UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology
Department of Computer Science

Arctic HARE

A Machine Learning-based System for Performance Analysis of Cross-country Skiers

—

Tor-Arne Schmidt Nordmo

INF-3981 Master's Thesis in Computer Science, Spring 2018

Abstract

The advances in sensor technology and big-data processing enable performance analysis of sport athletes. With the increase in data, both from on-body sensors and cameras, it is possible to quantify what makes a good athlete. However, typical approaches in sports performance analysis are not adequately equipped for automatically handling big data.

This thesis presents *Arctic Human Activity Recognition on the Edge*, a machine-learning based system that aims to provide live performance analysis of cross-country skiers. *Arctic HARE* uses on-body sensors and cameras to capture movement of the skier, and provides classification of the perceived technique. We explore and compare two approaches to classifying data, in order to determine optimal representations that embody the movement of the skier.

The viability of *Arctic HARE* is substantiated through a working prototype. We ascertain how to optimally capture the movement of the skier and we qualitatively compare the two approaches through experimental evaluation. Our results reveal we can achieve higher than 96 % accuracy for real-time classification of cross-country techniques.

Acknowledgements

I would like to express my gratitude to my advisor Robert Pettersen for always helping me in any way, and being available for discussions both related and unrelated to this thesis. Your knowledge, feedback, and availability has been invaluable during the writing of this thesis. And to my co-advisor Professor Dag Johansen for pushing me and fueling my academic ambitions.

Special thanks to Aril Bernhard Ovesen for being a dear friend and colleague these past five years.

Thank you to André Pedersen for his friendship and knowledge, particularly during these 5 busy months.

Thank you to my fellow students, particularly the others in Corpore Sano, for interesting discussions and experiences.

Finally, thank you to my family for their love and support throughout my studies and in my life.

Contents

List of Figures

List of Tables

List of Abbreviations

1SK	V2/One skate
2SK	V2 Alt/Two skate
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	comma-separated values
dGNSS	Differential Global Navigation Satellite System
DIA	Diagonal Stride
DP	Double Poling
DPK	Double Poling with Kick
fps	Frames per Second
GDPR	General Data Protection Regulation
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
HAR	Human Activity Recognition
HTTP	Hypertext Transfer Protocol

IaaS	Infrastructure as a Service
IMU	Integrated Measurement Unit
IOT	Internet of Things
LSTM	Long Short-Term Memory
MLP	Multi-layer Perceptron
NLP	Natural Language Processing
NTP	Network Time Protocol
OS	Operating System
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGX	Intel Software Guard Extensions
SLA	Service-level Agreement
SLOC	Source lines of code
V1	Offset/V1 skate

1 Introduction

The number of commercial on-body sensors, known as wearables, or gadgets with integrated sensors has increased tremendously in the last few years [1]. These devices store, aggregate and analyze information from the data collected from the user. This includes simple counting of steps, through low-level activity recognition, to health and sports-related recognition applications that can help a user with health and fitness tracking. There are many possible applications that can utilize the data generated by on-body sensors.

Analyzing the performance of athletes is becoming easier with the growth of technology. Performance analysis of athletes is the act of quantifying sports performance in order to develop an understanding that can inform the conscious or unconscious choices done by the athlete in order to enhance their performance [2]. Multiple products allow coaches and athletes to review their performance manually [3, 4]. Physiological data is also often obtained through on-body sensors [5, 6]. Sports produces very large amounts of data that can be analyzed, however it can be difficult to make general analysis software due to the differences between the sports and differences between data sources.

Jim Gray spoke of the fourth scientific paradigm, *eScience*, which focuses on data exploration in response to the exponential increase in data [7]. And indeed, this increase in volume, speed and dimensionality of data in the world is what we now call *Big Data*. The emergence of big data has called for new methods of dealing with data in general. Solutions range from new database solutions that move past SQL [8], to parallel computing tools such *Apache Hadoop* [9] and *Spark* [10], running on large computer clusters. Due to the increase in wearables, the amount of personal data is also increasing exponentially.

Traditional analytical methods are usually not adequate for analyzing big data [11]. Machine learning has increase in recent years due to computational progress and the increase in data [12]. This makes it perfect for analyzing big data in order to learn underlying information. Big data provides the data foundation, both in terms of amount and dimensionality, that machine learning models are dependent on in order to provide good, general models for analysis. For large machine learning systems, storage of the data and the computational demands of building machine learning models is often cloud-based due to affordability and convenience [13].

However, privacy can be a problem. It becomes possible to analyze personal data, from the increased amount of sensors, which can be used to profile users and learn information that was not explicitly shared [14]. The new EU regulation called the General Data Protection Regulation (GDPR) is known as the most important data privacy regulation in the last 20 years [15]. GDPR aims to give data subjects more control of their data by giving them more rights regarding their data. For example, data subjects will have the right to know who has their data and what it is used for. GDPR will also make manufacturers and companies obliged to design products with privacy and security in mind[16].

There are also issues with the typical cloud-based solution for big data analysis. Service-level Agreements (SLAs) such as latency and throughput can be difficult to maintain. This can be a problem if timing is critical. Doing analysis closer to the *edge* of the system can therefore be more appropriate by employing upstream evaluation [17]. Edge analytics is gaining popularity because it can circumvent some of the issues that arise from cloud-based solutions by sending less data or desensitizing the data before sending the data [18].

1.1 Problem Definition and Goal

This thesis presents *Arctic HARE* which is a system that utilizes machine learning, both in the cloud and on the edge, to do efficient performance analysis on cross-country skiers. The system will utilize a camera and on-body IMU sensors which are electronic devices that contain accelerometers, gyroscopes and magnetometers.

The thesis can be described as follows:

> *This thesis will explore two approaches to automatic performance analysis of cross-country ski athletes. One is based on IMU sensors where we will try to minimize the number of sensors and still achieve acceptable accuracy. The other is based on video data and will evaluate the viability of the approach as a real world application using multiple methods of preprocessing.*

The reason for minimizing the number of sensors used is twofold; we want to reduce the dimensionality of the data and we want to lessen the equipment load of the skier. High-dimensional data can be more difficult for machine learning models to learn from and contain irrelevant features of the data. Sensors can interfere with an athlete both due to its weight and also be in the way of the athlete's movement. This will be discussed further in Section 5.4.

We also want to compare the analysis methods based on video vs. IMU data, both in terms of objective accuracy and in terms of ease of use. Therefore we will explore methods that allow us to quantify the movement of a cross-country skier that will help us analyze their performance.

1.2 Requirements and Limitations

This thesis will design and implement a system to investigate the thesis stated in Section 1.1. The system should be able to perform classification of cross-country ski sub-techniques on a mobile computation device using IMU sensor data, and on a cloud server for video data. The machine learning models used for classification should be trained in the cloud due to training being computationally expensive. However the IMU-based model should be possible to retrieve from the cloud and be used on the mobile computation device for edge analysis.

The following is a list of limitations and system features that are assumed to be out of the scope of this thesis.

1. The performance analysis feedback given to the user will initially consist of the automatic classification of the sub-technique and the cycle length. The system will however provide means of presenting higher-level metrics to the user based on sub-technique and cycle length. More detailed feedback requires specific domain knowledge.

2. The data collected for training the machine learning models used by the

system was obtained from athletes on a skiing treadmill. The ability to control the conditions (speed and incline) gives us more control regarding uniformity of data. The applicability of the system out in the field will not be tested due to varying amounts of snow.

3. The cloud system used for training is made up of two independent nodes that are used for video and IMU data respectively. Therefore, the scalability of the system will not be explored. It will, however, be discussed.

4. Possible privacy-preserving solutions will be discussed in Section 6.2, but not explored.

1.3 Methodology

According to the final report of the ACM Task Force [19], the discipline of computer science can be divided into three major paradigms: theory, abstraction, and design. Theory deals with the mathematical principles and properties of what is to be studied. Abstraction stems from the experimental science that is performed and analysis of the models created. Finally, design is the engineering process that uses a systematic approach to construct systems that solve specific problems.

The focus of this thesis is on the design of the system and therefore mainly adheres to the design paradigm. The system presented in this thesis is both a proof-of-concept and -of-applicability. By this we mean that the system first will be designed and implemented so that it can be used in general for performance analysis for all sports and general movement. Then it will be applied to cross-country skiing and the specific implementation choices related to this will be discussed. The prototype will then be evaluated in a series of experiments to determine optimal design choices for real-time performance.

1.4 Context

In a larger context, this system is relevant to the work undertaken by the Corpore Sano Centre [20]. We focus on inter-disciplinary research within sports science, computer science and medicine with a goal of providing new knowledge and research tools for these fields. The *Arctic HARE* system is therefore extremely relevant because of its applications as a tool within cross-country skiing and sports in general.

Previous works from Corpore Sano range widely; *Vortex* [21] is an Operating System (OS) that uses the *omni-kernel* architecture to provide resource control. In security, *Fireflies* [22] is an overlay network protocol which provides intrusion tolerance in a network. In video streaming, *DAVVI* [23] allows for video distribution over HTTP with search-based composition and recommendations. And in the Artificial Intelligence (AI) domain, *StormCast* [24], which is a distributed AI application used for severe storm forecasting.

Corpore Sano also collaborates with the soccer club Tromsø IL and the national soccer team. Within this collaboration multiple systems for sports analysis have been developed. *Bagadus* [5] integrates a sensor system, a soccer analytics annotation system, and a video processing system using multiple cameras. With *Bagadus* it is possible to track individual players and get stitched panorama video summaries. *Muithu* [25] is a system that allows coaches to annotate live soccer matches and provides a social network for the players and coach. This makes it possible for the coach to track players' nutrition and training. The privacy of the players was preserved with *Code capabilities* [26] that embeds executable code fragments in cryptographically protected capabilities. This realizes flexible access control in the cloud.

1.5 Summary of Contributions

This thesis makes the following contributions:

- We build and evaluate a prototype of *Arctic HARE* that can be used for performance analysis of cross-country skiers.
- We determine the optimal distribution of IMU sensors for quantifying the movement of the skier during skiing sub-techniques.
- We apply video-based machine learning to perform performance analysis of cross-country skiers. This appears to be a novel approach in the field.
- We apply and compare multiple feature extraction methods on the video data in order to determine which gives the best representation of the data.

1.6 Outline

The rest of this thesis is structured as follows:

Chapter 2 gives background information and and overview of related work related to machine learning methods and activity recognition and sports analysis.

Chapter 3 presents the IMU and camera system. Then the applications running on the mobile device and the cloud are presented.

Chapter 4 describes the data collection process and preprocessing of the data. Afterwards it presents the machine learning models used and implementation details of the applications which specifically relate the system to cross-country skiing.

Chapter 5 This chapter presents multiple experiments used to evaluate the system and respective results. The results are then reflected upon.

Chapter 6 presents the conclusion and potential future work for the thesis.

2 Background and Related Work

This chapter gives an introduction to machine learning in general before describing the relevant methods and technologies which are used in the *Arctic HARE* system. After this, edge analytics is introduced along with how it solves some of the problems arising from cloud computing.

The chapter will also present the current state of activity recognition and its applications, both in general and in the sports domain. Then cross-country ski performance analysis is presented. Finally we present related work that utilize similar methods to what has been explained in this thesis.

2.1 Machine learning

Machine learning, or pattern recognition, is the scientific discipline whose goal is to classify objects into distinct categories. The objects are usually called *feature vectors*, and the classified categories are usually called *classes*. The objects are organized data points made up of different attributes, called *features*, that uniquely identify a certain pattern when combined [27]. There are several classes of machine learning algorithms to consider; ensemble methods, neural networks, clustering methods, etc. They are mainly classified into two classes: supervised and unsupervised. Supervised methods require training data in order to learn the differences between the different classes, while unsupervised methods do not [27]. Machine learning methods are dependent on a large amount of data in order to produce a generalized model that can predict accurately. This also means that they can be computationally expensive and be dependent on cloud computing solutions [28, 29].

Transfer learning is an area of machine learning where models are created by utilizing previously trained models. Thus one can utilize layers of a pre-trained model in order to extract features from a dataset, and then either train a new model on those features, or tweak the pre-trained model to handle data related to other classes [30].

2.1.1 Recurrent Neural Networks

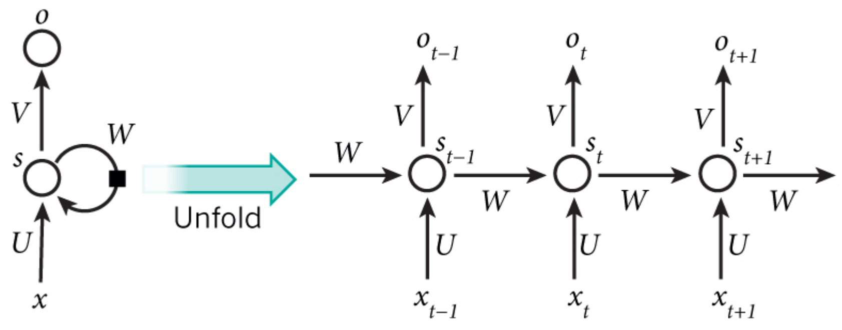


Figure 2.1: Illustration of how RNNs work where x_t refers to the input feature vector, o_t to the output vector and s_t to the hidden state, at time t. U, V and W are matrices that are multiplied at their respective steps [31].

A Recurrent Neural Network (RNN) is a neural network method that specializes in sequence learning. It is a modified version of the feed-forward neural network called RNN which utilizes feedback loops to retain state over time. A sequence is a series of feature vectors that are related in a way, either through time or, in Natural Language Processing (NLP), are parts of the same sentence. This makes them perfect for time series problems and spatially-connected problems [32]. In Figure 2.1 one can clearly see the interdependence of sequential feature vectors, which is saved in the hidden state s_t seen in Figure 2.1.

Long Short-Term Memory (LSTM) units are used as nodes in RNNs as an improvement that makes it easier for the network to remember old feature vectors in a long sequence. LSTM units do this by having specific learning algorithms that make them only remember data relevant to a specific sequence, and forgetting data from previous sequences [33]. In Figure 2.2 σ represents a sigmoid operation that transforms the vector input elements to lie between 0 and 1. This effectively decides which elements to keep from the top input line and what information to store in the current unit. LSTM units also have W matrices which contains the weights that are learned in order to determine which values should be remembered, and which should be forgotten between units [34]. The weights correspond to the connections between the components in Figure 2.2.

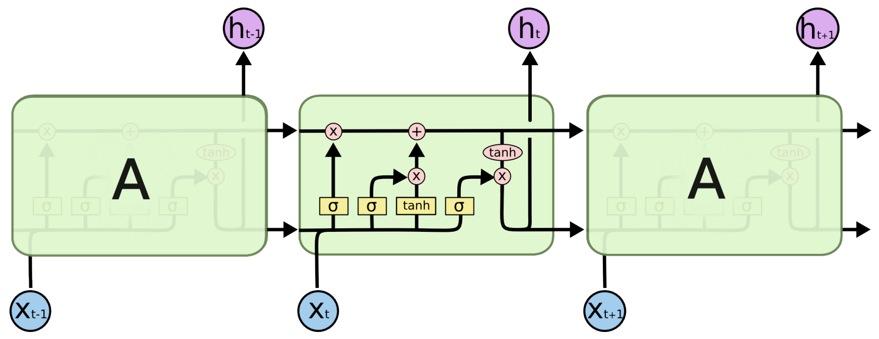


Figure 2.2: Illustration of how Long Short-Term Memorys work and how each unit passes information [34].

2.1.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is another modified version of a feed-forward neural network that works particularly well on image recognition. They are generally made up of three different types of layers: convolutional layers, pooling layers, and fully-connected layers. These can be seen in Figure 2.3. The convolutional layer contains one or more filters with trainable weights as matrix values. The filters in a CNN manage to learn hierarchical features from images, so the filters in early layers learn simple features like where the edges in the image is, while later layers learn more specific features like where faces are located in the image [35].

The convolutional layer is connected to local sub-regions in the input because it is expected that nearby inputs are highly correlated, while inputs further away are less correlated. It is usually followed by a function that introduces non-linearity in the output such as a Rectified Linear Unit (ReLU) or sigmoid function. The pooling layer acts as a dimensionality reduction layer and pools the values of multiple convolutional neurons into a single value. For example, the maximum or the average of the input values effectively downsampling the input. The final layer to consider is the fully-connected layer where each neuron has connections to all of the outputs of the previous layer. This layer is usually used at the end to compute which class the input belongs to [36].

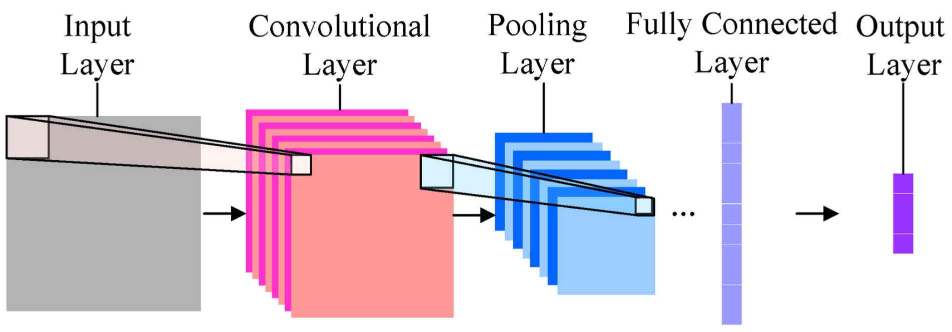


Figure 2.3: Illustration of how Convolutional Neural Networks are constructed, showing the different types of layers [37].

One of the advantages of using CNNs as opposed to standard neural networks is due to the fact that CNNs scale more easily with image dimension size. This is because of the convolutional layers only being connected to local subregions, and parameters can be shared over different filters. Therefore, there are significantly less weights to train in a CNN as opposed to a fully-connected neural network [38].

CNNs are an often used technique in the field of computer vision. The goal of computer vision is to devise models which can gain a high-level understanding of images and video, similar to that of the human visual system, so that tasks dependent on humans can be automated [39]. One example of such a model is OpenPose [40]. OpenPose is a real-time multi-person system that detects keypoints in human body poses, hand gestures and facial expressions. It calculates and outputs the pixel locations of body parts within an image or the frames of a video.

OpenPose utilizes multiple CNNs to accomplish body pose estimation of an image. One of them is called a Convolutional Pose Machine that has a sequence of CNN-based classifiers where at each stage the corresponding classifier predicts the probability of body part locations [41]. The second method does the same but also predicts the orientation of limbs in order to help with the pose estimation. It achieves a high frame rate and has the ability to detect the poses of multiple people in a frame [42]. These, combined with a third method that detects hand keypoints [43], give OpenPose the ability to detect many different body parts, which can be seen in Figure 2.4, and therefore intricate poses at a high frame rate.

Inception-v3 is a CNN architecture which was trained on the ImageNet Large

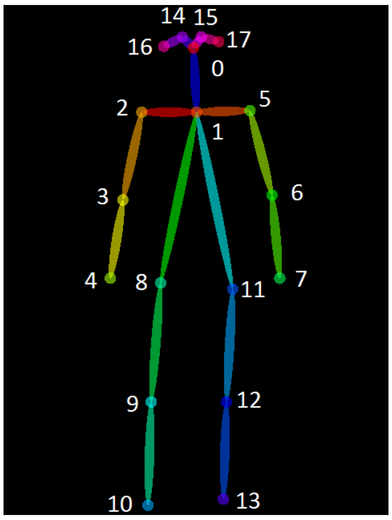


Figure 2.4: Illustration of OpenPose keypoint overlay and output format order [44].

0. Nose
1. Neck
2. RShoulder
3. RElbow
4. RWrist
5. LShoulder
6. LElbow
7. LWrist
8. RHip
9. RKnee
10. RAnkle
11. LHip
12. LKnee
13. LAnkle
14. REye
15. LEye
16. REar
17. LEar

Visual Recognition Challenge dataset [45] from 2012. It achieves state-of-the-art accuracy in recognizing general objects from 1000 classes [46]. How the Inception-v3 is structured can be seen Figure 2.5. It was built according to these design goals:

1. Avoid representational bottlenecks by gently reducing the dimensionality of the data through the network.

2. Localize processing of higher-dimensional representations.

3. Increase the amount of dimensionality reduction due to the lack of loss in representational power.

4. Balance the width and depth of the network.

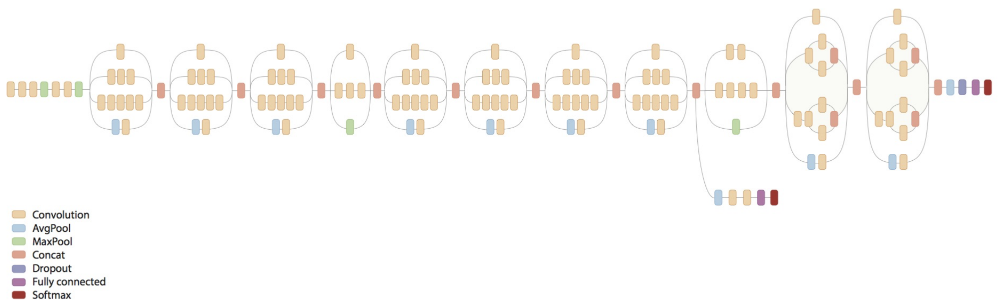


Figure 2.5: Illustration of the architecture of Inception-v3.[47]

Inception-v3 was built in an attempt to improve on the inception architecture of GoogleNet [48], while also keeping the increase in computational cost to a reasonable minimum.

2.2 Edge analytics

Cloud computing is the use of elastic distributed services for computational power, storage and other applications through a cloud service platform. It allows a user or a company to use cloud services on a subset of the computers that are part of the cloud. This makes it easy for users to scale their applications [49]. Machine learning methods are dependent on large amounts of data and computing power in order to train models. Cloud computing solutions are therefore increasing in popularity for machine learning and the Artificial Intelligence (AI) domain in general. Google, Amazon and Microsoft now provide such solutions, both as Infrastructure as a Service (IaaS) and dedicated services [50, 51, 52].

However, there are also problems regarding the cloud. The privacy issues related to cloud computing are two-fold; both the cloud provider and cyber-attacks aimed at the provider are possible risks that users expose themselves to when using cloud-based services. The cloud provider can either be affected by insider breaches like the Vodafone breach [53] and the Snowden leaks [54] or mine personal data themselves, and cyberattacks are occurring more frequently [55, 56, 57]. These can be avoided by utilizing edge computing to allow system designs that can limit the amount of data sent to the cloud or encrypt it locally before sending it for storage in the cloud. Analysis can also be off-loaded onto the edge which allows for upstream evaluation to reduce service latency tremendously [58, 57, 17].

Edge analysis and computing, also known as ubiquitous computing or pervasive computing, are terms describing the use of embedded computational power in general items that is not a desktop computer or a cloud service. It has gotten increasingly attention as a solution for problems arising in cloud computing. The technological advances which give more computationally powerful appliances and mobile phones, which also has given rise to the Internet of Things (IOT), has made these devices more capable of performing computations on the edge [58]. Wearables are capable of performing computations on data from their sensors which can be used for activity recognition.

2.3 Activity Recognition and Sports Performance Analysis

Recognizing complex human activities is a challenging area of research. It has been approached in two different ways; either by using external sensors [59], such as cameras, or by using on-body sensors. Human activity recognition is applicable in multiple areas such as healthcare for monitoring fitness [60] and patient behavior [61], and military scenarios where knowledge about the movement of troops is important for their safety and tactical strategy [62]. There are multiple facets of human activity recognition which make it very challenging, such as concurrent activities like talking to someone while walking, and subtle differences between different persons activity patterns [59]. The solution to these issues demand a large database containing many different activities from many different people in order to create a model that can be used by the general public out-of-the-box. After such a model is created it could then be tweaked to better fit an individual's activity patterns.

Within the sports domain activity recognition can be used both for team sports [63] and individual sports [64, 65]. It is used in order to quantify and analyze performance so that the athletes can improve. Cross-country skiing is a highly competitive sport that is dependent on various aspects, such as technique. There are multiple race variants that utilize cross-country skiing as the main mode of transport, such as biathlons and marathons [66]. The performance of a ski athlete can depend on anything from the strength of the skier to their individual technique to their physiological state, and any combination of these.

2.3.1 Classical and Skating sub-techniques

Here we will describe a subset of both classical and skating sub-techniques. They are the most frequently used in a professional setting [67] and subsets of these are often used in research in this domain.

Diagonal Stride (DIA) is a classical technique where the skier moves their arms and legs in opposition, similar to how you walk. It is used mainly on uphills [68]. It is the only one of these techniques where the arms move asymmetrically.

Double Poling (DP) is another classical technique used while going slightly downhill or at high speeds. It is done by only pushing against the snow with the poles at the same time, with very little movement of the legs [68].

Double Poling with Kick (DPK) is a classical technique similar to DP, but it also involves a kick. The kick alternates between the left and right foot. This technique is used for traveling across rolling terrain for long distances when conditions are too fast for DIA, but too slow for DP [68].

Offset/V1 skate (V1) is a skating technique, though it is quite different from other skating techniques. It is regarded as the best way to go uphill. It is done in sequence by first pushing the poles down, then planting one ski before planting the other ski [68].

V2/One skate (1SK) is another skating technique. It is called "one skate" because there is one poling action for every leg push. This technique is often used on gentle terrain. It is also known as gear 2 [68].

V2 Alt/Two skate (2SK) is a skating technique, named similarly to the one above because the is one poling action for every other leg push. This is a high speed technique. It is also known as gear 3 [68].

2.3.2 Typical Approaches to Performance Analysis in Cross-Country Skiing

Within cross-country skiing there exists multiple race variants where results are highly dependent on the performance of the athletes. General approaches in performance analysis in cross-country skiing can be split up into three major categories: physiological analysis, biomechanical analysis, and analysis based on technique [67].

Traditionally, the technique-based approaches have been done by visual inspection of video [69], IMU [67] or Global Navigation Satellite System (GNSS) [70] data. Recently, other methods have been explored such as empirically-based rules [71] or naive machine learning on IMU data [72, 73].

Automatic detection of cross-country skiing sub-techniques using machine learning has been done before, but is currently not a typical approach in the field. Gløersen *et al.* automatically detected cycle length (how far the skier moves during a cycle), cycle duration and sub-techniques (V1, 1SK, and 2SK) using Differential Global Navigation Satellite System (dGNSS) measurements of the head of the skier. They based the cycle on the lateral velocity of the skier. They achieved an accuracy between 98 % and 100 %, depending on how many skiers the model was trained on [72].

There are some drawbacks with the dGNSS approach. The need for stationary base stations that need time to calibrate, and need to be placed so all of them can communicate with each other [74]. Obstacles such as trees and buildings, and differences in elevation make this placement non-trivial.

Rindal *et al.* use two IMUs on the skier's arm and chest and a Multi-layer Perceptron (MLP) to do classification on the sensor data. The dataset they use consists of 10 skiers performing 6 techniques including tucking, herringbone and turning, as well as the 3 classical techniques described above. A cycle detection method is used to split the sensor data based on the cycles. The split data is then interpolated or decimated, to ensure equal length on all splits. They achieved good results with ≈93 % accuracy on a relatively large dataset (over 8000 cycles/feature vectors) [73]. The Arctic HARE system utilizes the same technique for cycle detection, but uses different machine learning methods, and explores different sensor distributions on the body of the skier to maximize accuracy while minimizing the number of sensors.

Rassem *et al.* employed deep learning algorithms on 3D accelerometer data from cross-country skiing. They tested CNN, different versions of LSTMs, and an MLP for classifying data from 1SK and 2SK skating. They segmented the accelerometer data by using a window over 1 second with 50 % overlap [75]. The Arctic HARE system also utilizes deep learning models for classifying the different sub-techniques, but explores different types of data from more classes and employs different preprocessing methods before training.

There is an issue with the the papers by Gløersen *et al.* and by Rindal *et al.* presented above that is due to either not describing their data fully or not having a uniform dataset. Having a non-uniform dataset and evaluating a model based on its accuracy can be deceptive due to the accuracy paradox [76]. These problems and how to circumvent them will be discussed in Subsection 5.1.2.

2.4 Summary

In this chapter we have given a short summary of the field of machine learning, with descriptions of relevant methods used and technologies based on these. Then we described the shortcomings of cloud computing in this context and how to circumvent these with edge computing. Afterwards, we present the field of human activity recognition and how it can be used for sports performance analysis. Finally, we detail how performance analysis is done in the cross-country skiing domain. We also present related work relevant to this thesis.

3 Arctic HARE Architecture and Design

This chapter will describe the architecture and the design choices pertaining to *Arctic HARE*. The system is comprised of an on-body IMU sensor suit, a mobile device, a camera, and a cloud server. These components are used for data acquisition, preprocessing and storage of data, and training of and classification with machine learning models.

3.1 Architectural Overview

The *Arctic HARE* system is an extended version of our previous work with Human Activity Recognition (HAR), a system called *HARE* [77]. *HARE* consisted of 4 sensors on the limbs connected to a mobile computation device that was used to perform general human activity recognition. The *Arctic HARE* system extends it by using one more sensor on the chest, integrating video data, utilizing more appropriate preprocessing techniques, and more powerful hardware. An overview of the architecture can be seen in Figure 3.1.

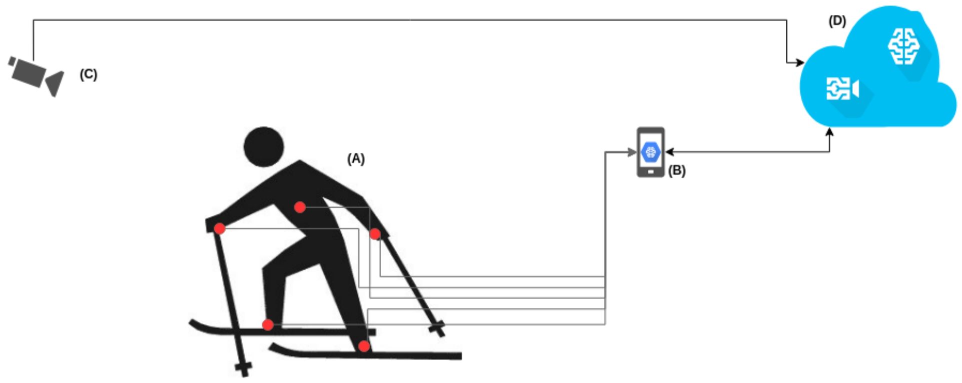


Figure 3.1: Architectural overview of the *Arctic HARE* system. (A) illustrates the distribution of the sensors on the body of the user. They are connected to the mobile device (B) which, along with the static camera (C), communicates with the cloud (D).

3.2 IMU Sensor Suit

A system was built to gather sensor data from the user's limbs, as can be seen in Figure 3.2. Five sensors were distributed onto forearms, calves and chest of the user which were connected to a Raspberry Pi via a multiplexer. The sensor on the chest is applied with a monitoring electrode and with velcro straps on the limbs. The Raspberry Pi uses the I2C protocol to read data from the IMU sensors. The multiplexer switches between the sensors, allowing the Raspberry Pi to communicate with multiple identical sensors via the limited pins on the Raspberry Pi itself.

The input/output pins of the Raspberry Pi, seen in Figure 3.3, that make it particularly easy to connect and interface with external devices and sensors. A ribbon cable that fits onto the Raspberry Pi was soldered to the multiplexer such that power source, ground, and I2C data and clock lines were connected to the master connectors. Then the slave connectors were soldered to the individual sensors using wires of appropriate length. The power and ground connectors on the multiplexer were also used for power and ground for the sensors, so the wires from all the sensors were twisted together with the wires through the multiplexer connectors. The sensors then needed a way to be fastened. Small plastic boxes that could fit the sensors were cut into with a small dremel tool in order to fit the wires and the velcro straps.

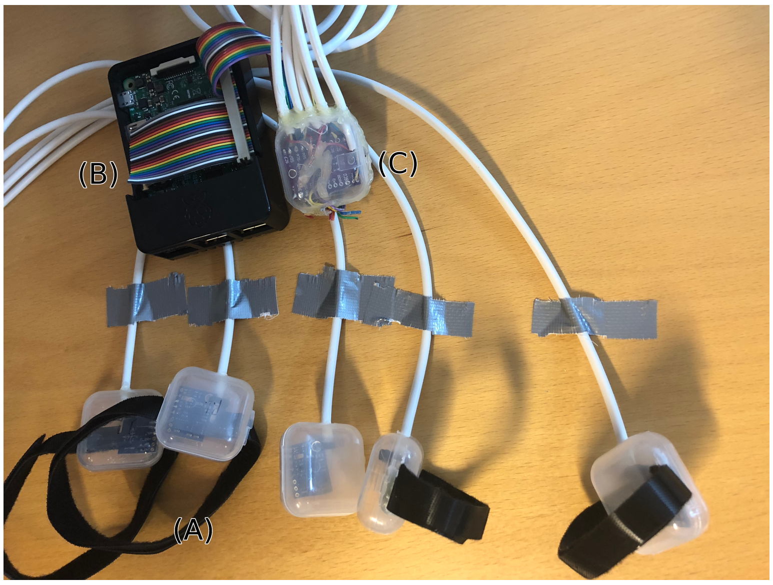
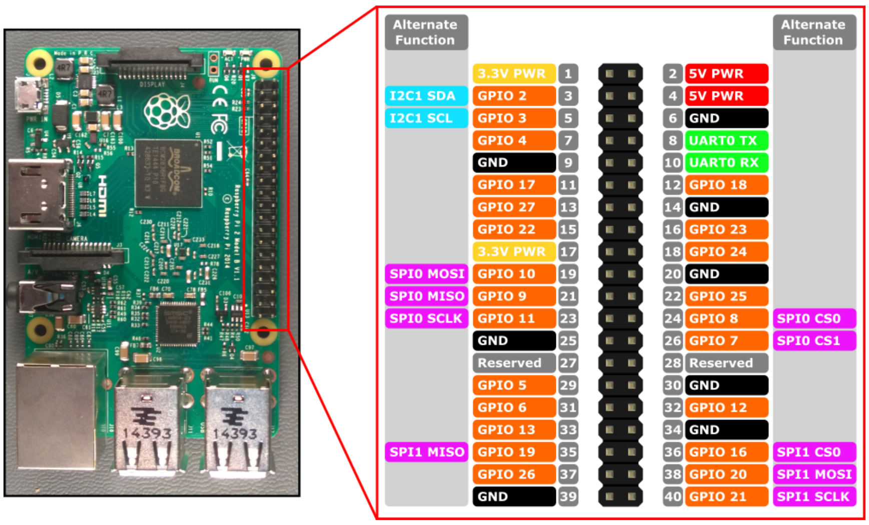


Figure 3.2: Sensor system with IMU sensors (A), Raspberry Pi (B), multiplexer (C) shown.



Alternate Function					Alternate Function
	3.3V PWR	1	2	5V PWR	
I2C1 SDA	GPIO 2	3	4	5V PWR	
I2C1 SCL	GPIO 3	5	6	GND	
	GPIO 4	7	8	UART0 TX	
	GND	9	10	UART0 RX	
	GPIO 17	11	12	GPIO 18	
	GPIO 27	13	14	GND	
	GPIO 22	15	16	GPIO 23	
	3.3V PWR	17	18	GPIO 24	
SPI0 MOSI	GPIO 10	19	20	GND	
SPI0 MISO	GPIO 9	21	22	GPIO 25	
SPI0 SCLK	GPIO 11	23	24	GPIO 8	SPI0 CS0
	GND	25	26	GPIO 7	SPI0 CS1
	Reserved	27	28	Reserved	
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
SPI1 MISO	GPIO 19	35	36	GPIO 16	SPI1 CS0
	GPIO 26	37	38	GPIO 20	SPI1 MOSI
	GND	39	40	GPIO 21	SPI1 SCLK

Figure 3.3: Illustration of Raspberry Pi I/O interface.[78]

The sensors were still moving around inside the boxes, so hot-melt was used in order to keep everything from moving around. It was determined that hot-melt is non-conducive and was therefore also used as an insulating coating on the power and ground wires. The Raspberry Pi and the multiplexer were placed in a belt bag with a hole cut out of it for the wires to the sensors. A power pack was placed in another compartment with a wire that connects to the Raspberry

Pi. The wires are strapped to the body using flexible velcro bands on the upper arms, thighs and chest. This is done to reduce swaying of the wires and to make sure they do not interfere with the skier. The complete system can be seen in Figure 3.2.

The sensors (Figure 3.4) themselves measure acceleration, magnetic field and orientation in three orthogonal directions, and are located on the limbs of the user. These sensors' values across the x, y and z directions comprise the total 46 features that the *Arctic HARE* system uses in its feature vectors, including a timestamp.

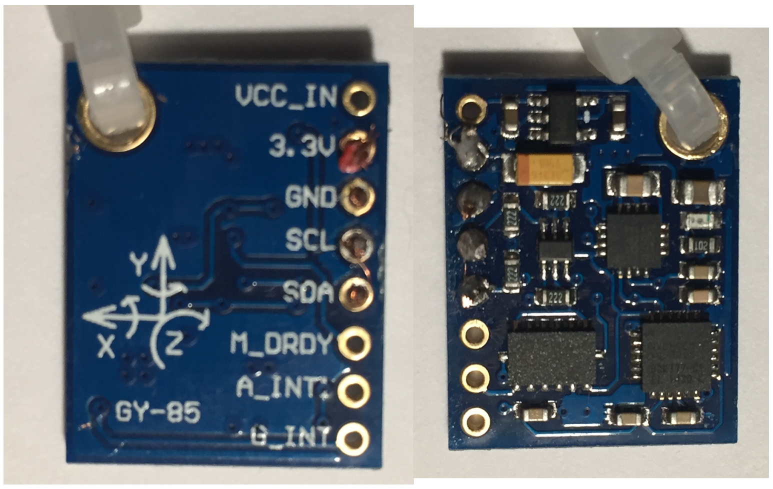


Figure 3.4: Front and back of IMU, containing an accelerometer, gyroscope, and compass.

The sensor locations were chosen to be on the limbs in order to properly quantify the movement of the user. The sensor on the chest was added because it is typically chosen in research [73, 79] and it captures the average overall movement of the user. It can also be used as a reference point for the other sensors to see how much they move with regards to the torso of the user.

The sensor data is generated at a variable rate due to the switching on the multiplexer and the mobile device's scheduler. The maximum frame rate after 10000 samples of the duration between two different measurements was determined to be at 48 Hz with an average of 40 Hz. The read rate was therefore throttled to the average rate in order to maintain a uniform read rate. As stated previously, the multiplexer has to switch between the different sensors. This means that the data from the different sensors are slightly skewed in time, however the differences are in nanoseconds and it can be argued that human movement does not change substantially at this scale.

3.3 Mobile computation device

The mobile device is an embedded system on the user that is connected to the IMU sensor suit. Two applications were created for the mobile computation device on the user; one for the data collection phase, and one that presents results from the classification to the user.

3.3.1 Collector

The data collection program reads data from the individual IMU sensors and concatenates them to create one data point. Each data point is appended to a file for storage. Each line in the file can be appended with the class it belongs to. The final data file is then used for training of the machine learning model. The collection program can also be configured to send data to the IMU training server instead of saving it to a file.

3.3.2 IMU Classifier

The IMU live classification program uses the model trained on the IMU data to classify new live data from the sensor suit. Due to the temporal form of the IMU data, sequences of the data are considered as feature vectors to be classified. Determining the sequence length is assumed to be dependent on the task that is performed. Therefore we went for a modular approach when dealing with the concatenation of data points. This allows the user to specify either a sliding window or something more elaborate for defining the sequences. The class a given sequence is predicted to be in and its cycle duration is then presented to the user.

The live IMU classification program allows for upstream evaluation that can give the user feedback. This feedback can be low-level information such as what activity you are doing and how long it took, or more high-level concepts. This high-level information can be constructed from the low-level information and knowledge in the specific domain in order to give the user a professional evaluation of their performance.

3.4 Video System

A camera was mounted on the wall next to the user at a single stationary angle, which was used for the video data. This equipment was supplied by the research lab at Alfheim. The camera was connected to a black box embedded system for storage that would automatically save video data to a USB stick if it was connected. The video data is at 42 fps with a resolution of 1440×1080.

The video approach was chosen due to the recent increase in interest and success of computer vision both in industry and academia [80]. Exploring multiple methods of recording human movement can also have advantages for different kinds of movement and in multiple scenarios. Thus comparing or combining the video approach with the IMU-based approach seemed like an interesting area to explore. This will be discussed further in Section 5.4.

3.5 Cloud system

The cloud system trains the respective models on the video and IMU training data. It is also possible to update a given model by training it on new data. This is called *online learning*. Two different applications were devised: a server that can be fed IMU sensor data for live training, and a server that handles preprocessing, training, and classification of video data.

The cloud system is easily scalable considering every model can be distributed in a parameter server fashion. This works by having a centralized server that stores and distributes the updated parameters of models to training servers, similar to what was done with Project Adam [81]. This combined with the mobile computing device makes it easy to scale the system if a public consumer version is to be considered.

3.5.1 IMU Training Server

A cloud server is dedicated to training of the models used for classification of the IMU data. It accepts request for model updates i.e. clients can request updated parameters for their models. The training server can also receive new training data from the collector program. The concatenation is then done on the server side before training the model. The server has an Hypertext Transfer Protocol (HTTP) RESTful Application Programming Interface (API). The *GET* command is used for update requests, and the *POST* command for sending new training data.

3.5.2 Video Classifier

The video classifier has two modes; one to classify new data and one to train a given model on training data. It accepts video data and performs preprocessing of the data before classification or further training. The video classifier pipeline can be seen in Figure 3.5. The preprocessing component is modular which allows the user to specify preprocessing methods that fit the given classification task. Multiple methods can also be used in sequence on the raw video data before classification. It is, however, important to use the same preprocessing methods as was performed on the training data in order to obtain accurate results. Choices regarding preprocessing are discussed in Section 4.3 and Subsection 4.5.2.

The video classifier also has a modular design when it comes to the way the data points are concatenated. One can for example use the IMU data in order to define the sequences, or it can be purely based on the video data. Specific solutions will be explored in Section 4.3.

The classification is performed on a sequence when its elements have been preprocessed and concatenated together. The resulting class prediction and the duration of the sequence are then displayed for the user. This can then be saved for further analysis if needed.

The video system is directly connected to the cloud system and therefore differs from the IMU data in the way it is collected. The training mode works by first specifying the class the movement belongs to. Then, while the user performs the movement, each data point that is collected is assumed to be part of the given class. The data points are then preprocessed and concatenated according to what the user has specified. Finally, the sequence is used to train the model or added to a batch before training depending on the training method.

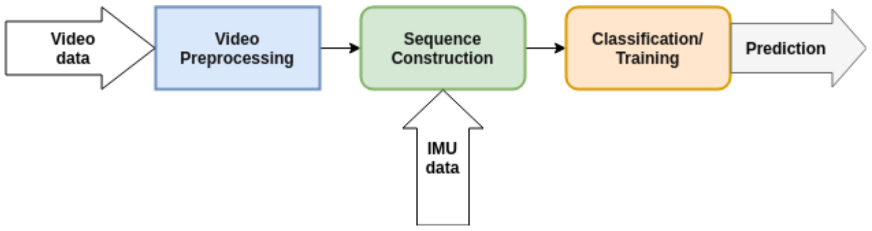


Figure 3.5: Illustration of of the video classifier pipeline. Note that the IMU input is optional due to the modular design of the sequence concatenation. Also, if the server is training a prediction is not outputted.

3.6 Summary

In this chapter we presented the system architecture and design of *Arctic HARE*. We described how the sensor suit was built and how it works. The mobile computation device that the sensor suit is connected to is also outlined, along with the applications that were implemented to run on it. Then the video system, and how it is connected to the cloud, is presented. Finally, the applications implemented for running on the cloud system is described.

4 Data Acquisition, Preprocessing and Implementation

In this chapter we will apply *Arctic HARE*, described in Chapter 3, on the concrete problem of performance analysis of cross-country skiers. First, we will explain the data acquisition process, how the data was preprocessed and how it was annotated so that it could be used as training data. Then, the machine learning methods that were used will be presented and how we chose the respective hyperparameters. Finally, the implementation of the specialized applications for cross-country skiing will be described.

4.1 Data Acquisition

During data acquisition a professional ski athlete wears the sensor suit and is filmed with a stationary camera while skiing with roller skis on a large treadmill. The skier then performs three classical and three skating sub-techniques for approximately 5 minutes each. The sub-techniques that were examined were the ones defined in Subsection 2.3.1; Diagonal Stride (DIA), Double Poling (DP), Double Poling with Kick (DPK), Offset/V1 skate (V1), V2/One skate (1SK), and V2 Alt/Two skate (2SK). They were chosen because they are most frequently used [67], they all utilize the entire body and they can be performed with roller skis on a treadmill. The rest of the sub-techniques are relatively static methods such as downhill tuck and turns, and techniques not often used professionally such as diagonal skating and the herringbone technique. The data was recorded from 7 young national-class male elite skiers skiing at their respective marathon speeds. The two Raspberry Pis that handle the IMU sensor data and the video data respectively are synchronized via the Network Time Protocol (NTP).

During the data acquisition the skiers' chosen speeds and inclines were relatively similar for the respective sub-techniques. The treadmill speed and incline was adjusted during transitions between sub-techniques. The incline also did not change dynamically as they would in the field. This causes a problem regarding the data width, i.e. the data cannot represent the full range of realistic speeds and inclines. However, due to the nature of the system, which allows for edge computing, it can easily be tested in the field and can acquire data from more realistic conditions.

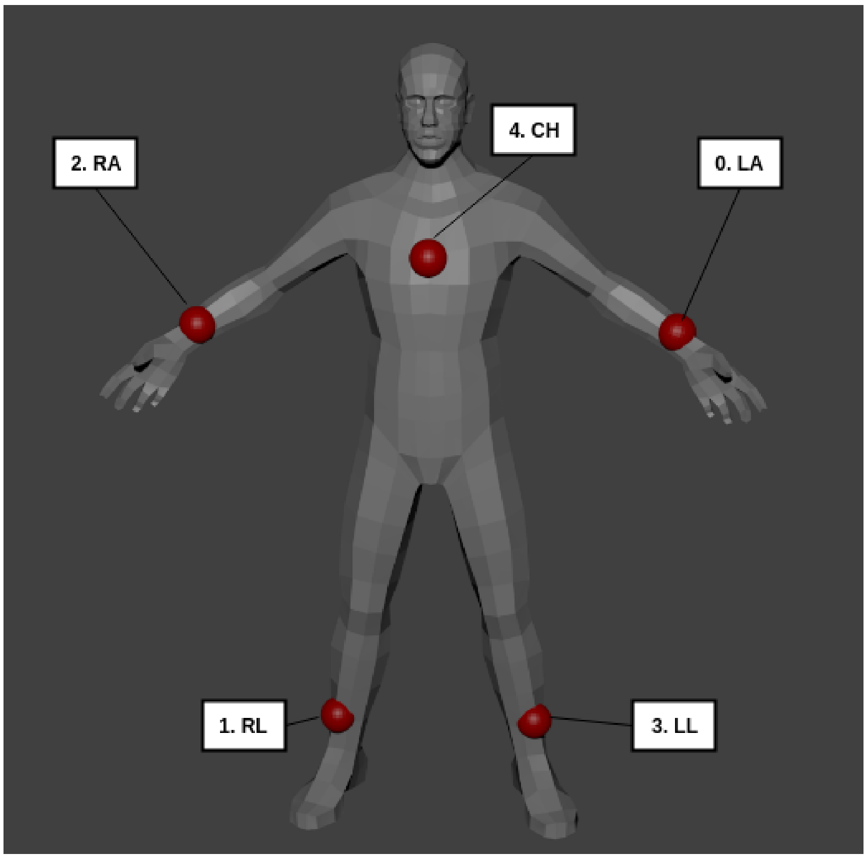


Figure 4.1: Illustration of how the IMU sensors are distributed across the body and the order of data output.

4.2 Sensor Data

The distribution of the IMU sensors and their order can be seen in Figure 4.1. The IMU sensor data is used to determine cycle length by detecting peaks in the data. The method used is similar as what is described in [82]. The z-axis data of the gyroscopic sensor (this is different from [82] due to a different orientation of axes, which is illustrated in Figure 4.2.) on the right arm is filtered using a gaussian low-pass filter over 15 samples to remove high-frequency noise. The peaks of the signal are then detected using a first-order difference approximation of the derivative to find where the slope is zero. The indices of these peaks are then saved in a file to be used for splitting both the IMU and video data into sequences. The length of the longest cycle is stored in the configuration file described in Section 4.5 in order for new data sequences to be padded to the appropriate length.

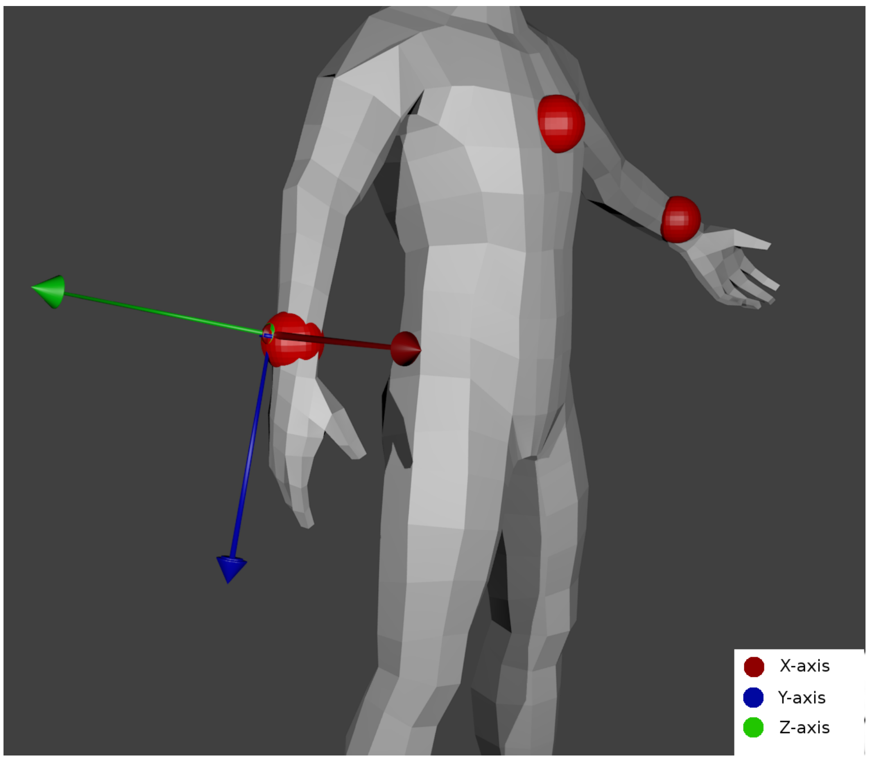


Figure 4.2: How the axes of the IMU (depicted as red orbs) are oriented on the arms. The z-axis of the gyroscope remains mostly orthogonal to the movement of the arms during skiing.

After this multiple data sets were generated, one for each of the sensor distributions. Each distribution was identified by a 5 bit code, where a 1 or a 0 indicated whether IMU sensor data from that specific sensor was present in the respective data set. An example of the code can be seen in Figure 4.3. These datasets were used for the comparison in Section 5.2.

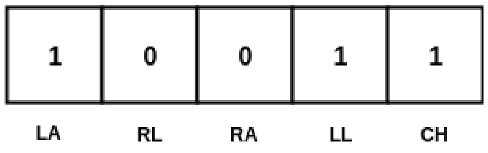


Figure 4.3: Example of 5 bit code corresponding to the sensor distribution that uses sensors on the left arm, left leg, and chest.

Cycle length changes based on which sub-technique is used. Measuring the sub-techniques for five minutes each is not a guarantee for a completely balanced dataset. However, as we can see in table 4.1, the number of cycles are approximately uniformly distributed.

Class	Number of cycles in dataset
DIA	1150
DPK	1087
DP	1232
V1	1235
1SK	1410
2SK	1035

Table 4.1: Table illustrating approximately uniform distribution of feature vectors over the different classes for the IMU training data.

4.3 Video Data

We did not have direct control over the Raspberry Pi connected to the camera, but we could connect USB drives to these cameras in order to get the video footage. The raw video files are stored as multiple MPEG transport stream (`.ts`) files that are then concatenated together to one MPEG-4 (`.mp4`) file for each skier. It was then split up into the respective classes based on the IMU sensor data splits. The video sequences are then split into individual frames which are resized to 256 by 192 pixels with 3 channels (R,G,B) and concatenated as sequences of $256 \times 192 \times 3$-tensors where the sequences correspond to cycles similar to what is done with the IMU data. This dependency on the IMU sensor data to create the video cycles can be circumvented by for example using OpenPose for detecting cycles instead, or using a set interval duration as a sliding window. Before training of the neural networks the tensor sequences are padded with zero-tensors (which corresponds to completely black images) so that all sequences are of the same length. The data is saved in `Numpy` arrays using the `uint8` data type to save space after preprocessing. This can be done because pixel color-values are stored as 8-bit integers. The splitting procedure is very parallelizable, therefore each step was parallelized over the video sequences by using the `multiprocessing` module of `Python`.

Class	Number of cycles in dataset
DIA	399
DPK	394
DP	453
V1	408

Table 4.2: Table that shows the distribution of training data for the video data. Note that this dataset is much smaller than the IMU dataset.

Due to a hardware error, the endings of multiple videos were corrupted. Therefore the video data only consists of the four first classes in order to maintain a balanced dataset. Also, due to the inability to access the hardware containing the files and a problem with the USB interface, we only got footage from 2 skiers. This reduced the amount of possible training data by a considerable amount, as can be seen in Table 4.2. It was therefore important to look at methods of reducing the dimensionality of the data, like the two feature extraction methods described below.

4.3.1 Inception-v3

The first method of feature extraction executed on the video data is Inception-v3. The output sequences from this are acquired after the splitting explained above. Inception-v3 was pretrained on the ImageNet dataset [45]. The fully-connected layers that are used for classification at the end of the network are discarded. In Figure 4.6 some of the outputs of the filters in the earlier layers of Inception-v3 are illustrated. The network extracts features of the individual frames, which are represented as $256 \times 192 \times 3$ tensors, and compresses them into a information-dense 2048-dimensional vector. The fact that it was pretrained on the ImageNet dataset means it should have learned many general features that can be useful in most image classification tasks. It also saves us from having to gather enough data to train it ourselves which also would take a considerable amount of time.

4.3.2 OpenPose

The second method of feature extraction on the video data is done via OpenPose. The OpenPose program runs directly on the scaled video data and produces a video of the resulting key-point skeleton overlayed over the original video. It also writes multiple `.json` files corresponding to the number of frames in the video. These contain the x and y data of body parts and a confidence score of the point corresponding to how sure OpenPose is of the body part being in that location in the specific frame.

OpenPose has the ability to detect the poses of multiple people, but it does not remember across frames which person is which. Therefore the keypoints corresponding to a person in the output can jump between different people in the video. The issue with this is that the video data contains both the skier and the one who controls the treadmill. This was solved by detecting which person had the average location of the keypoints closest to a specified point. Because, due to the camera being static and maintaining a set angle, the skier usually is located in a certain area in the frame. If OpenPose can't locate or interpolate based on spatial dependencies where a specific body part is it sets that point's value to zero. The `.json` files are parsed so that the (x, y)-points can be concatenated into a feature vector.

As stated above, if OpenPose cannot determine the location of a point it sets it to zero. This is not a major issue when a single or a few points are lost. However, due to the angle of the camera which cuts off the head of the skier, OpenPose sometimes has difficulties with interpolating where the body parts in view are located. Very rarely whole frames are lost. There are multiple ways of dealing with missing data in machine learning. One can for example set the missing value to zero or to the mean of the rest of feature vectors' corresponding values.

4.4 Data Annotation

Annotation of the IMU and video data is a crucial part in creating the dataset. It is the process of labeling your training data so that, when the models are being trained, they know what results they are expected to give. It is possible to do this by visual inspection, as can be seen in Figure 4.4, however this process can be slow and the classes can be difficult to discern from each other, so alternative methods were explored. The goal was to split and annotate the data automatically.

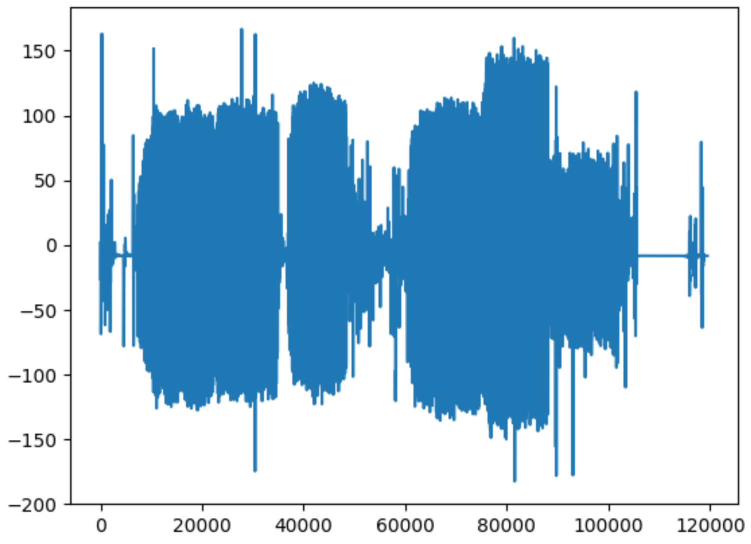


Figure 4.4: The raw output of the z-axis of the gyroscope on the right arm.

The first approach was to try and design an algorithmic scheme that could discern where the transitions between the different classes were in the dataset. Based on knowledge of the axes of the IMU sensors and the differences in the ski techniques, it was possible come up with a few rules to locate these transition points. Looking at the z-axis of the gyroscopes on the arms it was possible to determine if they peaked at approximately the same time, or were a half period apart. Thus we could determine whether the arms were moving symmetrically or asymmetrically. The only class where the arms move asymmetrically is during DIA, therefore we could find the transition from DIA to DPK. After this, the data from the gyroscopes on the legs were examined in order to look for the transition from DPK to DP. The differences between these two techniques is the kick which starts at the end of each cycle. This can be seen as a periodic movement in the data that ends when the skier transitions to DP.

The issue with this approach was that the differences in the skating techniques are more subtle and therefore it can be difficult to make generalized rules for the corresponding transitions. Thus we looked at other methods. Skiing techniques are made up of mainly periodic movements, therefore Fourier analysis seemed like a natural solution. Different sub-techniques should have slightly different spectral density, or distribution over the frequency spectrum. We assumed that the peaks in the spectrum would remain similar between skiers. Doing a Fourier transform of the signal of different gyroscopes proved that this was not the case. The peaks in the frequency spectrum changed between skiers and should also change when a skier moves at a different speed, therefore this wasn't feasible.

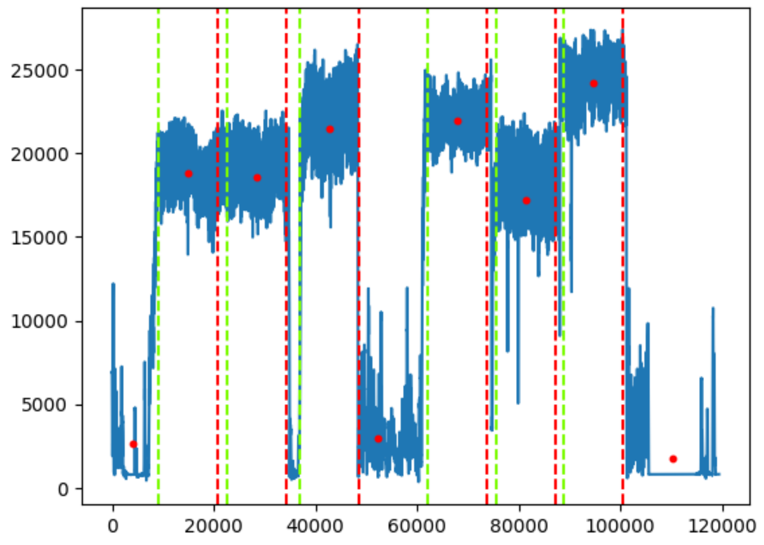


Figure 4.5: Output of annotation process of the same signal as in Figure 4.4. The red dots represent the cluster means and the green and red dashed lines represent the beginning and end of each class respectively.

Finally we looked at clustering, which is a class of unsupervised learning methods. K-means is one of the most well-known clustering methods that tries to assign the data points to k clusters continuously updating the means of the clusters based on the points closest to it [27]. We knew the form of the signal data. It starts with relatively little movement before the skiing started. Then continuous movement while performing the classical sub-techniques, followed a pause while the skier changes equipment, before finally performing the three skating techniques and a small rest at the end. Thus we know that there are six sub-techniques and 3 pauses. Therefore we used k-means to find 9 clusters based on all of the IMU data. The absolute value of the data is convolved with a constant 1-vector with 100 entries. This causes the means to move above zero for the the classes and decreases noise. Then, for each cluster, we looked at the value of the mean to determine if the cluster was located at a pause, or when the skier is performing a sub-technique. The interval of a certain sub-technique is determined to be at the mean of each cluster:

$$\text{Interval}_k = \mu_k \pm \frac{L}{2}$$

where L is the total duration of a sub-technique performance in number of measurements (≈ 12000). If the pauses are too long it can interfere with the clustering, however these can be trimmed or avoided during data acquisition by starting and stopping the program at times closer to the technique exercises. The final result can be seen in Figure 4.5.

4.5 Machine learning methods

All of the models for the IMU sensor data and video data were created using *Keras*[1], a neural network framework running on top of *TensorFlow*[2] [83]. *Keras* provides an Application Programming Interface (API) for constructing neural networks with a layer abstraction. Approximately 1800 Source lines of code (SLOC) where written in total, with around 600 SLOC for the final applications. The rest are composed of code written for testing, preprocessing, and prototyping.

The data was standardized before training and classification. This is done by subtracting the sample mean $\bar{x}_k$ of a feature x from all the corresponding features in the dataset and then dividing them by the sample standard deviation s_k.

$$x_k(\text{standardized}) = \frac{x_k - \bar{x}_k}{s_k}$$

This effectively moves the mean to zero and the variance to 1. Normalization (scaling the data to lie between 0 and 1) and standardization can make the training faster and reduces the chances of the model getting stuck in local minima instead of reaching the global minima due to saturation of the hidden neurons [84]. The mean and standard deviation is stored in a configuration file along with the model in order to standardize new data during classification. During training on new data it is possible to update these values without the original dataset.

A regularization technique was also applied to the network to make the model less prone to overfitting. The method used is called *Dropout* and can be applied to one or more layers in the network. It works by ignoring a neuron in the layer with probability p during a specific iteration through the network. This reduces the co-adaption between neurons by making the presence of a specific neuron unreliable and improves performance on unseen data [85].

Movement cycles in skiing, and human movement in general, will have varying

1. https://keras.io/
2. https://www.tensorflow.org/

durations. This leads to varying sequence lengths of both IMU and video data. There are multiple approaches for dealing with data sequences of varying length so that an Recurrent Neural Network (RNN) can deal with them. The batches can be made to be 1 sequence long. This is known as stochastic learning and works by updating the weights of the network after only a single sequence before proceeding with the next batch. This can be noisy because the weights of the network are continuously changed after propagating each sample sequence through the network.

Another method would be to batch sequences of similar length and pad them within the respective batches. This solves the noise issue of the previous method and does not require padding on meaningless values to the data. However, this can lead to non-uniform batches and learning skewed towards larger batches. This would be an issue if there is a correlation between sequence length and sub-technique.

Lastly, it is possible to pad the sequences so that all sequences are as long as the longest occurring sequence. An advantage of padding them is that all the batches will be uniform and that you can try out different batch sizes during training [86]. An issue that could occur when padding is outlier sequences that are much longer than the other sequences, leading to them having many zeros after padding. This was not an issue with the given data due to the robust cycle detection method and the fact that the skier was on a treadmill which enforced a constant skiing velocity during a specific sub-technique performance.

4.5.1 On IMU Sensor Data

The choice of using a RNN with LSTM units for classification of the IMU sensor data is based on good results on time series data [87] and results from our earlier work [77]. The IMU data was concatenated based on a cycle detection method described in Section 4.2 and padded before being classified by an LSTM network. Datasets for each sensor configuration was created and fed through the network to determine the best configuration (see Section 5.2). The model we looked at has two LSTM layers followed by a fully-connected layer and then a Dropout layer. The number of neurons of the LSTM layers was set to 50 while testing the different sensor configurations.

After a suitable sensor configuration was found, grid search was used to determine optimal hyperparameters while trying to make the model as small as possible, which was a requirement in order to make training and classification more efficient. The number of units in the LSTM layers were varied between 10 and 128 units and the dropout layer's rate varied between 0.1 and 0.7.

The number of epochs indicate how many times the entire dataset is propagated through the network. The dataset is shuffled before each epoch allowing the data to be presented to the network in a different order. The number of epochs was chosen to be 50, because we wanted to avoid overfitting to the training data and it gives a good trade-off [84]. Batch size controls how often the weights of the network are updated. A batch size of 32 was chosen due to its effect on the efficiency of training the model [88]. It is also possible to use different optimizers to calculate the gradient in order to minimize the cost function. Adam [89] was chosen due to its relatively good performance and low memory usage.

4.5.2 On Video Data

Convolutional Neural Networks (CNNs) have achieved state-of-the-art results on image data [46] and are widely used in industry today [90]. Combining this with RNNs was therefore expected to yield good results.

Two different CNN-based methods were used for feature extraction; one based on extracting general features of the frames and one that uses pose estimation from the frames to determine the locations of the body parts of the skier.

The first method is done by sending the processed video data sequences through Inception-v3. We use all the layers except the final layers which

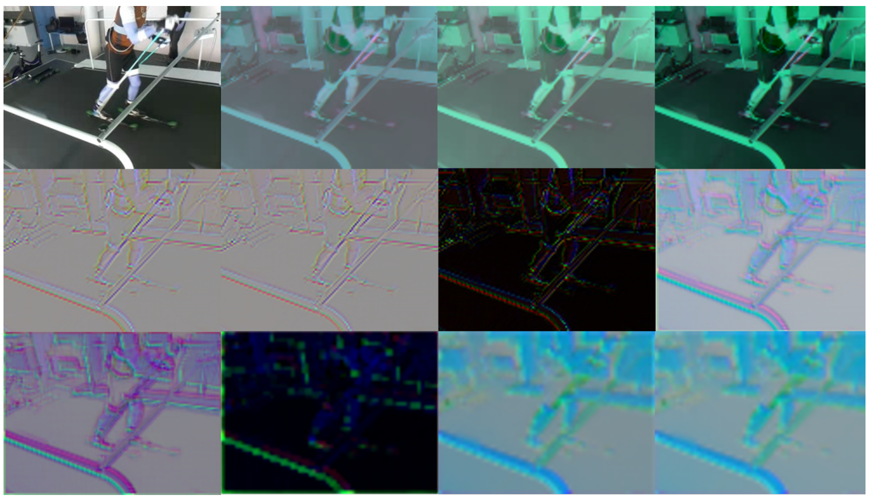


Figure 4.6: Output from some of the filters in the early layers in Inception-v3 using our video data. The layer outputs are ordered from left to right, top to bottom. Note the drop in resolution due to pooling layers and the detection of edges in the frame.

are fully-connected and used for classification. The frames are input into Inception-v3 and reduced to 2048-dimensional vectors. In Figure 4.6 you can see how Inception-v3 transforms the video data in the early layers. The vectors are combined into sequences similar to what is done with IMU data. Then these output sequences are run through an LSTM network which classifies the sequences into the different sub-technique classes.

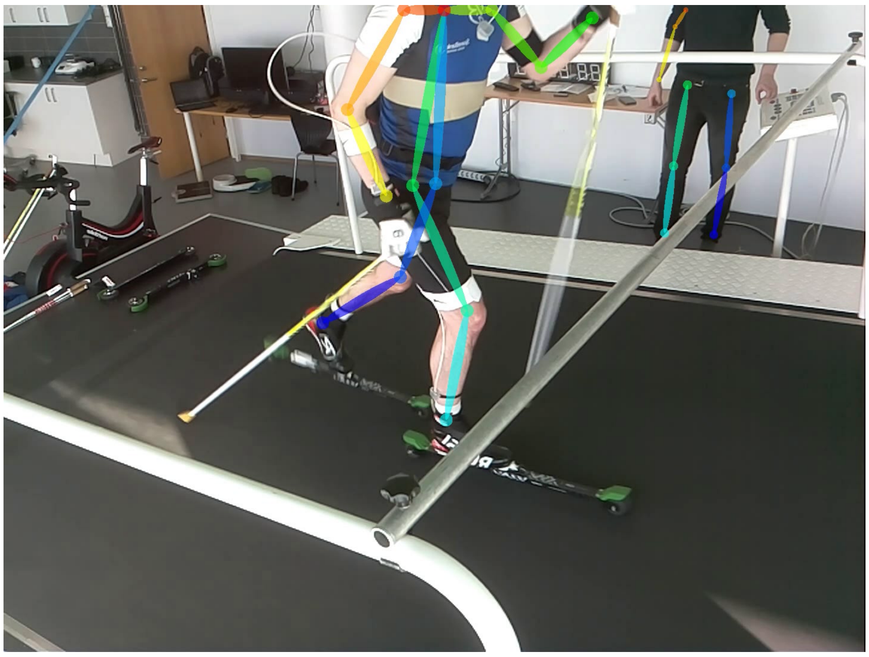


Figure 4.7: Output of OpenPose result overlayed over video. Note that the head is outside of the view of the camera and the lack of points on the treadmill controller. This is discussed further in Section 4.3.

Openpose was the second method used for feature extraction. Openpose extracts the positions of the limbs of the skier in the video frames. These positions, which can be seen in Figure 4.7 are used as features in a 36-dimensional vector which are combined into sequences, similar to what is done above.

The two methods of feature extraction are examples of transfer learning. Both transform the video data into feature vectors that are of lower dimension and contain more relevant data to the classification task. Then, these feature vectors are input into an LSTM network that learns based on the new representations of the data. These LSTM networks use the same layers as what was used in Subsection 4.5.1.

The LSTM architectures were also tuned by a grid-search, but were not constrained by trying to minimize the network size as what was done in Subsection 4.5.1. Due to the fact that the features of the training data are different and the feature vectors have a different dimensionality, one needs to determine which architecture works for the given data. This also means that two different LSTM networks were made for the different feature extraction methods.

There was an issue with the size of the preprocessed video data for training of the model. It would not fit in the main memory of a general purpose computer so a batch generator was used. This loads parts of the data into memory during the training process. This runs in parallel with the training of the model [91]. That means that it is possible to do real-time preprocessing of the video data concurrently with training the model. *Keras* also detects whether a Graphics Processing Unit (GPU) is available and, if so, utilizes it automatically in order to speed up the training process.

4.6 Applications

In this section we will describe the implementation details of the applications written for the mobile device and the cloud system. All applications were written in Python.

4.6.1 Mobile Device Applications

The collection program first specifies which input indices on the multiplexer correspond to the sensors. Then it has to calibrate the gyroscopes, offsetting them based on the average orientation of the limbs. The sensors are read from in sequence with short intervals and the values are placed in a flattened list. This list is prepended with a timestamp before being saved to a CSV-file.

The live classification program uses the machine learning model that was trained on the training data to classify new data while the user is skiing. This program needs to determine where the cycles are on the fly. The cycle detection described above is used to define the sequences. This was done by first simulating a data stream from the Raspberry Pi by using an IMU dataset, setting the sending rate at 40 Hz and letting a process act as the Raspberry Pi. The main process then receives each line in the dataset through a interprocess communication pipe and stores it in a buffer.

The data is stored in the main process buffer until it contains data spanning approximately 5.4 seconds. This duration was chosen because, on average, the cycles are shorter than this, thus the buffer should contain a peak. The buffer is then filtered with the gaussian filter and the peaks in the buffer are detected, similar to what is described in Section 4.2. The peak index values are given with respect to the buffer so they need to be converted to a general reference frame, in other words we want the index values to be with respect to the whole dataset. Therefore we add an offset value to all the peak indexes in a given buffer corresponding to the number of buffer iterations multiplied by the size of the buffer minus the current progress i.e. number of indexes traversed. The data corresponding to the indexes from the current location (0) to the next peak index is then stored in a list and the remaining peak indexes are subtracted by the next peak index in order to shift to the location of the next cycle.

The lists corresponding to cycles are converted to numpy arrays and are concatenated like in Section 4.2 before being classified in the trained model. The result of the classification, the timestamp of the beginning of the cycle, and the duration of the cycle are displayed for the user and saved in a file. The duration is defined as:

$$\text{duration} = \frac{\#\text{measurements in given cycle}}{\#\text{measurements per second}}$$

These cycles can also be sent to the cloud server for further training of the model if the user specifies the sub-technique they are performing. This will be regarded in the section below.

4.6.2 Cloud Applications

The RESTful HTTP server that can receive IMU data works similarly to the live classification program defined above. It accepts data points in a POST request concatenated based on the cycle detection method and appended by the class it belongs to. The concatenation is done on the client side. The model can then be updated in different ways; either by online training with a batch size of 1, or by batching them together before propagating them through the model. The server also accepts GET requests from clients that want to fetch the updated version of the machine learning model.

Due to the lack direct access to the Raspberry Pi that the camera is connected to we had to simulate the `.ts` file writing to create a realistic system for live classification. We use an already stored video dataset and artificially limit the read rate to 2 seconds, which is the duration of each `.ts` file. The cycles are determined by a simulated feed of IMU data, similar to what is done above. This can be argued to be a realistic solution to the cycle issue of the video data described in Section 4.3 due to the fact that most professional skiers wear pulse watches and these can be outfitted with a gyroscope in order to determine the cycle length.

When the video data is acquired by the cloud system it can be preprocessed based on the two feature extraction methods described in Subsection 4.3.1 and Subsection 4.3.2. If Inception-v3 is used, the frames of the video are converted to tensors and appended to a buffer. If OpenPose is used, a subprocess is created to process the video files into `.json` files containing the positions of the body parts. These files are then parsed and the positions are appended to a buffer. Then, for both methods, the indexes of the peaks are converted to corresponding indexes of the frame buffer (due to the IMU data and video data having different frequencies) and are used to define the sequences that will be used as feature vectors. The feature vectors are padded or truncated according to what is specified as the input shape to the trained LSTM model. Finally, they are classified with LSTM model and the result and sequence duration is displayed to he user.

The training mode is implemented in the same way as above with regard to preprocessing and sequence concatenation. The training works by first specifying the sub-technique that the skier is going to perform. After this, the class is appended to a vector for each new recorded sequence. Then, when a batch of sequences is available, it is used to train the model. When the program is terminated the model is saved to a `HDF5` file.

The delay of the process that feeds the video is longer than that of the process that feeds the IMU data. Therefore the main process receives the video data asynchronously in order to not overflow the sensor buffer and allow for peak detection and processing of video while waiting for a new video clip.

Preprocessing the video data is computationally expensive and is therefore dependent on a powerful computer or a cloud-based solution in order attain live classification. This is why we went for cloud-based classification of the video data compared to the edge-based approach used for the IMU data. It makes it possible to maintain a relatively high classification rate compared to running it on the edge. It would also be very taxing on the battery of the mobile computation device which would reduce the usage time of the system.

4.7 Summary

In this chapter we applied the *Arctic HARE* system to cross-country skiing. The chapter details the acquisition and preprocessing of both the IMU sensor and video data. How the training data was annotated was also described. The machine learning methods and corresponding hyperparameters were then presented. Finally, the implementation details for appropriating the applications for cross-country skiing was described.

5 Evaluation

This chapter will describe the evaluation of *Arctic HARE*. The first section will describe the experimental setup and the different ways of how to evaluate machine learning methods. Then, we will go over the experiments used to evaluate the system.

The number of features increase with the number of IMU sensors, and the raw video data also correspond with high-dimensional feature spaces. This leads to problems for the machine learning models, because they require more training data and can learn undesirable aspects from noise in the data. Therefore, we want to reduce the dimensionality of the data and generate a suitable representation that contains the essence of the movements of cross-country skiers. We also want to determine which, of the IMU and video approaches, is better both in terms of ease of use and efficacy.

To evaluate *Arctic HARE*, we seek to answer the following questions:

1. What is the minimum number of IMU sensors and how should they be distributed on the body of the skier so that they can give an acceptable classification accuracy?

2. Which feature extraction methods used for the video data will give the best representation of the data for classification?

3. Which of the IMU- and video-based methods are most appropriate for

performance analysis of cross-country skiers, and can they be combined?

By answering these we can determine how to best represent IMU and video data so that our system can give accurate performance analysis. It will also let us determine whether the video-based approach can be used for accurate and live classification. *Arctic HARE* was designed with multiple IMU sensors, and multiple feature extraction methods for the video data so that we could answer these questions.

5.1 Experimental Setup and Evaluation Metrics

To evaluate *Arctic HARE* we first need to describe the experimental setup i.e. the hardware used for testing the system. Then, we will describe different methods for evaluating machine learning models.

5.1.1 Experimental Setup

To collect data and perform experiments we used a Raspberry Pi 3 model B connected to five ADXL345 IMUs [92] as our mobile computation device. The Raspberry Pi ran Raspbian 4.4 and the server ran Ubuntu 16.04.4 LTS. A Raspberry Pi was chosen due to having similar hardware limitations, size and architecture to modern smartphones, but with a simple interface for connecting the wired IMU sensors. Two different cloud instances were used, one rack server with 16 GB of memory and two quad-core Intel Xeon X5355 CPUs running at 2.66 GHz and a 1 Gbit/s ethernet network interface for training the sensor-based model. A tower computer with quad-core Xeon E5-1620 CPU running at 3.70 GHz and a Titan Xp GPU with 12 GB RAM and 3584 CUDA cores running at a base rate of 1417 MHz was used for training and classification with the video-based model.

5.1.2 Evaluating Machine Learning Methods

A difficult aspect of evaluating machine learning models is to generalize them in the sense that it classifies correctly on new data and not just on the training data. Therefore we cannot simply train the model and then test the model on the same data. The model could then learn specific particularities of the training data. The data is therefore split up into 2 different sets in a ratio of 85/15; a training set which the model is trained on and a test set which is used to gauge the general accuracy of the model. The sets are built by randomly sampling the original dataset, but is stratified so the class distribution remains balanced. This is done by utilizing k-fold cross-validation with $k = 7$. K-fold cross-validation works by splitting the dataset up into k subsets and uses $k - 1$ as training data and the remaining kth subset for validation and testing. The cross-validation is then repeated 5 times and the results are averaged.

K-fold cross-validation is a popular approach to evaluating how good a particular model performs when classifying unknown data. Compared to a training/test split approach which can be biased, you can utilize all your data in order to more accurately ascertain whether a model is generalized. There is however an issue with bias of the variance estimation [93] which leads to high variance. This can be reduced by repeating the cross-validation multiple times and then averaging the results. The subsets are then resampled for each iteration.

As was stated in Subsection 2.3.2 there is a problem with evaluating a model's accuracy when you have non-uniform training data. This is known as the accuracy paradox which basically says that "Predictive models with a given level of accuracy may have greater predictive power than models with higher accuracy" [94]. With very unbalanced datasets, i.e. a 95/5 % split of training data in respective classes, classifying all of the feature vectors to belong in the first class gives an accuracy score of 95 %. The models ability to classify the two classes would be horrible, despite the high accuracy. This is however not an issue for our models because our data is approximately balanced.

Nevertheless, there are multiple other metrics for evaluating the performance of machine learning models that avoids the accuracy paradox. For a binary classification problem where you have a positive and a negative class the following definitions hold [95]:

True Positive is when the model predicts correctly that the feature vector belongs to the positive class.

False Positive is when the model incorrectly classifies the feature vector as belonging to the positive class.

True Negative is the outcome where the model correctly classifies the feature vector as belonging to the negative class.

False Negative finally is the case in which the model incorrectly classifies the model as belonging to the negative class.

In the multi-class case each class C_i have corresponding classifications of predictions similar to those above. However *true negative* would specify the outcome where a feature vector from any other class $C_{j \neq i}$ does not classify as belonging to C_i. Similar for *false negative*.

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision is defined as what proportion of the feature vectors classified as C_i were actually classified correctly. Recall is defined as what proportion of feature vectors that were belong to class C_i were correctly classified as such. A high recall value is important in for example medicine where tumor classified as a false negative can have catastrophic effects [96]. The *F1-score* is the harmonic mean of the precision and recall. It gives a single score that can be used to evaluate a given model, and will always be between the precision and recall values. For multi-class classification a *Confusion Matrix* is also often used. The rows represent the true class distribution and the columns represent the predicted class distribution. Therefore it is possible to determine which classes the model is having difficulties with.

5.2 Comparing Sensor Distributions

The first question related to our goals was to look at an optimal distribution of sensors on the body of a ski athlete that minimizes the number of sensors. In the field of machine learning the size of the training dataset needs to scale exponentially with the number of dimensions l of the dataset in order for the model to achieve acceptable performance. This is known as the *curse of dimensionality* [27]. Training time for neural networks also increases exponentially with l due to the increasing number of weights to train. Another issue that can arise with a large l is that the model can overfit to features that actually are irrelevant to the classification task. Finally, reducing the number of sensors will make the sensor suit easier to apply and will interfere less with the skier.

First we performed an exhaustive search over the entire grouped feature space. The space is grouped because subsets of the features belong to different IMU sensors. As was defined in Section 4.2, the different sensor distributions are represented by a 5-bit code.

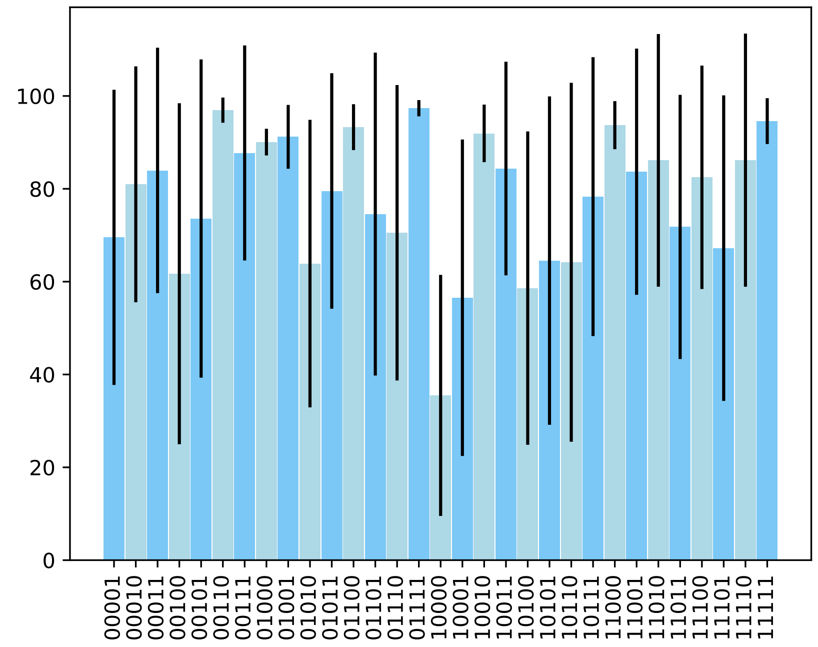


Figure 5.1: Graph of the different sensor distributions and corresponding mean accuracy and standard deviation after running a (non-repeated) 7-fold cross-validation. This was run with a (50,50)-LSTM layer configuration. Note that every other bar represents a sensor configuration that includes the chest-mounted IMU sensor.

The (50, 50) layer configuration used for the results above contain a total of 34, 306 parameters that need to be trained. The top 5 sensor configurations that contain less than 4 IMU sensors, with corresponding accuracies, from Figure 5.1 are then:

Sensor Configuration	Mean Accuracy	Standard Deviation
00110	96.94 %	2.71 %
11000	93.71 %	5.18 %
01100	93.28 %	4.94 %
10010	91.94 %	6.20 %
01001	91.20 %	6.87 %

Table 5.1: Table of the top 5 sensor configurations with the best accuracy.

The best sensor configuration seems to be 00110, which corresponds to the IMU sensors on the left leg and on the right arm. Therefore this configuration will be explored further.

We can also see that the second index, which corresponds to the IMU sensor on the right leg, appears in three of the five configurations in Table 5.1. If we look at Figure 5.1 we can also see that, from the one-sensor distributions, that specific sensor gives the highest accuracy. So combining it with either the left arm or the right arm, which give the top 2 and 3 accuracies respectively, will also be explored further.

Sensor Configuration	**Accuracy**	**F1-Score**
00110	96.67 %, 2.88 %	96.72 %, 2.81 %
11000	91.38 %, 10.23 %	87.75 %, 15.13 %
01100	91.16 %, 12.22 %	85.16 %, 7.95 %

Table 5.2: Table of the top 3 sensor configurations with means and standard deviations of accuracy and f1-score. Results from 5 times repeated 7-fold cross-validation.

Interestingly, the accuracy results in Table 5.2 are worse than in Table 5.1, particularly for the last two configurations. The F1-score is also much lower. This can imply that the two last sensor configurations are slightly dependent on the order the training data was presented in. Nevertheless, we will now continue with grid search to determine an optimal number of LSTM units and dropout rate.

The grid search space is between 10 and 128 for the two LSTM layers, and between 0.1 and 0.7 for the dropout rate.

Hyperparameters	**Accuracy**	**F1-Score**
20,10,0.1	79.25 %, 13.95 %	70.92 %, 27.34 %
32,64,0.2	82.45 %, 10.99 %	78.52 %, 29.17 %
50,50,0.1	96.11 %, 3.98 %	96.33 %, 4.29 %
64,64,0.2	95.87 %, 5.48 %	92.49 %, 3.31 %
128,64,0.4	96.77 %, 2.26 %	94.74 %, 4.02 %

Table 5.3: Table of the top hyperparameter choices with means and standard deviations of accuracy and f1-score. Results from 5 times repeated 7-fold cross-validation.

It is important to note how the hyperparameters scale the number of weights that need to be trained and used for calculating predictions; a $20, 10$ network consists of 4426 trainable parameters, a $50, 50$ network consists of 34306 trainable parameters, and a $128, 128$ network consists of 207622 trainable parameters. We want to reduce the size of the model by minimizing the hyperparameters as well. Based on the results in Table 5.3, we want to keep the 50, 50 configuration due to the high accuracy and f1-score and the fact that it is a relatively small network.

Finally, we will test the classification rate using the final sensor distribution. This efficacy test will be done to determine if the IMU-based approach can handle real-time feedback. The live classification application fills a buffer corresponding to approximately 5.3 seconds before detecting the peaks and concatenating the feature vectors. The time from getting the first feature vector in the buffer to classification was determined to be 5.32 s ± 0.11 s. Thus the time it took to process and classify the feature vectors is inconsequential to the rate of classification and the true bottleneck lies in the fact that classification cannot occur before the buffer is full.

5.2.1 Discussion

The large standard deviations in Figure 5.1, particularly for the sensor distributions that give worse accuracies, can imply that the model is dependent on which subset is used for validation. This is an unwanted trait, we want a model that is accurate irrespective of how the data is presented to it. An interesting thing to note is that using all five sensors gives worse accuracy than using the four sensors corresponding to 01111. More features can possibly lead to an unstable cost function with more local minima due to specific combinations of features.

The right leg appeared in multiple configurations that got the best accuracies. The reason for this might be that the skiers most likely are all right-handed, and therefore probably favor the right leg as well. This can be significant, particularly for less symmetric movements such as Offset/V1 skate (V1).

The best configurations include IMU sensors on a leg and an arm. Combining the IMU sensors on a leg and an arm makes sense considering most movements are highly symmetrical and those limbs encompass the entire movement well.

5.3 Comparison of the Video-Based Methods

Both of the methods tested on the video data are examples of transfer learning. We wanted to determine which method would produce the best features to use in order to classify the video data correctly. It is important to extract the relevant information from the video so that the models are trained correctly. This is related to the curse of dimensionality. We do not want the models to learn irrelevant particularities about the video data. Therefore it can be theorized that OpenPose will give a better result, given that Inception-v3 gives a more general representation of the entire frame.

First we determined the best LSTM layer configuration for the respective feature extraction methods. This was done via a grid search as explained in Subsection 4.5.2.

Layer Configuration	Mean Accuracy	Standard Deviation
32,128	92.52 %	10.9 %
64,128	96.66 %	4.34 %
256,32	97.81 %	3.23 %
256,128	96.94 %	7.58 %
256,64	90.89 %	12.9 %

Table 5.4: Table of the top 5 LSTM layer configurations that give the best accuracy on the video data run through Inception-v3. The numbers were produced by using the (non-repeated) 7-fold cross-validation process described above.

Layer Configuration	Mean Accuracy	Standard Deviation
64,128	93.24 %	2.54 %
64,256	90.57 %	5.64 %
128,64	90.83 %	7.19 %
128,128	92.93 %	3.47 %
256,128	90.43 %	5.62 %

Table 5.5: Table of the top 5 LSTM layer configurations that give the best accuracy on the video data run through OpenPose. The numbers were produced by using the (non-repeated) 7-fold cross-validation process described above.

In Table 5.4 and Table 5.5 you can see the top 5 configurations of the LSTM network that give the best accuracy for the two different feature extraction methods.

The cross-validation done to produce Table 5.4 and Table 5.5 was not repeated in order to save time, but also probably led to the relatively high values of standard deviation. However, now that we have the best LSTM layer configurations these will be examined further and repeated cross-validation will be used.

Based on the results above we chose (256, 32) and (64, 128) as the number of units respectively for the LSTM layers after Inception-v3 and OpenPose feature extraction. The accuracy and F1-scores using both feature extraction methods is given in Table 5.6.

Feature extraction	**Accuracy**	**F1-score**
Inception-v3	94.33 %, 4.90 %	93.32 %, 7.18 %
OpenPose	86.69 %, 13.13 %	87.94 %, 17.23 %

Table 5.6: Accuracy and F1-scores with corresponding standard deviations using the different feature extraction methods.

Confusion matrices were also calculated for the different feature extraction methods. Because k-fold cross-validation uses the kth subset of the data for validation the confusion matrices based on every subset were added together to form a matrix that encompasses the entire dataset. This was repeated and the confusion matrices were averaged and the values were rounded to the nearest integer. The results can be seen in Table 5.7 and Table 5.8.

	DIA	**DPK**	**DP**	**V1**
DIA	386	0	5	8
DPK	3	381	7	2
DP	5	12	433	3
V1	8	2	3	392

Table 5.7: Confusion matrix from using Inception-v3 as feature extraction method.

	DIA	**DPK**	**DP**	**V1**
DIA	345	3	16	34
DPK	5	339	33	14
DP	11	27	409	5
V1	38	11	18	339

Table 5.8: Confusion matrix from using OpenPose as feature extraction method.

Finally we wanted to look at the efficacy of the two methods to determine which one could handle the throughput needed for approximately live classification.

Feature extraction	**Processing Time per File**
OpenPose	20.55 s ± 0.08 s
Inception-v3	10.44 s ± 0.12 s

Table 5.9: Table showing the average time of processing a 2 second video file using the different feature extraction methods. The results were acquired by scaling and processing 100 files.

The results in Table 5.9 correspond with 4.87 Frames per Second (fps) and 9.58 fps respectively.

5.3.1 Discussion

The accuracies, f1-scores and confusion matrices indicate that the Inception-v3 is the best feature extraction method. It appears to create a representation of the data that is easier to classify. This contradicts with our theory in the beginning of this section. However it is interesting to note that both confusion matrices are similarly distributed. It seems both models have a slight difficulty with distinguishing certain cycles of Diagonal Stride (DIA) vs. V1 and Double Poling (DP) vs. Double Poling with Kick (DPK). The second pairing makes sense because the methods are almost the same, with the exception being the kick. The first pairing is less pronounced, however V1 has a slightly asymmetric arm movement, similar to that of DIA. It is possible that, if we had the video data for the other skating techniques, that these models would have more difficulty in classifying the different skating techniques.

OpenPose gives a 2D representation of the positions of the limbs, therefore it can be difficult to differentiate between certain movements. Additionally, due to the skier being slightly cropped out of the frame, multiple data points were lost. These were most likely the reasons why feature extracting with OpenPose gave worse results. However, it is important to note that Inception-v3 uses information from the entire frame to get a more compact representation of the data. Therefore, it might be more susceptible to changes in the background in comparison to OpenPose.

There are multiple problems that arise from having a small dataset, such as high variability. The skier has the ability to move closer and farther away from the camera due to the size of the treadmill. There are also height differences and, albeit slight, technique differences between the skiers. Due to the size of the dataset, it is possible that the models have overfit to the specific skiers that were used for training. It is, however, possible to artificially generate more data from the training data we have by slightly scaling or rotating the frames, which also would make both models more robust.

For the efficacy test it is important to note that we didn't try to optimize the code of OpenPose by removing unnecessary features which most likely affected the results negatively. It is also possible thhat OpenPose resizes the video data again, thereby making the first resizing redundant.

5.4 IMU vs. Video

The IMU-based and video-based methods are trained and tested on a different number of classes and for differently sized datasets. However, comparing these two different approaches to performance analysis in cross-country skiing can still be enlightening. So here we will give a qualitative comparison of the two approaches.

5.4.1 Discussion

There are multiple advantages for both methods. The IMU sensors always generate the data points with respect to the body of the skier, as opposed to the video-based models which are dependent on the angle of the camera and distance from the subject. This is why the IMU sensor system is much easier to test in realistic settings. The sensor suit isn't dependent on the environment it is used in because the sensors are on the body of the skier. Based on the efficacy tests of both the IMU- and video-based methods, the IMU-based method is also more apt for real-time performance analysis and feedback.

On the other hand, cameras do not interfere with the skier's technique, while IMU sensors can be restrictive both due to weight and to the wires being in the way. As was discussed in Section 5.3, the video data could be transformed to artificially generate more data. This could be used to make the video classifier more robust to changes in camera positioning. However, the video-based approach can be applied in certain situations. According the International Ski Federation, both the entire start and the finish line have to be filmed with multiple cameras [97]. These areas should be flat or nearly flat. Therefore, the models only need to be robust with regard to scaling of the subject in the frame. They also need to be able to distinguish between multiple skiers. This can be done manually or with region-based convolutional networks that can localize and classify objects in an image [98].

The biggest issue with the video-based approach is the classification rate, which is significantly slower than the IMU-based approach. Therefore, the video-based approach cannot be used for time critical performance analysis with this hardware, but can be used for post-race analysis of critical moments such as at the finish line. It is also possible to scale the preprocessing stage out to more servers before classifying the sequences. Another solution would be to process less frames, effectively reducing the frame rate of the video. However information could possibly be lost making the classification task more difficult.

5.5 Summary

In this chapter we evaluated the *Arctic HARE* system. In the beginning of the chapter we set out to answer three questions by experimentally evaluating what form of the system is more fitting in the field of cross-country skiing. By this we mean that we looked at the adaptable components of the system, such as how many IMU sensors to consider and what video preprocessing method to use. We also qualitatively compared the IMU- and video-based approaches. Our results provided answers to these questions, and we determined the optimal distribution of IMU sensors and the optimal feature extraction method for the video data. We also determined that, due to the time it takes to process the video data, that the video approach is not viable for continuous live classification, but can be more suitable for post-race analysis near the finish line.

6 Concluding Remarks

We conclude this thesis by summarizing goals and contributions, and describing some possible future work.

As part of this thesis we have designed, implemented and experimentally evaluated *Arctic HARE*, a machine learning-based system for performance analysis of cross-country skiers. In Section 1.1 we stated our problem definition and goal. We wanted to explore two different approaches to automatic performance analysis of cross-country skiers. We wanted to minimize the number of IMU sensors while achieving acceptable accuracy, and see whether the video-based approach is viable by exploring feature extraction methods. In Chapter 5 we set out to answer some questions related to this:

1. What is the minimum number of IMU sensors and how should they be distributed on the body of the skier so that they can give an acceptable classification accuracy?

2. Which feature extraction methods used for the video data will give the best representation of the data for classification?

3. Which of the IMU- and video-based methods are most appropriate for performance analysis of cross-country skiers and can they be combined?

We built the system with this mind by creating the sensor suit with five IMU sensors distributed across the body, and using multiple feature extraction methods on the video data. We also implemented several applications for data acquisition and live classification in order to test the system.

We determined from the results of the experimental evaluation of the system that, for the task of performance analysis of cross-country skiers, it is possible to use two IMU sensors to achieve high classification accuracy. These sensors need to be located on the right arm and left leg. We also determined which feature extraction method that gave the best representation of the video data for the purposes of classification, which was Inception-v3. Finally, we resolved the question of which approach was better. Based on the processing time needed for the video data and the fact that the IMU sensors can easily be used in multiple environment, the IMU-based approach is more fitting for the task. However, the video-based approach can be used for specific areas, such as post-race analysis of the finish line.

6.1 Conclusion

Based on the work presented in this thesis, we draw the following conclusions:

- The *Arctic HARE* system can accurately provide performance analysis in real-time for cross-country skiers.

- For the IMU-based approach we determined a minimal distribution of sensors that reduces the number of features in the data. The model based on the reduced number of sensors achieves 96.11 % accuracy.

- For the video-based approach we determined the best feature extraction method of the two considered. Based on results of their processing efficacy and the corresponding models accuracy, Inception-v3 was more applicable to the task.

- The viability of the video-based approach for real-time performance analysis is dependent on acquiring more data and distributing the pre-processing of the video data. However it is possible to utilize it for more specific cases such as at the starting and finishing lines.

6.2 Future Work

A summary of this thesis will be submitted to the 1st International Workshop on Multimedia Content Analysis in Sports, which is part of the ACM Multimedia Conference 2018. The paper is currently in submission.

There are multiple improvements and functionality that was out of the scope of this thesis that is worth looking into:

More high-level feedback Cycle duration and sub-technique are important determinants of skiing performance [72]. However, it would be interesting to combine other inputs, such as physiological monitors or Global Navigation Satellite System (GNSS) in order to provide more high-level performance feedback to the user.

Realistic conditions The system needs to be tested in more realistic conditions. Currently, the training data is only from the treadmill environment, therefore the models can have difficulty with in-field data. It would also be more appropriate to use wireless sensors for such use-cases.

Privacy solutions The privacy of the user is important to consider when utilizing cloud-based services. This is particularly important due to recent research [99] that states that it is possible to extract information of feature vectors from the models. Therefore, it is important to either anonymize the feature vectors, using methods such as [100], or utilize Intel Software Guard Extensions (SGX) [101] or similar technology to protect code and data from disclosure.

Other extraction methods With two cameras OpenPose can estimate the positions of the limbs of a human in 3D. Vnect [102] is a similar system to OpenPose, but it can extrapolate 3D pose estimation from a single camera source. It is not open-sourced and therefore was not used in this thesis. It would be interesting to see if it would give better results.

Distributed video processing The problem with both of the feature extraction methods were their efficacy, i.e. the rate of preprocessing video data was too slow for live classification. As was stated in Section 5.4 distributing preprocessing of video before concatenating them for classification could make the video-based approach more viable.

Individualize models For professional applications, It could be insightful to have individual models for different user. Then, based on feedback metrics, such as cycle duration, we could determine what makes certain skiers better than other.

Bibliography

[1] P. Lamkin, "Wearable Tech Market To Double By 2021." https://www.forbes.com/sites/paullamkin/2017/06/22/wearable-tech-market-to-double-by-2021/#4e53aff3d8f3, 2017. Online; Accessed 14.05.2018.

[2] P. O'Donoghue, "Research Methods for Sports Performance Analysis," 2010.

[3] "Hudle: One platform to help the whole team improve.." https://www.hudl.com/products/hudl. Online; Accessed 21.05.2018.

[4] "Quintic software: Biomechanical 2D Video Analysis." https://runkeeper.com/. Online; Accessed 21.05.2018.

[5] P. Halvorsen *et al.*, "Bagadus: An integrated real-time system for soccer analytics." https://www.researchgate.net/publication/262365510_Bagadus_An_integrated_real-time_system_for_soccer_analytics, 2014.

[6] I. FitnessKeeper, "Runkeeper." https://www.quinticsports.com/software/. Online; Accessed 21.05.2018.

[7] J. Gray and A. Szalay, "eScience, the Next Decade Will be Interesting!." https://jimgray.azurewebsites.net/talks/ETH_E_Science.ppt, 2006.

[8] techtarget, "Guide to NoSQL databases: How they can help users meet big data needs." https://searchdatamanagement.techtarget.com/essentialguide/Guide-to-NoSQL-databases-How-they-can-help-users-meet-big-data-needs, 2018. Online; Accessed 14.05.2018.

[9] "What is Apache Hadoop?." https://www.oreilly.com/ideas/what-is-apache-hadoop. Online; Accessed 21.05.2018.

[10] I. Pointer, "What is Apache Spark? The big data analytics plat-

form explained." `https://www.infoworld.com/article/3236869/analytics/what-is-apache-spark-the-big-data-analytics-platform-explained.html`. Online; Accessed 21.05.2018.

[11] "Why do Machine Learning on Big Data?." `http://www.skytree.net/machine-learning/why-do-machine-learning-big-data/`, 2017. Online; Accessed 14.05.2018.

[12] J. Brownlee, "Machine Learning is Popular Right Now." `https://machinelearningmastery.com/machine-learning-is-popular/`. Online; Accessed 21.05.2018.

[13] D. Linthicum, "Machine learning in the cloud: How it can help you right now." `https://techbeacon.com/machine-learning-cloud-how-it-can-help-you-right-now`. Online; Accessed 29.05.2018.

[14] S. Pearce, "Wearable tech and the privacy issue." `https://www.itproportal.com/features/wearable-tech-and-the-privacy-issue/`. Online; Accessed 22.05.2018.

[15] "GDPR Portal." `https://www.eugdpr.org/eugdpr.org.html`. Online; Accessed 21.05.2018.

[16] "GDPR Key Changes." `https://www.eugdpr.org/key-changes.html`. Online; Accessed 21.05.2018.

[17] A. Carzaniga and A. L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service." `http://www.inf.usi.ch/carzaniga/papers/crw_tocs01.pdf`, 2001.

[18] F. Marr, "Will 'Analytics On The Edge' Be The Future Of Big Data?." `https://www.forbes.com/sites/bernardmarr/2016/08/23/will-analytics-on-the-edge-be-the-future-of-big-data/#2a22a15d3644`. Online; Accessed 22.05.2018.

[19] D. Comer *et al.*, "Computing as a discipline." https://dl.acm.org/citation.cfm?id=63239, 1989.

[20] "Corpore Sano - Life Sciences: A Centre for Sport and Health Technology." http://www.corporesano.no/.

[21] Åge Kvalnes *et al.*, "Omni-Kernel: An Operating System Architecture for Pervasive Monitoring and Scheduling." https://ieeexplore.ieee.org/abstract/document/6919315/, 2014.

[22] H. D. Johansen *et al.*, "Fireflies: A Secure and Scalable Membership and Gossip Service." https://dl.acm.org/citation.cfm?id=2701418, 2015.

[23] D. Johansen *et al.*, "Search-based composition, streaming and playback of video archive content." https://link.springer.com/article/10.1007/s11042-011-0847-5, 2011.

[24] G. Hartvigsen and D. Johansen, "Co-operation in a distributed artificial intelligence environment—The StormCast application." https://www.sciencedirect.com/science/article/pii/095219769090046O, 1990.

[25] D. Johansen *et al.*, "Muithu: Smaller Footprint, Potentially Larger Imprint." https://munin.uit.no/handle/10037/5120, 2012.

[26] R. van Renesse *et al.*, "Secure Abstraction with Code Capabilities." https://arxiv.org/abs/1210.5443, 2012.

[27] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 4 ed., 2009.

[28] A. Woodie, "Five Reasons Machine Learning Is Moving to the Cloud." `https://www.datanami.com/2015/04/29/5-reasons-machine-learning-is-moving-to-the-cloud/`, 2015.

[29] "Cloud AI." `https://cloud.google.com/products/machine-learning/`, 2017.

[30] J. Yosinski *et al.*, "How transferable are features in deep neural networks?." `https://arxiv.org/abs/1411.1792`, 2014.

[31] D. Britz, "Reccurent Neural Networks Tutorial, Part 1 - Introduction to RNNs." `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`, 2015. accessed 10.02.2018.

[32] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," 1997.

[34] R. Bartyzal, "Understanding LSTM Networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. accessed 12.05.2018.

[35] M. Chivers, "Differential GPS Explained." `http://www.esri.com/news/`

arcuser/0103/differential1of2.html, 2012.

[36] A. Karpathy, "Convolutional Neural Networks (CNNs / ConvNets)." http://cs231n.github.io/convolutional-networks/, 2017.

[37] A. Jain, "Convolutional Neural Networks and Image Recognition." https://www.linkedin.com/pulse/convolutional-neural-networks-image-recognition-aarush-jain/, 2018. accessed 04.04.2018.

[38] J. Gonzales, "Use of Convolutional Neural Networks for Image Classification." https://www.apsl.net/blog/2017/11/20/use-convolutional-neural-network-image-classification/, 2017.

[39] T. Huang, "Computer Vision: Evolution and Promise." http://cds.cern.ch/record/400313/files/p21.pdf, 1996.

[40] G. Hidalgo, "OpenPose: Real-time multi-person keypoint detection library for body, face, and hands estimation." https://github.com/CMU-Perceptual-Computing-Lab/openpose, 2017.

[41] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[42] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[43] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[44] G. Hidalgo *et al.*, "OpenPose Format." https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md, 2018. accessed 05.03.2018.

[45] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge." https://arxiv.org/abs/1409.0575, 2014.

[46] C. Szegedy *et al.*, "Rethinking the Inception Architecture for Computer Vision." https://arxiv.org/abs/1512.00567, 2015.

[47] C. Olah, "Multi-label image classification with Inception net." https://towardsdatascience.com/multi-label-image-classification-with-inception-net-cbb2ee538e30, 2017. accessed 05.03.2018.

[48] C. Szegedy *et al.*, "Going Deeper with Convolutions." https://arxiv.org/abs/1409.4842, 2015.

[49] "What is Cloud Computing?." https://aws.amazon.com/what-is-cloud-computing/. Online; Accessed 16.05.2018.

[50] "Cloud AI." https://cloud.google.com/products/machine-learning/. Online; Accessed 16.05.2018.

[51] "Azure Machine Learning Studio." https://azure.microsoft.com/nb-no/services/machine-learning-studio/. Online; Accessed 16.05.2018.

[52] "Machine Learning on AWS." https://aws.amazon.com/machine-learning/. Online; Accessed 16.05.2018.

[53] J. Kollewe, "Vodafone fined £4.6m for serious breaches of consumer protection rules." https://www.theguardian.com/business/2016/oct/26/vodafone-fined-46m-for-serious-breaches-of-consumer-protection-rules, 2016. Online; accessed 12.05.2018.

[54] BBC, "Edward Snowden: Leaks that exposed US spy programme." http://www.bbc.com/news/world-us-canada-23123964, 2014. Online; accessed 12.05.2018.

[55] S. Angeles, "8 Reasons to Fear Cloud Computing." https://www.businessnewsdaily.com/5215-dangers-cloud-computing.html, 2013. Online; accessed 12.05.2018.

[56] J. Fruhlinger, "What is a cyber attack? Recent examples show disturbing trends." https://www.csoonline.com/article/3237324/cyber-attacks-espionage/what-is-a-cyber-attack-recent-examples-show-disturbing-trends.html, 2018. Online; accessed 12.05.2018.

[57] R. Roman, J. Lopez, and M. Mambo, "Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges," 2016.

[58] P. G. Lopez *et al.*, "Edge-centric Computing: Vision and Challenges." https://dl.acm.org/citation.cfm?id=2831354, 2015.

[59] E. Kim, S. Helal, and D. Cook, "Human Activity Recognition and Pattern Discovery," 2010.

[60] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga, "Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing

and Sports Applications: A Survey," 2010.

[61] V. Osmani, S. Balasubramaniam, and D. Botvich, "Human activity recognition in pervasive health-care: Supporting efficient remote collaboration," 2008.

[62] Óscar D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," 2013.

[63] C. Direkoğlu and N. E. O'Connor, "Team activity recognition in sports," in *Computer Vision – ECCV 2012*, 2012.

[64] B. J. Reily, "Human Activity Recognition and Gymnastics Analysis through Depth Imagery." `https://dspace.library.colostate.edu/handle/11124/170153`, 2016.

[65] K. Soomro and A. R. Zamir, "Computer vision in sports." `https://link.springer.com/chapter/10.1007/978-3-319-09396-3_9`, 2015.

[66] R. Bryhn and K. A. Tvedt, "Kunnskapsforlagets idrettsleksikon (Encyclopedia of Sports)," 1990.

[67] H. Myklebust, "Quantification of movement patterns in cross-country skiing using inertial measurement units." `https://brage.bibsys.no/xmlui/handle/11250/2422117`, 2016.

[68] K. McKenney, "Cross-country ski techniques with video examples." `http://crosscountryskitechnique.com/`, 2014.

[69] R. Zory *et al.*, "Effect of fatigue on double pole kinematics in sprint cross-country skiing." `https://www.researchgate.net/publication/23299580_Effect_of_fatigue_on_double_pole_kinematics_in_sprint_cross-country_skiing`, 2008.

[70] M. Swarén *et al.*, "Using a Real-Time Location System to Track And Analyse Performance in Cross-Country Skiing." `https://www.researchgate.net/publication/311714461_Using_a_Real-Time_Location_System_to_Track_And_Analyse_Performance_in_Cross-Country_Skiing`, 2016.

[71] T. M. Seeberg *et al.*, "A multi-sensor system for automatic analysis of classical cross-country skiing techniques." `https://brage.bibsys.no/xmlui/handle/11250/2463584`. accessed 28.03.2018.

[72] Øyvind Gløersen and M. Gilgien, “Classification of Ski Skating Techniques using the Head’s Trajectory for use in GNSS Field Applications.” https://www.researchgate.net/publication/311735263_Classification_of_Ski_Skating_Techniques_using_the_Head%27s_Trajectory_for_use_in_GNSS_Field_Applications, 2016.

[73] O. M. H. Rindal *et al.*, “Automatic classification of sub-techniques in classical cross-country skiing using a machine learning algorithm on micro-sensor data.” https://www.ncbi.nlm.nih.gov/pubmed/29283421. accessed 5.01.2018.

[74] A. L. Katole *et al.*, “Hierarchical Deep Learning Architecture For 10K Objects Classification.” https://www.researchgate.net/publication/281607765_Hierarchical_Deep_Learning_Architecture_For_10K_Objects_Classification?_sg=Y_9YhUV0C2e__lanG9cxw8d2a-1od0fp-xaH60fS5B7avvY8Io-xg45aZa_Gzos1pAulkGf5sA, 2015.

[75] A. Rassem *et al.*, “Cross-Country Skiing Gears Classification using Deep Learning.” https://arxiv.org/abs/1706.08924, 2017.

[76] R. Lao, “Machine Learning | Accuracy Paradox.” https://www.linkedin.com/pulse/machine-learning-accuracy-paradox-randy-lao, 2017. Online; Accessed 16.05.2018.

[77] T.-A. S. Nordmo, “HARE: Human activity recognition on the edge,” 2017. Capstone Project.

[78] “Raspberry Pi 2 and 3 Pin Mappings.” https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi, 2011. accessed 25.01.2018.

[79] P. Casale *et al.*, “Activity Recognition from Single Chest-Mounted Accelerometer.” https://www.researchgate.net/publication/260425987_Activity_Recognition_from_Single_Chest-Mounted_Accelerometer, 2014.

[80] N. Kishor, “Top 8 Technology Trends for 2018 You Must Know About.” http://houseofbots.com/news-detail/2653-4-top-8-technology-trends-for-2018-you-must-know-about, 2018. Online; Accessed 15.05.2018.

[81] T. Chilimbi, “Project Adam: Building an Efficient and Scalable Deep Learning Training System.” https://www.usenix.org/node/186213, 2014.

[82] O. M. H. Rindal *et al.*, "Automatic Classification of Sub-Techniques in Classical Cross-Country Skiing Using a Machine Learning Algorithm on Micro-Sensor Data," 2017.

[83] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning." https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf, 2016.

[84] Y. LeCun, L. Botton, G. B. Orr, and K.-R. Müller, "Efficient backprop," 1998.

[85] N. Srivastava *et al.*, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf, 2014.

[86] R2RT, "Recurrent Neural Networks in Tensorflow III - Variable Length Sequences." https://r2rt.com/recurrent-neural-networks-in-tensorflow-iii-variable-length-sequences.html, 2016.

[87] A. Graves, "Supervised sequence labeling with recurrent neural networks," 2012.

[88] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures." https://arxiv.org/abs/1206.5533, 2012.

[89] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." https://arxiv.org/abs/1412.6980v8, 2014.

[90] F. Lambert, "Tesla's new head of AI and Autopilot Vision comments on his new role." https://electrek.co/2017/06/21/tesla-ai-autopilot-vision/. Online; Accessed 25.05.2018.

[91] "Keras: The python deep learning library." https://keras.io/. accessed 10.03.2018.

[92] "Adxl345." http://www.analog.com/en/products/mems/accelerometers/adxl345.html. accessed 01.03.2018.

[93] Y. Bengio and Y. Grandvalet, "No Unbiased Estimator of the Variance of K-Fold Cross-Validation." http://www.jmlr.org/papers/volume5/grandvalet04a/grandvalet04a.pdf, 2004.

[94] T. Afonja, "Accuracy Paradox." https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b, 2017.

[95] "Classification: True vs. False and Positive vs. Negative." https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. Online; Accessed 19.05.2018.

[96] "Classification: Precision and Recall." https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. Online; Accessed 19.05.2018.

[97] I. S. Federation, "THE INTERNATIONAL SKI COMPETITION RULES (ICR): BOOK II CROSS-COUNTRY." http://www.fis-ski.com/mm/Document/documentlibrary/Cross-Country/02/95/69/ICRCross-Country2013_clean_English.pdf, 2013.

[98] R. Girshick *et al.*, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation." https://ieeexplore.ieee.org/document/7112511/, 2016.

[99] N. Carlini *et al.*, "The Secret Sharer: Measuring Unintended Neural Network Memorization and Extracting Secrets." https://arxiv.org/pdf/1802.08232.pdf, 2018.

[100] C. Feutry *et al.*, "Learning Anonymized Representations with Adversarial Neural Networks." https://arxiv.org/pdf/1802.09386.pdf, 2018.

[101] "Intel® software guard extensions (intel® sgx)." https://software.intel.com/en-us/sgx. accessed 15.05.2018.

[102] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," vol. 36, 2017.