgety movements. It is an advantage that healthy subjects are well tracked so that the network have good data to train on and learn from. The number of bad tracked subjects are also of great interest since images generated from these are adjusted in the attempt of fixing misleading points. Table 4.5 shows that there are a higher percentage bad tracked subjects in the validation and test set, than in the training set. This means that the systems had a larger share of the non-labelled and well-tracked subjects, which in turn results in training on fewer edited images and greater opportunities for learning the important fidgety movements.

6.2 Data Representation

Discussion about outlier detection and correction

The CIMA-tracker has two other situations that often results in bad data. The first one is when there is something in the outskirts of the image that disturbs the tracker, as shown in figure 6.1a. The other situation is when one of the seven tracked body parts moves towards one of the other, or is interpreted as one of the other. In these cases there are two main issues. If the body parts moves toward each other, they might have some overlapping movements. Since the tracker reads 2D data, it can not follow both the overlapping body parts. This often results in noisy data. When one body part is interpreted as another, the CIMA-tracker gets

confused, and as a result it jumps between the two body parts. This is illustrated in figure 6.1b.

(a) External disturbances. (b) Internal disturbances.

Figure 6.1: The two most common types of errors from the CIMA-tracker.

Working with data that includes both types of tracking errors makes the outlier detection and removal challenging, especially when using cluster analysis as outlier detectors. If a cluster representing the motion of a body part have a data point far from the cluster mean, all other data points will look normal compared to that point. This is illustrated in 6.1a.

Tracking errors inside a body part cluster are hard to correct. Visualisation of these errors makes it extremely clear that there is no way a human body can move that fast. But how is this best corrected? Even a clinician will have trouble with predicting the next move of an infant, so there is no way the outlier correction will be 100% right. The challenge is then changed to how one can make the correction as correct as possible. In this master thesis clustering method were mainly used for outlier detection and correction. When detected, each outlier is handled by replacing it with the mid value of the previous and next normal points. Since each data point represents the movement done in 1/24th or 1/29th second, this data normalisation will work. But if too many outliers are detected in a row, it will result in a new straight line of "fixed" points. This line is no more representative for the infants movement than the original outliers, and the point with outlier detection is gone.

As mentioned in section 4.2.1, the provided dataset includes data with varying density, in which make choice of parameters for the outlier detection algorithms difficult. The data sent into the deep learning network will still include some outliers, both due to outliers that are not detected by the algorithm, but also due to wrong correction in the reprocessing. This might have an impact on the network, which in turn might learns features that does never really were done by the infants. In section 2.3.1 we mentioned that there is another member of the InMotion project

that has been looking into mathematical and statistical methods for fixing outliers. Her methods have been applied to the data points representing, with time along the x-axis, and it would be interesting to use this as prepossessing before turning data into images. With such methods one eliminates the problem with datapoints compared to the cluster mean, and both datapoint with an Euclidean distance far from the cluster mean and closer points can be handled.

Evaluation of representation

Another aspect of the data representation is whether the representation suits its purpose. The proposed data representation, with movement trajectories of 10 seconds in one image, is easy to understand for humans, and was the main reason for the choice. For clinicians, that are going to be the end users of the CP prediction application, it is important that the proposed method is somewhat understandable. A prediction that comes from an application feels more reliable when the process is explainable, which this data representation is. The InMotion-project were also a part of the decision, as they already had this data representation as something they wanted tested. In our Specialisation Project [10][p.18-19] we concluded that we waned to look more into 2D-representation, as we found some appealing convolution methods during our literature study.

The results of the different classifications tested in this thesis indicate that the proposed data representation is hard to interpret for our autoencoders as well as for the baseline method VGG and a tweaked DenseNet. Our literature study [10] revealed two methods where time series are represented as images, as discussed in section 3.2.2. These methods are too complex for humans to understand, but they might provide useful information for a deep learning network.

6.3 Autoencoder and Pretraining

The results of the autoencoder do show some promising reconstructions and high accuracy but the results of the classification afterwards is not equally promising. It is hard to make a firm conclusion on whether using an autoencoder does not work at all but in this case we think that it is the data and the representation of data that makes it challenging for the autoencoder to pull out a good set of features that represents movement patterns separating the instance of having CP or not having CP.

There was not a huge difference in pretraining on degree of fidgety or on simply

not having CP but in general No-CP was slightly better. We think with the great imbalance, it is more intuitive and accurate to pretrain on No-CP. It is a better representation of the full span of movements that healthy infants show, especially when dividing into time intervals. We do not know which interval is showing the relevant movements, and the subjects are not necessarily doing fidgety movements constantly throughout the recording.

We do not want to eliminate the possibility of an autoencoder that may serve as a good feature extrator, but we believe it requires more research and dedicated time. The autoencoder we proposed here did take a great deal longer to train than the complete classification and it requires it's own tuning and searching of hyperparameters. It is also likely that with more data and another data representation the autoencoder may get more fruitful results. The autoencoder might also benefit from reducing the feature space even more, and we theorise that experimenting more with stride lengths and diluted convolution might help in reducing the feature space without increasing the depth unnecessary.

6.4 Classifier Model and Prediction

We saw that the simple classifier did not perform well on the results from the autoencoder simply because it was too shallow. Flattening the features from the encoder resulted in a number of features that were still too big and the features from the autoencoder were not discriminating enough to divide between the classes. It required a deeper network with more pooling to reduce the feature space. The extended classification model was an attempt at that, and though it seemed to improve a little, the results were practically the same. Because we could not run denseNet directly on top of the autoencoder, we attempted a reduced version that would fit with the memory constraint and the feature reduction that we thought was required.

It is worth noticing that even though using mean squared error in general gave very similar accuracy, it also often gave a slightly higher sensitivity value because it predicted more samples to have CP. The same tendensies is what happened with denseNet. It guessed more samples to have CP so even when accuracy was lower the sensitivity was higher. Unfortunately it meant the specificity decreased, because the overall number of correctly classified samples did not change remarkably.

We also did some test with the extended classifier without an autoencoder, and though we did not include the results her, it was interessting to see that it made less of a difference than we thought. The autoencoder did improve the results a

little but not enough to exclude the possibility that if the classification algorithm is improved upon then the autoencoder might be excessive.

Our baseline denseNet did perform better than our proposed model, but not nearly as good as it would on a "standard" classification task, which only proves how important the training data is. We did not attempt to make the denseNet architecture into an autoencoder and pretrain that on the fidgety or No-CP datasets, but we imagine it could be a possibility, if not only to see if the structure of the autoencoder should be different. Even though the pretrained weights from imagenet is generally improving the performance of denseNet, in our case it did not make a big difference whether denseNet used the pretrained weights or random initialisation.

Even though we called this a classification task, it is not an easy classification because of the imbalance and the few samples that actually have CP. It can be debated whether it is closer to an anomaly detection task were the CP class is counted as anomalies, but anomalies are more often than not detected over a time period and our data does not have a label in time, meaning we do not know at which time step the movement patterns that indicate CP is. We do not have the knowledge to conclude whether anomaly detection methods would work or not but it might be something to investigate. Alternatively, it might be useful to look at ways to train the classification with a more symmetric distribution of the two classes, thus balancing out huge difference and maybe mitigate some of the issues that this brings.

Chapter 7

Conclusion and Future Work

We considered to test out the representation from [41] but as the authors point out, the ordering of the features matters with this approach, and that is something we wanted to avoid. For our purpose, it does not matter if the one specific body part is on the top line of the image or not, the movement of one body part does not necessarily have a connection with the movement in another body part. Doing all the permutations of body parts with both x and y coordinates was too time-consuming for this project now and might be even more so if the tracking model is being extended to include more body parts or joints in the future. The representation of motion and joint-joint representation in [42] could be more promising regarding the movement from time step to time step, but as they were less detailed with their implementation of the image encoding we cannot be certain whether the same issue with the ordering is present or not.

Another suggestion is to combine more statistical methods as preprocessing. The data from the tracking is not perfect and by "cleaning up" some of the mistakes in more ways than we have in this report might help improving the data representation. We also think it might be worth treating the data as 1D channels and either do 1D convolution on each, or do try using RNN since it is still the most used architecture with sequential data.

We also think it could be valuable to have the temporal aspect more prominent. When putting all the time steps into one image, the overlap of point makes it impossible to keep the temporal dynamics completely. One way would be to make one image per timestep, resulting in a 3D representation that keeps the temporal aspect and the positions.

It is also possible to train on each body part individually and combine them into a joint model. The advantage of this is that it would be easier to do preprocessing on

each individual cluster as there are a significant difference in movements between for example the arms and leg which can have very varying clusters versus torso which in general has a denser cluster.

References

[1] Tan, Steinbach, and Kumar, *Introduction to Data Mining*. Pearson, 1 ed., 2014.

[2] Gringer, "File:Linear visible spectrum.svg." `https://commons.wikimedia.org/wiki/File:Linear_visible_spectrum.svg`. Accessed: 2019-06-26.

[3] "NN-SVG." `http://alexlenail.me/NN-SVG/LeNet.html`. Accessed: 2019-06-24.

[4] P. Rosenbaum, "Cerebral palsy: what parents and doctors want to know," *BMJ*, vol. 326, pp. 970–974, 5 2003.

[5] M. Bosanquet, L. Copeland, R. Ware, and R. Boyd, "A systematic review of tests to predict cerebral palsy in young children," *Developmental Medicine & Child Neurology*, vol. 55, no. 5, pp. 418–426, 2013.

[6] M. Burger and Q. A. Louw, "The predictive validity of general movements–a systematic review," *Eur J Paediatr Neurol*, vol. 13, no. 5, pp. 408–20, 2009.

[7] H. T. Myrhaug, K. G. Brurberg, L. Hov, K. Håvelsrud, and L. Reinar, "Prognose for og oppfølging av ekstremt premature barn: En systematisk oversikt," 2017.

[8] A. N. Datta, M. A. Furrer, I. Bernhardt, P. S. Huppi, C. Borradori-Tolsa, H. U. Bucher, B. Latal, S. Grunt, and G. Natalucci, "Fidgety movements in infants born very preterm: predictive value for cerebral palsy in a clinical multicentre setting," *Dev Med Child Neurol*, vol. 59, no. 6, pp. 618–624, 2017.

[9] I. Novak, "Evidence-based diagnosis, health care, and rehabilitation for children with cerebral palsy," *J Child Neurol*, vol. 29, no. 8, pp. 1141–56, 2014.

[10] M. L. Wiik and M. Theisen, "Diagnosis of CP Using Machine Learning," 2018.

[11] P. Rosenbaum, N. Paneth, A. Leviton, M. Goldstein, M. Bax, D. Damiano, B. Dan, and B. Jacobsson, "A report: the definition and classification of

cerebral palsy april 2006," *Dev Med Child Neurol Suppl*, vol. 109, pp. 8–14, 2007.

[12] M. Hadders-Algra, "The assessment of general movements is a valuable technique for the detection of brain dysfunction in young infants. a review," *Acta paediatrica (Oslo, Norway : 1992). Supplement*, vol. 416, pp. 39–43, 11 1996.

[13] C. Einspieler and H. F. R. Prechtl, "Prechtl's Assessment of General Movements: A Diagnostic Tool for the Functional Assessment of the Young Nervous System," *Mental Retardation and Developmental Disabilities Research Reviews*, vol. 11, no. 1, pp. 61–67, 2005.

[14] H. F. Prechtl, C. Einspieler, G. Cioni, A. F. Bos, F. Ferrari, and D. Sontheimer, "An early marker for neurological deficits after perinatal brain lesions," *The Lancet*, vol. 349, no. 9062, pp. 1361 – 1363, 1997.

[15] P. J. Brockwell, *Introduction to Time Series and Forecasting*. Springer Texts in Statistics, Springer International Publishing, 2016.

[16] R. E. Walpole, R. H. Myers, S. L. Myers, and K. E. Ye, *Probability and Statistics for Engineers and Scientists*. Pearson, 2014.

[17] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson, 4 ed., 2018.

[18] N. C. B.-S. . (https://creativecommons.org/licenses/by sa/3.0)], "File:RGB cube.jpg." `https://commons.wikimedia.org/wiki/File:RGB_cube.jpg`. Accessed: 2019-06-30.

[19] R. C. B.-S. . (https://creativecommons.org/licenses/by sa/3.0)], "File:RGB Colorcube Corner White.png." `https://commons.wikimedia.org/wiki/File:RGB_Colorcube_Corner_White.png`. Accessed: 2019-06-30.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Adaptive computation and machine learning, Cambridge, Mass: MIT Press, 2016.

[21] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[22] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. C. Platt, and T. Hoffman, eds.), pp. 153–160, MIT Press, 2007.

[23] M. aurelio Ranzato, C. Poultney, S. Chopra, and Y. L. Cun, "Efficient Learning of Sparse Representations with an Energy-Based Model," in *Advances in*

Neural Information Processing Systems 19 (B. Schölkopf, J. C. Platt, and T. Hoffman, eds.), pp. 1137–1144, MIT Press, 2007.

[24] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[25] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[26] "LeNet-5." `http://yann.lecun.com/exdb/lenet/`. Accessed: 2019-06-30.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[28] "ImageNet." `http://image-net.org/`. Accessed: 2018-12-07.

[29] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv e-prints*, Sept. 2014.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv e-prints*, Dec. 2015.

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, p. 533, 1986.

[33] D. Groos and K. Aurlien, "Infant Body Part Tracking in Videos Using Deep Learning: Facilitating Early Detection of Cerebral Palsy," Master's thesis, Norwegian University of Science and Technology, 2018.

[34] 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.

[35] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR Time Series Classification Archive," July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[36] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and

G. Batista, "The UCR Time Series Classification Archive," October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[37] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Web-Age Information Management* (F. Li, G. Li, S.-w. Hwang, B. Yao, and Z. Zhang, eds.), pp. 298–310, Springer International Publishing, 2014.

[39] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[40] L. P. Jin and J. Dong, "Ensemble deep learning for biomedical time series classification," *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

[41] I. Hajime, H. Naohiko, M. Yohei, O. Makoto, M. Masato, and F. Yoshinori, "Exercise classification using cnn with image frames produced from time-series motion data," *International Journal of Computational Intelligence Systems*, vol. 4, no. 1, pp. 18–21, 2017.

[42] Z. Rostami, M. Afrasiabi, and H. Khotanlou, "Skeleton-based action recognition using spatio-temporal features with convolutional neural networks," in *4th IEEE International Conference on Knowledge-Based Engineering and Innovation, KBEI 2017*, vol. 2018-January, pp. 0583–0587, Institute of Electrical and Electronics Engineers Inc., 2017.

[43] "Keras." https://keras.io/. Accessed: 2019-06-28.

[44] "TensorFlow." https://www.tensorflow.org/. Accessed: 2019-06-28.

Appendix 1

Analysis of the results of the CIMA-tracker

Method:
1) For each video recording representing one infant, every 10th second of movement coordinated generated by the tracker were plotted in an image.
2) Looked through all images representing one infant and labelled the images as *well tracked* or *bad tracked,* and if none of these categories fitted – no label was given.
3) Aggregated all labels representing the same infant into one label, listed in the table below.

FM	Fidgety Movements
0	Absent
1	Sporadic
2	Intermittent
3	Continual
4	Exaggerated

ID nummer	CP score	FM	MOS	CP utkom	Well tracked	Bad tracked
001-1-1	-0.4974603175	2	16	0		
002-1-1	-0.1334814815	3	16	0		
003-1-1	0.2410493827	2	14	0		
004-1-1	-0.3437654321	2	14	0		x
005-1-2	0.493037037	2	16	0		
007-1-1	-0.2988888889	2	16	0		
009-1-1	-0.2287654321	2	14	0		
010-1-1	-0.5501234568	2	14	0		
011-1-1	0.1062962963	2	14	0		x
012-3-1	0.4879012346	1	14	0		x
013-2-1	-0.9611111111	2	14	0		
014-1-1	-0.6325185185	0	6	1		
015-1-1	-0.2138888889	1	14	0		
016-1-1	-0.9079765131	0	5	1		x
017-2-1	-0.5772839506	2	14	0		
018-1-1	-0.8233333333	1	14	0		x
019-1-1	-0.9543809524	0	6	1		
020-1-1	0.2635802469	3	14	0		x
021-2-1	-0.6276190476	0	11	1		
022-1-1	-0.7788888889	0	10	1		
023-2-1	0.6004938272	3	14	0		x
024-2-1	-0.2187513228	1	14	1		
025-1-1	0.05654320988	2	14	0		x
026-1-1	-0.0350617284	2	16	0		
027-1-1	-0.4482716049	2	16	0		
028-1-1	-0.7248994709	0	9	1		
029-2-1	-0.2968994709	0	7	1		
030-2-1	0.4301234568	2	14	0		x
031-2-2	-0.2411111111	0	5	1		
032-1-1	-0.1546984127	2	16	1		
033-1-1	0.7216049383	2	16	0		x
034-2-1	0.200308642	3	16	0		x
035-1-1	-0.9094789705	0	10	1		
036-1-1	0.7714197531	2	14	0		
037-1-1	0.1620987654	2	14	0		
038-1-1	-0.3749074074	2	16	0		
039-1-1	0.1460493827	2	16	0		
040-2-1	0.4667901235	3	16	0		x
041-2-1	-0.487962963	2	16	0		
042-1-1	0.510308642	2	14	0		
043-2-2	0.8446296296	2	13	0		x
044-1-1	-0.7449470899	0	6	1		
045-2-1	0.3037654321	2	14	0		
046-1-1	0.6171604938	2	16	0		x
047-1-1	-0.1276296296	0	5	1		
048-2-1	0.1449382716	1	14	0		x
049-1-1	0.4785185185	2	14	0		
050-1-1	0.5952777778	1	14	0		
051-1-1	0.8373148148	2	16	0		
052-2-3	0.7783333333	2	16	0		
053-1-2	0.3267283951	1	14	0		
056-2-1	0.5495061728	0	14	0		
057-2-1	0.01324074074	2	16	0		

058-1-1	0.6774691358	2	16	0		
060-3-1	0.447037037	0	7	0		
062-1-1	0.1750617284	2	11	0		x
064-1-1	-0.4663809524	2	16	0		
065-1-1	-0.502345679	0	14	0		
066-1-1	-0.2358024691	2	14	0		
067-1-1	-0.7542222222	0	6	1		
068-2-1	0.4162962963	2	14	0		
069-2-1	-0.1308024691	2	14	0		
071-1-1	-0.1728395062	2	16	0		x
072-1-1	-0.06259259259	2	16	0		
073-1-1			*missing data*	0	x	
075-2-1	-0.6688395062	2	16	1		
077-1-1	0.4145061728	0	9	0		
078-1-1	0.5235802469	0	11	0		
079-1-1	0.9989506173	3	14	0		
080-1-1	0.4991358025	1	14	0		
081-1-1	0.1417901235	2	16	0		x
082-1-1	0.7701851852	3	14	0		
083-2-1	0.1530864198	2	16	0		
084-1-1	0.3037777778	2	16	0		
087-1-2	0.6249382716	2	16	0		x
088-1-1	0.3814814815	2	14	0		
089-1-1	0.001172839506	2	16	0		
090-2-1	0.6471604938	3	16	0		
091-2-1	0.5327777778	3	16	0		
092-1-1	0.03968339307	2	14	1		
093-1-1	0.4710582011	2	16	0		
094-1-1	0.517654321	2	14	0		
095-2-1	0.6787037037	2	14	0		
096-1-1	0.5664550265	2	16	0		x
097-1-1	0.7384126984	0	9	0		
099-1-1	0.00538647343	1	16	1		x
100-1-1	0.4917901235	2	14	0	x	
101-1-1	0.8502469136	4	14	0	x	
102-1-1	0.7717901235	2	16	0		
103-1-1	-0.3154938272	2	16	0	x	
104-2-1	0.5575132275	2	16	0	x	
105-2-1	0.3932716049	0	14	0	x	
106-1-1	-0.1583068783	2	14	0		x
107-1-1	0.7184567901	0	14	0		
108-2-1	-0.6775132275	0	11	1		
109-2-1	0.07833333333	2	14	0		
110-2-1	-0.08100529101	2	16	0		
111-1-1	-0.5904360812	0	14	1		
114-1-1	0.1779541446	2	16	1		x
115-2-1	0.427037037	2	14	0		x
116-1-1	0.6166049383	2	11	0		
117-1-1	-0.3061375661	1	14	0	x	
119-1-1	0.3872839506	2	14	0	x	
120-2-1	0.8393333333	2	16	0		x
121-2-1	-0.212037037	2	14	0		
122-1-1	0.8144444444	3	16	0		
123-1-1	0.2782222222	4	12	0		

124-2-1	0.702962963	2	14	0		
125-1-1	0.6752469136	2	16	0		
126-2-1	0.4151851852	2	14	0		
127-2-1	-0.3425507766	0	7	1		
128-1-1	-0.2764197531	1	14	0		x
130-1-1	-0.6575090777	0	12	1		
131-1-1	0.5063703704	0	14	0		
132-1-1	0.3856613757	2	14	0		x
133-1-1	0.02918518519	2	14	0		x
134-1-1	-0.2870840682	2	14	1	x	
135-2-1	-0.2138518519	1	14	0	x	
136-1-1	0.8277160494	2	14	0		
137-1-1	0.847345679	4	14	0		
138-1-1	0.3104938272	2	14	0		x
139-2-1	0.7494179894	2	14	0		x
140-1-1	0.9449382716	3	16	0		
141-2-1	0.6825925926	2	14	0		x
142-2-1	0.8926455026	2	16	0		
143-1-1	0.5335555556	2	14	0	x	
144-2-1	0.6545185185	2	16	0		
145-1-1	0.2891975309	2	16	0		x
146-1-1	-0.1375185185	2	16	1	x	
147-1-1	0.7225925926	0	14	0		
149-2-1	0.1375308642	2	16	0		x
151-1-1	0.7884567901	2	16	0		x
152-1-1	0.7017989418	2	12	0		
153-1-1	0.1062962963	2	16	0	x	
154-1-1	0.5875925926	3	14	0	x	
155-2-1	0.7791851852	2	16	0		x
156-2-1	0.2808465608	2	14	0		
157-1-1	0.7174074074	2	14	0		x
158-1-1	0.5759876543	3	16	0		x
159-1-2	0.6840740741	3	16	0		
160-1-1	0.6144444444	2	11	0		x
161-1-1	0.5668518519	2	12	0		
162-1-2	0.6255555556	2	16	0		
163-1-1	0.7132804233	3	16	0		x
164-1-1	-0.3258641975	2	12	0	x	
165-1-1	0.9435185185	3	16	0		
166-2-2	0.6101851852	2	16	0		x
167-1-1	0.2675925926	2	16	0	x	
168-1-1	-0.02080246914	1	14	0		
300-1-1	0.6369135802	2	14	0		x
302-1-1	0.7745679012	2	16	0		
303-1-1	0.4789506173	2	12	0		
304-1-1	0.2757407407	2	16	0		x
305-1-1	0.7682407407	2	16	0		
307-1-1	0.6849382716	2	16	0		x
LCH_001-1-1	0.197968254	0	14	1		x
LCH_003-1-1	0.3650358423	1	14	1	x	
LCH_004-1-1	0.1063580247	2	16	0	x	
LCH_005-1-1	0.01043209877	0	12	0	x	
LCH_006-1-1	0.3869753086	2	16	0		x
LCH_007-1-1	0.3022222222	2	12	0	x	

LCH_008-1-1	0.2344444444	2	14	0		x
LCH_009-1-1	0.8727777778	3	16	0	x	
LCH_011-1-1	0.07953703704	2	14	0	x	
LCH_012-1-1	-0.2125925926	1	14	0		x
LCH_015-1-1	0.08703703704	2	14	0		
LCH_016-1-2	0.3530864198	4	14	0		
LCH_017-1-1	-0.01597371565	0	14	1		x
LCH_018-1-1	0.4217283951	2	14	0		
LCH_019-1-1	0.1434259259	0	14	0	x	
LCH_020-1-2	0.0249382716	2	14	0	x	
LCH_021-1-1	-0.1366666667	2	16	0	x	
LCH_024-1-1	0.1616049383	2	14	0	x	
LCH_025-1-1	0.2156790123	2	16	0	x	
LCH_027-1-1	0.5592592593	1	14	0	x	
LCH_028-1-1	0.7292592593	3	16	0		x
LCH_029-1-1	-0.1661481481	2	14	0		x
LCH_030-1-1	-0.2983273596	0	14	1		
LCH_031-1-1	0.5981481481	2	16	0		
LCH_032-1-1	0.06672839506	0	14	0	x	
LCH_033-1-1	0.7300925926	2	14	0		
LCH_034-1-1	0.467037037	2	11	0		
LCH_035-1-2	0.5755555556	2	16	0		x
LCH_036-1-1	0.347654321	0	14	0	x	
LCH_038-1-1	0.3375925926	3	14	0	x	
LCH_040-1-1	0.1098765432	2	14	0		x
LCH_043-1-1	0.7381481481	3	16	0		
LCH_044-1-1	0.6884444444	3	14	0		x
LCH_045-1-1	-0.3695679012	2	14	0		
LCH_046-1-1	-0.2332098765	1	14	0	x	
LCH_047-1-1	0.3992592593	0	14	0		
LCH_051-1-1	0.6965185185	3	16	0		
LCH_052-1-1	0.1369135802	2	16	0		x
LCH_053-1-1	0.9206481481	4	7	0		
LCH_054-1-1	-0.6071604938	1	14	0	x	
LCH_055-1-1	0.2490123457	2	14	0		x
LCH_056-1-2	0.3563580247	2	12	0		x
LCH_057-1-1	0.5253333333	2	14	0	x	
LCH_058-1-1	0.9145679012	3	16	0	x	
LCH_059-1-1	-0.379691358	2	16	0		
LCH_061-1-1	0.1882098765	2	16	0		x
LCH_062-1-1	0.0312345679	2	16	0	x	
LCH_063-1-1	0.5185802469	2	16	0		
LCH_064-1-1	0.4666666667	2	16	0	x	
LCH_065-1-1	-0.3785802469	0	12	0		x
LCH_066-1-1	0.3428395062	2	16	0	x	
LCH_067-1-1	0.8911111111	1	12	0		x
LCH_068-1-1	0.8451234568	4	14	0	x	
LCH_069-1-1	0.4519444444	2	16	0		
LCH_070-1-2	-0.7737037037	2	14	0	x	
LCH_072-1-2	0.7164197531	1	12	0		
LCH_073-1-1	-0.7450358423	0	4	1		
LCH_075-1-1	0.3734259259	2	12	0		
LCH_076-1-1	0.7548148148	2	16	0		x
LCH_077-1-1	0.6382098765	2	12	0	x	

LCH_078-1-1	-0.6065791246	0	5	1		
LCH_079-1-1	0.7180246914	0	12	0		x
LCH_080-1-1	-0.2646723647	2	14	1		
LCH_081-1-1	0.6074691358	1	14	0		x
LCH_082-1-1	0.4662345679	2	16	0		
LCH_084-1-1	0.6563580247	3	16	0	x	
LCH_085-1-1	0.4425308642	2	14	0		
LCH_087-2-1	-0.4899223417	2	12	1		x
LCH_088-1-1	0.4934567901	2	16	0	x	
LCH_091-1-1	0.1951851852	2	16	0	x	
LCH_092-1-1	0.5547530864	1	14	0	x	
LCH_093-1-1	0.3392592593	2	16	0	x	
LCH_094-1-1	0.6627777778	2	16	0		x
LCH_095-1-1	0.7517777778	4	14	0		
LCH_096-1-1	0.4680246914	0	14	0	x	
LCH_097-1-1	0.7871604938	3	16	0	x	
LCH_098-1-1	-0.2885802469	2	14	0	x	
LCH_099-1-1	0.1697530864	2	12	0	x	
LCH_100-1-1	0.3453703704	2	16	0		
LCH_101-1-1	0.8315740741	2	14	0	x	
LCH_102-1-1	0.03364197531	2	16	0		
LCH_106-1-1	0.3044444444	2	16	0		
LCH_110-1-1	0.7951111111	2	16	0		x
LCH_111-1-1	0.7823703704	2	14	0		
LCH_114-1-1	0.4375308642	2	16	0		
LCH_117-1-1	-0.3108641975	2	16	0		
LCH_119-1-1	0.395	2	16	0		x
LCH_121-1-1	0.5811111111	2	14	0	x	
LCH_122-1-1	0.8418518519	3	12	0	x	
LCH_124-1-1	0.1722839506	2	16	0	x	
LCH_126-1-1	-0.07302469136	2	16	0	x	
LCH_129-1-1	0.7820740741	2	16	0	x	
LCH_132-2-1	0.3016049383	2	14	0	x	
LCH_136-1-1	0.1203703704	2	16	0	x	
LCH_137-1-1	-0.282654321	2	14	0		
LCH_138-1-1	0.2250617284	2	16	0		x
LCH_139-1-1	0.5297037037	2	16	0	x	
LCH_140-1-1	-0.102	2	16	0		
LCH_141-1-1	-0.04827160494	2	16	0	x	
LCH_142-1-1	0.06303703704	2	14	0		
LCH_143-1-1	0.8712962963	2	16	0	x	
LCH_144-1-1	0.4836419753	3	16	0		x
LCH_145-1-1	0.2082716049	2	14	0	x	
LCH_148-1-1	0.9985185185	3	16	0	x	
LCH_149-1-1	0.6759259259	2	14	0		
LCH_150-1-2	0.4300617284	2	14	0		
LCH_152-1-1	-0.2167144564	0	12	1		x
LCH_153-1-1	0.1267283951	2	14	0	x	
LCH_154-1-1	0.08111111111	3	16	0		x
LCH_155-1-1	0.2908641975	2	16	0	x	
LCH_157-1-1	1	2	12	0		
LCH_159-1-1	0.739382716	2	16	0		
LCH_160-1-1	0.7940740741	3	16	0		
LCH_162-1-1	0.4819135802	2	16	0	x	

LCH_167-1-2	-0.4392894936	0	10	1	x	
LCH_171-1-1	-0.193948626	2	12	1		
LCH_174-1-1	0.590308642	2	14	0	x	
LCH_175-1-1	0.03141975309	2	16	0	x	
LCH_176-1-1	0.7662962963	2	14	0	x	
LCH_179-1-1	0.2955555556	2	16	0		
LCH_180-1-1	0.5574814815	2	16	0		
LCH_181-1-1	-0.4984156379	0	6	1		x
LCH_182-1-1	-0.3219883041	0	6	1		x
LCH_184-1-1	-0.3325925926	2	14	0		x
UoC_001_1_1	0.437037037	3	16	0		x
UoC_002_1_1	0.05385185185	0	12	0	x	
UoC_005_1_1	0.5586666667	0	14	0		
UoC_006_1_1	-0.138863779	0	8	1	x	
UoC_007_1_1	0.1362345679	0	14	0		
UoC_008_1_1	0.2764197531	2	16	0		
UoC_009_1_1	0.3490123457	0	14	0	x	
UoC_012_1_1	0.715037037	2	14	0	x	
UoC_014_1_1	-0.410962963	0	7	0	x	
UoC_015_1_1	0.5840740741	2	14	0	x	
UoC_016_1_1	0.6443209877	2	16	0		
UoC_017_1_1	0.02435185185	0	14	0	x	
UoC_018_1_1	0.3638888889	1	12	0		
UoC_019_1_1	-0.8433333333	0	9	1	x?	
UoC_020_1_1	0.1432716049	2	14	0	x	
UoC_021_1_1	-0.06932098765	1	14	0	x	
UoC_023_1_1	0.4109876543	2	16	0	x	
UoC_024_1_1	0.7856296296	2	16	0		
UoC_025_1_1	0.5321604938	2	11	0	x	
UoC_026_1_1	0.8509259259	2	16	0		
UoC_028_1_1	0.5830246914	2	14	0	x	
UoC_029_1_6	0.3288888889	1	14	0	x	
UoC_031_1_1	0.03197530864	2	14	0	x?	
UoC_032_1_1	0.1974074074	1	14	0	x	
UoC_033_1_1	0.9266666667	2	14	0	x?	
UoC_034_1_1	0.2525	2	16	0		
UoC_035_2_1	0.8645679012	2	16	0	x	
UoC_036_2_1	0.5094444444	2	14	0	x	
UoC_038_1_1	0.1372222222	0	12	0		
UoC_040_1_3	-0.8640448343	0	5	1		
UoC_041_1_1	0.7695679012	2	16	0		x
UoC_042_1_1	0.7416666667	2	16	0		x
UoC_043_1_1	0.3874074074	2	16	0	x	
UoC_044_1_2	0.9679012346	2	16	0		
UoC_045_1_1	0.5298148148	3	16	0	x	
UoC_047_1_1	0.1582716049	1	14	0	x	
UoC_049_1_1	0.8351851852	2	14	0		
UoC_050_1_2	0.6866666667	2	14	0	x	
UoC_054_1_3	0.3078395062	2	16	0		x
UoC_055_1_1	0.875037037	3	16	0	x	
UoC_057_1_1	0.8443209877	3	16	0		
UoC_059_1_1	0.5656790123	1	14	0		
UoC_060_1_1	0.01222222222	2	16	0	x	
UoC_061_1_1	0.6372839506	3	16	0		

UoC_062_1_1	0.9337037037	3	16	0	x	
UoC_063_1_1	0.5011728395	3	16	0		
UoC_064_1_1	0.5851851852	3	16	0		
UoC_066_1_1	0.7648148148	2	16	0		x
UoC_067_1_1	0.5230555556	3	16	0	x	
UoC_068_1_1	0.7718518519	2	16	0		
UoC_069_1_1	0.5866666667	3	16	0	x	
UoC_071_1_1	0.6231481481	0	9	0		
UoC_072_1_1	0.8486419753	3	16	0	x	
UoC_073_1_1	0.4093518519	0	11	0		x
UoC_074_1_1	-0.01734567901	2	16	0		
UoC_075_1_1	0.6231481481	2	16	0		
UoC_076_1_1	0.4425308642	2	14	0		
UoC_077_1_1	0.1051234568	2	16	0		
UoC_078_1_1	0.05617283951	2	16	0	x	
UoC_079_1_1	0.7941975309	2	16	0		
UoC_080_1_1	0.4124074074	2	16	0	x	
UoC_082_1_1	0.7647407407	3	16	0	x	
UoC_083_1_1	0.07395061728	2	16	0		
UoC_084_1_1	0.7717283951	2	14	0	x	
UoC_085_1_1	-0.2940123457	0	12	0	x	
UoC_086_1_1	0.6769907407	2	11	0		
UoC_087_1_1	0.7714814815	2	14	0		
UoC_088_1_1	0.6674074074	2	16	0	x	
UoC_089_1_1	0.1617901235	2	16	0	x	
UoC_090_1_1	0.647962963	2	16	0		
UoC_091_1_1	-0.2225925926	2	12	0		
UoC_093_1_1	0.6283333333	2	14	0	x	
UoC_094_1_1	0.597654321	2	16	0		
UoC_096_1_1	0.4746666667	2	16	0		x
UoC_097_1_1	0.2636419753	1	14	0	x	
UoC_098_1_1	0.0362962963	2	16	0		x
UoC_099_1_1	0.3347530864	2	16	0		
UoC_100_1_1	0.03104938272	2	16	0	x	
UoC_101_1_1	0.7890740741	3	16	0	x	
UoC_102_1_1	0.5712345679	3	16	0		x?
UoC_104_2_1	0.1651234568	2	11	0		
UoC_106_1_1	0.1738888889	2	11	0		
UoC_107_1_1	-0.1107407407	2	16	0		x
UoC_108_2_1	-0.03191358025	2	16	0		
UoC_109_1_1	-0.3331959379	0	14	1	x	
UoC_110_1_1	0.1885802469	2	16	0		
UoC_111_1_1	-0.07543209877	2	14	0	x	
UoC_112_1_1	0.5667283951	3	16	0		x
UoC_113_1_1	-0.08625	2	12	0		
UoC_114_1_1	-0.2116666667	2	12	0		x
UoC_115_1_1	0.7566666667	3	16	0	x	
UoC_116_1_1	0.8125925926	3	16	0		x
UoC_117_1_1	0.02518518519	2	16	0		
UoC_118_1_1	0.734962963	3	16	0	x	
UoC_119_1_1	0.6167592593	2	16	0	x	
UoC_120_1_2	0.4922839506	2	16	0		
UoC_121_1_1	0.4438888889	2	16	0		x
UoC_122_1_1	0.3034567901	3	16	0	x	

UoC_123_1_1	0.0362962963	2	16	0		

NTNU
Norwegian University of
Science and Technology