university of groningen

faculty of science and engineering

Swimmer Classification using an Artificial Lateral Line and Time Series Approaches

Daan Lambert

s2974576

university of groningen

faculty of science and engineering

University of Groningen
Faculty of Science and Engineering

Swimmer Classification using an Artificial Lateral Line and Time Series Approaches

MSc Thesis
Artificial Intelligence

Supervisors: Dr. S. M. van Netten
Dr. B. J. Wolf

Author:
Daan Lambert
s2974576

December 2021

ABSTRACT

The lateral line system is a sensory organ most commonly found in aquatic vertebra. These systems transduce movement of the water into a stimulus. Artificial analogues of such systems can be constructed to replicate this conversion from movement to signal. In this thesis the signal produced by an artificial lateral line (ALL) is used for classification purposes. Recordings were made of 5 swimmers passing by an ALL. We showed success in identifying the swimmers through the utilisation of several classification techniques. We tested two transformation techniques as data representations along side the raw data. The two data transformations which were tested are the spectrogram, and the discrete wavelet transform of the data. Both of these data representations incorporate the frequency domain as well as the time domain. For the classification we used three different machine learning approaches. These are a nearest neighbours approach, a time series forests approach, and a deep learning approach. All three of these approaches represent a different niche in the field of time series classification. We show that the raw data representation outperforms the spectrogram in all situations. The wavelet transform outperforms the raw data depending on the classifier used. The classifiers produce similar results on the data set, when comparing the best performing data representations. Each of the classifiers can be useful depending on circumstances and aim of the system. Future research should be conducted to strengthen the conclusion made by this thesis.

Contents

Chapter 1

Introduction

This focuses on classification of swimmer styles based on the data gathered by an artificial lateral line (ALL). We aim to accomplish this through the use of various machine learning (ML) approaches specifically focusing on time series classification. In this introduction we will first give a short overview of the main ideas behind this thesis. Afterwards we will make the aim of this thesis more concrete. Then in the next section we will move onto the theoretical background of this paper, looking at the specifics of the ALL and the different ML approaches.

1.1 Lateral lines

The lateral line is a sensory system found in certain species of aquatic vertebra, from the common goldfish to various amphibians. Even some species of cephalopods seems to posses a comparable organ, although the structure of the system diverges [1]. The system is sometimes described as akin to hearing, both detecting movement of the surrounding medium, although some researchers prefer to characterise it as "touch at a distance" [2]. The lateral line, similarly to our sensation of touch, is not centralised, but spread out across the body of the animal. The lateral line in particular occupies an interesting space in in relation to other sensory systems, as it has no direct human analogue, meaning the signals are harder to interpret. This system has a long history of biological and physiological research, which we will detail in section 2.1.1.

1.1.1 The artificial analogue

More recently research has also been conducted into simulating these lateral lines through mechanical means. Sensory systems are often a place where many types of research meet, and this is no different for the ALL. ALL's are a new intersection of several different areas of research. These areas include biology, engineering, fluid dynamics and machine learning. Such research has a large pool of studies to pull from, and could lead to breakthroughs in multiple fields.
Research on the artificial lateral line mostly surrounds the tasks of localisation, identification and navigation. These are good first steps to understand the capabilities of mechanical analogues. Other research can build on this by either making the current tasks more specific, or by incorporating these tasks in more complex systems. The tasks usually contain three components: mechanical design, signal processing, and machine learning. Research in this area balances their focus on these parts differently. In the next section we will discuss the aim of this thesis, and how it relates to these three components.

1.1.2 The topics of this thesis

This thesis will focus on the task of identification. The aim is to utilise an artificial lateral line to identify people swimming by an ALL. A set of swimmers will swim by an array of sensors that form an artificial lateral line. The swimmers will also vary the style of swimming they use, either swimming barefoot, with flippers or with a monofin. The data the ALL records will then be used to create a system which aims to identify the swimmer who came by. Initially, the aim of this thesis also included the localisation of the swimmer, during data processing however, it became clear this could not be reliably done with the gathered data.

As for the three components we mentioned earlier, we will first discuss mechanical design. This project will be utilising earlier work in regards to the design and implementation of the the ALL. Although this is an integral part of the work I present, the specifics of the ALL have been explained more comprehensively in other works [3]. That is not to say this project will not be talking about any mechanical components, as it is inseparable from the system that was produced. We will however look at most of the mechanical design through the lens of the signal that was produced.

The second component of the task, signal processing, takes up a larger part of this thesis. Signal processing techniques will be used to transform the signal produced by the ALL's into different forms. These transformations produce a representation of the data that is more straightforward to understand and work with. Such a form could be a signal representing the movement, or a Fourier transform of this signal. The representations aim to be informative of different aspects of the movement in the water.

The last component of the task we will talk about is machine learning. This component will enjoy the biggest focus in this thesis. In particular we will discuss classification tasks in detail. There are many ways to characterise machine learning tasks, and classification more specifically. The most high level aim of these types of tasks is to transform data to an identifying class label. If we compare this with the goal of this thesis, it becomes apparent this is we aim to perform a classification task. This thesis focuses on a version of the transformation that uses data generated by an ALL and generates identifying label corresponding to the identity of a swimmer.

The data was generated at multiple points along a length of time. Such data is commonly known as time series data. Our data also incorporates multiple streams of information from different sensors, which means our representations will be multidimensional. Because of these two points our task is most accurately described as a multidimensional time series classification task.

1.2 Research question

As stated earlier in the introduction, our aim is to be able to identify swimmers through the use of an artificial lateral line. When formalised into a research question this becomes:

- *Can artificial lateral lines be utilised to classify swimmers in a real world scenario*

Earlier research done into similar topics indicates that this should be possible [4].

1.3 Thesis outline

The thesis will have the following sections: Theoretical background, Methods section, Results, and Discussion. In the theoretical background the focus will be on the earlier research and the wider context of the techniques used. It starts with an overview of the historical background of the biological and artificial lateral line. It then covers time series in general, before moving on to the data representations and the classification methods utilised in this thesis.
The methods section will discuss the specifics of the data collection. Afterwards the implementations of the data transforms and classification methods are explained. Lastly the methods section will explain the evaluation criteria used in this thesis.
The results section will present the results of the different methods used in this thesis, as well as a comparison between the different methods. The discussion section explains the results, as well as implications for the wider field. Lastly the discussion will finish with a brief conclusion.

Chapter 2

Theoretical background

This section of the thesis covers the basic structure and functioning of the lateral line. We discuss its functioning and important characteristics to gain a better understanding of the system. Afterwards, the focus will shift to research done on lateral lines, starting with historic and biological research and transitioning to research on the artificial lateral line. Lastly the topics of machine learning and time series classification in the context of the artificial lateral line are introduced. This will provide a general introduction to time series, as well as background information on the transforms and classification methods used in this thesis.

2.1 Lateral lines

The lateral line's basic unit is the neuromast, a sensory structure that is responsible for the translation of movement to a neural signal [5]. These neuromasts come in two variants: Superficial neuromasts, which are located on the exterior of the animal, and canal neuromasts, which are located in fluid filled canals. Both types of neuromast are comprised of two components: the cupula and the hair cells. The cupula is gelatinous structure which sticks out into the surrounding fluid. It detects the movement of the fluid and transfers this movement to the hair cells. The hair cells are located in the surface of the skin or scales, with hair bundles which protrude into the cupula. Through the deflection of the hair bundles, the hair cells can transform the movement of the cupula into an electrical signal. Each hair bundle is tuned to detect the movement of the fluid in a specific direction. This is accomplished through the orientation of the hair bundles, as well as tip links present near the top of the hairs [3].

The deflection of the hair bundles produces a change in membrane potential, causing an impulse. As already stated before hair cells are tuned to a movement in a certain direction. Through the specific structuring of the these hair cells in the cupula and the innervating effects of afferent nerves, a single neuromast is sensitive to movement along a specific axis [7, 8]. This characteristic allows us to make an abstraction of the underlying mechanics such that a neuromast represents a sensor of movement along a single axis. This simple representation is easily replicated through artificial means discussed in later sections.
The two different types of neuromasts are tailored to detect different types of signals. The superficial neuromast are better suited to detect constant flow speeds or DC effects in the fluid, because they are are located at the surface and are minimally affected by surrounding structures. The canal neuromasts are more adapted to detect changes in the fluid flow or the AC component, owing to their positioning in recesses in the skin of the animal. This distinction allows the two types of neuromasts to be sensitive to the direction of a constant flow, as well as any perturbations in the flow that would otherwise

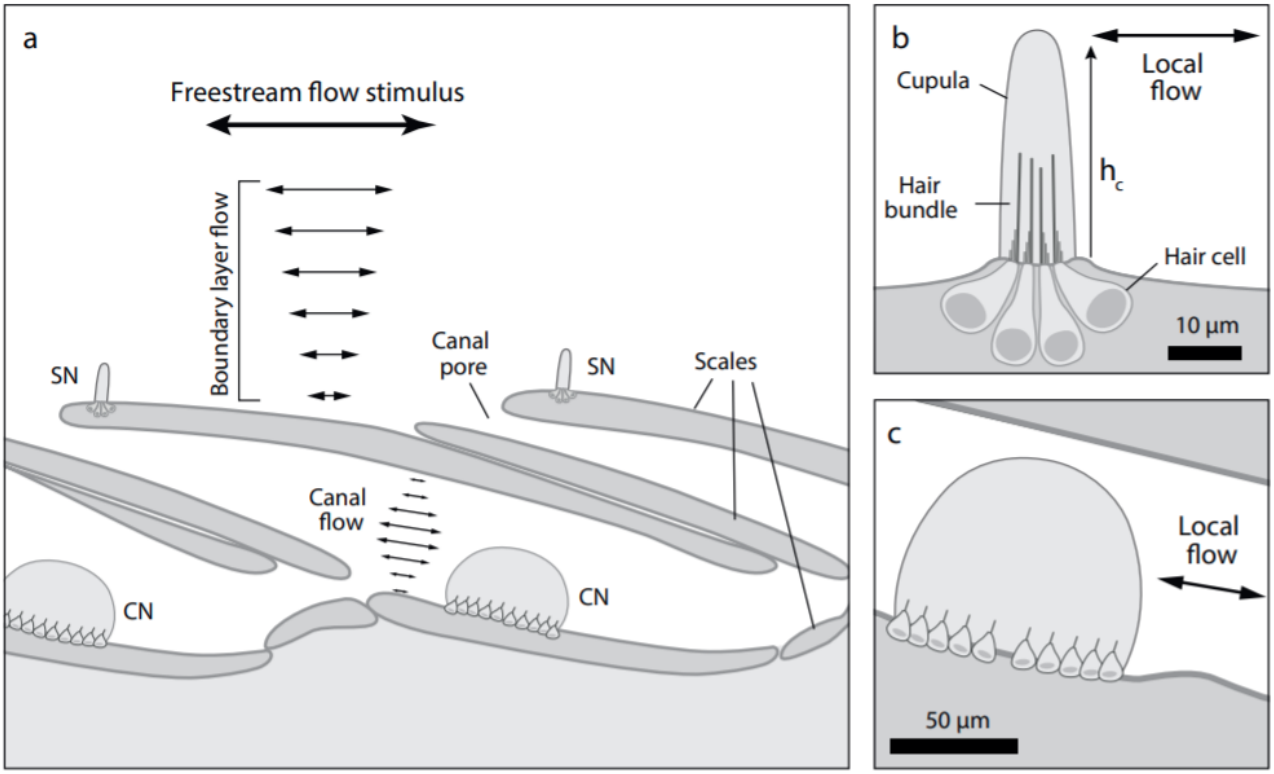

Figure 2.1: a) shows a representation of the different types of neuromasts and how they are situated on the skin of the animal. b) and c) show a detailed views of the superficial neuromast and canal neuromast respectively. Image adapted from [6].

be masked [9]. Again this allows artificial analogues to focus on both the DC as well as the AC components of the signal, without straying from the biological basis.

2.1.1 Historical research

The lateral line has a long history in biological research. It is thought the earliest account of the organ is by Stenonis in 1664 [10]. At this point the organ was thought to produce mucus. It was not until the 19th century that this view changed, and the view of this system as a sensory organ was established [2]. After this point more biological and physiological research has focused on the lateral line. Since this point, research has uncovered that the lateral line is involved with predator and prey detection, schooling, object detection, entrainment, and rheotaxis [5, 11, 12]. This research aided in our understanding of hair cells and efferent systems, as well as the filtering of noise [13].

Research into the functioning of the lateral line distinguishes two modes of operation: *active* and *passive* sensing. The *active* mode detects object in the environment of the animal while moving. Movement through the fluid produces a signature flow. Any object in the environment that perturbs this flow can be detected by the lateral line through changes in the flow. The *passive* operational mode detects movement of object through the motion of the fluid. For this mode the animal does not have to move, but detects the flow signatures produced by other objects.

2.1.2 The Artificial Lateral Line

Recently research into artificial lateral lines (ALL's) has been expanding. As stated earlier in this thesis, we distinguish three part: mechanical design, signal processing, and machine learning. We will treat each of these components separately, by presenting a condensed history in terms of papers, and corresponding techniques employed in this thesis.

Mechanical design

When looking at the mechanical components for the artificial lateral line, we see some variety in the design. The earliest related work is into underwater flow detection. Here techniques such as thermal anemometry and Doppler frequency shift are used as well as indirect inferences from pressure differential [14]. To detect Doppler frequency shifts necessitate an active acoustic launcher, while measuring pressure differentials requires bulky systems that are not fit to form distributed arrays [14]. Thermal anemometry has to content with boundary layer effects, and has therefore lost favour in recent years [15].

Recently the sensors have been based on the measured deflection of a cantilever or lamella [15]. A lot of research used microelectromechanical systems (MEMs) [16]. This allowed for a reduction in cost, power consumption, and size. The size reduction in particular produced some favourable characteristics such as higher reliability and faster responses [17]. Because of their electrical components they can be sensitive to unwanted noise. To combat this all-optical neuromasts where developed, which have to contend with less interference [15].

In this thesis we will be utilising these sensors, developed in cooperation with the LAkHsMI Consortium [18]. The design of this type of sensor is based on a cantilever sensor. Any sensor needs to have an interaction with the fluid flow. The sensors employ a spherical body for this. This spherical body is connected to a fibre support. This elastic support can then be modelled as a cantilever where forces enact primarily on the end. Through the use of Bernoulli's beam equation, the strain on the different points of the fibrous support can be modelled in terms of the deflection of the cantilever [15].

A characteristic of this system is that the strain in the support increases on the opposite side of the direction of deflection, but the strain also decreases on the side of deflection. This mimics the inhibiting effects of the paired hair cells in biological neuromasts.

The sensors have made the translation from deflection to strain in the fibrous support. Through the use of Fibre Bragg gratings (FBGs) the strain of the fibre can be measured. These gratings reflect a certain range of wavelength, while allowing other wavelengths to pass through. The strain in the fibre produces a effect on the wavelengths that are reflected. The shift in these wavelength can then be used to determine the strain on the FBGs. This relation can be seen in equation 2.1.1 [19].

$$\frac{\Delta\lambda_B}{\lambda_B} = (1 - p_e)\varepsilon \tag{2.1.1}$$

In this equation we see how the Bragg wavelengths (λ_B) is related to the stress (ε). p_e is the photoelastic coefficient. This equation does not address the effect of temperature on the shift in Bragg wavelengths. In the sensors this compensated for by utilising a differential strain configuration in which the temperature is a common factor. [15].

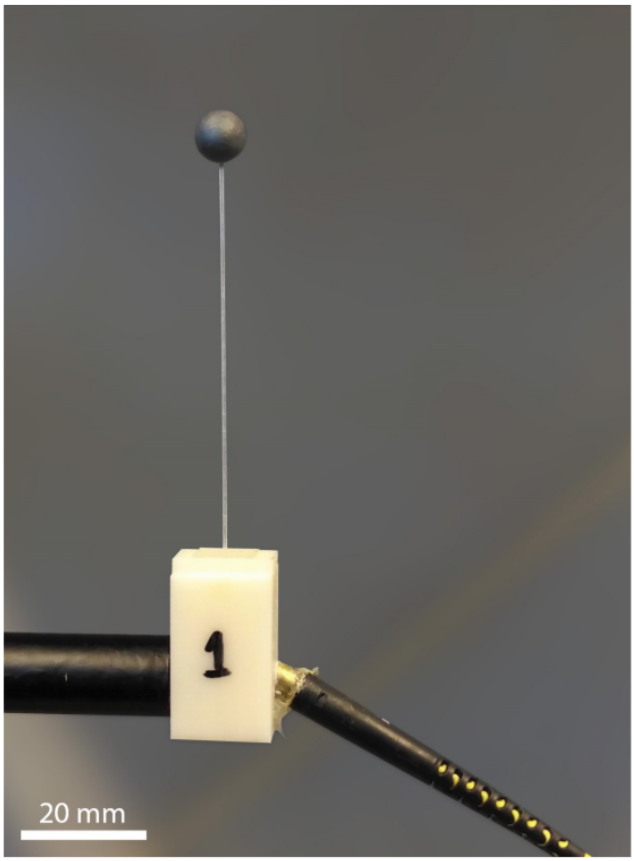

Figure 2.2: The LAkHsMI sensors. Image adapted from [3]

This necessitates the use of multiple FBGs in the fibre support. Each fibre support contains four FBGs. This allows an accurate prediction about the deflection of the sensor to be made through measuring the shift in reflected wavelengths. A sideview of two neuromasts under different deflection conditions is given in figure 2.3. The configuration using four gratings also allows us to determine the deflection along two orthogonal axis.

To conclude this section we recap the different steps that transform the flow into an electrical signal. The first step is deflection of the sensor through the force enacting on the spherical body. This deflection causes strain in the fibrous support of the spherical body. This strain shifts the wavelengths deflected in the four FBGs inside the fibrous support. This shift of the wavelength can then be used to determine the deflection of the spherical body, and therefore the fluid flow.

Signal processing and Machine learning

We will now discuss the historical research of the signal processing and machine learning methods utilised in relation with ALLs. Because these signal processing methods are often used alongside the machine learning methods, we will look at both these aspects congruently. In the next section we will focus on the theoretical background of the techniques found in the method section.

Early work in the field can be found in the work by Yang, Chen, Engel et al. [20]. This work focused

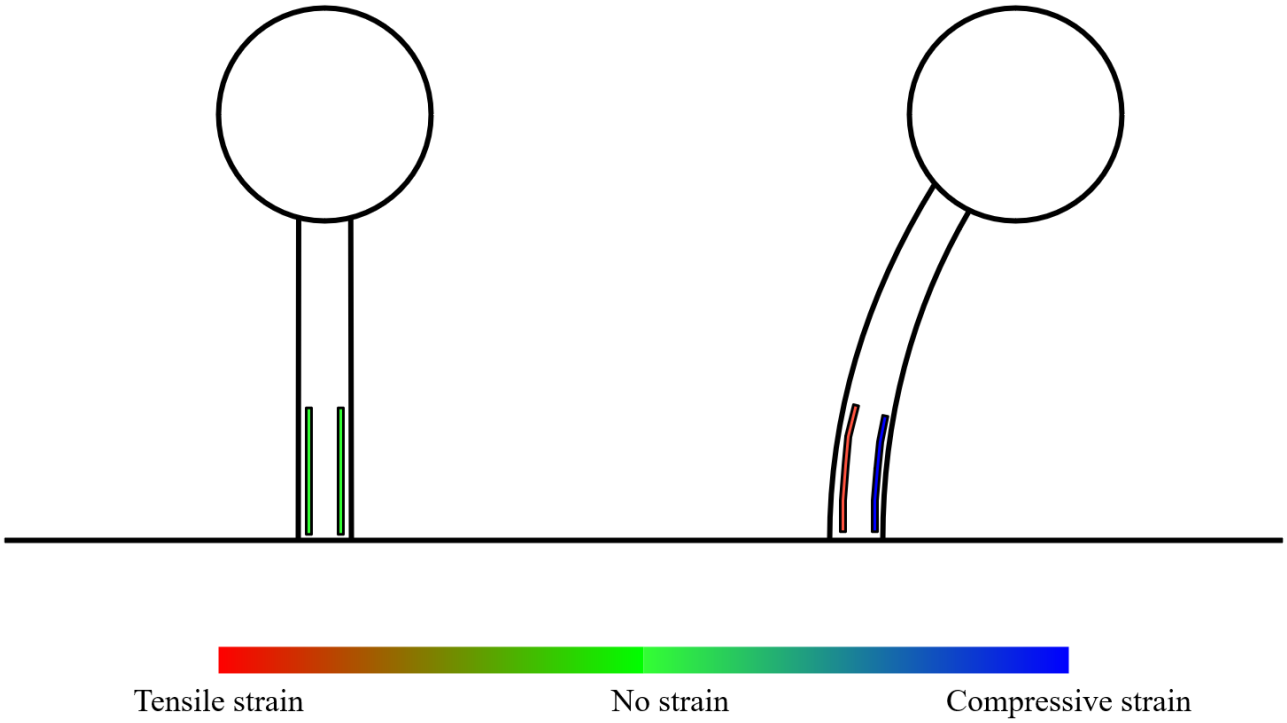

Figure 2.3: A representation of the artificial neuromasts. The neuromast on the left shows the neuromast in centre position, while the right shows a deflected neuromast, and the corresponding strain effects in the fibre Bragg gratings.

on the dipole source localisation and hydrodynamic wake detection. That work used two approaches based on signal processing. Such approaches utilised template or model matching through maximum likelihood estimations [21]. Other early work showed that a wavelet transform can be used for reconstruction of dipole sources [22].
Later work by Yang, Nguyen, Chen et al. expanded on that research by introducing a beamforming algorithm to the ALL [23]. They determined the location of a dipole source as well as the tail flicking of a crayfish. Later research incorporated more machine learning methods to detect moving objects [24]. It covered the performance of different machine learning approaches such as a multilayer perceptron (MLP), an echo state network (ESN), and an extreme learning machine (ELM).

Recently other methods have been employed for source localisation. Such methods include long short-term memory networks (LSTM) [25] and a quadrature method [26]. The LSTM method uses a neural network with recurrent links, incorporating a form of memory into the system. The quadrature method employs the two dimensional output of the ALL to produce an orientation independent curve.

Early work on classification tasks in the domain of the ALL was preformed by by Bouffanais, Weymouth and Yue [27]. That research tried to determine location, size and shape of objects. Other early work was performed by Fernandez, Maertens, Yaul et al. in 2011 [28]. The research harnessed principal component analysis to determine the size of cylinders.

Research in object classification using simple neural networks was performed by Liu, Wang, Wang et al. [29]. That research produced some high accuracy scores, but was only focused on a binary classification task. More recent work on shape classification was conducted by Wolf, Pirih, Kruusmaa

and Van Netten. That work showed classification of five different shapes could be achieved with an accuracy of 95,8% [30].

Most of said research revolves around either localisation or classification of dipole sources or simple moving objects. Research into complex sources or shapes seems under explored at this moment. The classification and characteristics of swimmers specifically is more often researched through the lens of wearable technology [31].

This thesis builds on recent research done by Lucia Baldassini which focused on the localisation and classification of swimmers [4]. For the classification an extreme learning machine as well as an random forest approach were evaluated. The research provided a good baseline with decent results. This thesis will iterate on this by evaluating on a bigger data set, and utilising methods more adapted to time series classification.

2.2 Features and Machine learning methods

In this section of the paper, the feature representations that were used in our research will be covered. The start of this section will cover the general concept of time series. We will then analyse the two different feature representations that were applied to the data, the dynamic wavelet transform (DWT) and a spectrogram. After this we will cover the machine learning approaches that were used in this thesis. This will cover the main transformation of time series classification as well as the specifics methods employed in this paper.

2.2.1 Time series

As stated in the introduction the type of task we are trying to solve is a multivariate time series classification task. All of the parts of the definition of our task bring their own advantages and problems. We will go through each of the parts of the definition starting with the idea of time series.

A time series is collection of data points with as specific relation. We can represent all our data points in a the form of $\mathbf{x} = [x_1, x_2, x_3, ...x_n] \in \mathbb{R}^N$ where $\mathbf{x}$ is our time series of N data points and x_i is the ith data point. A time series ensures that subsequent time series follow each other in time, such that for two data point x_i and x_j if $i < j$ then x_i represents an earlier point in time than x_j. Usually time series are evenly spaced, such that the time between points x_{i-1} and x_i is the same as between points x_i and x_{i+1}. This does however not always have to be the case. In this thesis we will only work with evenly spaced data points. The exact time point that each data point is captured at depends on the start of the observation and the frequency of capturing data points.

We can find time series in a lot of applications, from measured temperatures to stock prices. The types of time series that enjoy the most research from the view of machine learning are of these types. That is to say the series of data points only relates to a single variable. Only the variables of a single thermometer or the price of a single stock are considered. This type of time series is called Univariate, as opposed to multivariate. Because an Artificial lateral line comprises of multiple artificial neuromasts, it produces multiple variables for a single point in time. This type of multivariate data can be represented as a time series of the collection of data points captured at a certain time. A multivariate time series $\mathbf{X}$ with M dimensions is then of the following form: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ...\mathbf{x}_n] \in \mathbb{R}^{M \times N}$ where

each $\mathbf{x}_i$ is collection of M data points.

Lastly we cover what the importance of being a classification task is to the way the time series is processed. Earlier we gave the examples of stock prices and temperatures. These are prime examples of time series that are not studied in the context of classification but instead in the context of prediction tasks. It is usually not the aim to determine the type of stock, but what future values can be expected. To accomplish this many methods are available that range from statistical methods, to classical machine learning techniques and deep learning applications [32, 33].

2.2.2 Data transformations

In this thesis we will focus on two types of data transformation. These transformations are meant to help the classification methods to find discerning features in the data. A data transformation in the context of classification problems should represent the data in such a way that the differences between the classes are easy to detect.

Spectrogram

The first transformation we will discuss is the transformation into a spectrogram. As a data representation a spectrogram contains both information about the time and the frequency domain. While a Fourier transform converts the time series entirely to the frequency domain, a spectrogram only partially converts to the frequency domain. While a time series only represents the magnitude of a signal at a specific time point, the Fourier transform represents the magnitude of the frequencies at all time points. Therefore the spectrogram represents an approximate frequency magnitude at an approximate point in time. The specificity of the time and frequency representation are inversely related, so any intermediate representation is forced to lose specificity in both the time and frequency domain as compared with the respective pure representations.

The transformation to a spectrogram is accomplished by applying a Fourier transform to short windows in time. This is similar to the short-time Fourier transform(STFT). The relation between these transformations is actually quite direct, as the spectrogram is the magnitude squared of the STFT, as seen in equation 2.2.1.

$$Spectrogram\{\mathbf{x}(n)\} = |STFT\{\mathbf{x}(n)\}|^2 \tag{2.2.1}$$

In this equation, we find our time series as $\mathbf{x}(n)$. To find out how the spectrogram is computed we can look at the definition of the STFT. First we look at the definition for the STFT derived from the Fourier transform. This definition can be found in equation 2.2.2 [34].

$$X[n,\lambda) = \sum_{m=-\infty}^{\infty} \mathbf{x}[n+m]w[n]e^{-j\lambda m} \tag{2.2.2}$$

In this equation we find the STFT as $X[n,\lambda)$. This definition shows that the STFT can be interpreted as Fourier transform of the complex function $\mathbf{x}[n+m]w[n]$. $\mathbf{x}[n+m]$ is the shifted time series, and $w[n]$ is a window function. The shape and size of this window function determines what is incorporated into each time interval of the STFT.

We can also see that the definition works with a continuous variable λ for frequencies. To work with the spectrogram a discrete variable is preferred. To this end we can look at a definition of the STFT

based on the discrete Fourier transform. Making sure we sample for both the time and frequency. This definition can be found in equation 2.2.3.

$$STFT\{\mathbf{x}(t)\} = X[n,k] = \sum_{m=0}^{L-1} \mathbf{x}[n+m]w[m]e^{-2\pi jkm/F} \tag{2.2.3}$$

In this definition k represents the discrete variable for the frequencies. We also see L to be the window length and F to be the number of equally spaced frequency slices. This definition together with equation 2.2.1 allows us to turn our time series $\mathbf{x}(n)$ into a spectrogram $X[n,k]$. For our application, we will have to apply this transformation to all available dimensions separately.

We chose the spectrogram representation to include in this thesis, because it incorporates frequency information with some time specificity. A Fourier transform does not allow you to determine the sequencing of different frequencies. A spectrogram could allow any classification algorithm to garner good results when the distinguishing feature is a specific sequence of frequencies.

In the context of the this thesis, a spectrogram of the data can be interpreted most easily as a set of subsequent intervals. For each interval the spectrogram shows how much of the power of the signal comes from each frequency band. That is to say how much of the motion of the tips of the sensors is due to the high or low frequency motion. This shows us something about the different types of waves produced by the swimmers. A slow movement of the flipper might produce a low frequency wave, while movement of the body through water could produce a wake effects expressed as higher frequency waves. The essence of this transform is to indicate approximately when high and low frequency components of the signal are created by the swimmer.

Wavelet transform

The other data transformation this thesis incorporates is the wavelet transform. This transformation performs a similar role as the spectrogram. The wavelet transform is a representation of how much the signal corresponds with an analysing wavelet. This wavelet is varied in two dimensions to obtain the transform. The first is shifting of the wavelet along the signal. From this it is possible to determine at what point in the original signal, the wavelet is present in which measures. The second is the time dilation or stretching and shrinking of the analysing wavelet. This is a scaling of the frequency of the wavelet. This allows the multiple frequency versions of the wavelet to be captured. It also allows us to utilise a variety of different waves as an analysing wave. Again we will first look at the definition of the continuous wavelet transform given in equation 2.2.4 [35].

$$WT(a,b) = \frac{1}{\sqrt{a}} \int \psi\left(\frac{n-b}{a}\right) x(n)dn \tag{2.2.4}$$

This definition immediately shows that the analysing wavelet ψ is changed by a, the time dilation, and b, the translation along the signal. As with the STFT this is a continuous function so for our purposes we look to the discrete wavelet transform (DWT). The DWT is generally computed through the use of a filter bank with two complementary filters. To obtain a more direct example one can look at the Mallet's multiresolutional algorithm [36]. One filter g computes the signal detail $\mathbf{D}_j$, while another filter h computes the approximation of the signal $\mathbf{A}_j$. Both these filters are down sampled by a factor of two. Then the approximate signal is given as input to the filters g and h to produce another set of outputs. Three levels of this process can be seen in figure 2.4. This process is repeated a number of

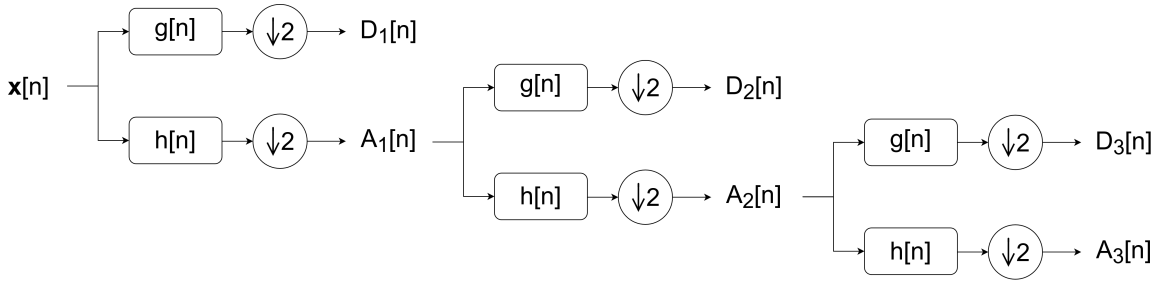

Figure 2.4: Discrete wavelet transform filterbank. A signal enters on the left, and is transformed by two filters g and h. This produces signal detail $\mathbf{D}_1$ and signal approximation $\mathbf{A}_1$. $\mathbf{A}_1$ is then used as input for the filters for the second level of the transform.

times J, until we have a set of coefficients $\{\mathbf{D}_1, \mathbf{D}_2, ..., \mathbf{D}_J\}$ and $\mathbf{A}_J$. This set is can then be used as the transformed signal.

Most application do not separate out the convolution with the filter and the down sampling, but shift the wavelet along the signal with a stride of two. Only a select group of filter pairs can be used for this transformation.

The first level of the transformation produces an array of values related to the high frequency wavelet. Every subsequent level produces coefficient corresponding to a wavelet with a time dilation factor of two, because the approximate signal is down sampled. This also means that for each subsequent level the frequency specificity increases, while the time specificity decreases.

The reason to include this transformation in this thesis is twofold. Firstly, we have seen earlier work utilising the continuous wavelet transform to extract information from ALL data [22]. This shows that a wavelet transform can encode useful details in the context of the ALL. secondly, it fulfils a similar role to the STFT, as both give time as well as frequency information about the signal. The wavelet transform however adapts time and frequency specificity for different frequency bands. This allows methods adapted to time series to utilise many time points for the lower levels corresponding to the high frequency components of the signal. It does this without sacrificing the ability to show the presence of low frequency waves in the higher levels of the transform.

For the context of ALL sensor data this means we see how well the motion of the sensor head matches the analysing wavelet. We expect motion of the sensor to be wavelike, as it is driven by waves propagating through water. Each level of the transform should show when waves of a certain frequency passed by. A person wearing flipper would most likely produce slower waves, which would show up as higher values in a higher level of the transform. A person swimming in the same way without flippers moves faster, producing faster waves and resulting in higher values in the lower levels of the transform. This property is helpful in the context of this thesis, as our data includes recording from both barefoot swimming as well as swimming with flippers. A more detailed description of the data will be given in section 3.1.2. In essence the wavelet transform aims to show approximately when high and low frequency waves were created by the swimmer.

2.2.3 Machine learning methods

In this section we will be examining the theoretical framework surrounding the classification methods we will be testing in this thesis. Subsequently, we will have a broad look at different approaches that classifiers employ. After this we will discuss the theoretical background of the specific classification methods We will look at three techniques that were implemented: Firstly, a simple approach, that has long been used as a benchmark, utilising dynamic time warp. Secondly, a random forest based approach called time series forests. And lastly a state of the art deep learning approach called InceptionTime. Each approach is situated in a different corner of the field of time series classification, and will therefore give us a general idea how all these areas compare.

As stated earlier in this thesis, any classification task tries to predict the correct class for an input. In our case the output class y is a single value, corresponding to the identity of the swimmer. This means our classifiers have to make the following transform: $\mathbf{x}[n] \in \mathbb{R}^{M \times N} \rightarrow y$. One does not necessarily need to look at this classification problem strictly in the context of time series. Any classifier that accepts unstructured data could work, although such methods cannot take advantage of the inherent characteristics of a time series.

Dynamic time warp

The first classification method we will look at is one utilising dynamic time warp as a distance measure. Dynamic time warping is a technique designed to accommodate the comparison of time series with similar phase characteristics but differing time scales. When a series is directly compared to a down sampled version of this signal, a direct distance function might calculate a relatively high distance, while similar features are present in both signals. This can be remedied through a global dilation of the signal. But similar problems arise when subsets of a time series do not match in the time scale. This is not solvable by a simple global dilation of the signal.

To circumvent this, dynamic time warping of the series allows a one to many comparison between points in the time series. By comparing one point of time series **a** to many points of time series **b**, this section of time series **b** is dilated. This means we can better compare series, by locally dilating some sections. An example of the distances calculated between two time series can be found in figure 2.5. In this figure you can see how the some points can be linked to multiple points on the other time series.

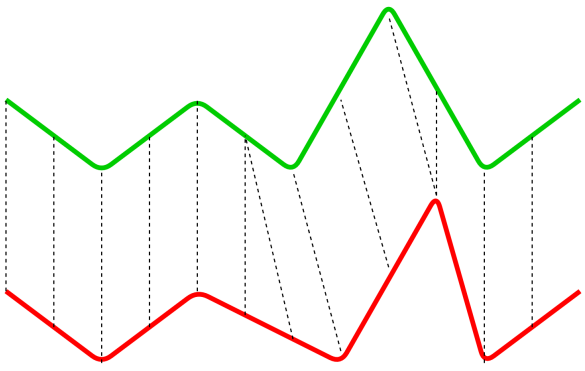

Figure 2.5: A simple example of the way in which dynamic time warping determines the distance between two time series (red and green). The dotted lines represents the distance which is calculated.

The method of dynamic warping of datasets for comparison is originates with spoken word recognition [37]. In this context it was used for comparisons between feature vectors representing sentences. The application of DTW to time series was highly influenced by a early paper by Berndt and Clifford [38].

This distance measure can then be utilised by a simple classification algorithm to determine similarities. A common combination is using DTW with a nearest neighbour algorithm for time series classification, known as nearest neighbour DTW (NNDTW) [39–41]. K nearest neighbour classification uses a distance metric to find the K closest data points in a training set, and uses their classes to make a prediction of the to be evaluated time series. A simple representation of the algorithm can be found in figure 2.6

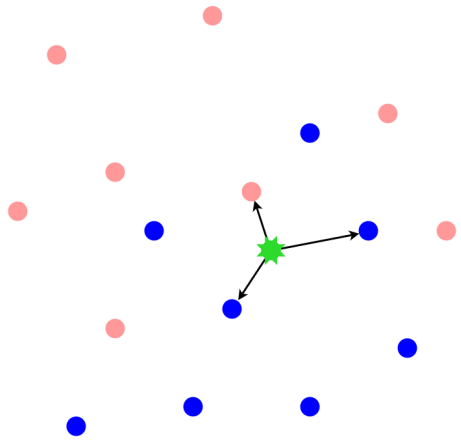

Figure 2.6: A simple illustration of how KNN determines the class of a point. The lighter red and darker blue points represent training examples of two classes. The newly introduced green star will determine its closest neighbours. For the case of three nearest neighbour classification, the darker blue class is predicted as 2 out of its 3 closest neighbours are blue.

This method was chosen because it provides a good baseline to compare other methods against. As already stated it is widely used in this context. It also allows us to gain some insight how informative different aspects of the data are. As the dynamic wavelet transform looks mostly at the phase pattern of the data, and not at the speed at different points along the time series. This could be very informative, if the defining characteristics of the swimmers are in their phase pattern, and they vary in speed across the runs. It could however also be the opposite, where the phase patterns of all the swimmers match to a high degree, because they all perform a similar activity. By comparing the results of this approach to other approaches, we will hopefully be able to gain insight in which characteristics are informative.

Time series forests

The second method we will be using is called time series forests (TSF). This method was first introduced by Deng et al. in 2013 [42]. It was explicitly designed to focus on the temporal characteristics of a signal that are neglected by NNDTW. TSF iterated on the ideas of interval based features for time series classification. Earlier research by Guerts and by Rodriguez et all. both proposed classification methods based on intervals, but these methods had some definite drawbacks [43, 44].

The idea behind interval based classification, is to separate the time series into intervals and calculate local features for these intervals. These local features can then be used for a decision making algorithm. TSF uses a simple set of local features to compute. It computes the following features:

- *mean*
- *standard deviation*
- *slope*

The *mean* and *standard deviation* are self explanatory, while the *slope* is the slope of the least squares regression line over the points in the interval. This is an easy to compute set of features which produces a relatively fast algorithm. Of course a more complex set of features could be used for this such as the recently introduced *catch22* feature set. [45]. Recent research iterated further on the initial design of TSF, and utilised this set of features [46].

These features are then used to produce a time series forests. This is made up of multiple time series decision trees. Each tree in turn is comprised of nodes. In each node a split is made of the current set, passing the two subsets to different branches of the tree. This is repeated until a leaf node is reached in which the set is comprised of a single class. Any to be evaluated data point can follow the tree down until they reach a leaf node. The uniform class of the leaf node is used as the prediction.

In each node a split is made based on a decision function, which should produce more homogeneous subsets. Formally the decision function is characterised as follows: $f_n(t_i,t_j) > \tau$. Where $f_n(t_i,t_j)$ is feature n over interval (t_i,t_j). τ is the decision threshold. A representation of such a decision tree is given in figure 2.7. In the decision function we can vary 3 variables to get different decision functions. From all these variations we want to select the function which is most helpful for separating the classes. We will first talk about which variations are possible, after which we will discuss the selection criterion.

Of course the most simple variation is the feature f_n. In the case of the original paper, this is just the three features [42]. The time interval (t_i,t_j) can have a large possibility space, as both the length of the interval as well as its starting position can vary. To combat this, an approach similar to random forests is taken. For a time series of length N, $\sqrt{N}$ intervals sizes are randomly sampled, as well as $\sqrt{N}$ distinct starting points. This produces a set of N intervals to test. Lastly the threshold τ is varied. To produce a faster algorithm, the entire range of the feature output is determined, and 20 values for τ are evenly spaced in this range. The value of 20 is taken from the original paper, but can of course be adapted. An increased value would mean a slower algorithm, as there are more possibilities to check. So our decision function has $3 \times M \times 20$ possible variations.

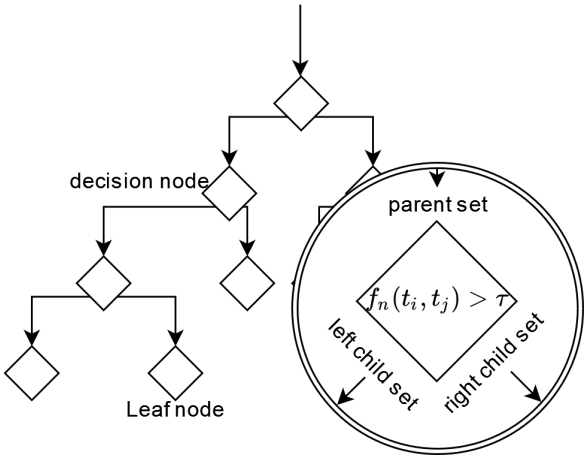

Figure 2.7: A abstract representation of a decision tree, with one node enlarged to show the splitting criteria. all instances in the parent set that do not meat the decision criterion are send to the left child set, while the remaining are passed on to the right child set.

Now that it has been explained which possible decision function there are, we will move on to the criterion by which they are evaluated. The decision functions should separate different classes based on the local features. The evaluation metric of the decision function generally uses the entropy gain. The entropy gain is based on the entropy of the parent set of the node, and the two child nodes. The definition of entropy of a set is given in equation 2.2.5.

$$Entropy = -\sum_{c}^{C} p_c \; log \; p_c \tag{2.2.5}$$

In this equation we see that the entropy is defined as a negative sum over all classes C present in set. The p_c term defines the proportion of instances of the class c present in the set. From this we can define the entropy gain ΔE, as done in equation 2.2.6.

$$\Delta E = E_{parent} - \sum_{i}^{children} w_i \, E_i \tag{2.2.6}$$

In this equation we see that the entropy gain is the difference between the entropy of the parent set E_{parent} and a weighted sum of the entropy of the children sets E_i. The weight factor W_i is the proportion of instances of the parent set present in the child set. In our descriptions of decision trees, we mostly refer to binary trees, with just two child nodes, but this does not always have to be the case.

In the original paper on TSF, the authors introduced a new measure called entrance, a portmanteau of entropy and distance. This adds another factor called margin to the entropy. The margin is the distance between a candidate split and the nearest feature value. The margin factor is weighted extremely light. This in turn means it only influences the decision when the entropy of two decision functions is equal.

With this entrance criterion all the decision functions can be evaluated to provide the most informative node. This is then recursively applied to the child sets to build the entire tree. The time series forest is then constructed by assembling multiple trees, to form an ensemble which decides the prediction

through majority voting.

A strength of this algorithm is the temporal importance curves that can be calculated from the generated time series forests. The importance curve can be constructed per feature. In the original paper, equation 2.2.7 is given [42].

$$IMP_k(t) = \sum_{t_1 \leq t \leq t_2, v \in SN} \Delta E(f_k(t_1, t_2), v) \tag{2.2.7}$$

In this equation, SN is the set of all decision nodes. $\Delta E(f_k(t_1,t_2),v)$ is the entropy for decision function $f_k(t_1,t_2)$ at node v. Essentially this is an expression of how much any point of time is used to make informative splits and how informative these splits are.

This importance curve provides insight into the temporal characteristics of the signal. Specifically how informative each time point in the signal is for each feature. This would be helpful to determine defining characteristics in the signal of each swimmer.

InceptionTime

The last classification method we will look into is InceptionTime. It was introduced by Ismail Fawaz et al. in 2019 [47]. InceptionTime was introduced to replicate the impact of AlexNet [48] for the domain of time series. That is to say, a deep learning approach that brought about a mayor shift in the types classifications, and is widely used as a benchmark. It is however difficult to determine the impact of the system as of now, as few years have passed. The system is explicitly situated in competition to HIVE-COTE. HIVE-COTE is an hierarchical voting ensemble method, proposed in 2016 by Lines et al. which has long served as the state of the art for time series classification [49]. Ismail Fawaz et al. showed that InceptionTime was able to produce similar results with a drastically lower time complexity.

Before the structure of the InceptionTime is explained, the next sections will focus on a brief explanation of the structure of artificial neural networks, and their functioning. These concepts provide an introduction to functioning of deep learning methods, as well as the problems that can arise.

Artificial neural networks are most easily explained by looking at the functioning of a multilayer perceptron. A multilayer perceptron is a network of connected perceptrons. The perceptron is a simple unit of computation first proposed by RosenBlatt [50]. A schematic overview of a perceptron can be found in figure 2.8.
A perceptron is used to produce a binary output from a set of inputs. It can be used to solve binary classification tasks that are linearly separable. The perceptron takes the dot product of an input vector and a weight vector. A bias is added to the input vector. If the results of the dot product is above zero the output is one, otherwise the output will be zero. The weight vector can be adapted to achieve any linear separation of possible inputs. This basic idea of classifiers build from nodes with associated weights is leveraged in artificial neural networks.

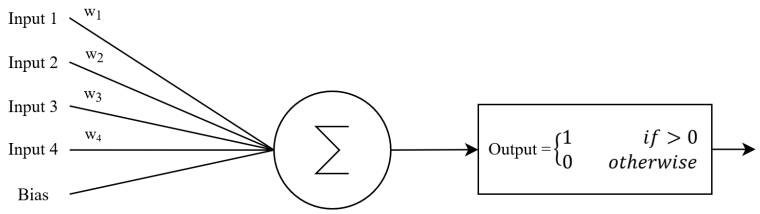

Figure 2.8: A simple representation of a perceptron. The inputs are multiplied by the weights, and summed with the bias. If this sum is above zero, the output will be one, otherwise the output will be zero.

The multilayer perceptron combines multiple perceptrons to achieve more complex classifications. These can be ordered in layers of multiple perceptrons to achieve non-linear separation of the classes. The output of a perceptron in the previous layers can be used as input for one or more perceptrons in the next layers. The idea of a network of nodes through which activation flows is still used as the basis for modern artificial neural networks. Deep learning specifically focuses on the non linear aspects that large amounts of layers can provide to a classifier. Modern neural networks often include an activation function in the nodes, allowing each node to output a range of values instead of a single binary output.

To turn any artificial neural network into a functional classifier, the right values for the weights need to be selected. To achieve this most modern approaches rely on a version of gradient descent. This

technique is based on the idea that the difference between the current output of a network and the target output of a system can be described as the loss. This loss increases when the network performs worse. The loss of the network can be determined for each possible configuration of the weights, so the loss can be described as a function of the weights of the system. The aim is to find the minimum of this loss function.

Gradient descent can take the derivative of the loss function with respect to the weights. This derivative shows what the loss will do when the weights are changed. Since our loss should be low to obtain a good performance of the network, gradient descent adapts the weights such that the loss will decrease. A simplified version can be seen in figure 2.9. For more complex networks with more weights, the loss function becomes a complex manifold. Gradient descent determines the direction in which the gradient of the loss function decreases. How much the weights are adapted is based on the learning rate, a parameter that can be set. [51]

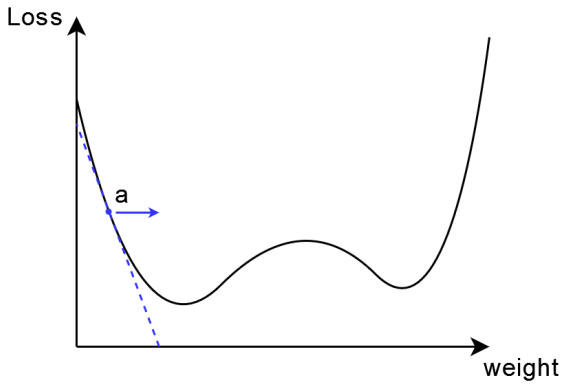

Figure 2.9: Example simple of a loss function with a single weight. For point a, gradient descent can determine the gradient, seen as the blue dotted line. The direction in which the single weight needs to be updated, can then be determine, seen here as the blue arrow.

Modern research has also adapted the configuration of the nodes and their connections. Besides the fully connected networks seen in early work, convolutional neural networks have come to play an important role. These networks utilise kernels that are shifted along the input to create a feature map. The advantages of this are that the same kernel can be reused for the entire input, allowing only a small set of parameters to be trained.

Now that these basic concepts have been discussed, the specific structure of the InceptionTime will be explained.

The InceptionTime network is build up of three parts. First is a set of inception modules, secondly a global average pooling layer, and lastly a fully connected layer. The inception modules consists of two separate parallel paths through which activation in the network flows. The first path is a bottleneck layer, followed by a set convolutions with varying filters. This path first reduces the amount of dimensions of the data before expanding it again. The second path is a combination of a max-pooling layer combined with another bottleneck layer. This reduces the time dimension before reducing the other dimensions of the input. These inception modules are then stacked in blocks of three to produce residual blocks. The input of a residual block is added to the input of the next block. This aims to circumvent the vanishing gradient problem.

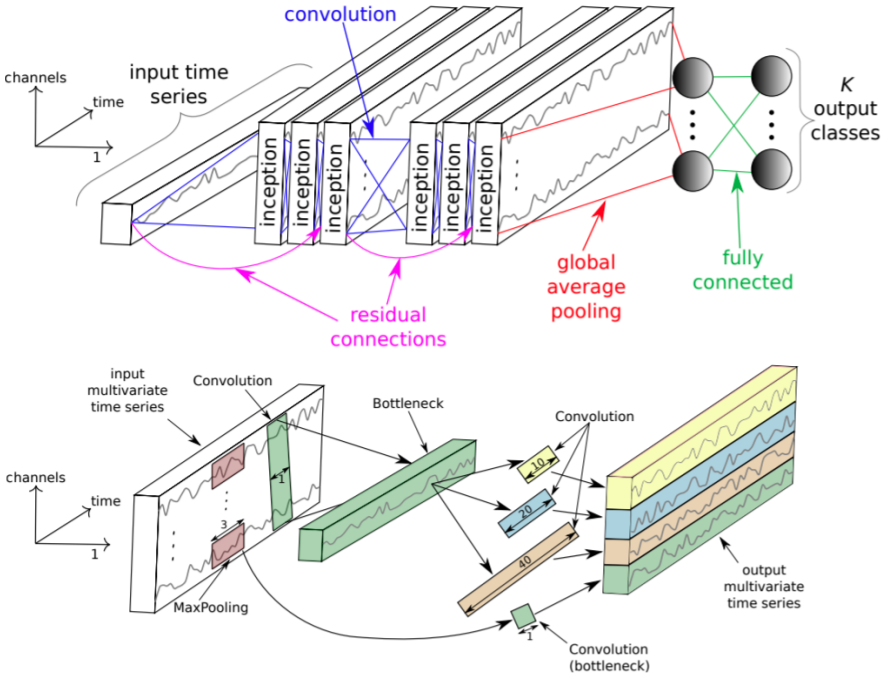

Figure 2.10: The structure of the InceptionTime network. The structure of the entire network is given at the top, while the bottom image shows the structure of an Inception Module. Adapted from [47]

The idea behind this structure is that the inception modules transforms from multidimensional time series to multidimensional time series. With each inception module, the representation can incorporate more complex characteristics. Through the use of the bottleneck layer, the time series reduces in dimensions. The convolutions then indicate the presence of more complex features. A convolution can be understood as a detector of a specific characteristic present in the input. The output of these convolutions can then be used to detect the presence of more complex characteristics based on the simpler characteristics.
In earlier layers, this could be the presence of a specific wavelet, while in later layers, it could represent a complex set of wavelets. Convolutions are widely used in image processing, where a first convolution can detect lines, while later on they detect shapes. In our domain of time series the detected features will be different, the closest example would be the classification of a piece of music. The first convolutions would detect specific notes, while later convolutions combine these in chords or harmonies, which would in turn be used to detect simple melodies. In this way the Inception modules produce an signal that represents the presence of ever more complex characteristics.

This architecture incorporates methods to reduce the effect of overfitting on the data set by incorporating bottleneck layers. Overfitting is a common problem especially in deep learning applications. It occurs when a system does not base its predictions on intrinsic aspects of the phenomenon which produces the data, but on characteristics present in the data set included by chance or improper data collection. Such a characteristic would make it easy for the system to classify the swimmers, without producing a system that generalises well. The bottleneck layers lessen the effect that a single value of the input has on the output, reducing the chances of overfitting because of a value of the input that is highly informative for a specific data set.

Chapter 3

Methods

This chapter of the thesis will consist of three parts. The first part covers the collection of the data, and the processing of the signal into a usable format. The second concerns the data transformations presented in section 2.2.2. Lastly we will detail how the different types of classifiers specified in section 2.2.3 were build and evaluated. This should give a clear image of how our results were produced.

3.1 Data collection and processing

3.1.1 Experimental setup

The data was collected before the start of this thesis. Part of the this data set was used by Baldassini in 2020 [4]. The data was collected in the swimming pool of the Willem Alexander sportcentrum of the Hanze hogeschool. An array of eighth optical sensors described in section 2.1.2, was suspended in the water. The sensors were arranged in a straight line with a spacing of 50 cm. Video of each run was captured with a GoPro camera. To synchronise the captured footage with the data recorded by the sensors, a LED light was present in the pool which flashed to indicate the start of the sensor recording. The swimmers were instructed to swim by the array lengthwise at approximately the same height as the sensors. The swimmers wore LED markers, to indicate their location. An example of the experimental setup can be found in figure 3.1

3.1.2 Data sets

The collected data can be organised into two different data sets. The first is a set where two different swimmers perform two runs each for three different swimming styles. The swimming styles were barefoot, flippers, and monofin. The only deviation was that one swimmer performed three runs of barefoot swimming. This gives us a total of thirteen runs. This dataset varies in both the identity of the swimmer as well as the style used.

The second data set consists of five different swimmers all swimming with flippers. The four runs with flippers included in the first data set are also included in this set. Besides this we have 2 more people performing 4 runs, and a last participant performing 5 runs. This gives us a total of seventeen runs. This data set only varies in the identity of the swimmer.

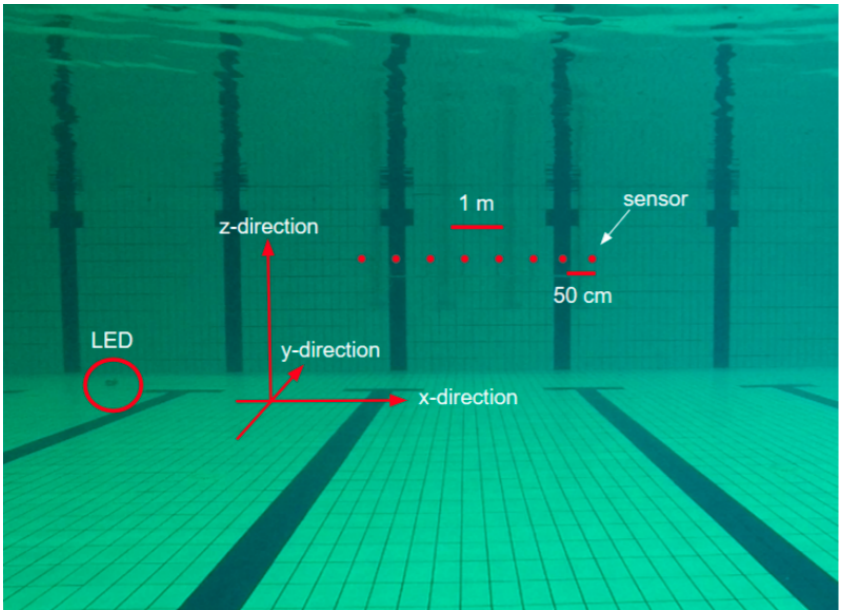

Figure 3.1: An image of the experimental setup used for the data collection. Adapted from [4].

3.1.3 Sensor data

The sensor data is collected in the form of peaks in the Bragg wavelengths. The shift in these wavelengths is used to construct the deflection of the sensors using the method detailed in section 2.1.2. First the shifts in the peaks for every time point was determined. Then the shift of these four gratings were used to determine the motion of the object in the X and Y directions. This was done by utilising the shift in wavelength from a known deflection of the sensor, which was recorded beforehand. These transformations were made by a Matlab script, constructed for earlier research.

The data now represented the motion of the sensor in two dimensions for each of the eight sensors. The start and end of the sensor recording do not precisely coincide with start and end of swimming. To determine a usable range for the sensor data, the location data gathered from the videos was used.

3.1.4 Video data

The videos were used twofold. First to determine the usable range for the sensor data. Secondly, to gather the exact location of the swimmer. To accomplish the first task we needed to determine the range of frames with proper data and what range of the sensor data those frames corresponded too. We considered two types of ranges. A wider range which started as soon as the swimmer had reached the appropriate depth and was swimming in a regular pattern, as judged by eye. The second narrower range was determined as the frames where the swimmer had any part of their body in front of the sensor array. We decided to use the second narrower range. This was done because the starting and ending points of the swimmers varied significantly in the wider range, which could produce an easily classifiable effect in our data. A classifier could classify based on the starting point of a run, not intrinsic properties of the swimmer's style. The narrower range did reduce the amount of data we could utilise. For every run the range in regards to frames was then determined.

To convert this to a sensor range, we used the LED light to determine the start of the sensor recording. We could then determine the range in the sensor data, based on offset from the frame in which the LED

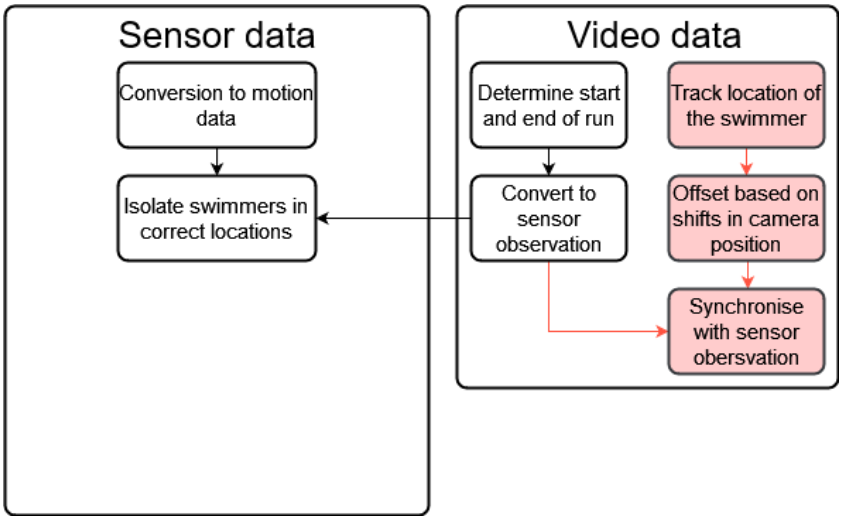

Figure 3.2: Overview of the processing pipeline of the videos and the sensor data.

light showed, and the recording frequencies of both the camera (119.88 fps) and the sensor (5000 Hz).

We also tracked the position of the swimmer through the LED lights on their body. For this both the position of the swimmer in the image, as well as the some stationary background points were tracked. With the background we could correct for any shifts in the camera view between runs. For both these video processing tasks we used Blender [52]. Blender automated some of the tracking, while also allowing manually adjust where it was necessary. Then through the LED flash in the video, we could synchronise this tracking data to the sensor data. An overview of the steps of the pipeline up to this point is given in figure 3.2

During this process it became clear that for a majority of the runs in the second data set, the five different swimmers, the LED lighting up could not reliably be pinpointed. This produced a problem for synchronisation between the tracking and the sensor data. As both the selection of the proper range, as well as the synchronisation between the tracking data and the sensor data, relied on this LED flash. In the end we chose to manually make an estimate of the usable sensor range, as small variation in this would produce a still usable range for the classification task.
The manually selected sensor range was based on two factors. The length of frames that the swimmer was in front of the array in the video, as well as the characteristics of the sensor output. We compared the sensor output of runs with a known range, to the sensor output without a clear LED flash. This helped us determine an approximate range for the sensor data. We did not continue using the location data for these runs, as a small deviation in starting range could have a large effect on any location predictions. In short we could not use this data for location detection purposes, as the exact locations of the swimmers could not be reliably determined. We did however use this data for the classification task, as an exact location is not necessary for this. The steps which could not be performed because of this are coloured red in figure 3.2. This did limit the scope from the original two tasks of localisation and classification to just classification.

3.1.5 Further processing

After determining the correct ranges, a python program was used to extract these ranges. The runs where then sub-sampled to create three different versions of the dataset. A not downsampled version with a sampling frequency of 5000hz, and two downsampled version with sampling frequencies of 500hz and 100 hz. The lower the sampling frequency, less data points along the time dimension. Downsampling removes high frequency components of the signal, including high frequency noise present. This means the lower frequency signals are influenced less by high frequency noise. The results of these sets should show us if any sampling frequency outperforms the others. This could give us insight as to what kind of features of the data are informative, and how well the algorithms handle higher or lower dimensional data. Sub-sampling was done through the use of the *scipy* library [53].

Because the limited amount of runs, we used sampling to get more training and testing examples. From every run a certain number of ranges with a specific length are taken. This produces multiple mini-runs for every run of the swimmer. In this way we could obtain a data set with more data points, all of which have the same size. It reduces the dimensions of the input data, but equalises them for each data point. We take ranges such that each data point represents half a second of sensor recording. It is good to keep in mind that this means different lengths for the different downsampled sets. An overview of the data processing pipeline up to this point is given in figure 3.3.

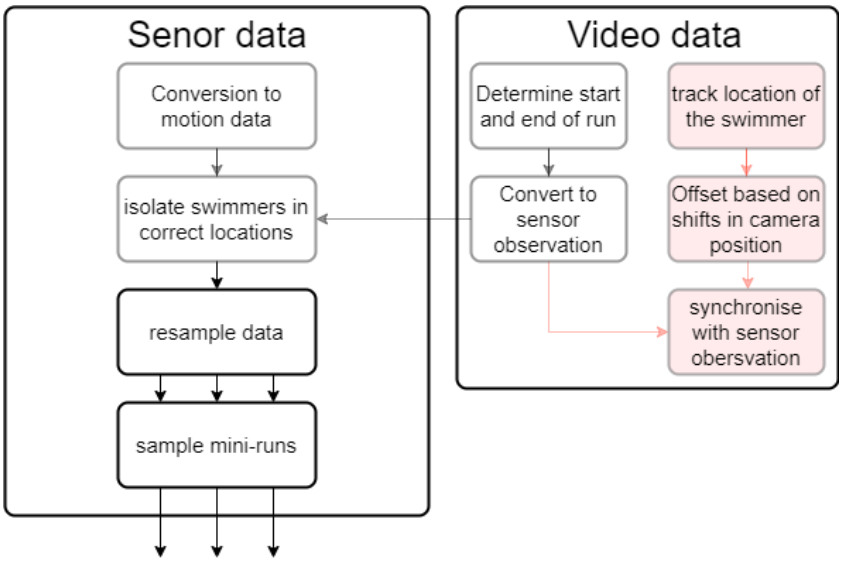

Figure 3.3: The sensor array and video data parts of the processing pipeline.

3.2 Data transforms

This section will cover the two types of data transform we will test. As the main ideas have already been discussed in earlier sections, this will focus on the implementation used, and any applicable parameters.

3.2.1 Spectrogram

The spectrogram was implemented with the use of the *SciPy* library for python [53]. We chose this transform because it incorporates features of the frequency domain without disregarding all information in the time domain. This transform applies a discrete Fourier transform to short intervals of the data. The transform produces a matrix of data, where one axis corresponds to the time intervals, and the other axis to frequency bands. The values of this matrix represent the presence of a signal component of the corresponding time and frequency interval, in the original signal. In our case of multidimensional data, we will have to perform the transform on each dimension individually. This increases the dimensions of the data, as each frequency band produced in the spectrogram will be its own dimension. If the spectrogram of each of the sixteen dimensions produces only two frequency bands, we already have thirty-two dimensions.

In our implementation we used *SciPy*'s `signal.spectrogram` function. We can tune this function through the use of several different parameters. The most important of which is the amount of points per segment, which can be controlled through the `nperseg` parameter. This parameter allows us to control the dimensions of the spectrogram. The resolution of the time and frequency domain are inversely proportional and can be adjusted through the points per segment. Increasing the points per segment decreases the time resolution, as less time intervals are present. This increases the frequency resolution, as from more points in a more frequency bands can be distinguished. For our data sets we aimed for an approximately equal amount of time and frequency intervals. To accomplish this we adjusted the number of points per segment. This was done as to not lose too much resolution in either of the domains. Because we have three different data sets with different frequencies and series lengths, the `nperseg` needs to set differently for each data set. The exact numbers as well as the dimensions of the data are can be found in table 3.1. It is good to note that the dimensions of each data set are the 16 separate signal created by the sensors, times the amount of frequency bands produced by the spectrogram. The length is the amount of time intervals produced by the spectrogram.

Other parameters are the window function, which we kept as the default Tukey window with a shape parameter of 0.25. This allows some overlap between the windows. Another important parameter is the scaling function, which we set to spectrum to obtain the power spectrum instead of a power spectral density. Other parameters were left as default and are omitted from this explanation for brevity's sake.

An example of the spectrogram of a single dimension can be found in figure 3.4. It shows a complex signal, which is difficult to interpret. This is however not discouraging, as a complex signal can provide discernible features. We can also see that no part of the spectrogram has a significantly higher spectral power than other parts, meaning that all time and frequency intervals seem to contribute somewhat.

Frequency of the dataset (Hz)	5000	500	100
`nperseg`	75	25	10
Transformed dimensions (examples, dimensions, length)	(170,608,37)	(170,208,11)	(170,96,5)

Table 3.1: The points per segement and dimensions of the different transformed data sets.

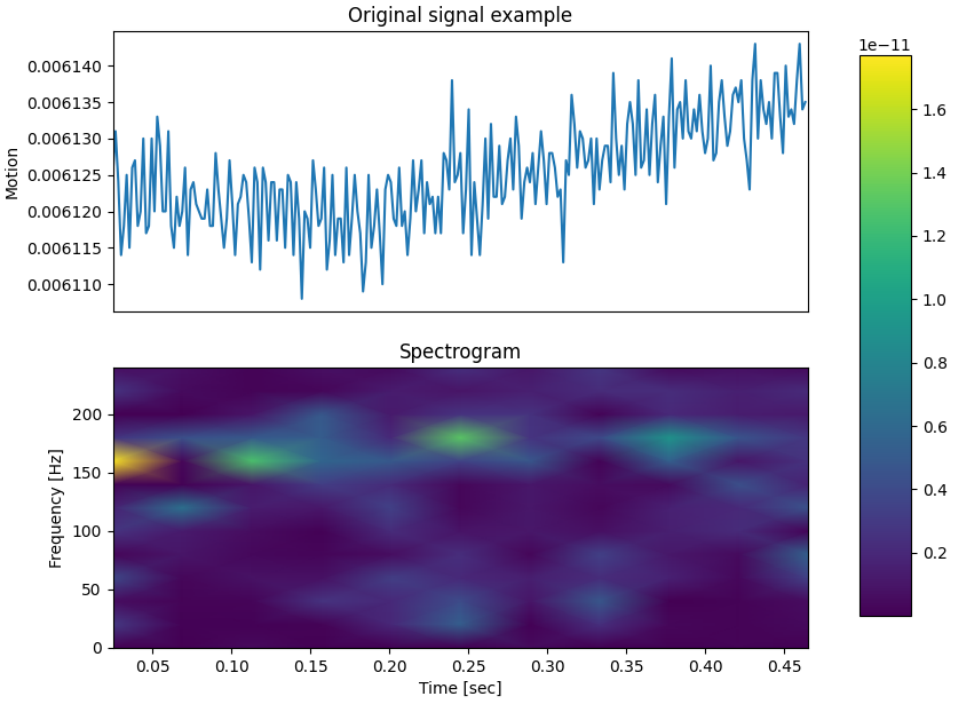

Figure 3.4: One dimension of an example mini-run and corresponding spectrogram.

3.2.2 Discrete wavelet transform

The implementation of the discrete wavelet transform was accomplished with the *sktime* library for python [54]. It implements a Haar wavelet transform. The Haar wavelet is a simple wavelet which can be seen in figure 3.5. It was introduced by Alfred Haar in 1910 [55]. This wavelet function allows for quick computation, as the filters functions associated with the Haar transform are simple. The filter providing details is a difference function, while the approximating filter is an averaging function. The simplicity of these filters allow for fast computation of the transform.

For this transform we had only one parameter to tune, which was the amount of levels. We decided to keep this to the default of four. This results in a transformed signal where half the coefficients are from the first level, a quarter from the second level, and one eighth from third level. The remaining eighth is the approximation of the signal. Our procedure follows the steps seen in figure 2.4, with filters relating to the Haar wavelet.
The effective frequency of the Haar wavelet differs in in each level. For the first level, the analysing wavelet has a frequency of half the signal frequency. For each subsequent level the frequency of the

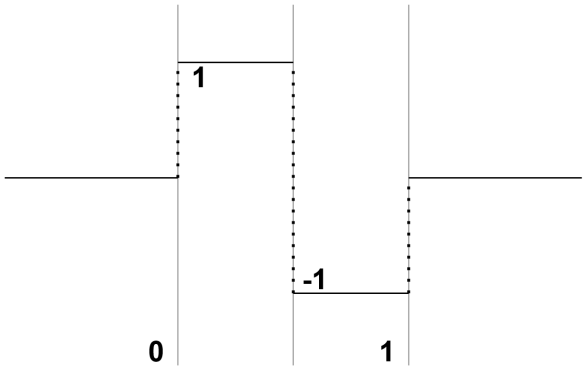

Figure 3.5: simple representation of a Haar wavelet.

Haar wavelet is halved. The Haar wavelet for the 5000 Hz signal has frequencies of 2500 Hz, 1250 Hz and 625 Hz for levels 1, 2 and 3 respectively.

An example of the transform can be seen in figure 3.6. In the figure, the divisions between the different coefficient levels have also been marked. The first level coefficient take up the largest part and are located the farthest right. These coefficients represent higher frequency Haar wavelets. Farthest to the left we can find the approximation of the signal. If we compare this approximation to the original above, we can see it recognise it as a down sampled version. We chose a signal from the 500Hz data set as an example, as this showed the transformation more clearly.

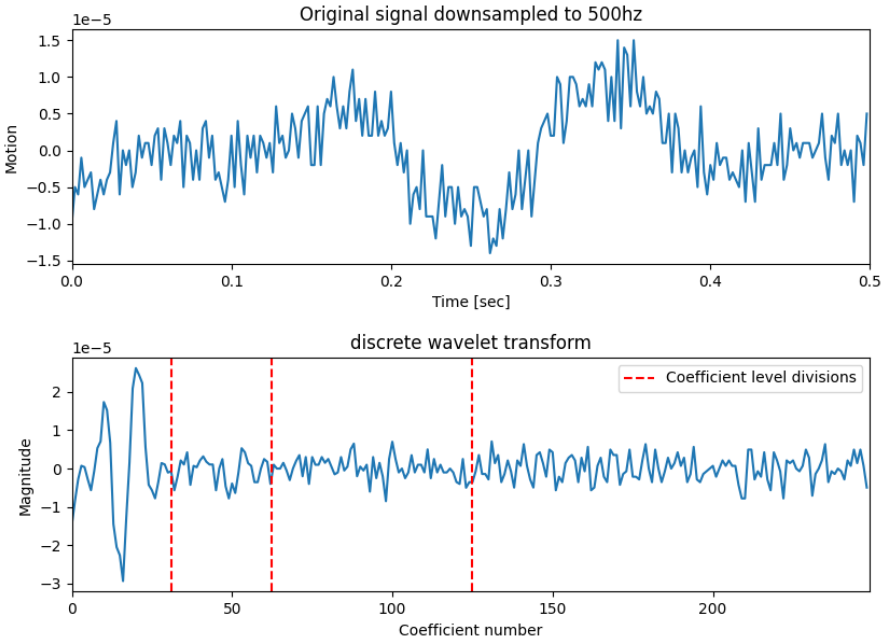

Figure 3.6: One dimension of an down sampled example run and corresponding dynamic wavelet transform.

3.3 Data sets

The combination of the raw data as well as the two transforms and the 3 frequencies means we end up with 9 data sets. The dimensions of these data sets can be found in table 3.2. We can see that each data set still has the same number of examples. It is also good to notice that the wavelet transform produces a data set with similar dimensions to the raw data, while a spectrogram has has an increased number of dimensions and a reduced length. An overview of the pipeline up to this point can be found in figure 3.7.

type	frequency	examples	dimensions	length
raw	5000 Hz	170	16	2500
	500 Hz	170	16	250
	100 Hz	170	16	50
spectrogram	5000 Hz	170	608	37
	500 Hz	170	208	11
	100 Hz	170	96	5
wavelet	5000 Hz	170	16	2499
	500 Hz	170	16	249
	100 Hz	170	16	49

Table 3.2: Data dimensions for the different data sets.

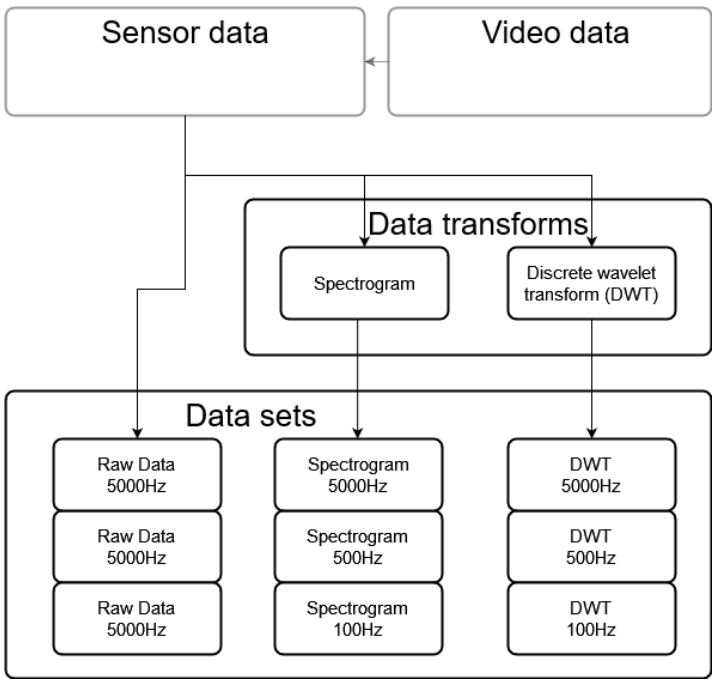

Figure 3.7: An overview of data transform and data set parts of the pipeline.

3.4 Classifiers

In this section we will discuss the implementation of our classifiers, and the specific parameters that were used to generate the results. For all methods we will provide a quick description of the method, before looking at the specifics of implementation. A more broad view of the theory behind these methods is found in section 2.2.3.
As an initial note, all three of the classifiers were implemented using the *Sktime* library [54].

3.4.1 Dynamic Time Warp distance metric

The first type of classifier we will test is a simple nearest neighbours (NN) classifier with dynamic time warping distance metric. This method uses a K nearest neighbours approach, which uses the a distance metric to determine the distance to the points in the data set. The K closest points are then used to make a prediction about the new point. Dynamic time warping produces a distance metric which is invariant to local changes in speed in the time series.

The `KNeighborsTimeSeriesClassifier` method in the *Sktime* library was utilized as classifier. The method allows the use of dynamic time warp as a distance metric. The parameter K can be set, this is the number of neighbours that the algorithm bases the prediction on. In this thesis this parameter was set to 1 or 10. 1-NN makes it such that the predicted value is the same as the closest time series in the data set. The value of 10 was chosen for the algorithm to utilises a small subset of the training set for the prediction. This parameter was not varied further to limit the amount of produced results.
The weighting scheme was set to distance, meaning the influence of each of the K neighbours is inversely proportional to the distance from the new point. This setting was chosen as our data set is of limited size. Because of the limited size, the examples will only sparsely populate the input space. In such situations it is more common for only a few highly predictive neighbours are near, with other neighbours being much more distant. In this case the prediction should be made based on the few neighbours who are very close, as the class membership of the distant neighbours is more likely to be due to random sampling. To mitigate this effect a distance based weighting was chosen, which will value the class closer neighbours higher than distant neighbours.

We set the distance metric to dynamic time warp, and set the search algorithm to use a brute force strategy. The choice for the first is self evident, while the second setting is fine for our small data set. If a bigger data set is used, the search space increases, and a different searching algorithm could provide significant speed ups.

In total two classifiers will be produced. One for the 1-NN parameter setting and one for the 10-NN parameter setting.

3.4.2 Time series forest

A time series forest approach utilises features over intervals of the time series to build a forest of decision trees. Each tree is made of a set of decision nodes that split the training set into the output classes. This method is interval based, and can be adapted with different sets of features.

The implementation used is the `CanonicalIntervalForest` method from *Sktime*. This method implements the catch22 feature set in addition to the mean, slope and standard deviation features of the

TSF approach [45] It also uses the margin property for tie braking [42]. A overview of the catch22 feature set can be found in table A.1 in the appendix. This method also includes several improvements such as sampling 8 out of the total 25 features for each tree [46].

The parameters we can vary can be divided into two groups. Parameters relating to the estimators and parameters relating to the intervals. For the estimators we can vary the amount through the `n_estimators` parameter. This will be varied between the values *100, 200* and *300*. The type of `base_estimator` will be set to *"CIT"* to use the entrance metric as well as to allow the extraction of importance curves. The parameters relating to the interval will not be varied, as the default values will suffice.

This will produce three sets of classifiers. One for each setting of the `n_estimators`. We can evaluate every classifier, and after training a feature importance curve can be extracted from each classifier.

3.4.3 InceptionTime

InceptionTime is a deep learning approach which uses a sequence of Inception modules and fully connected layers to produce a classifier. The method leverages a complex network to accurately distinguish between the different classes. This method benefits from and is susceptible to all the traditional perks and pitfalls of deep learning approaches.

The `InceptionTimeClassifier` method from the *Sktime_dl* companion package for *Sktime* was used for the implementation. This method allows a wide range of parameters to be set. The parameters relevant for the algorithm can be found in table 3.3.

Parameter	Description	Value type
`nb_filters`	Number of filters used in the Inception modules	Integer
`use_residuals`	Does the network use residual layers	Boolean
`use_bottleneck`	Does the network use bottleneck layers	Boolean
`depth`	The amount of subsequent Inception modules in the network	Integer
`kernel_size`	Size of the 1D convolution window	Integer
`batch_size`	Size of the training batches	Integer
`bottleneck_size`	Size of the bottleneck layer	Integer
`nb_epochs`	Amount of rounds of training	Integer
`callbacks`	Callbacks used to adjust parameters or values during training	Callback objects

Table 3.3: Parameters of the InceptionTime network. Only parameters relevant to the functioning of the algorithm were included.

The use of residuals and bottleneck layers was turned on to mirror the original algorithm. We set the batch size to a value of 8. This meant some performance loss, but aims to avoid a loss in generalisation caused by larger batch sizes [56]. We left the size of the bottleneck layer set to 32. We set our training epochs to 500, as gains flattened before this point. This choice will be supported through the

training curves provided in the discussion.

In our first attempts we kept the depth set to 6. This was based on the hyper-parameter study in the original study conducted by Fawaz et all. [47]. During the experiments this was found out to be insufficient. Subsequently we tested classifiers with the depth varied across 10 and 15. A more in depth explanation of this will be given in the discussion.

We initially varied the number of filters across (8, 16 and 32). The kernel size was varied across (5, 10 and 50). For later tests this was reduced to varying across (16 and 32), and (50 and 100) respectively. This was done to reduce the amount of possible training configurations when the depth was also varied.

During the study it also became clear that the learning rate did not decrease enough to converge to a minimum of the cost function. The default implementation includes a mechanism to reduce the learning rate when the convergence plateaus. It reduced the learning rate to a minimum of $1e-4$. This mechanism was adapted through a callback which changed the minimum learning rate to $1e-6$.

Our initial parameter settings produces 9 classifiers, as we varied the number of filters and the kernel size across 3 values simultaneously. When the variation of the depth was introduced adding 9 more classifiers to evaluate became unmanageable, so we reduced the amount of parameter variations. In the end 8 classifiers were tested. The number of filters, the kernel size as well as the depth was varied across 2 values. The parameter settings for these tests can be found in Table 3.4.

Besides the classifiers, characteristics of the training process can be extracted to provide insight in the algorithm and data. In this case the results will be supplemented with loss and accuracy curves of the training process. These metrics have been invaluable during the initial phases of testing. A more complete description of these metrics will be given in the discussion section.

Parameter	**Values or characteristics**
`nb_filters`	(16, 32)
`use_residuals`	True
`use_bottleneck`	True
`depth`	(10, 15)
`kernel_size`	(50, 100)
`batch_size`	8
`bottleneck_size`	32
`nb_epochs`	500
`callbacks`	reduced learning rate
Total number of classifiers	**8**

Table 3.4: The parameter used to create the classifiers

3.5 Other methods

During the initial investigation into what methods to use, additional options were found. In this section we will give a brief overview of other methods that were not included in this thesis, as well as an explanation for the exclusion.
We start with a shapelet transform based method [57]. This method finds short distinctive intervals, called shapelets, in the time series. These shapelets are then shifted along the time series to produce a distance between shapelet and time series. The distances can then be utilised to determine class membership. This approach hinges on the presence of class specific shapelets. The method was omitted because other methods based on both distance metrics and intervals were already included.

Other options considered were HIVE-COTE, a hierarchical voting ensemble method [49]. HIVE-COTE is widely seen as a state of the art approach, because of its outstanding performance. This method was omitted because of two reasons. The first has to do with the time complexity of HIVE-COTE. The system is build as an ensemble of many classifiers, each of which needs to be trained. This means training could take excessive amounts of time. The second reason for exclusion is the type of classifier. As HIVE-COTE is an ensemble method it provides less straightforward insight into which types of classifiers perform best on this specific task. As not much research into time series approaches for artificial lateral line data has been done, this information could prove valuable.

Other methods outside the domain of time series classifications were also considered. Previous research focused largely on more standard deep learning methods such as long short-term memory systems, recurrent neural networks and extreme learning approaches [4, 25, 30]. These methods do not exploit the temporal relationship between between points in a time series as effectively, because they were not designed for this purpose. In this thesis we explicitly focus on the time series approaches, as this was a little researched area.

3.6 Evaluation

To evaluate we had two primary data sets to consider. The three different swimmers with different styles, and the five different swimmers all wearing fins. Initial research was based on the three-swimmer data set, but the main results we will present is based on the five-swimmer data set. The five swimmer data set allows stronger conclusions to be drawn, as the amount of data is larger, and there are more classes available.

We have 9 available data sets to evaluate the classifiers with. These can be subdivided by the three data representations: the raw data, the spectrogram and the wavelet transform. For each of these data representations we have 3 variations based on frequency: the 5000 Hz, the 500 Hz and the 100 Hz sets. We also have three different types of classifier, each with their own range of hyper-parameters to evaluate. This adds up to $3 \times 9 = 27$ total combinations of classifier and data set.
We evaluated each of these 27 combinations to obtain our results. To do this we first split each data set into a training and testing set. A quarter of each set was reserved for testing. We then used the training set for a hyper-parameter search, to find the optimal hyper-parameter settings for the classifier. For this search we used a cross validation scheme with three folds. After we found the optimal hyper-parameters for a classifier, the whole training set was used to train that classifier with optimal settings. The test set was then used to score this optimal classifier. This scheme is repeated for each combination of classifier and data set, so 27 total hyper-parameter searches, and testing scores. This setup was chosen as the cross validation is most effective for limited data. An schematic representation of the evaluation scheme is given in 3.8.

This scheme produces multiple metrics for evaluation. The most important is a single accuracy score on the test set for each combination of classifier and data set. The cross validation on the training set also produces an accuracy score for each of its folds. In the results section we report the testing accuracy as well as the mean cross validation accuracy for each pair. For one of the data sets per classifier we also include a confusion matrix, showing which classes are most difficult for that classifier.

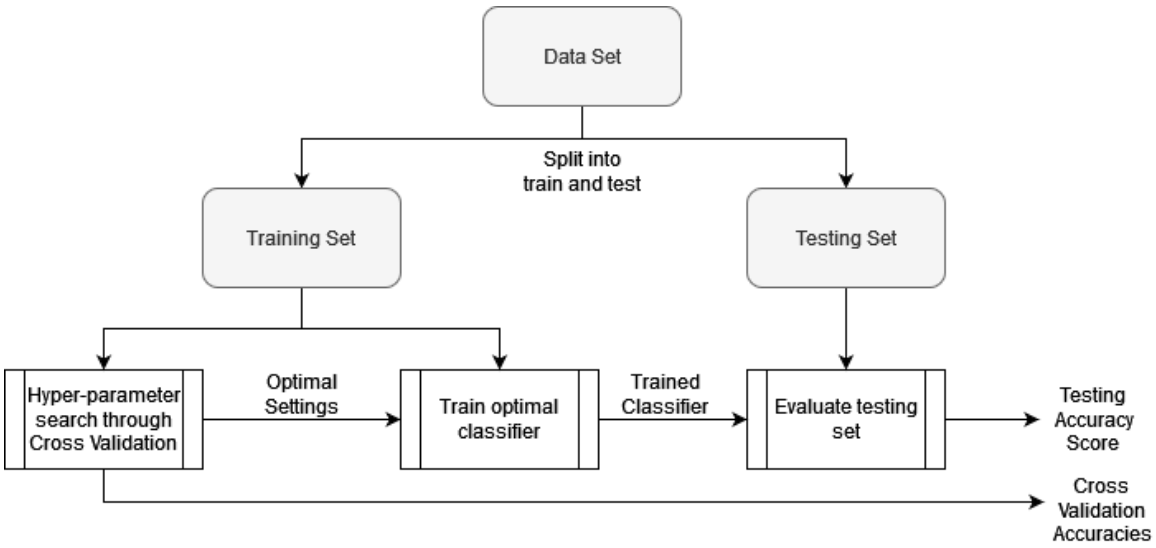

Figure 3.8: Overview of the evaluation scheme used in this thesis. This process is repeated for each of the classifiers and data sets.

We chose cross validation for the hyper-parameter search because it is more robust in situations with a small amount of data. A recursive cross validation scheme also including cross validated testing would produce even more robust results, which can be generalised more reliably. We did not perform a full cross validation in this thesis because it would triple the time taken for evaluation. Due to the need to evaluate 27 combinations of classifier and data set, this extra time investment pushed it outside the scope of this thesis.

To increase the generalisability of our results we performed an additional four fold cross validation on the whole data set. To limit the time investment, we only ran the hyper-parameter search once, reusing it for all the folds. For full cross validation the hyper-paramater search should be repeated each fold. This single hyper-parameter search, reduces the time of cross validation drastically, but it has a problem. In this scheme the hyper-parameters search is done on the same data present in the testing set. This could distort results somewhat, as the test set can influence the characteristics of classifier used to evaluate the test set. Because of this we decided to use our initial non cross validated testing scheme as our main results, and added the cross validated results as an additional resource at the end of the results section. They will also sparingly be used to draw conclusions from.

Chapter 4

Results

In this section of the thesis the results of evaluations detailed in the previous section are given. The results will be structured as follows: First the results of the hyper-parameter search will be presented for each classifier type separately. With this the a single confusion matrix is presented. The data set chosen, is one on which the classifier preforms relatively well, but not so well that the confusion matrix provides no insight. We will also present other metrics or features particular to that type of classifier, such as the importance curves for the time series forest. After this has been completed for the three types of classifiers, the tests scores gathered through the test sets will be compared.

4.1 K Nearest Neighbours with Dynamic Time Warp

For the nearest neighbour classifiers, two variations were tested: A 1-NN classifier and a 10-NN classifier. These two options were cross validated for each of the 9 data sets. The results can be found in table 4.1. The classifiers build from the optimal settings were evaluated against the test set. The test set results can be found in figure 4.1. The detailed results of the cross validation are given in appendix A.2. A confusion matrix of the results on the 5000 Hz raw data set is also given in figure 4.2.

		optimal K setting	Mean training accuracy	standard deviation
raw	5000 Hz	1	0.97	0.01
	500 Hz	1	0.98	0.01
	100 Hz	1	1.00	0.00
spectrogram	5000 Hz	1	0.39	0.05
	500 Hz	10	0.50	0.08
	100 Hz	1	0.50	0.04
Wavelet	5000 Hz	1	0.31	0.00
	500 Hz	1	0.49	0.04
	100 Hz	1	0.54	0.11

Table 4.1: cross validation results of the KNN algorithm. The optimal setting for the number of neighbours is presented. For this setting, the mean accuracy and standard deviation across the cross validation folds is included.

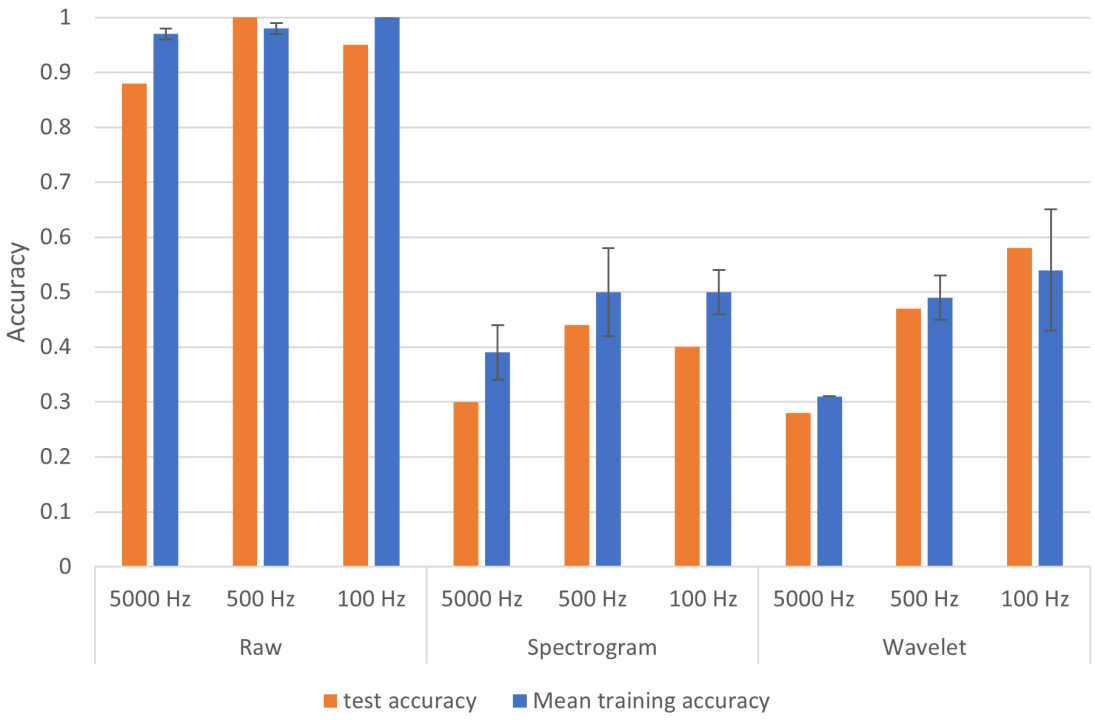

Figure 4.1: Test and mean training accuracy of the K nearest neighbour algorithm with dynamic time warp distance measure.

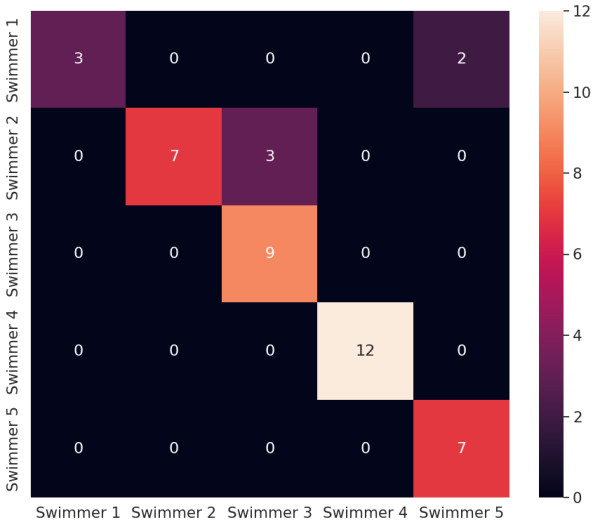

Figure 4.2: Confusion matrix of the for the K nearest neighbour approach on the 5000 Hz raw data set. The columns represent the predicted results, while the rows represent the actual results, meaning the top right entry represents runs from swimmer 1 classified as swimmer 5.

4.2 Time series forest

For the time series forest approach we chose to implement the canonical interval forest [46]. For this methods we decided to change only a single parameter, the number of estimators used. This is the amount of trees used for the ensemble. The results of the cross validation can be found in table 4.2.

As seen in this table, the spectrogram results for the 100 Hz data set is absent. This is because the time series in this data set have a length of only 5. The algorithm has trouble with time series of such a short length. Any random interval will overlap over the same points of time. This means the algorithm would in essence become a global feature classifier instead of an interval feature classifier. Because of this the algorithm was not adapted for shorter time series, and the results were omitted.

The classifier build with these parameter settings were evaluated against the test sets, producing the results seen in figure 4.3 The detailed results of the cross validation are given in appendix A.3. A confusion matrix of the results on the 5000 Hz spectrogram data set is also given in figure 4.4.

		optimal number of estimators	Mean training accuracy	std dev training acc
raw	5000 Hz	100	0.98	0.02
	500 Hz	100	1.00	0.00
	100 Hz	200	1.00	0.00
spectrogram	5000 Hz	200	0.73	0.11
	500 Hz	200	0.54	0.09
	100 Hz	n/a	n/a	n/a
Wavelet	5000 Hz	50	0.96	0.05
	500 Hz	100	0.96	0.01
	100 Hz	50	0.81	0.06

Table 4.2: cross validation results of the Canonical Interval Forest. The optimal setting for the number of estimators is presented. For this setting, the mean accuracy and standard deviation across the cross validation folds is included.

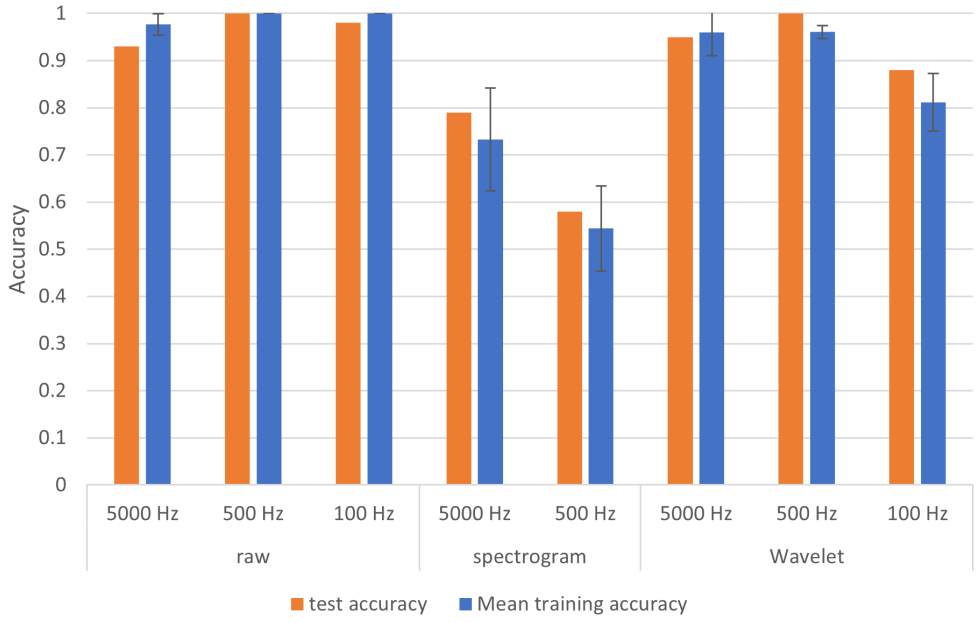

Figure 4.3: Test and mean training accuracy of the Canonical Interval Forest.

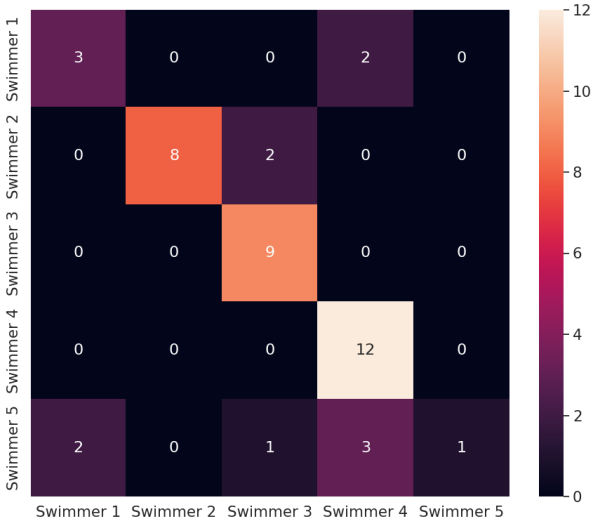

Figure 4.4: Confusion matrix of the for the Canonical Interval Forest approach on the 5000 Hz spectrogram data set. The columns represent the predicted results, while the rows represent the actual results, meaning the top right entry represents runs from swimmer 1 classified as swimmer 5.

4.2.1 Temporal importance Curves

The time series forests approach also allows us to construct the temporal importance curves. Each feature produces a separate curve for each dimension, meaning we can extract $25 \times 16 = 400$ curves. Here the 16 dimensions correspond with the x and y output of the 8 sensors. We aggregated these curves per dimension, figure 4.5, as well as per feature, figure 4.6. These curves were extracted from

the optimal classifier. Each separate line represents a different curve, meaning the each subplot in figure 4.5 has 25 curves, and each subplot in figure 4.6 has 16 curves.

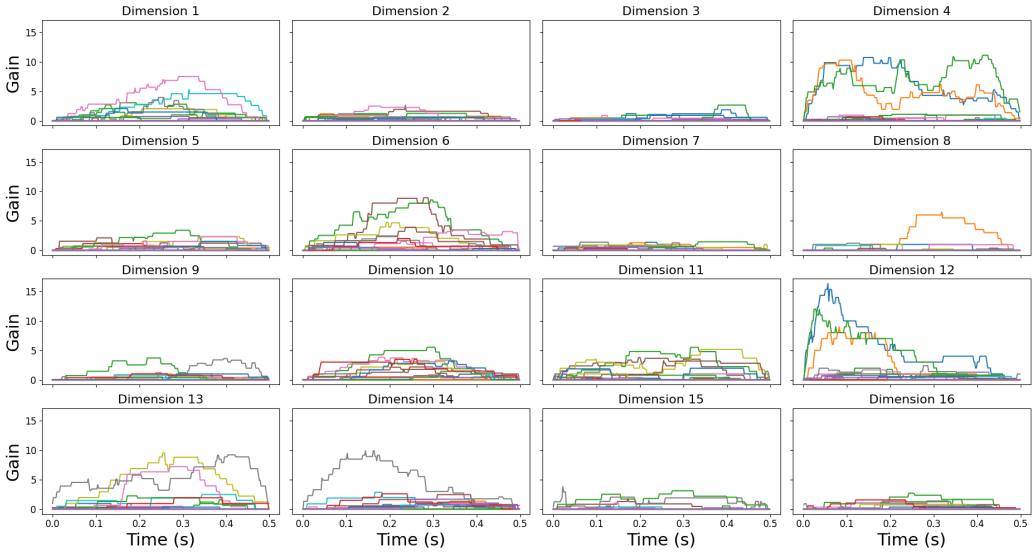

Figure 4.5: Temporal importance curves aggregated along the dimensions.

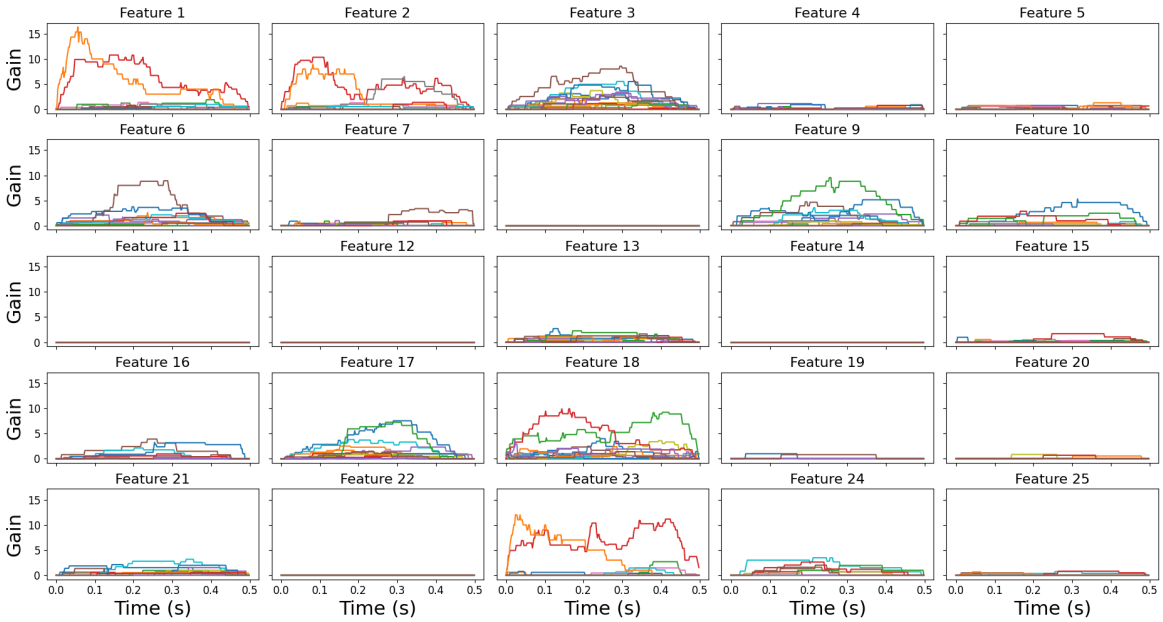

Figure 4.6: Temporal importance curves aggregated along the features. The first 22 features are part of the Catch22 feature set, while the last three are the mean, slope, and standard deviation features.

These figures show us how the importance curves are distributed across time. We can however also sum the gain for every point in time. This shows the information gain that the classifier gets per dimension or feature. Aggregating the gain makes it easier to determine which features and dimensions are informative. These results can be found in figure 4.7 and 4.8.

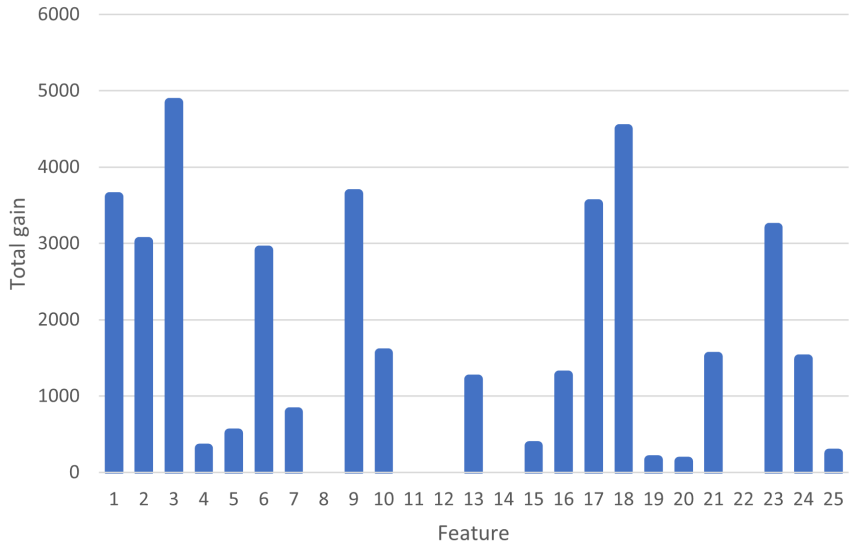

Figure 4.7: Total gain summed over time and dimensions. Gain was aggregated such that each bar represents the total gain corresponding to a feature.

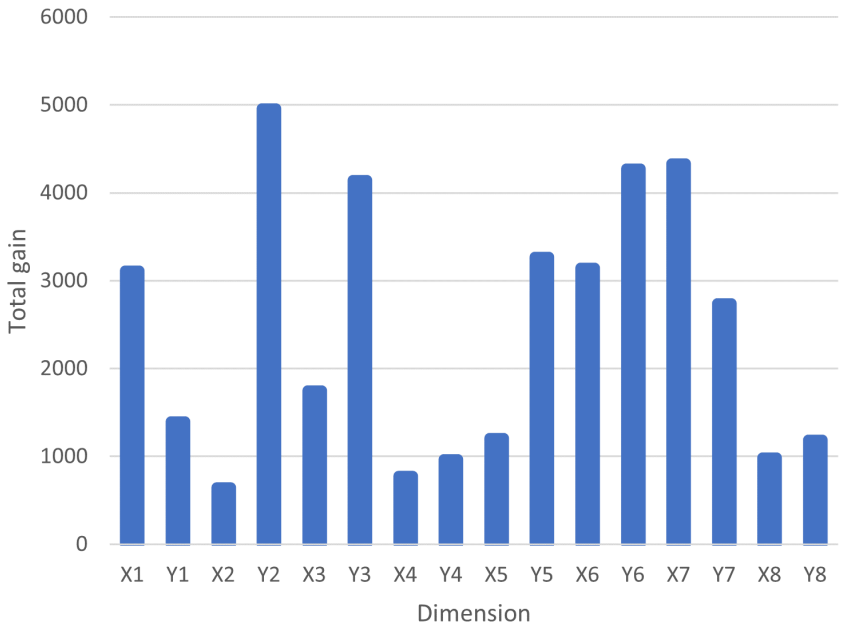

Figure 4.8: Total gain summed over time and features. Gain was aggregated such that each bar represents the total gain corresponding to one dimension of a single sensor.

4.3 InceptionTime

For the InceptionTime network, three settings were varied. The results of the cross validation for each of the nine data sets can be found in table 4.3. The optimal setting for the depth, kernel size and number of filters is given for each data set. Classifiers trained on the training set were evaluated on the testing set. These results can be found in figure 4.9. The detailed results of the cross validation are given in appendix tables A.4 to A.6. A confusion matrix of the results on the 100 Hz wavelet data set is also given in figure 4.10.

		Depth	Kernel size	number of filters	Mean training accuracy	Standard deviation
raw	5000 Hz	10	50	16	0.91	0.11
	500 Hz	15	100	16	0.94	0.11
	100 Hz	10	50	16	0.91	0.08
spectrogram	5000 Hz	10	50	16	0.30	0.04
	500 Hz	10	50	16	0.30	0.04
	100 Hz	10	50	16	0.30	0.04
Wavelet	5000 Hz	10	50	16	1.00	0.00
	500 Hz	10	50	32	0.95	0.02
	100 Hz	10	50	32	0.77	0.07

Table 4.3: Cross validation results of the InceptionTime network. The optimal parameters for each data set is given. For these settings, the mean training accuracy and standard deviation across the validation folds is presented

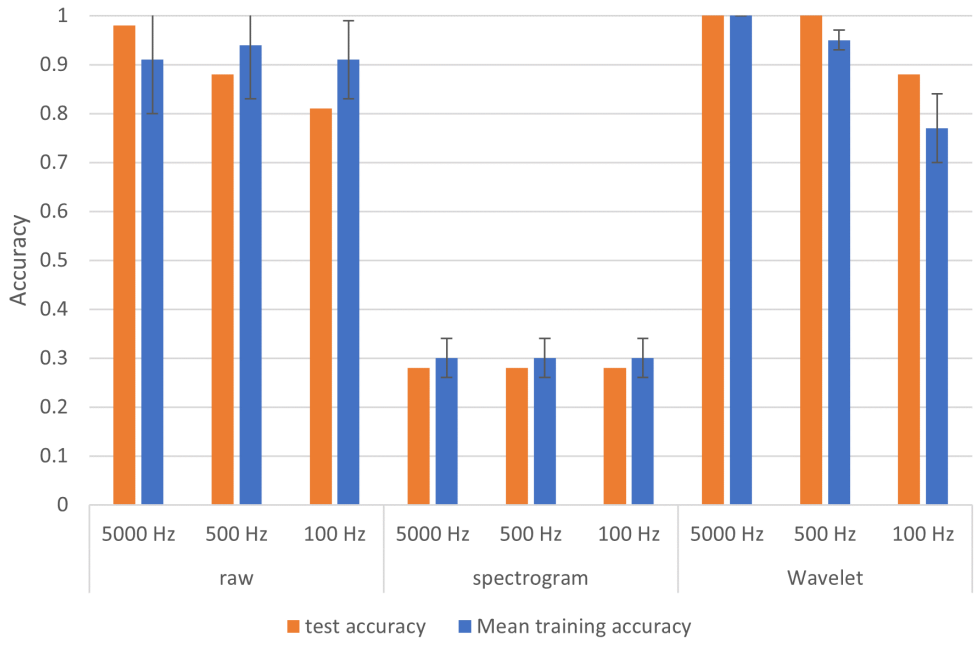

Figure 4.9: Test and mean training accuracies of the InceptionTime network.

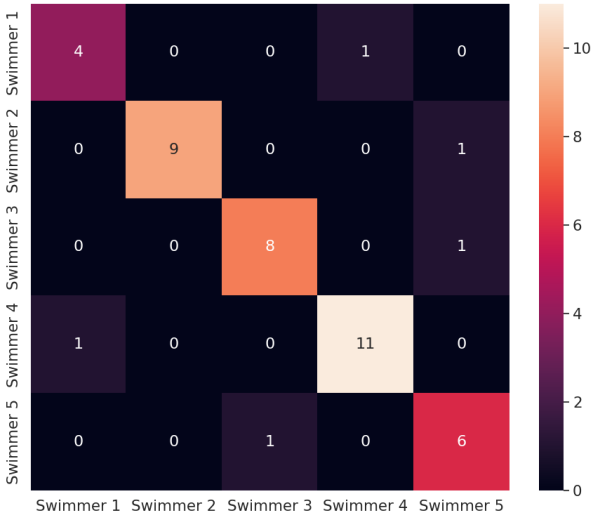

Figure 4.10: Confusion matrix of the for InceptionTime network on the 100 Hz wavelet data set. The columns represent the predicted results, while the rows represent the actual results, meaning the top right entry represents runs from swimmer 1 classified as swimmer 5.

4.3.1 Training metrics

In this section we will present some metrics that can be recorded during training of the InceptionTime network. We present the loss and accuracy curves throughout the training. These curves show the performance of the the network on the training set. We also included a validation loss and accuracy. These measures are based on the loss and accuracy on the test set. These metrics were recorded when a classifier with the optimal parameters was trained on the entire training set.

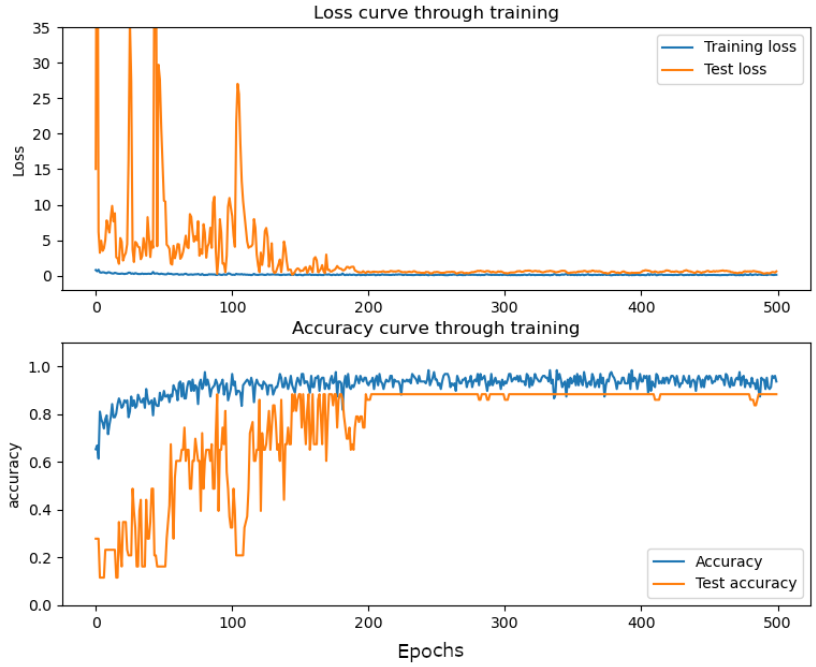

Figure 4.11: Loss and accuracy curves for the InceptionTime classifier during training on the raw 500 Hz data set.

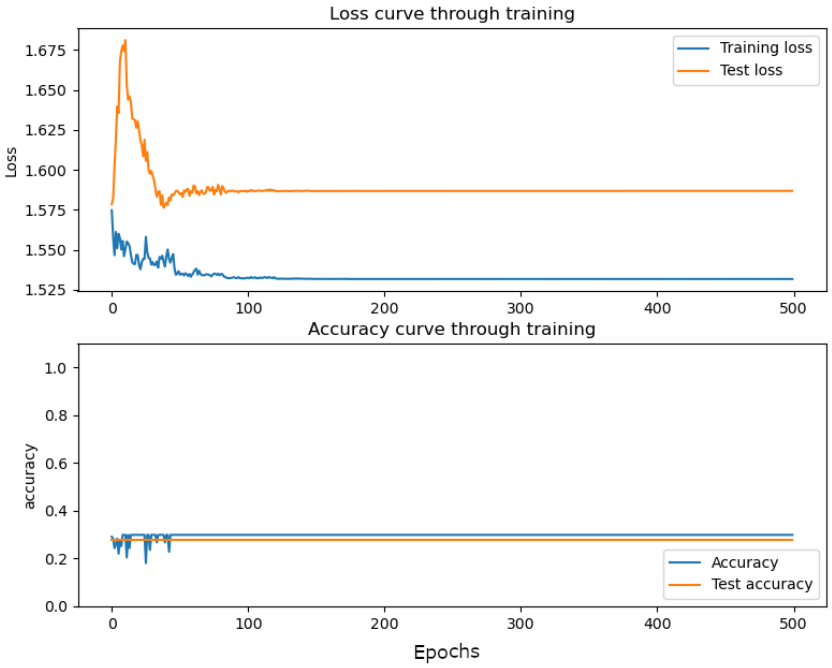

Figure 4.12: Loss and accuracy curves for the InceptionTime classifier during training on the spectrogram 500 Hz data set.

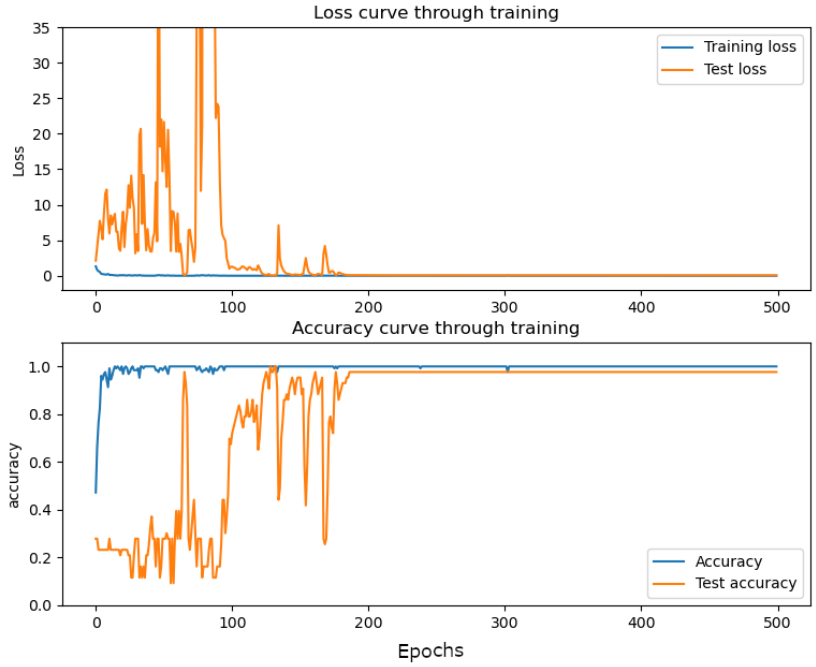

Figure 4.13: Loss and accuracy curves for the InceptionTime classifier during training on the dynamic wavelet transform 500 Hz data set.

4.4 Comparisons

In this section, results from previous sections will be aggregated to compare. For each of the three types of classifier, we determine the optimal classifier per frequency data set. Including one classifier for each frequency value allows any effects of frequency to be easily discerned. Another comparison presents these results, but included one classifier for each data transform, allowing us to compare the data representations across classifiers. For each of the categories the mean cross validation accuracy is given as well as the testing accuracy. These results can be found in figure 4.14 and 4.15

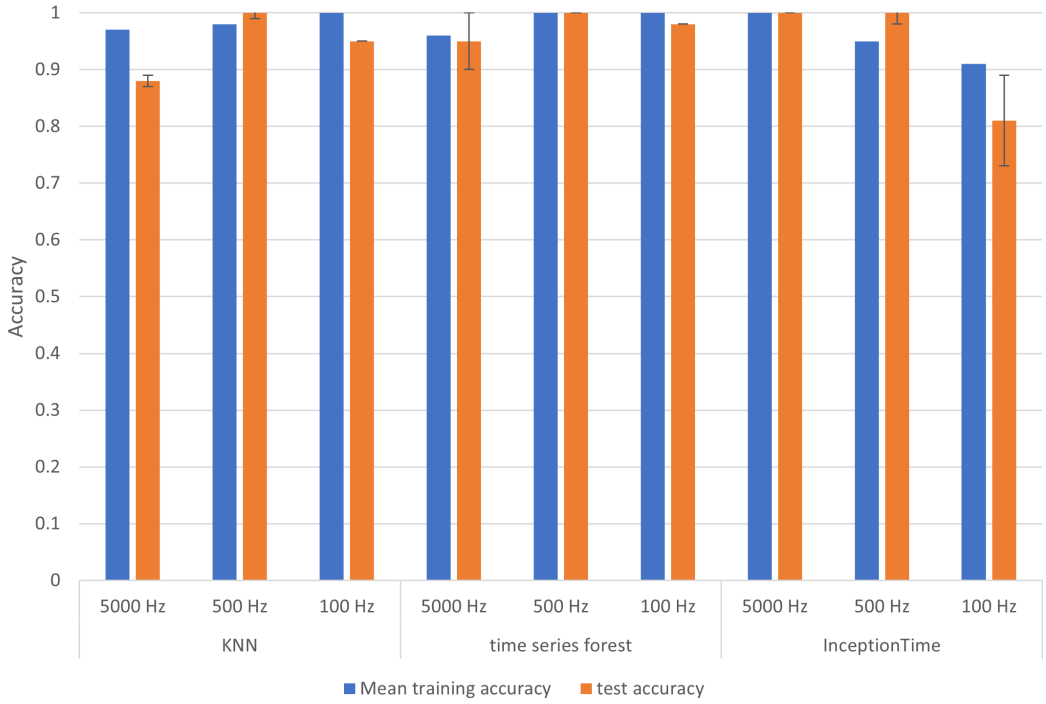

Figure 4.14: Comparison between the results of the best performing classifiers. For each of the three classifiers we included the best classifier for the three different frequencies (5000 Hz, 500 Hz and 100 Hz).

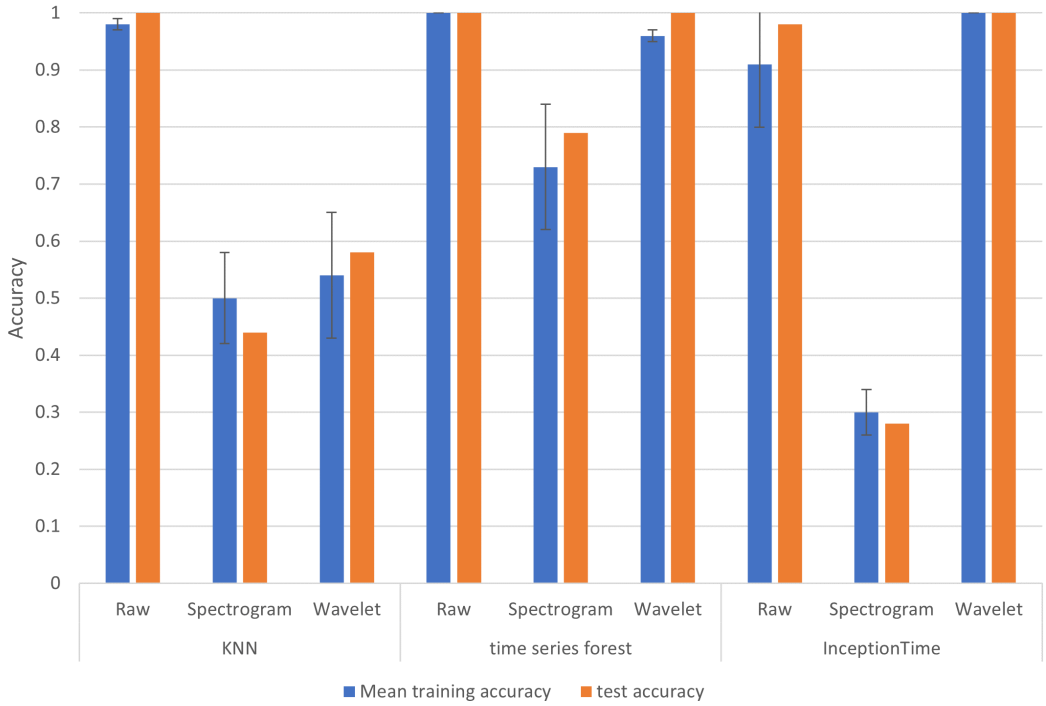

Figure 4.15: Comparison between the best results of the best performing classifiers. For each of the three classifiers we included the best classifier for the three different data representations (Raw, Spectrogram, Wavelet).

4.5 Supplemental cross validation

In this section we will be presenting the results of the extra cross validation performed on the entire data sets. These are meant to serve as supplementary results and not the main results of this thesis due to problems described in section 3.6. The results show the mean cross validation accuracy and standard deviation, as well as giving the test accuracy presented in the previous results as a reference. These results can be found in figures figs. 4.16 to 4.18

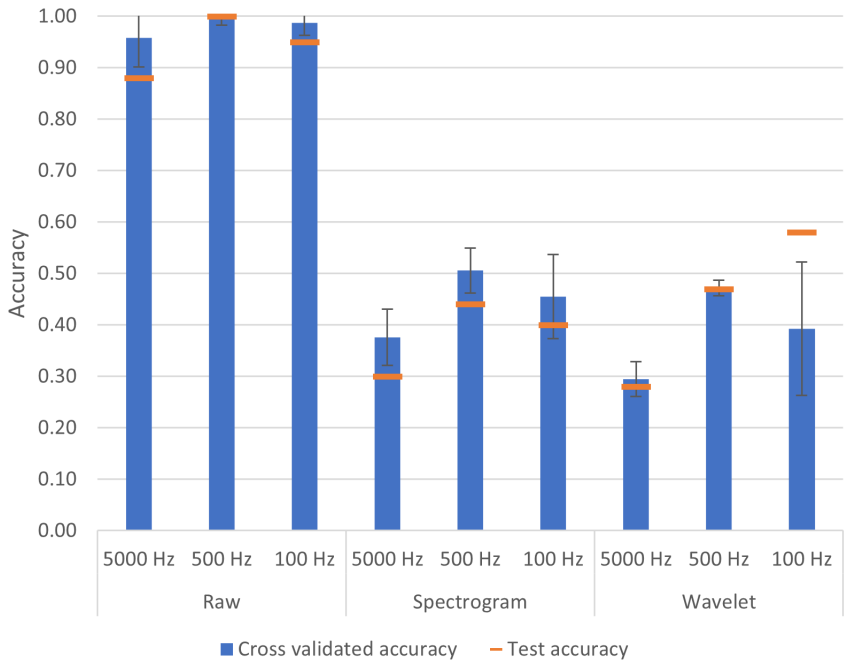

Figure 4.16: Additional cross validation results for K-Nearest neighbours with dynamic time warp. The orange lines show the results on the test set, while the blue bars show the mean of the cross validation accuracy over the whole data set.

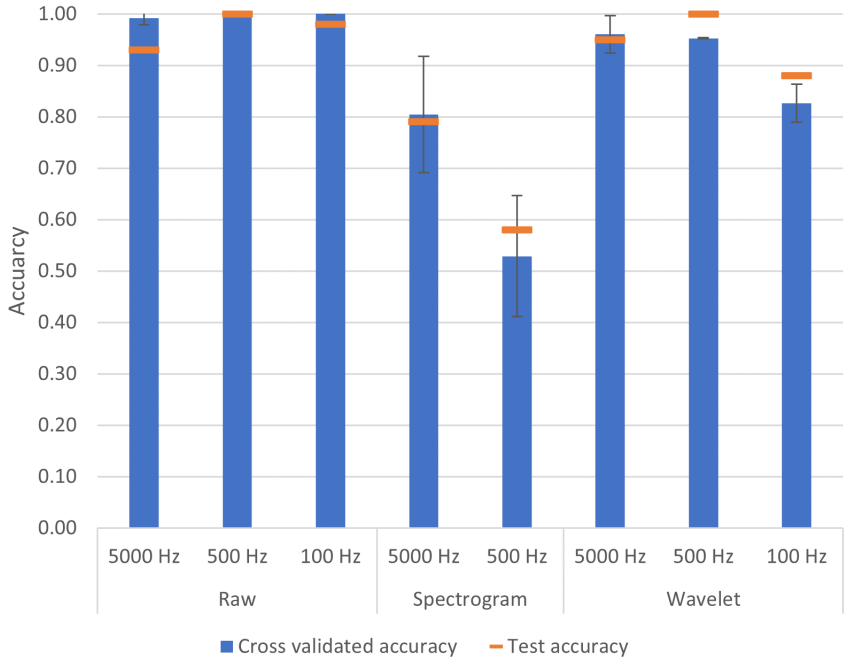

Figure 4.17: Additional cross validation results for the time series forest approach. The orange lines show the results on the test set, while the blue bars show the mean of the cross validation accuracy over the whole data set.

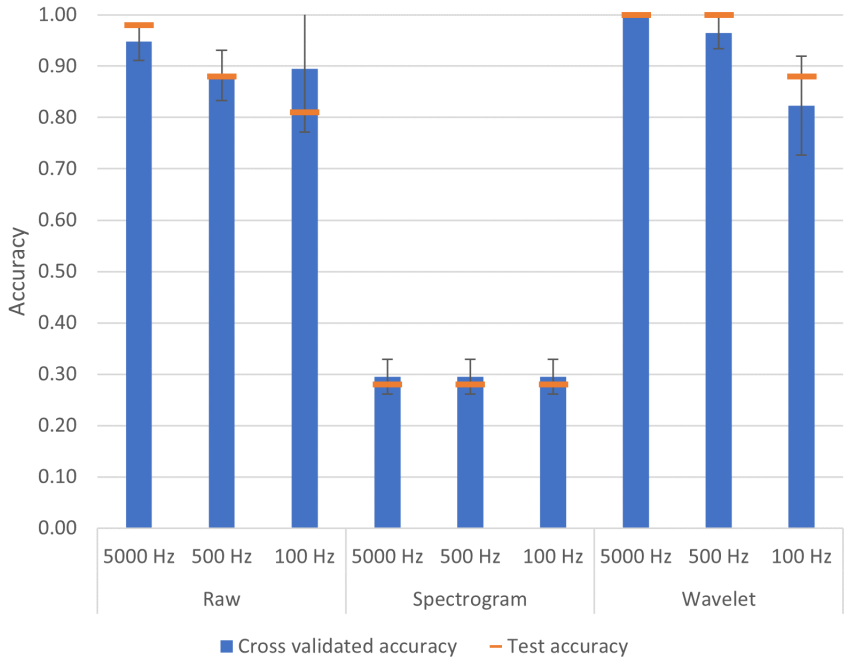

Figure 4.18: Additional cross validation results for the InceptionTime network. The orange lines show the results on the test set, while the blue bars show the mean of the cross validation accuracy over the whole data set.

Chapter 5

Discussion

5.1 Results on the task

5.1.1 Nearest neighbours with dynamic time warp

When looking at the results in table 4.1, we see a clear preference for a K-value of one. The results on the test sets in figure 4.1 also indicate a clear preference of for the raw data. The spectrogram and the wavelet data seem to perform similarly, both being outperformed by the raw data. We can also see a performance increase as the frequency decreases for both the spectrogram and the wavelet data sets.

The preference for the single neighbour classification can have multiple reasons. It seems that the closest neighbour is a better indication than the ten closest neighbours. It could be that the nearest neighbour is by far the closest, where all other neighbours are at such a distance that each class is equally represented in this set of further neighbours. However the usage of the distance based voting scheme should alleviate this situation. As the distance from the second to tenth neighbour increases, the voting becomes more similar to the 1-NN case, as the weights associated with the other neighbours decreases.

The raw data sets outperform the spectrogram and wavelet data set, which was to be expected. The transforms trade in resolution in the time domain for resolution in the frequency domain. If a component of the signal has a shift in frequency this is represented in a different way in each data set. In the raw data set, surrounding values will be slightly changed. In the case of the spectrogram, the change can be found along the frequency dimension, not the time dimension. For the discrete wavelet transform, the change can be found in the coefficients of next or previous level. Dynamic time warp only handles changes along the time axis well. One could develop a method which allows warping along the frequency dimension as well, but this falls outside the scope of this thesis.
That performance increases when the sampling frequency decreases can also be explained through the same reasoning. As the frequency decreases the time interval which each point in the time domain represents increases. For the transforms, this loss in time resolution also means a loss in frequency resolution. The loss in frequency resolution means each point of the transformed data set represents a larger frequency interval. A shift in the frequency or time point of a component is more likely to still be encoded by the same point of the spectrogram, allowing for easier matching. In essence a small change doesn't change the spectrogram of the lower frequency data much, but does effect the spectrogram of the higher frequency data. This allows easier more effective matching between spectrograms of the lower frequency data.

5.1.2 Time series forest

The result of the time series forest cross validation found in table 4.2 seems to show no clear preference for a high or low number of estimators. The classifiers trained on wavelet data seem to have a slight preference for a lower number of estimators, and the spectrogram classifiers for a higher number of estimators.
On the test set we see the classifiers trained on the raw data set performing the best. Figure 4.3 shows these classifiers narrowly outperforming the wavelet classifiers. The classifiers trained on the spectrogram seem to perform significantly worse than the classifiers trained on the two other types of data sets.

The absence of a clear effect of the number of estimators can be explained by one of two reasons. Either no effect is present, or the effect is not large enough to be detected in the configuration of the experiments. In both of these cases the number of estimators does not function as a significantly limiting factor on the performance of the classifiers. A lower number of estimators seems more appropriate, as this decreases trainable parameters, system complexity and increases generalisation of the system.

The classifiers seems to be perform well on the raw data sets. The classifiers trained on the spectrogram data set performing significantly worse can be ascribed to the change of data representation that the transform represents. The features of the catch22 set are most applicable to time series data. These features are less informative when taken over the power spectrum of the signal.

The performance on the wavelet data sets indicates the difference in representation between the two transforms. The discrete wavelet transform preserves more of the local characteristics of the data, through the repeated separation into detail and approximation of the signal. The simplicity of the Haar wavelet transforms the data in such a way that the catch22 features seem informative. A more complex wave function could decrease the performance, although further research must be done to confirm this.
For both the spectrogram as well as a wavelet transform with a more complex base waveform, the catch22 feature set is most likely not informative enough. One could define a set of features that provides more information when applied to these data representations. Because of this, it would be wrong to discount the performance of the spectrogram completely in comparison to the other data representations.

The temporal importance curves give an indication of the information each time point provides to the classifier. Each curve corresponds to the gain per feature per dimension over the length of the time series. Both figure 4.5 and 4.6 show these importance curves aggregated along the dimensions and features respectively.
The shape of these curves are as expected. Temporal importance curves are naturally biased towards values in the middle of the time series, as more intervals includes these time points over time points at the ends of the series [42]. Any deviation from this half circle shape indicates that the discriminatory properties of the signal are not uniformly distributed along the time axis. We see this for some dimensions and some features, such as dimension 12 and feature 1, both of which are biased for the

start of the time series. We refrain from drawing any conclusion based on these shapes, as no clear pattern is visible across dimensions or features.

When we look at the aggregation of the curves, we can however compare the performance across the feature or dimensions. For the features we see this in figure 4.6 and more clearly in figure 4.7 which are more and less compact version of the same data. We see features 8, 11, 12, 14 and 22 do not provide any gain to the system. A description of the features can be found in table A.1, in the appendix. Feature 8 relates to the lower fifth of the frequency power spectrum. This seems to indicate low frequency bands are not informative. We will come back to this later on in the discussion section, as it seems counter to some results found in the literature. Feature 11 and 12 relate to how time independence of the signal. Feature 11 is a time reversibility statistic and feature 12 is an automutual information feature. As the swimmers move through the pool during their swimming, the signal will most likely not be time independent, so features of this type will not provide much discernibility between runs. Feature 14 relates to the ratio of the successive differences greater than 0.04 of the standard deviation. As this feature is not informative, this ratio is most likely similar for all time series. Feature 22 is a periodicity measure. As the features are calculated over random intervals of time series, the periodicity is difficult to capture. The intervals are most likely not large enough to capture the periodicity of the signal. [45]
When looking at figures 4.6 and 4.7 again we can also see that features 1, 3, 9, 17 and 18 provide the most gain to the system. These features are not all of the same type, and are related to different metrics. This variance in the informative features supports the idea that the system is not overfitting too much. If a single characteristic is deciding, we would expect the most of the gain to be contributed by a single or small set of features.

When we look at the importance curves aggregated along the dimensions provided in figures 4.5 and 4.8, we see all dimensions contribute to the total gain of the classifier. This again supports the idea that no overfitting occurred, as no single sensor provides a decisive characteristic.

5.1.3 InceptionTime

During the cross validation of the InceptionTime network three hyper-parameters which were evaluated. Before we look at the optimal values for the hyper-parameters it is important to take the context of the training accuracies into account. Caution should be taken when drawing conclusion from optimal hyper-parameters of an under performing data set, as the hyper-parameters are most likely not the performance bottleneck. Table 4.3 shows the classifiers trained on the spectrogram under performing when compared to the raw and wavelet data sets. When looking at the table A.5 in the appendix we see that no configuration of the hyper-parameters settings had any effect on the outcome.
For the raw and wavelet data sets, we see the classifiers prefer a lower depth and kernel size. The raw data sets prefers a lower number of filters, while the wavelet data sets seem to prefer a higher number of filters.

When looking at the results of the optimal classifiers on the test set, we see the same trend as the mean cross validation. The spectrogram data sets perform significantly worse than the raw and wavelet data sets. The wavelet data sets seems to perform slightly better than the raw data sets. This relation between the data sets mirrors the same relation we saw for the time series forest approach. For both the

raw, as well as the wavelet data sets we see a slight trend of performance loss as the frequency of the data set decreases.

The optimal settings favouring a smaller depth and kernel size could indicate the network performance is not restricted by its size. Less expansive networks are capable of extracting the necessary information to perform well on the classification. A smaller kernel reduces the amount of parameters the network adapts during training. A smaller depth would mean the gradient in the update step of the network decreases less throughout the network. Both these factors entail a smaller network is favoured in the absence of other effects on performance. Because the data sets are small it is difficult to determine whether this limit to the network complexity is due to inherent characteristics of the data, or the lack of variations in the data set.

Spectrogram performance

The underlying reason for the lower performance of the spectrogram data sets is not immediately clear. Figure 4.12 shows training loss decreasing only slightly before staying on a plateau. The most likely explanation is related to the number of dimensions present in the spectrogram transform. The transform reduces the length of the time series, while increasing the dimensions. InceptionTime is a approach designed specifically for time series. The strength of this approach lies in the kernels, which can be shifted along the signal. This allows the kernels to be adapted to all the data points in the time series. The spectrogram transform decreases the time series length, which reduces the training points of each kernel. Thus the update step of the parameters is less efficient. The increase in dimensions, produce more parameters which need to be trained. The spectrogram increases the amount of training parameters, and decreases the amount training efficiency.
This increased parameter space means the network is more likely to start far away from the global minimum of the loss. Here the gradient loss function is more likely to be small. Combining this with the reduced efficiency of the update step, means the network is more likely to descend into a local minimum far away from the optimal parameters. This possibility fits with the loss curve we see in figure 4.12.

The trend of decreased performance when for lower frequency data sets, is more easily explained. As this trend is visible for both the raw and the wavelet data sets, it is most likely that the high frequency components are informative for the network. Another explanation for this could again be the kernels shifting along the signal. The higher frequency data sets allow the kernel to be shifted along more points, as a the time series is simply longer.

When looking at the accuracy and training curves of the raw and wavelet data sets, we see the training loss sometimes spike. This is important to keep in mind, because it could potentially be a sign of overfitting. The spiking is most likely due to the small test sets, as the loss is only calculated on a small set of examples.

Early iterations

In early stages of this research two problems with the performance of the InceptionTime network became apparent and were subsequently fixed. The first related to the settings of the depth parameter. We initially set the depth to a value based on the hyper-parameter study done by the original paper

of InceptionTime [47]. This turned out to be insufficient for our purposes. The network performance was only slightly better than random guessing. The most probable explanation is that the depth is needed to extract informative features from the data. Another option would be the network is overfitting when the depth is increased. This problem with increasing the depth is given in the original paper. We however would expect the test performance to not grow with the training performance once the depth was increased. We would also expect some improvement in the training accuracy, but not quite so big a difference, as well as the spectrogram performance increasing similarly to the raw and wavelet performance.

Another problem we encountered with the default implementation had to do with the learning rate. The default implementation utilises a method to reduce the learning rate, to make sure gradient descend does not bounce around the minimum of the loss. An example of the loss and accuracy curves under this default setting was given in figure 5.1. In this plot we can see that the loss follows an erratic path. The most likely cause of this is the learning rate not reducing enough. The gradient update step adjust the trainable parameters too much, overshooting the loss minimum. When we allowed the loss to reduce even further the classifiers performed better, and produced the loss curves visible in figure 4.13. From this we can conclude that the minima of our loss function are highly localised, surrounded by steep gradients. A small adjustments of the parameters is likely to reduce the accuracy of the network drastically. This could be inherent to the type of classification task, or this could be an indication of a system that does not generalise well. Without more data it is hard to discern which is more likely.

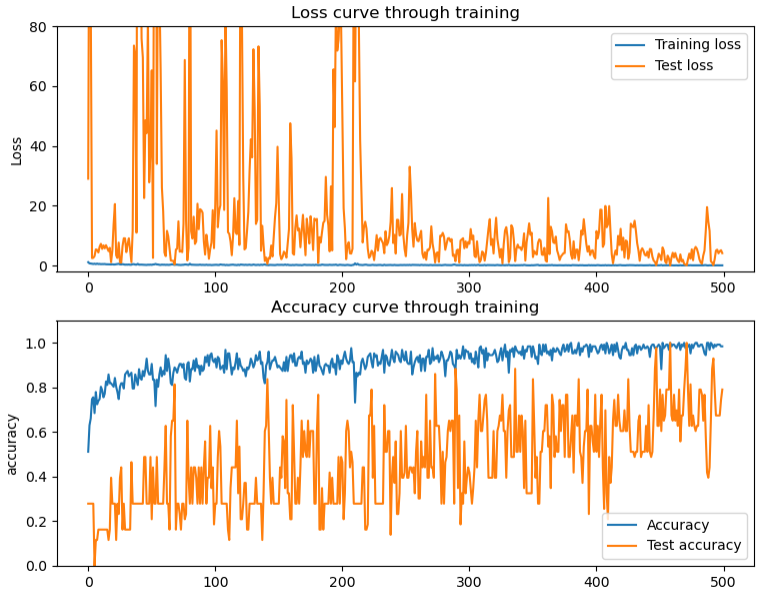

Figure 5.1: Loss and accuracy curves for the InceptionTime classifier during training on the discrete wavelet transform 500 Hz data set. The reduced loss settings were not adapted in this test.

5.1.4 Comparisons

We will now compare the performance off the different networks on the different frequency data sets. This allows us to compare how well the different methods would perform under situation where not much bandwidth is available for data collection. Afterwards we will look at the different transforms and how much they add to the performance of the networks.

Networks

When the performance of the different approaches is compared such as in figure 4.14 we see all the approaches perform relatively well. The InceptionTime network seems to work best on the 5000 Hz data set. The time series forest approach performs best on the 500 and 100 Hz data sets. This comparison shows that the choice of classifier is not apparent from just their performances.

The largest influence on choice of classifier will be outside factors, not their performance. If a method without much initial overhead is required, the KNN can be used. This method has problems with scalability, as its size increases with the training set. The InceptionTime network is most suitable for high frequency data, and will benefit from further advances in the realm of deep learning. The time series forest approach allows for easy adaptation to domain of application through changing of the features used. The temporal importance curves can be invaluable when explanations of the decision making process are required.

Transforms

An overview of the performance of the transforms, can be found in figure 4.15. When we compare the transform we see that the spectrogram is not as well adapted to the classification methods we used in this thesis. The inherent strength of the time series approaches, is that they can utilise the time dimension to its advantage. The spectrogram likely sacrifices too much of the time dimension to obtain frequency resolution. Another explanation could be the frequency bands that a spectrogram represents. A Fourier transform over a short interval has difficulty distinguishing between low frequency signals. In the context of the ALL this means more difficulty in discerning between low frequency waves produced by the swimmers, such as low frequency shedding effects. Other research has pointed to low frequencies being important for classification based on ALL signals [30]. Thus this type spectrogram might not be suitable to distinguish between the swimmers.

When comparing the discrete wavelet transform to the raw dataset, we see similar performance for the time series approach and the InceptionTime network. The performance is significantly worse for the KNN approach. This transform is characterised by the base wavelet used. A base wavelet such as a Daubechies wavelet could perform even better, as suggested by some studies [58]. This would increase the time the transform takes, but the the more complex waveform could represent the movement of the water more clearly. In this thesis there was not a clear advantage of using the wavelet transform over the raw data, as both attained comparable results.

One important note when comparing the transforms to the raw data is the type of classifier evaluated in this thesis. All methods were adapted for time series data. The data transforms necessarily change this data representation. Thus conclusions about the different data representations will most likely be biased toward the raw data representation.

5.2 Supplemental Cross validation

The results of the additional cross validation seems to be in line with the performance on solely the test set. The result which deviated the most was the 100 Hz wavelet data set for the KNN approach. This results is a bit lower under the cross validation scheme, bringing it more in line with the other results for the wavelet data sets.
The results strengthen the generalisability of our other results, which is important as we have a small data set. We will not be drawing any other conclusions from these results however, as explained in section 3.6.

5.3 Data set

During the discussion section, multiple references have been made to the size of the data sets as a limiting factor. It is the largest factor that casts doubt on any conclusions drawn in this thesis. We have seen multiple effects which could be the result of the small data set, such as the distribution of the training examples for the KNN approach, or the loss spiking observed during the training of the InceptionTime network. A larger data set would clarify a whether any other underlying effects are present.

A more insidious problem is also present in this data set. Normally the testing examples provide a clear indication of the generalisation and overfitting of the approaches. In this case the testing data is so closely related to the training data, that classification based on characteristics inherent to the runs instead of to the swimmers will increase the performance on the testing data as well. Under normal circumstances this would constitute overfitting and would decrease the performance on the test set. In the context of this thesis this would be expressed as decrease in generalisation, as performance on the test set is still high. To mitigate this, we would ideally separate the training and testing set based on the runs of the swimmers. Using a single run of the swimmers to construct the testing set, while using the other for the training. This was however not done in this thesis as this would mean only half of the data set could be used for training purposes. Since not much data was available this risked incorrect conclusions as the classifiers could not be properly trained based on this data.

This is not too say all signs point to bad generalisation. When we look at the confusion matrices we can see another characteristic of the data. We see most of the incorrect predictions are in the outer ring of the confusion matrix. This means they have to do with either swimmer one or five. Such an effect was expected, as these were the two swimmers with the least amount of data. Therefore, discerning between these and other classes is more difficult. What we do not see however is a large part of the errors being located in the top right and bottom left cells. These would be errors confusing swimmers one and five. We would expect to see more of these errors if the classifiers were predicting based on a characteristic of the sessions. The runs for swimmers one and five were recorded in a separate session from the other runs. If the characteristics of a session were used for classification, we expect to find many errors to be present confusing runs within the same session, and less for runs in other sessions. Since this effect is absent, we can reasonably conclude that this was not a problem.

5.4 Machine learning and time series classification

In the context of machine learning and time series classification, this thesis fulfils an interesting niche. The data is a small real world data set. These types of problems are usually the most difficult to evaluate. During this thesis we saw what problems this can cause when trying to draw solid conclusions.

Another interesting factor of any artificial lateral line data, is its status as sensory data. While most sensory data can easily be translated to human senses, this data cannot. This means we cannot rely on human intuition about the data. This can produce obstacles in research, an example of this is augmentation of the data to produce additional training data. Data augmentation of other sensory data can be evaluated based on the human perception. If an image still depicts the same after data augmentation, then that type of augmentation is applicable. Data produced by an artificial lateral line does not enjoy this benefit.
The necessity to overcome such obstacles can be a good driving force for innovation. Findings in this area can then be applied to other domains. In essence, artificially lateral line research forces another perspective, as we have no frame of reference for this type of sensing data.

During this thesis we also uncovered a promising confluence around the topic of wavelets. We saw the wavelet transforms being applied in the domain of time series classification [58], as well as being utilised in the context of more classical artificial lateral line research [22]. A technique that enjoys support from both time series research as well as ALL research, should be explored more in the context where these fields meet. Further research into this topic could provide insight on the utilisation of the wavelet transform for time series classification. Additionally it can be helpful in the wider context of using expert knowledge of the application domain to design features for classification. A wavelet transform more closely matching the natural wave produced by swimmers could mean significant performance boosts.

5.5 Artificial Lateral Lines

In the context of artificial later lines classification we have seen that relatively good performance can be achieved on real world data. The classification task also deviated for the more standard object detection tasks, by focusing on the movement of swimmers. As already stated at multiple points in the discussion, the data set used was too small to draw any definitive conclusions. Nevertheless this does support the idea that more complex classification can be attempted from ALL data.

This seems to be in line with other research in this area. The methods improved on earlier research on swimmer classification using an ALL [4]. This thesis also extends other research in a similar setting which showed that shape classification using ALL data is possible [30]. In that research Wolf et al. showed that low frequency components of the signal were more important for shape classification. This thesis does not support such a preference for low frequencies, and for some configurations even indicates a preference for higher frequency signals. The cause of this could be twofold. The lower frequency signal components could be determined mostly by the overall shape of the object. Effects such as the periodicity of vortex shedding would be indicative of the contour of an object in this case. Conversely the higher frequency components would be determined by the small differences between swimmers and their movements. Another explanation for the preference for high frequency signals could have to do with the training methods employed. The higher frequency signals has more time

steps, allowing the training methods to exploit more time steps for training. This second explanation seems more likely, as InceptionTime, seems to gain most from higher frequencies. This method can leverage the increased time steps more effectively than an interval based approach such as time series forests, where the higher frequency data set performs worse.

We have also seen that multiple techniques attain adequate performance. This allows for some choice based on the specific application domain. One example could be a sensor array in an inaccessible location without a good connection. This would force either a lightweight approach or an approach suited to lower frequency data. One would not choose the nearest neighbours approach if a small amount of storage is available and the InceptionTime network is not the preferred choice if only lower frequency data is available.

5.6 Further research

During this thesis we saw quite some possibilities for further research. The most apparent is an increased data set, as this allows stronger conclusions to be drawn. This extends not only to the amount of data, but also the type of data. This thesis focused on the identification of swimmers, but identification of the style of swimming could provide valuable insights.
when looking at the transformation and classification methods, we can also identify some areas of interest. An example is the use of more complex wavelet transforms in the context of the artificial lateral line. This topic has been approached from multiple angle in terms of research, so unifying this more thoroughly could be promising.
Another topic for further research could be the design of features specific to the context of ALL research. The time series forest approach could benefit greatly from features adapted to the ALL. These features could also be utilised by other machine learning methods.
Lastly we have seen that the classification methods cannot utilise the transforms to their fullest, as they are designed for time series data. Further research could adapt these methods to utilise the frequency domain, without losing the exploitation of the time domain.

5.7 Conclusion

This thesis aimed to research the classification of swimmers through measurements made by an artificial lateral line. The results showed good performance on the data set for different classification configurations. Because of a limited data set, generalisation of these results should only be done conservatively.

Bibliography

[1] B. Budelmann and H. Bleckmann, "A lateral line analogue in cephalopods: Water waves generate microphonic potentials in the epidermal head lines of sepia and lolliguncula," *Journal of comparative physiology. A, Sensory, neural, and behavioral physiology*, vol. 164, pp. 1–5, 12 1988.

[2] S. Dijkgraaf, "A short personal review of the history of lateral line research," in *The Mechanosensory Lateral Line*, S. Coombs, P. Görner, and H. Münz, Eds. New York, NY: Springer New York, 1989, pp. 7–14.

[3] B. J. Wolf, "Hydrodynamic imaging with artificial intelligence," Ph.D. dissertation, University of Groningen, 2020.

[4] L. Baldassini, "Hydrodynamic imaging of swimmers using an artificial lateral line and machine learning," Master's thesis, University of Groningen, 2020.

[5] H. Bleckmann and R. Zelick, "Lateral line system of fish," *Integrative Zoology*, vol. 4, no. 1, pp. 13–25, 2009.

[6] S. M. van Netten and M. J. McHenry, *The Biophysics of the Fish Lateral Line*. New York, NY: Springer New York, 2014, pp. 99–119.

[7] H. Münz, "Functional organization of the lateral line periphery," in *The Mechanosensory Lateral Line*, S. Coombs, P. Görner, and H. Münz, Eds. New York, NY: Springer New York, 1989, pp. 285–297.

[8] A. Flock and J. Wersäll, "A study of the orientation of the sensory hairs of the receptor cells in the lateral line organ of fish, with special reference to the function of the receptors," *The Journal of Cell Biology*, vol. 15, pp. 19 – 27, 1962.

[9] J. Engelmann, W. Hanke, and H. Bleckmann, "Lateral line reception in still- and running water," *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, vol. 188, pp. 513–26, 09 2002.

[10] B. F. Kingsbury, "The lateral line system of sense organs in some american amphibia, and comparison with the dipnoans," *Transactions of the American Microscopical Society*, vol. 17, pp. 115–154, 1896.

[11] A. Sutterlin and S. Waddy, "Possible role of the posterior lateral line in obstacle entrainment by brook trout (salvelinus fontinalis)," *Journal of the Fisheries Research Board of Canada*, vol. 32, pp. 2441–2446, 04 2011.

[12] J. Montgomery, C. Baker, and A. G. Carton, "The lateral line can mediate rheotaxis in fish," *Nature*, vol. 389, pp. 960–963, 1997.

[13] S. Coombs and H. Bleckmann, *The Gems of the Past: A Brief History of Lateral Line Research in the Context of the Hearing Sciences*. New York, NY: Springer New York, 2014, pp. 1–16.

[14] Z. Fan, J. Chen, J. Zou, D. Bullen, C. Liu, and F. Delcomyn, "Design and fabrication of artificial lateral line flow sensors," *Journal of Micromechanics and Microengineering*, vol. 12, no. 5, pp. 655–661, jun 2002.

[15] B. J. Wolf, J. A. S. Morton, W. N. MacPherson, and S. M. van Netten, "Bio-inspired all-optical artificial neuromast for 2d flow sensing," *Bioinspiration & Biomimetics*, vol. 13, no. 2, p. 026013, feb 2018.

[16] F. M. Yaul, V. Bulovic, and J. H. Lang, "A flexible underwater pressure sensor array using a conductive elastomer strain gauge," in *2012 IEEE 25th International Conference on Micro Electro Mechanical Systems (MEMS)*, 2012, pp. 500–503.

[17] M. N. M. Nawi, A. A. Manaf, M. R. Arshad, and O. Sidek, "Review of mems flow sensors based on artificial hair cell sensor," *Microsystem Technologies*, vol. 17, pp. 1417–1426, 2011.

[18] "Consortium: Lakhsmi-project."

[19] K. Hill and G. Meltz, "Fiber bragg grating technology fundamentals and overview," *Journal of Lightwave Technology*, vol. 15, no. 8, pp. 1263–1276, 1997.

[20] Y. Yang, J. Chen, J. Engel, S. Pandya, N. Chen, C. Tucker, S. Coombs, D. L. Jones, and C. Liu, "Distant touch hydrodynamic imaging with an artificial lateral line," *Proceedings of the National Academy of Sciences*, vol. 103, no. 50, pp. 18 891–18 895, 2006.

[21] S. Pandya, Y. Yang, D. Jones, J. Engel, and C. Liu, "Multisensor processing algorithms for underwater dipole localization and tracking using mems artificial lateral-line sensors," *EURASIP J. Adv. Sig. Proc.*, vol. 2006, 12 2006.

[22] B. Ćurčić-Blake and S. M. van Netten, "Source location encoding in the fish lateral line canal," *Journal of Experimental Biology*, vol. 209, no. 8, pp. 1548–1559, 04 2006.

[23] Y. Yang, N. Nguyen, N. Chen, M. Lockwood, C. Tucker, H. Hu, H. Bleckmann, C. Liu, and D. Jones, "Artificial lateral line with biomimetic neuromasts to emulate fish sensing," *Bioinspiration & biomimetics*, vol. 5, p. 16001, 03 2010.

[24] L. Boulogne, B. Wolf, M. Wiering, and S. M. Netten, "Performance of neural networks for localizing moving objects with an artificial lateral line," *Bioinspiration & Biomimetics*, vol. 12, 07 2017.

[25] B. Wolf, S. Warmelink, and S. M. Netten, "Recurrent neural networks for hydrodynamic imaging using a 2d-sensitive artificial lateral line," *Bioinspiration & Biomimetics*, vol. 14, 06 2019.

[26] D. Bot, B. Wolf, and S. M. Netten, "The quadrature method: A novel dipole localisation algorithm for artificial lateral lines compared to state of the art," *Sensors*, vol. 21, p. 4558, 07 2021.

[27] R. Bouffanais, G. Weymouth, and D. Yue, "Hydrodynamic object recognition using pressure sensing," *Proceedings of the Royal Society A*, vol. 467, pp. 19–38, 01 2010.

[28] V. Fernandez, A. Maertens, F. Yaul, J. Dahl, J. Lang, and M. Triantafyllou, "Lateral-line-inspired sensor arrays for navigation and object identification," *Marine Technology Society Journal*, vol. 45, pp. 130–146, 07 2011.

[29] G. Liu, M. Wang, A. Wang, S. Wang, T. Yang, R. Malekian, and Z. Li, "Research on flow field perception based on artificial lateral line sensor system," *Sensors*, vol. 18, p. 838, 03 2018.

[30] B. J. Wolf, P. Pirih, M. Kruusmaa, and S. M. Van Netten, "Shape classification using hydrodynamic detection via a sparse large-scale 2d-sensitive artificial lateral line," *IEEE Access*, vol. 8, pp. 11 393–11 404, 2020.

[31] V. Camomilla, E. Bergamini, S. Fantozzi, and G. Vannozzi, "Trends supporting the in-field use of wearable inertial sensors for sport performance evaluation: A systematic review," *Sensors*, vol. 18, p. 873, 03 2018.

[32] V. Cerqueira, L. Torgo, and C. Soares, "Machine learning vs statistical methods for time series forecasting: Size matters," 2019.

[33] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning : A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, p. 106181, 2020.

[34] A. Oppenheim, R. Schafer, and J. Buck, *Discrete-time signal processing*, 2nd ed. Prentice Hall, 1999.

[35] M. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on Signal Processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

[36] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[37] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[38] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *KDD Workshop*, 1994.

[39] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *PVLDB*, vol. 1, pp. 1542–1552, 08 2008.

[40] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller, "Data augmentation using synthetic data for time series classification with deep residual networks," *CoRR*, vol. abs/1808.02455, 2018.

[41] D. Yu, X. Yu, Q. Hu, J. Liu, and A. wu, "Dynamic time warping constraint learning for large margin nearest neighbor classification," *Inf. Sci.*, vol. 181, pp. 2787–2796, 07 2011.

[42] H. Deng, G. C. Runger, E. Tuv, and V. Martyanov, "A time series forest for classification and feature extraction," *CoRR*, vol. abs/1302.2277, 2013.

[43] J. Rodríguez, C. Alonso, and H. Boström, "Boosting interval based literals," *Intell. Data Anal.*, vol. 5, pp. 245–262, 05 2001.

[44] P. Geurts, "Pattern extraction for time series classification," in *Principles of Data Mining and Knowledge Discovery*, L. De Raedt and A. Siebes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 115–127.

[45] C. H. Lubba, S. Sethi, P. Knaute, S. Schultz, B. Fulcher, and N. Jones, "catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis," *Data Mining and Knowledge Discovery*, vol. 33, 08 2019.

[46] M. Middlehurst, J. Large, and A. J. Bagnall, "The canonical interval forest (CIF) classifier for time series classification," *CoRR*, vol. abs/2008.09172, 2020.

[47] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *CoRR*, vol. abs/1909.04939, 2019.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[49] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 1041–1046.

[50] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[51] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[52] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

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

[54] M. Löning, T. Bagnall, M. Middlehurst, S. Ganesh, G. Oastler, J. Lines, ViktorKaz, F. Király, M. Walter, P. Rockenschaub, jesellier, T. Owoseni, Lovkush, AidenRushbrooke, oleskiewicz, Y.-X. Xu, P. Schäfer, RNKuhns, Hongyi, G. Bulatova, J. Large, A. Ansari, M. E. Gorelli, A. Wang, L. Zugasti, simone pignotti, S. M. Meyer, matteogales, A. Bostrom, and A. Seth, "alan-turing-institute/sktime: v0.7.0," Jul. 2021.

[55] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 69, pp. 331–371, 1910.

[56] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *CoRR*, vol. abs/1609.04836, 2016.

[57] A. Bostrom and A. Bagnall, "A shapelet transform for multivariate time series classification," 2017.

[58] D. Li, T. F. Bissyandé, J. Klein, and Y. L. Traon, "Time series classification with discrete wavelet transformed data: Insights from an empirical study," in *SEKE*, 2016.

[59] J. E. Mietus, C.-K. Peng, I. Henry, R. L. Goldsmith, and A. L. Goldberger, "The pnnx files: re-examining a widely used heart rate variability measure," *Heart*, vol. 88, no. 4, pp. 378–380, 2002.

[60] X. Wang, A. Wirth, and L. Wang, "Structure-based statistical features and multivariate time series clustering," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 351–360.

Appendix A

Appendix

In this Appendix we provide some supplemental explanations of our methods as well as additional results generated during the evaluation are provided. These serve to complete the results present in the results section.

A.1 Catch22 features

We start this appendix with an overview of the catch22 features as described in the paper by Lubba et al. [45]. These were used by the time series forest approach as described in section 3.4.2.

#	**Feature description**
1	Mode of z-scored distribution (5-bin histogram)
2	Mode of z-scored distribution (10-bin histogram)
3	Longest period of consecutive values above the mean
4	Time intervals between successive extreme events above the mean
5	Time intervals between successive extreme events below the mean
6	First $1/e$ crossing of autocorrelation function
7	First minimum of autocorrelation function
8	Total power in lowest fifth of frequencies in the Fourier power spectrum
9	Centroid of the Fourier power spectrum
10	Mean error from a rolling 3-sample mean forecasting
11	Time-reversibility statistic, $\langle (xt+1-xt)^3 \rangle_t$
12	Automutual information, m=2,τ=5
13	First minimum of the automutual information function
14	Proportion of successive differences exceeding 0.04σ [59]
15	Longest period of successive incremental decreases
16	Shannon entropy of two successive letters in equiprobable 3-letter symbolization
17	Change in correlation length after iterative differencing
18	Exponential fit to successive distances in 2-d embedding space
19	Proportion of slower timescale fluctuations that scale with DFA (50% sampling)
20	Proportion of slower timescale fluctuations that scale with linearly rescaled range fits
21	Trace of covariance of transition matrix between symbols in 3-letter alphabet
22	Periodicity measure of [60]

Table A.1: Overview of the catch22 feature set with names and descriptions. The features are seperated into categories given by Lubba et al. Adapted from [45].

		n_neighbours	split1_acc	split2_acc	split3_acc	mean train acc	std Dev
raw	5000 Hz	1	0.953488	0.97619	0.97619	0.968623	0.013107
		10	0.930233	0.928571	0.952381	0.937062	0.013293
	500 Hz	1	1	0.97619	0.97619	0.984127	0.013747
		10	0.953488	0.880952	0.904762	0.913067	0.036974
	100 Hz	1	1	1	1	1	0
		10	0.976744	0.952381	0.928571	0.952565	0.024087
spectrogram	5000 Hz	1	0.418605	0.333333	0.428571	0.393503	0.052346
		10	0.27907	0.309524	0.333333	0.307309	0.027199
	500 Hz	1	0.55814	0.47619	0.380952	0.471761	0.088677
		10	0.511628	0.571429	0.404762	0.49594	0.084434
	100 Hz	1	0.534884	0.5	0.452381	0.495755	0.041415
		10	0.325581	0.5	0.309524	0.378368	0.105642
Wavelet	5000 Hz	1	0.302326	0.309524	0.309524	0.307125	0.004156
		10	0.27907	0.309524	0.309524	0.299373	0.017583
	500 Hz	1	0.488372	0.52381	0.452381	0.488188	0.035715
		10	0.27907	0.285714	0.285714	0.283499	0.003836
	100 Hz	1	0.534884	0.642857	0.428571	0.535437	0.107144
		10	0.348837	0.309524	0.285714	0.314692	0.031877

Table A.2: Complete cross validation results of the nearest neighbours algorithm

		n_estimators	split1_acc	split2_acc	split3_acc	mean train acc	std Dev
raw	5000 Hz	50	0.93	0.98	0.95	0.95	0.02
		100	0.95	1.00	0.98	0.98	0.02
		200	0.95	1.00	0.98	0.98	0.02
	500 Hz	50	0.98	1.00	0.98	0.98	0.01
		100	1.00	1.00	1.00	1.00	0.00
		200	1.00	1.00	1.00	1.00	0.00
	100 Hz	50	1.00	1.00	0.98	0.99	0.01
		100	1.00	1.00	0.98	0.99	0.01
		200	1.00	1.00	1.00	1.00	0.00
spectrogram	5000 Hz	50	0.53	0.69	0.69	0.64	0.09
		100	0.56	0.83	0.67	0.69	0.14
		200	0.65	0.86	0.69	0.73	0.11
	500 Hz	50	0.21	0.52	0.48	0.40	0.17
		100	0.40	0.55	0.55	0.50	0.09
		200	0.47	0.64	0.52	0.54	0.09
Wavelet	5000 Hz	50	1.00	0.95	0.93	0.96	0.04
		100	1.00	0.98	0.90	0.96	0.05
		200	1.00	0.98	0.90	0.96	0.05
	500 Hz	50	0.95	0.95	0.93	0.94	0.01
		100	0.98	0.95	0.95	0.96	0.01
		200	0.98	0.95	0.93	0.95	0.02
	100 Hz	50	0.77	0.88	0.79	0.81	0.06
		100	0.77	0.79	0.88	0.81	0.06
		200	0.81	0.88	0.71	0.80	0.08

Table A.3: Complete cross validation results of the time series forest algorithm

	depth	kernel_size	nb_filters	split1_acc	split2_acc	split3_acc	Mean acc
5000 Hz	10	50	16	0.79	1.00	0.93	0.91
	10	50	32	0.81	0.90	0.93	0.88
	10	100	16	0.79	0.90	0.93	0.87
	10	100	32	0.79	0.90	0.93	0.87
	15	50	16	0.79	0.90	0.93	0.87
	15	50	32	0.79	0.90	0.93	0.87
	15	100	16	0.79	0.90	0.93	0.87
	15	100	32	0.79	0.90	0.93	0.87
500 Hz	10	50	16	0.81	0.90	0.93	0.88
	10	50	32	0.81	0.90	1.00	0.91
	10	100	16	0.81	0.98	0.98	0.92
	10	100	32	0.88	0.95	0.93	0.92
	15	50	16	0.77	0.90	0.93	0.87
	15	50	32	0.79	0.90	0.93	0.87
	15	100	16	0.81	1.00	1.00	0.94
	15	100	32	0.84	0.90	1.00	0.91
100 Hz	10	50	16	0.81	0.95	0.95	0.91
	10	50	32	0.81	0.95	0.95	0.91
	10	100	16	0.81	0.90	0.98	0.90
	10	100	32	0.81	0.90	0.98	0.90
	15	50	16	0.81	0.88	0.88	0.86
	15	50	32	0.81	0.90	0.98	0.90
	15	100	16	0.81	0.90	0.98	0.90
	15	100	32	0.81	0.90	0.93	0.88

Table A.4: full cross validation of the raw data for the InceptionTime network

	depth	kernel_size	nb_filters	split1_acc	split2_acc	split3_acc	Mean acc
5000 Hz	10	50	16	0.26	0.33	0.31	0.30
	10	50	32	0.26	0.33	0.31	0.30
	10	100	16	0.26	0.33	0.31	0.30
	10	100	32	0.26	0.33	0.31	0.30
	15	50	16	0.26	0.33	0.31	0.30
	15	50	32	0.26	0.33	0.31	0.30
	15	100	16	0.26	0.33	0.31	0.30
	15	100	32	0.26	0.33	0.31	0.30
500 Hz	10	50	16	0.26	0.33	0.31	0.30
	10	50	32	0.26	0.33	0.31	0.30
	10	100	16	0.26	0.33	0.31	0.30
	10	100	32	0.26	0.33	0.31	0.30
	15	50	16	0.26	0.33	0.31	0.30
	15	50	32	0.26	0.33	0.31	0.30
	15	100	16	0.26	0.33	0.31	0.30
	15	100	32	0.26	0.33	0.31	0.30
100 Hz	10	50	16	0.26	0.33	0.31	0.30
	10	50	32	0.26	0.33	0.31	0.30
	10	100	16	0.26	0.33	0.31	0.30
	10	100	32	0.26	0.33	0.31	0.30
	15	50	16	0.26	0.33	0.31	0.30
	15	50	32	0.26	0.33	0.31	0.30
	15	100	16	0.26	0.33	0.31	0.30
	15	100	32	0.26	0.33	0.31	0.30

Table A.5: full cross validation of the spectrogram data for the InceptionTime network

	depth	kernel_size	nb_filters	split1_acc	split2_acc	split3_acc	Mean acc
5000 Hz	10	50	16	1.00	1.00	1.00	1.00
	10	50	32	1.00	1.00	1.00	1.00
	10	100	16	1.00	1.00	1.00	1.00
	10	100	32	1.00	1.00	1.00	1.00
	15	50	16	0.98	1.00	0.98	0.98
	15	50	32	0.98	1.00	1.00	0.99
	15	100	16	1.00	1.00	1.00	1.00
	15	100	32	0.98	1.00	1.00	0.99
500 Hz	10	50	16	0.91	0.90	0.88	0.90
	10	50	32	0.93	0.95	0.98	0.95
	10	100	16	0.86	0.90	0.93	0.90
	10	100	32	0.81	0.88	0.90	0.87
	15	50	16	0.88	0.90	0.93	0.91
	15	50	32	0.88	0.98	0.93	0.93
	15	100	16	0.81	0.88	0.93	0.87
	15	100	32	0.88	0.86	0.88	0.87
100 Hz	10	50	16	0.58	0.86	0.71	0.72
	10	50	32	0.72	0.86	0.74	0.77
	10	100	16	0.51	0.79	0.83	0.71
	10	100	32	0.63	0.76	0.60	0.66
	15	50	16	0.51	0.79	0.74	0.68
	15	50	32	0.81	0.71	0.69	0.74
	15	100	16	0.40	0.71	0.69	0.60
	15	100	32	0.65	0.60	0.57	0.61

Table A.6: full cross validation of the wavelet data for the InceptionTime network