UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

The Development of a Horse Activity Recognition Algorithm

Maartje Huveneers

Bachelor Thesis
July 2021

Supervisors:
Hamed Darbandi
Jacob Kamminga

Creative Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

Abstract

Animal activity recognition is a growing field of research, which can help with the monitoring of wildlife and their habitats. For portability and accuracy, the monitoring can be done using devices that contain sensors, such as an Inertial Measurement Unit (IMU). To recognize the activities that the animals are performing from the extracted data from the IMU, a well-performing algorithm is needed. Thus, the following research question has been researched in this report: *Which animal activity recognition algorithm performs the best on IMU horse data, and what aspects lead it to outperform other algorithms?* To answer this question, a pipeline was developed with the IMU horse data as input and analyzed using various deep learning algorithms.

The used dataset comprises labeled data of six horses and six activities (walking with a rider, walking natural, trotting, grazing, standing and running). The data was collected from an IMU (including an accelerometer and gyroscope) mounted on the collar of the horses while they were performing different activities. Before training the deep learning models, the data was filtered, normalized and windowed. Then, the windows were split into training, cross-validation and testing sets. Three deep learning algorithms were used: a deep Neural Network (NN), a Long Short-Term Memory (LSTM) and a Multivariate Long Short-Term Memory Fully Convolutional Network (MLSTM-FCN), based upon the state of the art. The hyperparameters (number of layers and cells per layer) of each algorithm were tuned through a grid search. All testing was performed through leave one subject out validation. Additionally, each classifier was trained and tested with different combinations of sensor data and generated features, hereafter referred to as input set, to see which combination yielded the highest performance. Input set 1 consisted of 2-second windows of 6 signals, which were raw x, y and z signals of the accelerometer and gyroscope, input set 2 consisted of the l2-norms of the accelerometer and gyroscope and input set 3 consisted of the l2-norms of the accelerometer and gyroscope, along with the Root Mean Square value of the l2-norms. The NN yielded the highest performance with input set 1 and 3. The LSTM and MLSTM-FCN, however, yielded the highest performance with input set 2 and 3. Overall, using input set 3 yielded high performance results among the NN, LSTM and MLSTM-FCN. Afterwards, the three classifiers were compared with each other, each with input set 2 as their input. In conclusion, the LSTM (87.8% accuracy, 88.4% F-score) and MLSTM-FCN (88.5% accuracy, 87.6% F-score) yielded a higher performance than the NN (82.8% accuracy, 82.3% F-score) on all metrics (accuracy, balanced accuracy, F-score and MCC) and are thus recommended to use in future animal activity recognition projects.

Acknowledgements

I want to thank both of my supervisors for the time and effort they put into my thesis. First, my main supervisor, Hamed Darbandi, was always available to help, taught me more about academic writing and was always positive. Second, my critical supervisor, Jacob Kamminga, has taught me a lot about animal recognition, but also about thesis writing in general. Third, I want to mention the pleasant collaboration with Suzanne Spink, Kevin Folkertsma, Rosalie Voorend and Wannes Vanwinsen to create the animal activity recognition pipeline. Last, I want to thank all friends and family who supported me during my graduation project.

Contents

List of Figures

List of Tables

1 Introduction

Wild animals are threatened by human activities, such as the destruction of their habitats, illegal hunting and climate change. Worldwide, approximately 50% of all mammal populations are declining and 25% are facing extinction [1]. To ensure the survival of these species, biologists need to know more about the background of the issues and the well-being of these animals. Animal activity recognition is one of the methods that helps biologists to gather information about the animal's well-being and social behaviour and to improve their welfare [2]–[4]. Thankfully, tracking devices for animals have become increasingly better and more available. These tracking devices can comprise of different kinds of components, such as a Global Positioning System (GPS) sensor, camera and Inertial Measurement Unit (IMU), the latter consisting of an accelerometer, gyroscope and magnetometer [3].

1.1 Problem Definition

Tracking devices can deliver biochemical or physiological patterns of an animal to biologists. However, raw data from, for example, an IMU, will not provide them with any insights since they lack the knowledge to extract the right information from it. Thus, a system is needed to transfer this knowledge by an implementation on the data. This can, for example, be a conversion of the raw data from the IMU into the activities performed by the animals. This system consists of multiple steps, including an algorithm which can learn to recognize the right activity from the data. This data consists of animal activity measurements, along with a label, which specifies the performed activity at the time of the measurement. The algorithm must be able to accurately classify the activities from offline labeled data, assuming that the biologists have measured the data beforehand. Achieving a high recognition rate is very important since wrong predictions could lead to wrong conclusions from the data, which could cause problems for the animals involved.

1.2 Objective

This study aims to compare multiple activity recognition algorithms to see the impact of each algorithm on the recognition performance on the IMU data of horses and investigate which algorithm is the best-performing algorithm. Thus, the main research question is defined as follows:

Which animal activity recognition algorithm performs the best on IMU horse data, and what aspects lead it to outperform other algorithms?

To answer the main research question, two sub-questions have to be answered first:

Which animal and human activity recognition algorithms that have been developed in the last two years are the most promising to utilize for a horse activity recognition algorithm using IMU data?

What are the requirements for an animal activity recognition algorithm that can classify multiple activities with an adequate performance?

1.3 Approach

There are many different algorithms to use for recognizing animal activities and to choose the best, first a study will be conducted on algorithms used within the field of activity recognition to define the state of the art algorithms. Since human activity recognition (HAR) algorithms are generally quite similar to animal activity recognition (AAR) algorithms, and since HAR is a more evolved field of study, also the HAR studies will be considered within the state of the art. With this study, the most promising algorithms will be retrieved from the state of the art algorithms. With this, the subquestion *"Which animal and human activity recognition algorithms that have been developed in the last two years are the most promising to utilize for a horse activity recognition algorithm using IMU data?"* will be answered. Apart from that, a study will be conducted on the aspects that lead to different performance results. This study will help in answering the main research question on why an algorithm outperforms another algorithm. Afterwards, a requirement analysis will be performed, where requirements are written down, which will answer the subquestion *"What are the requirements for an animal activity recognition pipeline that can classify multiple activities with an adequate performance?"*. Furthermore, an activity recognition pipeline was developed with three Creative Technology students and one Interaction Technology student. Within this pipeline, three algorithms, based on the state of the art, were developed. The data used stems from a dataset measured by an Inertial Measurement Unit (IMU) on the collar of horses. Lastly, the developed algorithms will be compared based on their performance on the horse data and the causes for the differences in performance are further elaborated upon to answer the main research question.

1.4 Organization

First, machine learning definitions in AAR are explained in Chapter 2. Moreover, a number of machine learning algorithms in AAR and HAR is discussed. Afterwards, a study is presented on the state of the art activity recognition algorithms using accelerometer data in Chapter 3, in which a recommendation is given based on the performances of the algorithms in the studies. Next to that, Chapter 3 also answers the first sub-research question. In Chapter 4 the methodology for the development of the classifiers is explained, along with the settings and experiments performed in Chapter 6. Before the classifier is developed, requirements are set up in Chapter 5, which also answers the second sub-research question. In Chapter 6 an experiment is performed to see the influence of different input sets on the performance of each of the developed algorithms. Additionally, the classifiers are compared to each other in terms of performance. Lastly, conclusions on the research (sub-)questions are drawn in Chapter 7 and further improvements are discussed in Chapter 8.

2 Background Research

2.1 Human and animal activity recognition

As mentioned in Chapter 1, apart from animal activity recognition (AAR), also human activity recognition (HAR) is a rapidly evolving field of study. Since both fields often use IMU data, the algorithms used are often similar. However, there are some differences to keep in mind when evaluating AAR and HAR studies.

HAR studies are often restricted to using sensor data from appliances that people already use, such as smartphones and smartwatches, and cannot simply design a collar as humans will most likely not buy such a product. Next to that, the placement of the sensors also often differ. Even though both AAR and HAR studies both often use the leg as sensor location [5]–[10], other sensor locations are generally different. Where AAR studies also often use the neck [11]–[13] as sensor location, HAR studies use the arm [6], [9], [10], [14], wrist [10], [15]–[18], chest [7]–[9], [14] and/or waist [9], [10], [18]–[20]. Apart from that, HAR studies generally use more sensor locations in their studies than AAR studies, which can also be seen in Table 2. Even though these issues are slightly different between AAR and HAR, often similar algorithms are applied, making HAR also a relevant field to study. Moreover, more research has been done in the field of HAR than AAR, thus the most advanced algorithms might be found in the field of HAR.

2.2 Supervised, Semi-Supervised and Unsupervised Learning

Supervised, semi-supervised and unsupervised learning are types of machine learning algorithm, where each serves a different purpose. Supervised learning is a type of machine learning that uses fully-labeled data. Thus, all inputs map to a certain output. The goal of a supervised machine learning approach is to label all unseen inputs correctly. In unsupervised learning, however, a machine learning algorithm learns from unlabeled data. This is done by grouping similar inputs together and focusing on patterns instead of predicting a right output value. One of the goals of an unsupervised learning approach is to find groups of data that belong together, but it will not be able to assign the right label to it. Another goal of unsupervised learning, where it is also unable to assign a label, but instead groups datapoints together, is to recognize outliers in the data. In semi-supervised learning these two types are combined, where part of the data is labeled and another part is unlabeled. Semi-supervised learning systems use the unlabeled data to aid the learning process of the labeled data, making it able to reach the same results as a supervised learning system with less labeled samples [21]. In conclusion, supervised learning algorithms learn to classify from fully-labeled data, unsupervised learning algorithms learn to find patterns in unlabeled data and semi-supervised learning algorithms learn to find and label patterns in partly labeled data.

2.3 Online and offline systems

Machine learning systems can consist of two types: online and offline systems. In an online system the classifier performs while the activities are performed and then saves the data locally. On the other hand, for offline systems, the classification performed after data collection. The downside of offline systems is that all raw data needs to be stored on the tracking device until the tracking device is retrieved for data processing. This is especially a problem due to memory constraints. Apart from that, online systems allow the system to adapt to the activities of the animals. For example, when the animal is sleeping, the system can also go in a stand-by mode to save resources [22].

2.4 Online and offline learning

Machine learning can be done either online or offline. In online learning, the classifier learns from the data while the data is being gathered. On the other hand, in offline learning, the classifier learns from a dataset that has been gathered previously. While online learning has to be very time-efficient, offline learning has the advantage that it does not suffer from time constraints. Apart from that, it can be a challenge to also label the data immediately when it becomes available in online learning. On the other hand, online learning does allow the algorithm to adapt to changes in the data [22].

2.5 Overfitting and underfitting

A problem that often occurs in machine learning algorithms is overfitting and underfitting. Overfitting occurs when the algorithm has fit overly to the training data, making it unable to appropriately generalize it to the test data. An example of this can be seen in Figure 1. On the other hand, underfitting occurs when the algorithm is not complex enough to calculate an accurate solution for the problem. The result of underfitting is that the algorithm does not perform well on the training data and on the test data as well [23]. A method that helps with preventing overfitting and underfitting is using a validation set, which consists of part of the dataset. When using a validation set, the hyperparameters in the algorithm can be tuned with a range in complexity. After that, the performance can be checked on each of these settings and the best performing algorithm can be chosen. Next to using a validation set, other methods are available which can prevent overfitting. One method is called "early-stopping", which is based upon the fact that the algorithm stops learning after some point. At this point, the algorithm should be stopped, because continuing may lead to overfitting. However, determining the point of stopping is hard, since stopping too early may lead to underfitting. Next to that, there are also multiple methods for noise reduction, which may lead to overfitting. Lastly, other methods are based on the expansion of the training data, where either more data is acquired, or noise is added to it or new data is produced based on the training set [24]. In conclusion, overfitting and underfitting are often-occuring machine learning problems that can be minimized by various methods.

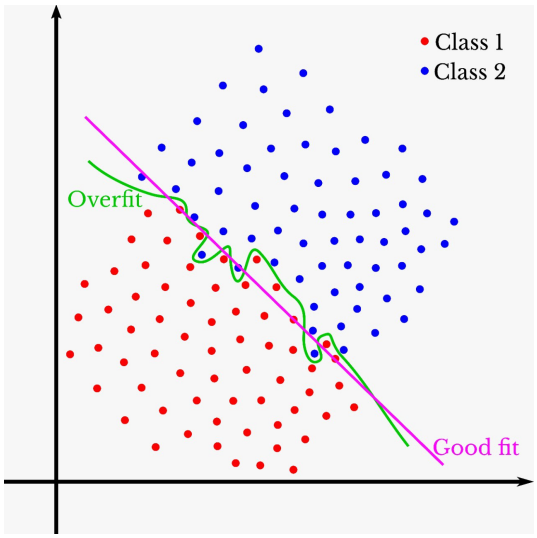


Figure 1: A solution by an algorithm with and without overfitting. [25]

2.6 Factors influencing an AAR pipeline's performance

There are many factors that can influence the performance of an AAR pipeline. In this section, a couple of these factors will be elaborated upon.

2.6.1 Number of subjects

The number of subjects in a training set can cause a difference in the performance of an algorithm. This is due to the fact that having a small number of subjects will increase the bias towards those subjects. For example, when training an algorithm on only one horse, this will lead to a biased algorithm, which has become very accustomed with how that horse performs all movements. However, when testing that same algorithm on other horses, it will probably not perform well, because it is only accustomed to the one training horse. Thus, the number of subjects in a training set can influence the performance of an algorithm.

2.6.2 Size of the dataset

The size of the dataset can also influence the performance of an algorithm. Multiple studies [26]–[30] have shown that the performance of an algorithm can be positively influenced by using a larger dataset. It is especially important to have a minimal amount of samples, as a higher number of samples than required will not negatively influence the performance of the algorithm. However, it may make the algorithm slower due to the overload of data. Conclusively, having a large dataset will positively influence the performance results of an algorithm.

2.6.3 Dataset imbalance

Another issue in the dataset size is class imbalance, where certain classes contain significantly less samples than other classes. This imbalance often causes a bias towards the classes that have many samples, resulting in a higher misclassification rate for classes with a low number of samples [31]. In conclusion, when the dataset is imbalanced, this may cause a bias towards certain classes, decreasing the performance of the classifier.

2.6.4 Sensor location and number of sensors

The sensor position influences the performance of an algorithm. For example when measuring whether a horse is wagging their tail or not, a sensor placed on the leg will not be sufficient for recognizing this behaviour. Thus, the sensor location, in relation to the activities to be recognized, is very important for the performance of the algorithm. Apart from that, there is also a difference between fixed and non-fixed sensors. Fixed sensors stay in place, without rotating, while non-fixed sensors can rotate and shift around the place they are attached. Additionally, when sensors are fixed their orientation is known. For example, a horse's collar may rotate around the neck, which will make it harder to recognize activities from the data. Apart from that, the number of sensors (in different locations) may affect the performance of an algorithm. When using sensors in different locations, this will increase the possibilities of having sufficient information about activities that are similar to each other. Thus, the location, number of sensors and whether they are fixed or not can influence the performance of the algorithm.

2.6.5 Number of activities

Evidently, the number of activities to be separated influences the performance of an algorithm. When an algorithm is made to recognize a high number of different activities, this is defined as fine-grained activity recognition, while recognizing a low number of activities is defined as coarse-grained activity recognition. With fine-grained activity recognition the chance is greater that there are activities that bring similar results. On the other hand, with coarse-grained activities, the algorithm can focus specifically on features that can decide upon those activities well. Thus, fine-grained activity recognition probably result in a lower performance score than coarse-grained activity recognition.

2.6.6 Train/Test-Split

All algorithms need some data to be trained on and some data to be tested on. The data from training and testing may not overlap, since that would cause unfair performance evaluation, called information leakage. When an algorithm is trained on certain data, it becomes accomplished with that data and it is taught what that data means. If the data appears in the test phase, leakage occurs. When leakage occurs, the chances are much higher that the algorithm will recognize this data than when new data is inserted. Therefore, it is

needed to split up the dataset in order to get a training dataset and test dataset, which do not overlap. There are multiple methods for splitting up that data, which are described in the following paragraph.

First, an often-used method is Single Split, in which a certain percentage of the data is randomly chosen to become the train data and a certain percentage is the test data [17], [18], [20], [32]–[36]. Secondly, in the leave one subject out method one (or multiple) subjects are chosen to be the test dataset and the others the train dataset [5], [10], [37]–[42]. In this method, the subjects chosen for the test dataset are switched with the subjects that are chosen for the train dataset, and the evaluation is performed multiple times. This allows the dataset to be used multiple times, making the performance results more complete. Apart from that, the Leave One Out method also aids in assessing how well the algorithm generalizes towards unseen subjects. Thus, the two most-used splitting methods for the train and test data are the Single Split and Leave One Out methods, where the Leave One Out method gives a more complete score on how well the algorithm performs on unseen subjects.

2.6.7 Other steps in the pipeline

Apart from the algorithm itself, there are also other steps in the AAR pipeline: the preprocessing and segmenting of the data and the feature extraction and selection. These steps are crucial in building a AAR pipeline with adequate performance, since they will perform the operations to feed the data in the right format into the algorithm. Evidently, when feeding the wrong type of data, the algorithm will also perform poorly. The preprocessing is one of these steps.

Preprocessing Apart from the algorithm itself, there are also other steps in the AAR pipeline, of which preprocessing is the first step. Within preprocessing, the data is adjusted so that it contains only valid values, meaning there are no more out-of-range or missing values. Apart from that, non-relevant data might also be removed from the dataset. This may be the case when removing unlabeled data in a study on fully-labeled data only. Apart from that, outliers may also be removed from the dataset during preprocessing. Lastly, methods may be performed to tackle imbalanced datasets, where some classes contain significantly more samples than other classes. Through these methods, preprocessing helps to prepare the data for the next steps. Without preprocessing the performance of the pipeline may decrease.

Segmenting Segmenting is the process of dividing the data up into segments, where each segment represents one continuous activity. This means that each window may be of a different size. Segmenting can be a hard process, because there is often no hard line within the performance of two activities. This is because activities often transition into one another. There are multiple methods for segmentation, including using a sliding window, energy-based segmentation, and using data from external sources. A fixed-width sliding window is a commonly used approach, where a window with a fixed size is moved over the data to

divide it up into segments that are all of an equal size. The step size of moving the window and the window size may differ, which will both influence the performance of the pipeline. When the step size is small, it will increase the computational load, while a big step size will decrease the performance of the pipeline. Next to that, a big window size may have a greater error rate when a transition is included in the window. A small window size may prevent the pipeline from seeing a full activity when the time of performing the activity is longer than the window size.

Energy-based segmentation uses the fact that activities often differ in intensity, and thus, also energy levels in the sensor measurements. This method uses a threshold for which segments can be defined that are likely part of the same activity. Next to that, segmentation based on additional data from external sources may be used. This method uses other sensors to provide the necessary information to segment the data. For example, GPS data may be incorporated to segment between standing and walking activities in AAR. However, often this external data is not available, making it unable to be used in a pipeline [43].

Feature Extraction and Selection A feature is "a noticeable or important characteristic or part" [44], but in the context of machine learning more specifically: a value extracted from a piece of data to characterize this piece of data. An example feature would be the mean value over a certain time frame, which characterizes that time frame. There are three types of features: features in the time-domain, frequency-domain or a combination of the two. The advantage of using time-domain features is that it costs less computational power and time. On the other hand, frequency-domain features can reach a high accuracy, but they are more computationally heavy [34]. Multiple studies [13], [34], [41] use a combination of the two types of features.

The approach for deciding upon the features differs per study. One method is simply defining many features and then reducing the number of features without losing useful information with Principal Component Analysis (PCA) [13]. Apart from that, a correlation coefficient may be calculated for all features. When the feature has a high correlation with a certain label, but low correlation with all other features, it is considered a useful feature [18], [37]. Apart from doing feature selection by hand, there are also possibilities to let algorithms themselves handle this step. When using a neural network, it can effectively learn from the raw data, so it will implicitly do feature extraction and selection in itself, without explicitly doing so. Letting an algorithm decide upon the features may also lead to better results than calculating the features by hand [10].

2.7 Machine Learning Algorithms

Within the section of activity recognition there are some algorithms that are used often, although studies generally still apply small changes to these algorithms. The most well-known algorithms used in the field of activity recognition will be discussed in this section.

2.7.1 Chaos Theoretic

While the Chaos-Theoretic (CT) is not a well-known approach, one of the surveyed studies involving animal activity recognition [37] used such a dynamical system approach, thus it is worth evaluating. CT assumes that the data used is chaotic, in this case meaning that the nature of the animals' behaviour is chaotic, thus being unpredictable over the long term [45]. They are based on dynamical systems, which are systems that "display nonlinear behavioral changes over time" [46]. However, CT assumes these seemingly random movements actually come from underlying patterns. It is, however, unknown whether animals also perform the behaviour that is described by chaotic systems.

2.7.2 Naive Bayes

Naive Bayes (NB) is a relatively simple classifier using Bayes' rule to estimate the probability of the classification of a certain activity. Bayes' rule is defined as:

$$P(y|x) = \frac{P(y) * P(x|y)}{P(x)} \tag{1}$$

In NB, x stands for an unlabeled data point and y stands for the label. This probability is calculated for all possible classes, after which the label y with the highest probability is chosen as classification for x [47]. In NB, Bayes' rule goes together with the assumption that all features are conditionally independent given the class. However, often this is actually not the case, since features are often dependent on each other. Even though this assumption is often false, the model is generally an easy fit and still works well. One of the advantages of using NB is that the probabilities can be updated later on if more data becomes available. Apart from that, it is relatively insensitive to noise [48]. Thus, NB uses probabilities to classify the data points, making it relatively insensitive to noise.

2.7.3 Decision Tree

A Decision Tree (DT) is a representation of the outcomes in the form of a tree, where each leaf represents a possible label and all other nodes represent the decision variables. The DT is especially based on an If-Then principle, in which the variables are compared to certain values and if they are high or low enough, a choice is made [49]. A simple example of this would be: if the mean of the accelerometer data is below a certain value, the horse is either standing or eating. The advantages of using a DT are that they are quite robust to outliers and errors in labels, they are easily interpretable and require a small computation time [50]. The disadvantages of the DT are that they are unable to produce multiple outcomes, they are often unstable, as even slight variations in the training data can already result in different classifications in the test data [51]. Thus, DT is a simple classification method which is quite robust to outliers, easily interpretable and requires a small computation time, but is also often unstable.

Bagged Decision Tree A Bagged Decision Tree is a Decision Tree that uses bagging to increase the accuracy of the DT algorithm. Bagging is a method through which parts of the dataset are selected randomly. In a Bagged DT, these random subsets each form their own DT and predict their results. After each DT has made a prediction, the most-often chosen prediction from all DTs is chosen as the overall classification of the Bagged DT. The advantage of this method is that it solves part of the stability problems that the DT suffers from, making it a more robust solution. The disadvantage is that it removes the property of easy visualization and interpretation that the DT has [52]. Thus, the Bagged DT is an adaption to the classic DT, which leads to an improved performance.

Random Forest The Random Forest (RF) algorithm can be seen as an advancement to the Bagged Decision Tree. RF also creates a collection of decision trees, which will classify an instance by the majority of votes. However, at each node in the tree a certain amount of features are selected randomly, after which the feature that provides the best split is chosen to perform the split on that node. The next node then again chooses a certain amount of features randomly and so on [53]. This is different with a Bagged DT in the sense that those algorithms use all features for each node [50]. Although each tree is generally classified as a weak learner, all trees together form a strong learner [54]. The advantage of using RF is that they intrinsically implement feature extraction, making them more robust to noise. Apart from that, they are also quite robust to outliers. [50]. In conclusion, the RF is an adapted version of DT, where RF is more robust to noise and outliers than DT.

Gradient Boosted Tree The Gradient Boosted Tree (GBT) is an adapted version of the Decision Tree with an iterative training process. In this training process, each iteration tries explain the errors from the last iteration. Through this process, the classifier learns to understand all previously occurred errors, making the classifier perform better each iteration [55]. The advantage of the GBT is that it is adaptable, easily interpretable and produces high-performance results. Next to that, they are less prone to overfitting than normal Decision Trees. However, the GBT are very computationally expensive and consume a lot of memory [56]. Conclusively, the Gradient Boosted Tree learns from the errors of previous iterations, making it produce high-performance results, but also computationally expensive.

2.7.4 Hidden Markov Model

The Hidden Markov Model (HMM) is based on the Markov chain, which is a chain of states, in which the next state depends on the current state only [57]. In the HMM these states are hidden. The HMM is based upon the fact that the system is always in one of the available states and on a certain time interval the system switches from that state to another state. The switching to a new state has a probability p. Apart from switching states, the states can also predict an outcome at any time. The HMM is especially known for temporal pattern

recognition, such as speech recognition, handwriting, musical scores, et cetera [54]. Thus, the HMM is based on a chain of states, which allows it to calculate probabilities for each possible outcome.

2.7.5 Support Vector Machine

The Support Vector Machine (SVM) is a learning method based on the recognition of patterns. The SVM first maps all inputs and their class as points in a space, after which it uses the principle of decision planes that define the boundary between two classes [58]. It finds an optimal hyperplane, which tries to divide the two classes into two groups, with the biggest distance between the nearest data points of both classes and the hyperplane itself. This process can also be seen in Figure 2.

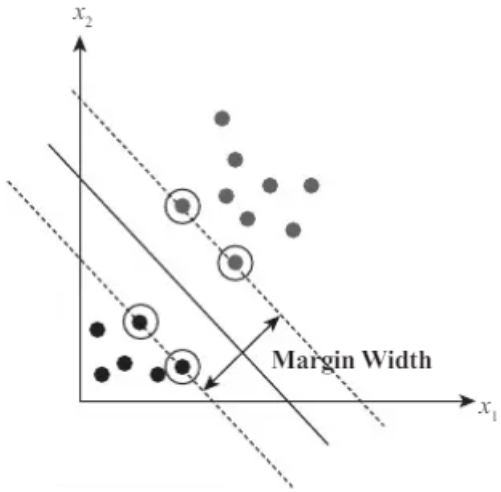


Figure 2: A hyperplane in which the boundary is defined. This boundary creates the biggest width between the two classifications. Adapted from [58]

One of the possible ways to extend this model to a multiclass classification model is to use the one-against-all (OAA) method. In this method, each of the possible classifications is mapped against all other classifications. For example, a mapping is made for "Walking" and all other classes, combined into "Not walking". After classification process has been finished for all classes, the class with the strongest prediction is chosen [54]. The advantage of using SVM is that it does not suffer the consequences of possible local minima, since it will always convert to the same best possible outcome. It can also handle a small set of sample data with a large number of dimensions [54]. Thus, SVM is generally made for pattern recognition, but can also be used for classification purposes, where it can handle a small set of samples with a large number of dimensions.

2.7.6 K-Nearest Neighbours

The idea of the K-Nearest Neighbours (KNN) algorithm is that it simply finds a certain number (k) samples that resemble the most to the sample that is being classified. Those samples are called the neighbours of the sample. The algorithm then counts the amount of classifications of the neighbours from each class and chooses the class that occurs most often. KNN is very simple, but the results are usually adequate [50].

2.7.7 Neural Network

A Neural Network (NN) consists of layers, in which each layer, in turn, consists of neurons which are connected to each other through specific weights. A graphical representation of these layers can be found in Figure 3.

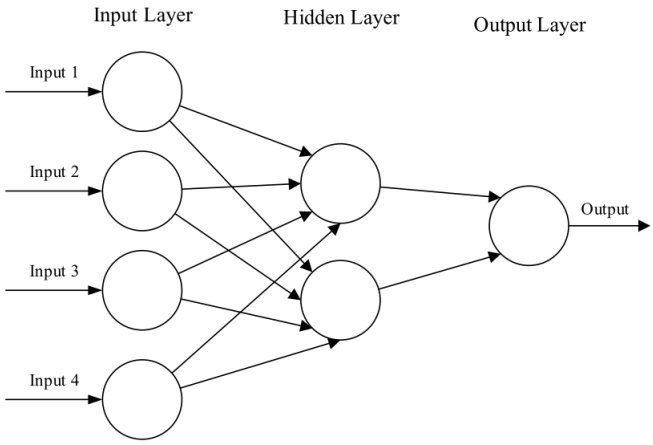


Figure 3: A graphical representation of the CNN structure. [59]

Convolutional Neural Network A Convolutional Neural Network (CNN) is a well-known class of the Neural Network. In comparison to other NNs, CNNs are able to handle much bigger tasks due to their ability to reduce the number of parameters. This also caused the interest in CNN to grow substantially over the past few years, especially in the field of pattern recognition. A CNN is comprised of layers of three types: convolutional layers, pooling layers and fully-connected layers [59]. An input layer will contain the original input, which is in this case the IMU data. The convolutional layer uses a weight vector, which slides over the input to generate a feature map. This action extracts the features from the input [60]. The pooling layer is used to reduce the complexity of the information for the next layers. An example of a pooling layer is the max-pooling layer, which generally compares the neuron outputs in a layer and takes the maximum value of those numbers. In this way, the dimensions are reduced to 25% of the original size. However, it is important to note the window in which the reduction is performed, because a window which is too big could massively decrease the performance of the algorithm [59]. In the fully-connected layer, every

node is connected to every node from the previous and the next layer. The output is calculated based on the dot product of the weight vector and input vector [60]. The way in which the layers are organized differs per algorithm, although definitely not all compositions will work for any data type [59]. The disadvantage of using a CNN is the low performance in handling long-term dependencies [61]. Apart from that, CNNs require a large amount of labeled training samples, along with a powerful GPU to handle all of that data quickly enough [62]. Thus, although it has been a popular approach in machine learning, CNNs are generally bad at handling long-term dependencies and they require a large amount of samples.

Fully Convolutional Network A Fully Convolutional Network (FCN) is a CNN, however, it replaces the often-used fully connected layers by convolutional layers. The advantage of this is that the number of model parameters decreases remarkably without decreasing the recognition rate. This is especially important when the algorithm has to be energy- or time-efficient and the training data is limited [28]. Thus, the FCN is a good approach for time- or energy-efficient systems, although it might cause a slight decrease in performance in comparison with CNN.

Adversarial Neural Network An Adversarial Neural Network (ANN) is based on the concept of a minimax [63]. In an ANN, two algorithms fight against each other. The first algorithm tries to generate fake samples, while the second algorithm tries to recognize the fake samples. Together they become more advanced in the generation and recognition of samples [64]. An advantage of an ANN is that it can address bias in the dataset and that it can select features well [63]. Thus, the ANN generates and recognizes fake samples, and with this approach becoming better at feature selection and addressing bias.

Recurrent Neural Network A Recurrent Neural Network (RNN) works similarly to the CNN in terms of that it also consists of a network of layers, consisting of neurons. However, it is designed to handle long temporal data. In RNNs, the output of neurons is not only passed forward, but also backwards as feedback for neurons in a previous layer at least once [23]. In general, RNNs are especially good at capturing long-dependence relationships [65]. However, RNN can have problems when it has long-time lags between the sending of a signal and receiving its feedback. When an error is returned as feedback to another neuron it then tends to either blow up or vanish completely. Blown up feedback can lead to unpredictable behaviour and oscillating weights, which is highly undesirable. On the other hand, when the feedback tends to vanish completely, the algorithm is unable to learn from the feedback, thus the training will be slowed down or stopped completely [66]. Thus, the RNN is especially made for long-term dependencies, but it also suffers from blown up or vanished feedback.

2.7.8 Long Short-Term Memory

The Long Short-Term Memory (LSTM) algorithm is based on a basic Recurrent Neural Network (RNN) with the ability to solve the blowing up and vanishing feedback problems. The LSTM cell, as shown in Figure 4, consists of constant error flow so that the error cannot vanish. This is done through the Constant Error Carrousal, which also causes the LSTM to have a short-term memory. Apart from that, there are multiplicative input and output gates to control the input and output of a cell. The input gate controls when to open and close and thus deciding whether the memory of that cell is allowed to be changed. The output gate works in a similar way, controlling when the cell is allowed to output information. These gates help to protect cells from unnecessary disturbances from other cells. However, the LSTM cells also need a forget gate, as they would become saturated over time if they did not have one, which means that the cell would not be able to memorize anything anymore. The forget gate resets cell states when they are no longer deemed necessary [66]. Thus, LSTM solves the feedback issues from LSTM, making it able to long-term dependencies well.

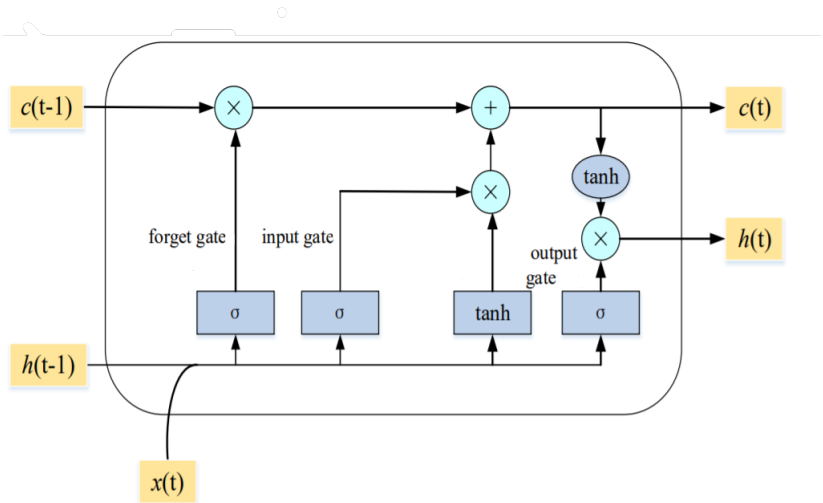


Figure 4: A graphical representation of the LSTM structure. Adapted from [67]

Bidirectional Long Short-Term Memory A Bidirectional Long Short-Term Memory (Bi-LSTM) algorithm works the same as a normal LSTM algorithm, except for that the data is used twice during training. The data is first fed to the algorithm in one way (for example from left to right) and afterwards fed the other way (for example right to left). According to Siami-Namini et al., using a Bi-LSTM reduces the error rates in the algorithm [68]. Thus, the Bi-LSTM is similar to the LSTM algorithm, although the error rates are lower for Bi-LSTM.

Hierarchical Long Short-Term Memory The Hierarchical Long Short-Term Memory (H-LSTM) algorithm consists of a network with two hidden layers, which in turn consist of LSTM neurons. Thus, instead of just one LSTM unit, the structure is a network. The

advantage of using a H-LSTM is that it is very good at the validation phase [34]. Thus, the H-LSTM is similar to the LSTM, but performs better in the validation phase than the LSTM.

2.7.9 Convolutional Neural Network with Long Short-Term Memory

The Convolutional Neural Network with Long Short-Term Memory (CNN-LSTM) combines both the CNN and the LSTM. It uses a Neural Network, of which some layers perform as CNN layers and some as LSTM layers. The advantage of such a network is that it can combine the strengths of both algorithms [17], [61]. Thus, the CNN-LSTM is able to recognize patterns well, as well as handle long-term dependencies.

3 State Of The Art

In this section, 28 studies will be compared to answer the sub-research question:

Which animal and human activity recognition algorithms that have been developed in the last two years are the most promising to utilize for a horse activity recognition algorithm using IMU data?

Within this research, only studies from 2019 and later are included. The studies have been acquired with the keywords "activity recognition" along with "IMU", "inertial sensors", "inertial measurement unit" or "accelerometer". When specifically searching for studies on AAR, the keyword "animal" was added to the search. Since the references in the studies are generally from before 2019, they were not included in the search. The results from these studies can be seen in Table 1. More information on the dataset these studies used can be seen in Table 2.

As can be seen from Table 1 and Table 2, the studies made different choices in terms of the subject type, number of subjects, sensor type, sensor location, number of activities, algorithms and train/test-split. Of the 28 studies that are evaluated, twelve studies focused on AAR, while the other sixteen focused on HAR. The amount of subjects from each study differs from one to 120 subjects. Ten of the surveyed studies [17], [33]–[36], [38], [42], [69]–[71] combined multiple datasets of activity data to train and test their algorithm on or trained and tested their algorithm separately on multiple datasets. Twelve of the surveyed studies [5], [10], [13], [20], [32], [37], [39], [40], [72]–[75] created their own dataset to operate on. The studies have been selected based on relevance, which is why all studies focused on accelerometer data. Studies focusing on other types of data, such as videography or photography have been excluded because the data type differs vastly from IMU data. Along with accelerometers, some studies also used gyroscopes, magnetometers, temperature sensors and heart rate sensors. According to Table 2, the sensors are located in different places, including the arm, back, chest, ear, foot, head, waist, leg, neck, phone, saddle, tail, wrist and hip. For the study of Wang and Liu, which used a phone as its sensor [34], it was unclear whether the phone was held by the participant or placed on the body somewhere else. Moreover, the amount of activities measured has a minimum of two activities up to a maximum of 21 activities. Apart from that, the sizes of the datasets differ a lot, ranging from 603 to 43930257 samples. In this study, the number of samples is defined by the amount of labeled and segmented datapoints, which generally consists of a standard timeframe, normally a couple of seconds. However, for one study [5] it was unclear whether the datapoints were segmented. Another study [40] only mentions the amount of datapoints before segmenting them. Some of the surveyed studies [6]–[8], [76] do not mention their sample size in their paper, but the sample size could be found in the UCI Machine Learning database [77]. However, this database mentions only the number of instances, without elaboration on the meaning of this. Thus, for many of the studies from the UCI Machine Learning database,

along with some other studies [7], [8], [15], [76], [78] the number of "instances" or "samples" is denoted without defining whether these are labeled and segmented. Thus, the surveyed studies differ in terms of the subject type, amount of subjects, dataset, sensor type, the number of activities and samples.

Activity recognition studies use different methods for splitting the data into a training and testing set. Thirteen of the surveyed studies [17], [18], [20], [32]–[36], [39], [72], [74], [79] used a Single Split, while eleven other surveyed studies [5], [10], [37], [38], [40]–[42], [70], [73], [75], [80] used the Leave One Out method. One study [13] used the Out-Of-Bag method, which is similar to the Single Split, where it leaves out a percentage of the samples as testing data. This method is especially used for evaluating Random Forest algorithms. Three of the surveyed studies [69], [81], [82] did not denote their train/test split, which makes it difficult for the reader to understand the pipeline and compare it to other studies. In conclusion, the Simple Split, Leave One Out and Out-Of-Bag methods are being used within activity recognition studies.

Table 1: A summary of the surveyed state of the art activity recognition studies and their results, sorted by algorithm.

Study	Subject type	Dataset used	Best performing algorithm	Other algorithms	Train/test-split	Accuracy	F-Score
Sturm et al. [37]	Calve	Own dataset (Sturm et al.)	CT	-	LOO	80%	-
Kamminga et al. [75]	Horse	Horsing Around	NB	-	LOO	81%	73%
Pirinen et al. [39]	Foal	Own dataset (Pirinen et al.)	NB	-	SS	89-96%	32-95%
Mojarad et al. [81]	Human	Opportunity	DT	RF, NB, SVM, DRBi-LSTM	-	96%	87%
Kleanthous et al. [13]	Sheep	Own dataset (Kleanthous et al.)	RF	-	OOB	99%	91-99%
Ayman et al. [69]	Human	Handy, Pamap2	RF	Bagged DT, SVM	-	99%, 99%	-
Priyadarshini et al. [82]	Human	Wrist-worn sensor ADL	GBT	DT	-	98%	-
Al-Frady et al. [80]	Human	WISDM	GBT	RF, KNN	LOO	93%	-
Ashry et al. [36]	Human	EJUST-ADL1, USC-HAD	HMM	LSTM, RF	SS	92%, 84%	91%, 83%
Conners et al. [72]	Albatros	Own dataset (Conners et al.)	HMM	-	SS	92%	-
Arablouei et al. [73]	Cattle	Own dataset (Arablouei et al.)	MLP	SVM, DT	LOO	96%	-
Chen et al. [41]	Human	UCI HAR	SVM	DT, RF	LOO	98%	-
Van den Berg [74]	Parakeet	Own dataset (Van den Berg)	NB, MLP, DT	KNN, SVM	SS	89%	55%
Casella et al. [32]	Horse	Own dataset (Casella et al.)	DT, KNN, MLP, SVM	-	SS	96%	-
Pucci et al. [79]	Seal	Seeing it All	ID-NN, SVM	-	SS	87%	89%
Eerdekens et al. [5]	Horse	Own dataset (Eerdekens et al.)	CNN	-	LOO	99%	-
Bocaj et al. [38]	Horse, Goat	Horsing Around, Goat Dataset	CNN	-	LOO	91%	79%
Wan et al. [35]	Human	UCI HAR, Pamap2	CNN	LSTM, Bi-LSTM, MLP, SVM	SS	92%	89%
Zhang and Zhang [40]	Human	Own dataset (Zhang et al.)	ANN	CNN	LOO	99%	-
Gil-Martìn et al. [42]	Human	Pamap2, Opportunity	CNN, CNN-LSTM	-	LOO	97%, 67%	96%, 63%
Karim et al. [71]	Human	UCI HAR, DailySports	MLSTM-FCN, MALSTM-FCN	LSTM-FCN, ALSTM-FCN	SS	97% 100%	-
Mutegeki and Han [17]	Human	UCI HAR, iSPL	CNN-LSTM	LSTM	SS	92%, 99%	-
Braganca et al. [70]	Horse	Multiple Horse Datasets	FCN	SVM, LSTM, Bi-LSTM, DT	LOO	97%	-
Chung et al. [10]	Human	Own dataset (Chung et al.)	LSTM	-	LOO	93%	67-86%
Zhou et al. [33]	Human	UniMiB SHAR, Position-aware dataset	LSTM	-	SS	96%	79%
Wang and Liu [34]	Human	UCI HAR, HHAR, DailySports	H-LSTM	DT, RF	SS	91%	-
Qi et al. [20]	Human	Own dataset (Qi et al.)	LSTM, Bi-LSTM	FR-CNN	SS	96%	-
Ashry et al. [18]	Human	CHAR-SW*	Bi-LSTM	LSTM	SS	94%	94-98%

Table 1: Data labeled with a dash (-) is unknown data or not applicable.

Algorithm type: ANN = Adversarial Neural Network, Bi- = Bidirectional, (C)NN = (Convolutional) Neural Network, CT = Chaos-theoretic, DR- = Deep Residual, DT = Decision Tree, FCN = Fully Connected Network, FR- = Fast and Robust, GBT = Gradient Boosted Tree, H- = Hierarchical, ID- = Input Delay, HMM = Hidden Markov Model, KNN = K-Nearest Neighbours, LR = Logistic Regression, LSTM = Long Short-Term Memory, MLP = Multilayer Perceptron, NB = Naive Bayes, RF = Random Forest, SVM = Support Vector Machine

Train/Test-split: SS = Single Split, LOO = Leave One (subject) Out, OOB = Out-Of-Bag

* Please note this study used multiple datasets, but only CHAR-SW is considered, since the other datasets used other sensor types.

Table 2: A summary of the datasets used by the surveyed state of the art activity recognition studies.

Dataset	Subject Type	Number of subjects	Sensor type	Sensor location	Number of samples	Number of activities
Own dataset (Conners et al.) [72]	Albatros	29	AM	B	319409	3
Own dataset (Sturm et al.) [37]	Calve	15	A	E	3600	9
Own dataset (Arablouei et al.) [73]	Cattle	10	A	N	6660	4
Goat dataset [11]	Goat	5	AGM	N	-	6
Own dataset (Pirinen et al.) [39]	Foal	11	A	T	74346	3
Own dataset (Casella et al.) [32]	Horse	2	A	SW*	2416	3
Own dataset (Eerdekens et al.) [5]	Horse	6	A	L	959075	7
Multiple Horse Datasets [70], [83], [84]	Horse	120	AGM	BHLS	-	8
Horsing Around** [12]	Horse	6	AGM	N	87621	6
Own dataset (Van den Berg) [74]	Parakeet	1	A	B	8277	6
Seeing it all** [55]	Seal	7	A	B	12692	2
Own dataset (Kleanthous et al.) [13]	Sheep	8	A	N	-	4
DailySports [6]	Human	8	AGM	CLW	9120	19
EJUST-ADL1 [15]	Human	3	AGR	W	603	14
Handy [16]	Human	30	AGM	W	992976	7
HHAR [76]	Human	9	AG	P	43930257	6
iSPL (not public) [17]	Human	4	AG	W	1590	3
Opportunity [7]	Human	4	AGM	CFW	2551	21
Pamap2 [8]	Human	9	AGHMT	CLW	3850505	18
Position-aware dataset [9]	Human	15	A	ACHIL	-	8
UCI HAR [19]	Human	30	AG	I	10299	6
UniMiB SHAR [14]	Human	30	A	ACH	11771	8
USC-HAD [78]	Human	14	AG	X	2311	12
Own dataset (CHAR-SW) [18]	Human	25	AGM	IW	-	10
Wrist-worn sensor ADL [85]	Human	16	A	W	979	14
WISDM [86]	Human	36	A	P	5418	6
Own dataset (Chung et al) [10]	Human	5	AGM	ACILW	-	9
Own dataset (Qi et al) [20]	Human	20	AGM	I	5088	12
Own dataset (Zhang et al) [40]	Human	8	AGM	F	1200000	5

Annotations to Table 2: Data labeled with a dash (-) is unknown data.

Sensor location: A = Arm, B = Back, C = Chest, E = Ear, F = Foot, H = Head, I = Waist, L = Leg, N = Neck, P = Phone, S = Saddle, T = Tail, W = Wrist, X = Hip

Sensors: A = Three-axis accelerometer, G = Gyroscope, H = Heart Rate, M = Magnetometer, R = Rotation, T = Temperature

* Please note the rider's wrist is meant in this study.

** Please note that only part of the Horsing Around dataset was used, so only the part that was used is denoted in this table.

3.1 State of the Art algorithms

3.1.1 Chaotic Theoretic

Sturm et al. [37] used a Chaotic Theoretic (CT) approach. For their study, they first created their own dataset with data from 15 calves, collecting in total 3600 unique labeled samples of 1 minute long, which is a relatively low number of samples. Sturm et al. equipped an algorithm to be used in their CT framework, for which six different classifiers (RF, SVM, NB, NN, KNN and LR) were compared. However, the paper doesn't mention which classifier performed best and was used in the CT framework. The performance denoted by Sturm et al. is outstandingly low (80% accuracy) in comparison to other studies. This may be caused by the relatively low number of samples and high number of activities. Since none of the other surveyed studies used the same method as Sturm et al., it is hard to verify whether the algorithm itself works well.

3.1.2 Naive Bayes

Naive Bayes is a well-known approach within the field of Machine Learning. For example, Pirinen et al. [39], Mojarad et al. [81], Van den Berg [74] and Kamminga et al. [75] used the Naive Bayes classifier. Pirinen et al. [39] measured the standing, lying and walking behaviour of eleven foals. The reported accuracies of the different activities were quite high (89 to 96%). However, the reported F-scores were very low (32%, 53%, 73% and 95%). These low scores were probably due to the fact that the sensor was attached to the tail, even though the tail made similar movements for standing and walking. Apart from that, they reported the selected features were insufficient to distinguish between having one large peak in the data (when the tail is slashed once while standing) or multiple peaks (when the tail is continuously slashed while walking). Kamminga et al. [75] collected activity data from horses, with which they yielded a relatively low accuracy (81%), but relatively high F-score (73%). The classifier was mainly developed to demonstrate the possibilities with the dataset. Mojarad et al. [81] created a robust framework to recognize activities with multiple labels. An activity with multiple labels is one that falls within multiple categories. For example, an activity can be labeled as standing and eating, instead of just standing or eating. This method can resolve many of the issues that researchers are currently facing within the activity

recognition sector, as activities can be described in more detail. Apart from a classifier, Mojarad et al. also included a Classification Error Detection algorithm and accompanying Correction Module. Mojarad et al. found that the Decision Tree performed better (with 97% accuracy) than the Naive Bayes classifier (with 95% accuracy). Furthermore, Van den Berg [74] classified activities of parakeets, finding that a Naive Bayes, Neural Network and Decision Tree performed similarly (with 89% accuracy). A great advantage of using NB is that it is relatively simple, and thus, fast in comparison to most other algorithms [87]. However, when computational speed is not a restraining factor, NB may not be the best approach for the highest performance, since the results of the surveyed studies are mixed.

3.1.3 Decision Tree

Next to the Naive Bayes classifier, multiple studies [32], [34], [41], [69], [73], [74], [81] adopted the (Bagged) Decision Tree. Ayman et al. [69] compared RF, Bagged DT and SVM classifiers, which all achieved surprisingly good results (99% for all algorithms for the Handy dataset and 99%, 98%, 98% respectively for the PAMAP2 dataset). However, it is important to note that the method for splitting the data was not elaborated upon in the study, making the denoted performance incomparable to similar studies. Mojarad et al. [81] observed similar results and denotes that the DT (97.3% accuracy) slightly outperforms the RF (96.7% accuracy), NB (95.2% accuracy), SVM (95.7% accuracy) and DRBi-LSTM (96.3% accuracy). Next to that, Casella et al. [32] equipped four different algorithms: DT, KNN, NN and SVM, but all algorithms achieved similarly high results (96% accuracy). This is, however, probably mostly due to the low number (3) of activities and the fact that a Single Split has been used as train-test data split, which has caused a bias towards the two horses it has trained on. Thus, if the algorithm were to be applied to a new horse, this would most probably lead to a much lower accuracy and might also lead to more evident differences in performance between the algorithms. On the other hand, Arablouei et al. [73] observed that the DT (with 92.7% accuracy) is slightly outperformed by MLP (with 93.4% accuracy). Along with Arablouei et al., Chen et al. [41] also found that the performance of the DT (97.9% accuracy) was slightly lower than with another algorithm, namely the SVM (98.3% accuracy). Lastly, Wang and Liu show a significant outperformance of the DT (86% accuracy) by an H-LSTM (92% accuracy). As well as Naive Bayes, the Decision Tree is especially interesting when a small computation time is a restriction [50]. However, when comparing the algorithm to others in terms of the performance results, the Decision Tree is probably not the best algorithm to use for AAR.

Random Forest As well as the normal Decision Tree classifier, also the Random Forest classifier got mixed results from the surveyed studies [13], [34], [41], [69], [81] that used it. Kleanthous et al. [13] reported good results (99% accuracy) with their RF classifier. However, they also focused severely on making the feature selection and extraction very

good, while this costs a lot of time to do by hand. Ayman et al. [69] reached similar results (99% accuracy).

Even though Kleanthous et al. and Ayman et al. denoted very high accuracies, some studies [34], [36], [41], [81] show an outperformance of RF by another algorithm. First, Chen et al. [41] states that the RF algorithm (98.0% accuracy) is slightly outperformed by the SVM (98.3% accuracy). Apart from that, Mojarad et al. [81] denotes a slightly higher accuracy with another algorithm than with the RF (96.7% accuracy), namely the DT (97.3% accuracy). Additionally, Wang and Liu [34] report a higher performance of the H-LSTM (92% accuracy) than the RF (91% accuracy) classifier. Lastly, Ashry et al. [36] also denotes a clear outperformance of an RF (79% and 82% accuracy) by an HMM (84% and 92% accuracy) on two datasets. In conclusion, the Random Forest classifier seems to lead to varying results.

Gradient Boosted Tree Two of the surveyed studies [80], [82] used the Gradient Boosted Tree classifier. Al-Frady et al. [80] focussed mainly on the feature selection method, namely the Sequential Forward Selection method, for their HAR classification. However, they applied this method to three classifiers, where the GBT (with 93.1% accuracy) slightly outperformed the RF (with 92.2% accuracy) and KNN (with 92.9% accuracy) classifiers. When comparing the GBT with other studies using the same dataset, the GBT (slightly) outperforms a DT (85.4% accuracy), NN (85.7% accuracy), SVM (90.5% accuracy) and CNN (93.0% accuracy). This outperformance may be attributed to the improved feature selection method or to the classifiers used. Next to that, Priyadarshini et al. [82] compared the GBT to a DT, where the GBT (with 98% accuracy) considerably outperformed the DT (92% accuracy). In conclusion, in both of the surveyed studies the Gradient Boosted Trees achieved high-performance results. However, due to the low number of studies in the past two years on this method, it is unclear whether the method generally works well for other datasets.

3.1.4 Hidden Markov Model

The Hidden Markov Model is adopted by two of the surveyed studies [36], [72]. Even though Conners et al. [72] built a classifier for only 3 activities, the denoted performance is relatively low (92% accuracy). Ashry et al. [36] report a similar performance (84% and 92% accuracy on two datasets). However, Ashry et al. did also compare the HMM with two other algorithms, which it outperformed: LSTM (79% and 87% accuracy) and RF (79% and 82% accuracy). On the other hand, Ashry et al. also denote that the computational complexity of an HMM may become an issue in other projects. This could, for example, be the case when using larger datasets than used by Ashry et al. Next to that, Gil-Martìn et al. [42] mention that an HMM is more robust than RF, but has a reduced performance when new subjects are introduced. Since both Conners et al. and Ashry et al. used the Simple Split method for splitting the data, their performance did not suffer from this problem. However, when applying the same algorithm to a new subject, the performance will probably decline. Since

only two of the surveyed studies adopted this method, it is hard to compare the performance of the model with other algorithms. Apart from that, it is unclear how much the performance of an HMM decreases when introducing new subjects.

3.1.5 Support Vector Machine

The SVM has been adopted by many activity recognition studies [32], [35], [41], [69], [73], [74], [79], [81], because it can handle a small set of sample data with a large number of dimensions [54]. Chen et al. [41] especially focus on the transitions between two activities, as they are often ignored in HAR research. To achieve this, they mainly focussed on selecting the best features for the data. Possibly due to this extensive research on feature selection, the three classifiers used by Chen et al. all perform very well. The SVM (98.3% accuracy) scores only slightly higher than the DT (97.9% accuracy) and RF (98.0% accuracy). Casella et al. [32] also used an SVM, DT, but also KNN and NN, which all denoted similar results (96% accuracy). Next to that, Pucci et al. [79] also found similar results among their algorithms, which were ID-NN and SVM, although the results were quite low for both algorithms (87% accuracy). On the other hand, many studies [35], [69], [73], [81] have also equipped SVM but reached better results with another algorithm. Mojarad et al. [81] reached slightly better results with a DT (97% accuracy) over an SVM (96% accuracy), Ayman et al. [69] did this with an RF (99% accuracy) over an SVM (98% accuracy), Arablouei et al. [73] with an MLP (93.4% accuracy) over an SVM (92.8% accuracy) and Wan et al. [35] with a CNN (93% accuracy) over an SVM (91% accuracy). In conclusion, the SVM seems to work just as well as some other algorithms (DT, RF, KNN, NN, ID-NN) in some of the surveyed studies, although others report an outperformance of SVM by other algorithms (DT, RF, MLP, CNN).

3.1.6 K-Nearest Neighbours

The K-Nearest Neighbours classifier is equipped by two of the surveyed studies [32], [74]. Casella et al. [32] denote the performance of KNN to be similar to a DT, NN and SVM (96% accuracy). Moreover, Van den Berg [74] denotes a higher performance with an NB, NN and DT than with KNN. One of the limitations of KNN is that the value of k affects the performance greatly, as a small k may lead to overfitting and a large k may lead to misclassifications because another activity is included in the subset [88]. Since this method is only used by two of the surveyed studies and reached varying results, KNN is not advisable for AAR.

3.1.7 Neural Network

The Multilayer Perceptron generally yield high performance results. Van den Berg [74], Casella et al. [32], Arablouei et al. [73] and Wan et al. [35] adopted the Multilayer Perceptron (MLP) in their study. Casella et al. [74] equipped the DT, KNN, SVM and MLP, which all performed similarly (96% accuracy). Additionally, Van den Berg [74] found that the NB, DT

and MLP performed similarly (with 89% accuracy), but outperformed the KNN and SVM. Arablouei et al. [73] measured cattle activities, reaching an accuracy of 93.4% with MLP, slightly outperforming SVM (92.8% accuracy) and DT (92.7% accuracy). On the other hand, Wan et al. [35] also adopted this method, but did not reach a great performance with MLP. In their study, MLP performed the worst (87% accuracy) compared to CNN, LSTM, Bi-LSTM and SVM (89 to 93% accuracy). Thus, the MLP seems to outperform the generally simpler algorithms (DT, NB, KNN), but may be outperformed by the more complex deep learning algorithms (CNN, LSTM, Bi-LSTM).

The Convolutional Neural Network (CNN) is a very popular approach in the field of HAR and AAR. Eerdekens et al. [5] used a CNN with two convolutional layers, max-pooling layers, two fully connected layers and lastly a softmax layer. With the best settings in the network, the algorithm reached an accuracy of 99%. They did not use any other algorithm to compare the CNN with. Bocaj et al. [38] compared different settings on CNNs to see which performs best, concluding that a high number of filters does not necessarily increase the accuracy of the algorithm due to overfitting. Their best CNN achieved an accuracy of 91% and an F-score of 79%. Next to that, Wan et al. [35] compared multiple algorithms, reaching higher results with CNN (93% accuracy) than with LSTM (89% accuracy), Bi-LSTM (89% accuracy), MLP (87% accuracy) and SVM (91% accuracy). Gil-Martín et al. [42] compared a CNN with a CNN-LSTM, reaching varying results with both algorithms, where the CNN scored better on the Pamap2 dataset and CNN-LSTM scored better on the Opportunity dataset. Interestingly, the performance on the Pamap2 dataset is much better (97% accuracy) than on the Opportunity dataset (67% accuracy). This is probably due to a combination of a having more samples, slightly less activities, more subjects and more sensor locations in the Pamap2 dataset. In conclusion, in general the results of Convolutional Neural Networks are very satisfactory, making it a promising method for AAR.

Even though these studies got quite good results with a CNN, two other neural networks have been proposed which might outperform the CNN. Zhang and Zhang [40] achieved better results with an Adversarial Neural Network (ANN) (98% accuracy) than with a CNN (94% accuracy). The study also focuses on adversarial training to reduce the effect of new subjects being unrecognized by the algorithm, which is done with unlabeled data. Braganca et al. [70] classified horse activities with multiple classifiers, where the Fully Connected Network (FCN) (97% accuracy) resulted in a better performance than SVM (96% accuracy), LSTM (96% accuracy), Bi-LSTM (96% accuracy) and DT (79% accuracy). The FCN consists of only an input layer, one hidden layer and an output layer. The ANN and the FCN yielded a great performance, although only one study has been conducted on each of those algorithms, making it hard to assume the algorithm will work well for other datasets.

Two other methods that have adapted CNN are FR-CNN and ID-NN. Qi et al. [20] adapted the CNN to become faster rather than adjust it for a better performance. Afterwards, the Fast and Robust CNN (FR-CNN) was compared to an LSTM and Bi-LSTM (both 96% accuracy), where the FR-CNN slightly underperformed (95% accuracy). This lower

performance result may be attributed to the focus on a more time-efficient classifier, over a higher performance classifier. Lastly, Pucci et al. [79] adopted an ID-NN, which reached similar results to an SVM. However, the performance of both algorithms was quite low, with only 87% accuracy. The FR-CNN and ID-NN did not yield outstandingly great performances and only one study has been performed on each algorithm, making it hard to assume these methods will work well in AAR.

3.1.8 Long Short-Term Memory

Multiple studies [10], [17], [18], [20], [33]–[36], [70], [81] adopted an LSTM in some form, whether a normal LSTM, Bi-LSTM or H-LSTM. Chung et al. [10] not only focused on achieving the highest accuracy, but also on a low execution time. The algorithm used is an LSTM with an RF as meta-learner, as this resulted in a high F-score and low execution time. The algorithm performs at an accuracy of 94%, although this could probably be improved with settings that are more time-consuming. Zhou et al. [33] used an LSTM classifier on partly labeled data. This method reached an accuracy of 96% and an F-score of 79%. These scores may have been improved if they had used fully-labeled data. In the study of Mutegeki and Han, the LSTM (96% and 91% accuracy) performed less than a CNN-LSTM (99% and 92% accuracy) in two datasets. Ashry et al. reported a higher performance with an HMM (84% and 92%) than LSTM (79% and 87%). In conclusion, the LSTM generally seems to yield good performance results, making it a promising technique in the field of AAR.

Multiple studies [18], [20], [34], [35], [70], [81] developed an adapted version of the LSTM (Bi-LSTM, H-LSTM and DRBi-LSTM), although the results are mixed. Qi et al. [20] equipped an LSTM and Bi-LSTM, which reached similar results (96% accuracy) and outperformed an FR-CNN (95% accuracy). However, it must be noted that this FR-CNN was an adapted version of a CNN, with a low execution-time as its goal, which may have led to a slightly decreased performance score. Next to Qi et al., also Ashry et al. [18] compared an LSTM with a Bi-LSTM, although the results did differ in this study, since the Bi-LSTM algorithm (96% accuracy) outperforms the LSTM algorithm (90% accuracy). In the study of both Braganca et al. [70] and Wan et al. [35] the LSTM and the Bi-LSTM classifiers perform similarly. However, they are outperformed by another algorithm in both studies (FCN in the study of Braganca et al. and CNN in the study of Wan et al.). Next to that, Mojarad et al. [81] developed a Deep Residual Bi-LSTM, although this algorithm did not perform as well as the, much simpler, DT. Lastly, Wang and Liu [34] equipped an H-LSTM and compared this algorithm to a DT and RF, performing better with their H-LSTM algorithm (91% accuracy) than with a DT (86% accuracy) or RF (90% accuracy). Conclusively, the added value of a Bi-LSTM, H-LSTM or DRBi-LSTM over a normal LSTM has not been shown.

3.1.9 Convolutional Neural Network with Long Short-Term Memory

The CNN-LSTM has been equipped by two of the surveyed studies, where it performed very well in both studies. As mentioned above, Gil-Martín et al. [42] compared a CNN with a CNN-LSTM, where the algorithms reached similar results (97% accuracy on Pamap2 and 67% accuracy on Opportunity). Secondly, Mutegeki and Han [17] compared a CNN-LSTM with a normal LSTM. The CNN-LSTM consists of a convolutional layer, a maxpooling layer, the LSTM network and a fully connected output layer. The CNN-LSTM (99% and 92% accuracy) outperformed the LSTM (96% and 91% accuracy) classifier in both datasets used. Next to those two studies, Karim et al. [71] equipped a Multivariate Long Short-Term Memory Fully Convolutional Network (MLSTM-FCN), which reached outstanding results as well. Next to combining the LSTM and FCN, they also added a squeeze-and-excitation block, which can map the interdependencies between channels and with this, increases the performance of the classifier. They also made an adapted version to this classifier, adding an attention mechanism to the LSTM block (named MALSTM-FCN), which is most often used within text-based machine learning, and yielded similar results to the MLSTM-FCN. The classifier yielded outstanding results (96.7% and 99.7%) on two HAR datasets. Another advantage of this state of the art algorithm is that the code has been publicly published. To conclude, even though not many of the surveyed studies have used this approach, the results have been very promising.

3.2 Comparison between HAR datasets

Some studies have used standard HAR datasets, which allows for an easier comparison based on their performance. The datasets that have been used by multiple of the surveyed studies are the Pamap2 [8], UCI HAR [19], Opportunity [89] and DailySports [6] datasets. A summary of these datasets, along with the best performance score from the surveyed studies, can be seen in Table 3.

3.2.1 Pamap2

Three of the surveyed studies [35], [42], [69] have used the Pamap2 dataset [8] for their activity recognition algorithm. Among the three studies, Ayman et al. denote the highest performance with an accuracy of 99% with their RF algorithm [69], scoring higher than Wan et al. (92% accuracy) with a CNN [35] and Gil-Martín et al. (97% accuracy) with a CNN and CNN-LSTM [42]. However, Ayman et al. did not denote their train/test-split method, making it harder to compare the performance score with the other studies. Thus, Ayman et al. denote the highest performance score among the surveyed studies using the Pamap2 dataset, although this score is hard to compare with the other studies due to an unknown train/test-split method.

Table 3: A summary of the datasets that have been used by multiple studies with their best performance scores.

	Pamap2	**UCI Har**	**Opportunity**	**DailySports**
Number of subjects	9	30	4	8
Number of sensors	3 (CLW)	1 (I)	6 (CFW)	5 (CLW)
Number of samples	3850505	10299	2551	9120
Number of activities	18	6	21	19
Best performance accuracy	100%	98%	96%	97%
Best performance F-score	-	-	87%	-

Annotation to Table 3:
Sensor location: C = Chest, F = Foot, I = Waist, L = Leg, W = Wrist

3.2.2 UCI Har

Five of the surveyed studies [17], [34], [35], [41], [71] equip the UCI Har dataset [19] for their activity data. Among these four studies, Karim et al. denote the highest performance (99.7% accuracy) with an MLSTM-FCN [71]. Next to that, Chen et al. also yield a very high performance (98% accuracy) with an SVM [41]. The other studies using this dataset score significantly lower, including Wang and Lui (91% accuracy) with an H-LSTM [34], Wan et al. (92% accuracy) with a CNN [35] and Mutegeki and Han (92% accuracy) with a CNN-LSTM [17]. The lower performance of Wang and Liu and Wan et al. could be attributed to the fact that they combined multiple datasets and reported the accuracy based on an average of all datasets. This means the other datasets have also affected the performance. For example, for Wang and Liu, their algorithm was also tested on the DailySports dataset [6], which includes a quite low number of samples, but a high number of activities. On the other hand, Karim et al. [71] also yield a higher performance on the DailySports dataset than Wang and Liu. In conclusion, Karim et al. yield the highest performance score among the surveyed studies using the UCI Har dataset.

3.2.3 Opportunity

The Opportunity dataset [89] is used by two of the surveyed studies [42], [81]. From the two studies, Mojarad et al. score significantly higher (with 96% accuracy and 87% F-score) with their DT [81] than Gil-Martín et al. (with 67% accuracy and 63% F-score) with their CNN and CNN-LSTM [42]. This is interesting, because DT is a much simpler algorithm than CNN/CNN-LSTM. However, it must be mentioned that Mojarad et al. do not denote their method for splitting the data, making it harder to compare the performance score with the score of Gil-Martín et al. Thus, Mojarad et al. yield the highest performance with their

DT algorithm, although the comparison with Gil-Martín is hard, since Mojarad et al. do not mention their train/test split.

3.2.4 DailySports

The DailySports dataset [6] is used by two of the surveyed studies [34], [71]. From the two studies, Karim et al. yield significantly the highest performance with 97% accuracy with an MLSTM-FCN. Wang and Liu, on the other hand, score a lower performance score with 91% accuracy with an H-LSTM. However, it must be noted that Wang and Liu combined multiple datasets, so this accuracy score may be affected by other datasets having less samples, more activities or more complex data overall. Conclusively, Karim et al. yield the highest performance score with an MLSTM-FCN.

3.3 Limitations

The limitations of many studies may be interesting to study, since they can show improvements for other studies as well. Denoted improvements are an increased number of subjects, samples and sensors. Four of the surveyed studies [5], [10], [32], [39] denote that their algorithm can be improved, especially in terms of robustness, with an increased number of subjects. Van den Berg [74] and Bocaj et al. [38], however, mention that the focus should be on increasing the number of samples to improve the performance of the pipeline. Apart from that, Bocaj et al. [38], Casella et al. [32] and Zhang and Zhang [40] mention that their pipeline may be improved by the exploration of different sensors. Thus, an activity recognition pipeline may be improved with an increased number of subjects, samples or the usage of different sensors.

Data preparation is an important step in an activity recognition pipeline, where unprepared and unbalanced data can lead to a poor performance of the classifier. Eerdekens et al. [5] extracts the features for their algorithm by hand. However, they recommends automatically extracting the features, instead of by hand, to improve the performance of their algorithm. Sturm et al. [37] point out that their framework could be improved by further data preparation, in order to filter it and cluster outliers. Apart from that, Sturm et al. [37] note that their dataset is very unbalanced, with one of the activities only occurring rarely (0.5% of the data). This makes the algorithm biased towards the other activities, making it hard to classify this rarely-occuring activity well. Conclusively, extracting features automatically, improving data preparation and having a balanced dataset may improve the performance of an activity recognition algorithm.

Combined factors, such as the sensor location in combination with the features used, can play a role in reaching great performance results. Pirinen et al. [39] claims that their sensor location is not optimal. They used the tail as their sensor location, but the tail shows almost no difference between standing and walking, making it hard to recognize the differentiate between them from tail activity alone. Apart from that, Pirinen et al. mentions

the feature selection process should be improved upon. The features used (mean, maximum and minimum) were not sufficient to distinguish between one large peak in the data (when the tail is slashed once while standing) and multiple peaks (when the tail is continuously slashed while walking). Using a different sensor location or selecting different features could resolve this issue.

Multiple studies [10], [13], [18], [34], [38], [72]–[74], [78], [80] denote the issue of activities being very similar. One of the examples for this is that resting and standing provide similar acceleration results, making them hard to distinguish from each other with accelerometer data only. Arabluoei et al. [73] and Van den Berg [74], both using an "other" category, show that this category is especially hard to classify accurately, since this category consists of many different activities. Thus, classifying activities that result in similar acceleration data, or having a very broad category, such as "other", may lead to a decrease in the performance score of an activity recognition algorithm.

3.4 Conclusion

26 studies have been compared based on the performance of their activity recognition algorithm to answer the question:

Which animal and human activity recognition algorithms that have been developed in the last two years are the most promising to utilize for a horse activity recognition algorithm using IMU data?

From the state of the art it seems that most recent studies use either a CNN or LSTM classifier. These classifiers also seem to be the most well-performing algorithms, along with the CNN-LSTM. CNNs have shown to have a great performance in large datasets. Apart from that, it is known for good generalization, which is an important feature in AAR [90]. Next to that, the LSTM is especially known for being temporally deep, which is important when processing long-time data, which is done in AAR [17]. Lastly, although the CNN-LSTM has not been equipped by many studies, it yielded very high results and has code available to equip this classifier. Thus, the CNN, LSTM and CNN-LSTM seem to be very promising algorithms within AAR, each with its own advantages. Some of the other algorithms (NB, DT, RF, HMM, SVM, KNN) reached varying results, in some studies performing very well, but are outperformed by other classifiers in other studies. Apart from that, some classifiers (CT, GBT, HMM, KNN) were not used very often, making it hard to evaluate the usefulness and robustness for an AAR application. In conclusion, the CNN, LSTM and CNN-LSTM classifiers are the most promising algorithms to use for an AAR pipeline using IMU data. This is why two out of these three classifiers were developed and tested in Chapter 6. The LSTM was chosen due to the high number of studies using it (10 out of 28) and its relatively high performances in each of these studies. Next to that, code was available to develop the LSTM easily. Moreover, the MLSTM-FCN, a variation of the CNN-LSTM, was chosen, which yielded high performances on different datasets and had clear code available online. It

was chosen not to develop the CNN due to time constraints. Apart from that, the CNN was used less often than the LSTM by the surveyed studies and it yielded lower results than the MLSTM-FCN. Thus, it was concluded that the LSTM and MLSTM-FCN were developed, due to these classifiers being the most promising algorithms to use, along with their code being available.

4 Methodology

4.1 Organization

In this chapter, the methodology of the experiments performed in this study will be explained. First, the dataset and its usage, is described. Next to that, the developed classifiers and their settings are described. Moreover, the full pipeline is explained, including the filtering, splitting, scaling, windowing, feature selection, shuffling, reshaping and encoding of the data. Afterwards, the evaluation methods are explained and the hyperparameters are decided upon. Lastly, the use of a database and other tools discussed. The evaluation of these experiments is discussed in Chapter 6.

4.2 Dataset

The data used in this Graduation Project has been gathered by Kamminga et al. [12]. This dataset consists of data gathered from 18 different horses and ponies across a period of seven days, during which the horses participated in both riding and free roaming activities throughout their pasture. The animals wore an inertial measurement unit (IMU), containing an accelerometer, gyroscope and magnetometer, in a collar. These IMUs made use of a 100 Hz sampling rate, recording a total of 1.2 million data samples of 2 seconds long. As the collar containing the IMU could still slightly move and rotate around the animal's neck, the dataset also includes l2-norm values for each of the sensors, which are orientation-independent. The dataset consists of labeled and unlabeled data. The used data as a whole is not equally labeled; only data from 11 subjects were labeled, of which six subjects and six activities were labeled more extensively, which can be seen in Figure 5 and 6. This was done so that leave one subject out validation can be done between those six subjects.

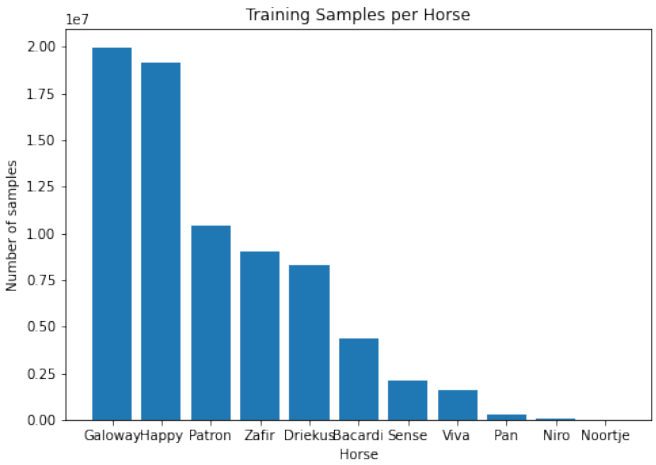


Figure 5: The distribution of labeled samples over the different horses for all activities.

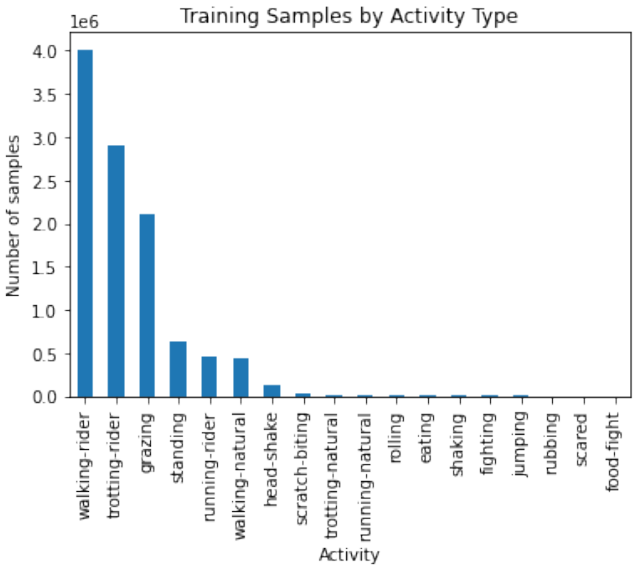


Figure 6: The distribution of labeled samples over the different activities for all horses.

The data is contained in CSV files, describing the x, y and z values of the accelerometer, gyroscope and magnetometer. Next to that, the subject, segment, label and date and time are denoted. In Figure 7 an example can be found of a sample for the activities eating, running-natural and shaking.

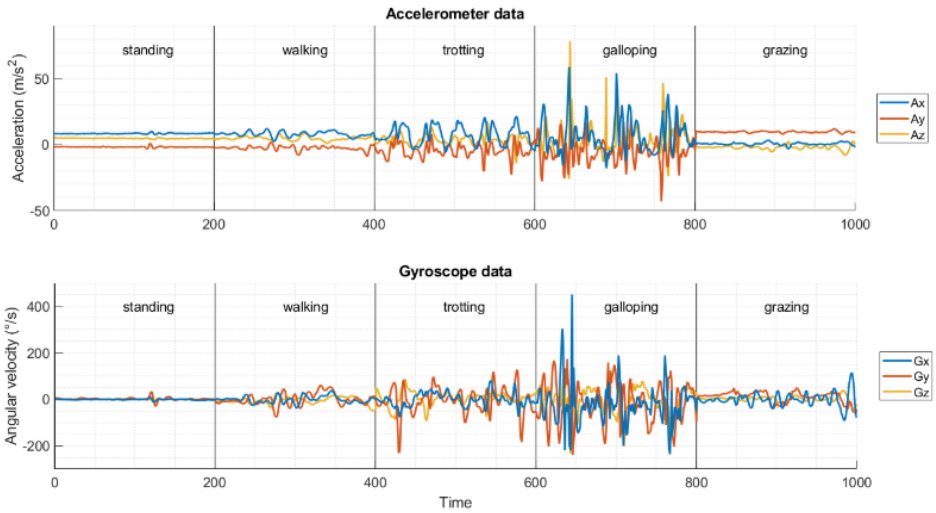


Figure 7: An accelerometer and gyroscope measurement of a horse standing, walking, trotting, galloping and grazing. [12]

4.3 Dataset usage

From the full dataset, only the labeled data was used. More specifically, only the six most extensively labeled subjects and activities were used for this project. The activities trotting rider and trotting natural were combined into one trotting activity. Next to that, the running rider and running natural were combined into one running activity. This was done because these activities are observationally similar and not both of the activities contained enough samples to be used as a separate activity to recognize. Thus, only data from the horses Galoway, Patron, Happy, Driekus, Zafir and Bacardi was used for the activities walking rider, walking natural, trotting (rider and natural combined), grazing, standing and running (rider and natural combined). The dataset containing these six horses and six activities contains 10022709 labeled (unwindowed) rows or 100227 seconds of data, since the sampling rate is 100Hz.

Sensor selection The dataset also contained three columns describing the magnetometer axes, which were dropped, as this data was found to be too prone to alterations as a result external disturbances, such as magnetic fields, and thus unreliable as a whole. The other two sensors, the accelerometer and gyroscope, were included, and measure the acceleration and angular velocity.

4.4 Classifier

As described in Chapter 3, a vanilla Long-Short Term Memory (LSTM) and Multivariate Long Short Term Memory Fully Convolutional Network (MLSTM-FCN) were chosen to be used in this study. Next to that, the pipeline was developed with a deep Neural Network (NN), thus also this classifier was compared with the LSTM and MLSTM-FCN. In Chapter 6 the performance of these classifiers will be reported and compared.

4.4.1 Deep Neural Network

A deep neural network was used as a standard for the AAR pipeline and is based upon a HAR tutorial on time-series data that uses a Sequential model from keras [91]. The optimal number of layers, either 1 or 2, was tested, which is described in Chapter 6. The structure of the model can be seen in Figure 8. This network consists of the following layers:

- Reshape layer, reshaping the data from an array in the shape of M_{NxWI}, in which M is the matrix, N the number of samples and WI the window size and number of inputs multiplied by each other, into the format M_{NxWxI}, in which M is the matrix, N the number of samples, W the window size and I the number of inputs.
- A number of Dense layers (1 or 2) consisting of a number cells with a ReLu activation function. In Chapter 6 the optimal number of layers has been tested. This layer has been repeated either once or twice to see which configuration performs best. Next to

that, the number of cells (100 or 200) in the Dense layer has been tested, which is further explained in Chapter 6.

- Flatten layer, used to flatten the data.
- Dense layer with a softmax activation function, serving as an output layer.

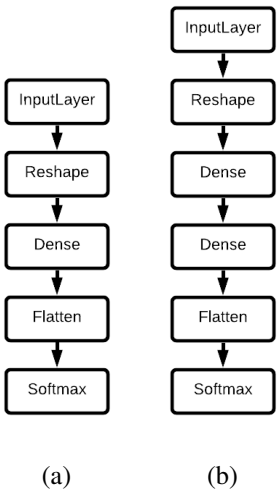


Figure 8: The structure of an NN with (a) one Dense layer and (b) two Dense layers.

4.4.2 Vanilla Long Short-Term Memory

The vanilla Long Short-Term Memory (LSTM) model consists of multiple layers, added in a Sequential model from keras. Since none of the LSTMs from the state of the art had clear and implementable code available, the model has been based upon a vanilla LSTM from a recent study of Elsworth and Güttel [92] on time-series data. This model consists of the following layers:

- LSTM layer, with either 100 or 200 cells in its layer. The optimal number of cells has been tested and is further explained in Chapter 6.
- Dropout layer, which ensures that part of the input is set to zero to prevent overfitting. Srivastava et al. [93] show that the optimal dropout parameter is set between 0.4 and 0.8, where the exact value does not yield a significant difference in performance. Thus, the parameter was set at 0.5, which is between those values and equal to the dropout parameter of another study on LSTMs [20].
- Dense layer with a softmax activation function, serving as an output layer.

Next to the number of cells per layer, also the optimal number of layers has been tested in Chapter 6. For a 1-layer LSTM network the LSTM and Dropout layers appears once and for a 2-layer LSTM network the LSTM and Dropout layers appear twice, thus the network consists

of the layers: LSTM, Dropout, LSTM, Dropout, Dense. The structure of these models can be seen in Figure 9.

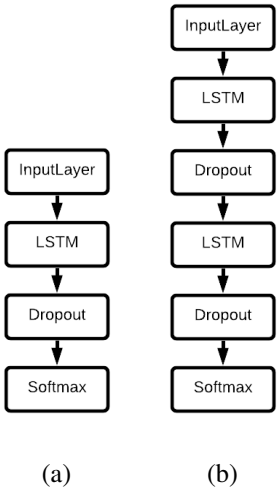


Figure 9: The structure of an LSTM with (a) one LSTM and dropout layer and (b) two LSTM and dropout layers.

4.4.3 Multivariate Long Short-Term Memory Fully Convolutional Network

The Multivariate Long Short-Term Memory Fully Convolutional Network (MLSTM-FCN) model consists of many layers and is based upon a state of the art model by Karim et al. [71]. The full model structure can be seen in Figure 10.

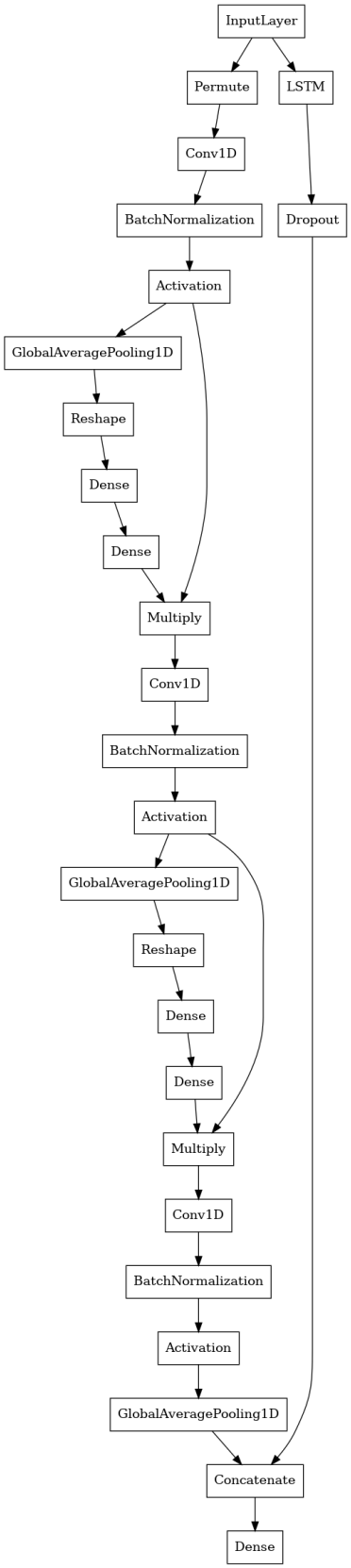


Figure 10: The structure of the MLSTM-FCN.

A brief description of each layer type can be seen below.

- Input layer, which simply instantiates a tensor from the input.
- LSTM layer, with either 100 or 200 cells in its layer. The optimal number of cells has been tested and is further explained in Chapter 6.
- Dropout layer, which ensures that part of the input is set to zero to prevent overfitting. The percentage of input that has been dropped depends on the dropout parameter, which is set at the same rate as the original study used, which is 0.8 [71].
- Permute layer, which permutes the inputted shape to the inputted pattern.
- Conv1D layer, which is a 1-dimensional convolutional layer.
- BatchNormalization layer, which normalizes the batch input.
- Activation layer with the Rectified Linear Unit activation function.
- GlobalAveragePooling1D layer, which performs global average pooling on the input.
- Reshape, which reshapes the input into a new format
- Dense layer with a ReLu activation function and 'he_normal' kernel initializer.
- Dense layer, with a softmax activation function serving as an output layer.

4.5 Horse activity recognition pipeline

The pipeline is split up into three classes: preprocessing of the data, the database interactions, and the main class, which also contains the classifier. The full pipeline can be seen in Figure 11. In the next subsections, each of the steps in the pipeline will be explained.

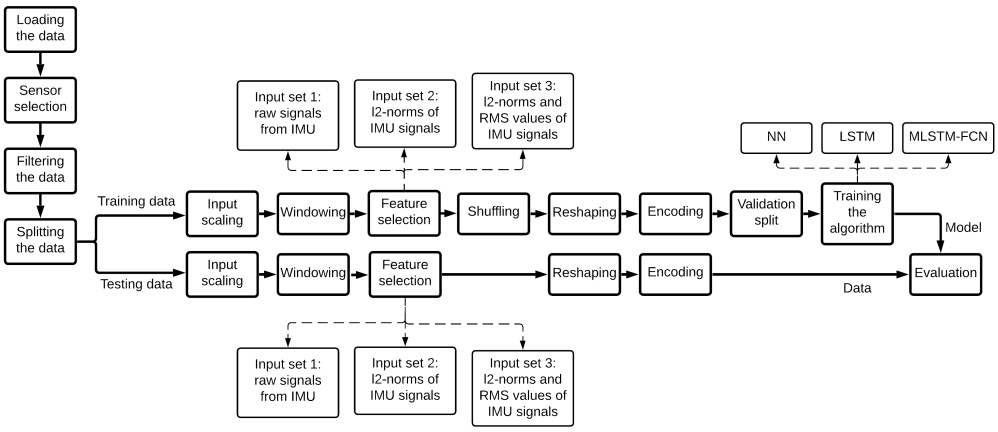


Figure 11: The steps performed in the pipeline.

4.5.1 Preprocessing data

The data sets from the six horses were combined into a single dataframe and the rows with missing values (appearing as NaN in the database instead of an actual value) were removed from the dataframe.

Data filtering The accelerometer and gyroscope measurements are inherently noisy. Thus, it is important to filter out high-frequency noise from the measurements. This is done with a low-pass Butterworth filter with a cut-off frequency of 30 Hz. Arablouei et al. [73] show that the most power in the signals ranges from 0 to 25 Hz, making 30 Hz a reasonable cut-off frequency for high-frequency noise. Next to that, Bosch et al. [83] also used this cutoff frequency, because the frequency of head movements and stride frequencies for horses are generally in the range of 0 to 4 Hz, making 30 Hz well-above the frequency signals coming from the the horses [94], [95].

Splitting the data The dataset has been split into a training and testing subset using the leave one subject out cross validation method. This method was also used by Kamminga et al. [96] for the same dataset, to make sure that the algorithm performs well on different subjects, and not just on one subject. Thus, one of the six horses was selected as the testing subject and the other five horses were included in the training subset. Next to that, the validation set was splitted from the training set, where 20% of the training data was used as validation data.

Input scaling Since some of the values were of a much larger size than others, it was important to scale them so that they are easily comparable. To do so, the accelerometer and gyroscope samples were divided by the highest value within the corresponding axis to obtain normalized values between 0 and 1. This scaling was performed separately for the train and test set, thus the training data was also not used to scale the test data.

Windowing The data has been segmented during the data acquisition. These segments contain data only about one activity. These segments were again split up into windows, each with a fixed size. The window size was set to 2 seconds, which is the same window size as Kamminga et al. [75] used with this dataset. A commonly used approach is to use a step size so that the windows following each other have 50% overlap [22], [34], [41], [74], [79]. Thus, the step size was set at 1 second.

Feature selection The l2-norms of the accelerometer and gyroscope were used as a standard input set, since they were also used by Kamminga et al. [75] on the same dataset. The main advantage of using the l2-norms is that it is orientation-independent. The l2-norms of the measurements from multiple axes can be calculated with:

$$l2(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2} \tag{2}$$

Here, l2 denotes the l2-norm, t the timestep, a the respective sensor (accelerometer or gyroscope) and x, y and z the axes of the sensor. Next to that, a test has been performed with using the 3 sensor axes from the accelerometer and gyroscope as input instead to evaluate whether this input produced better results for the classifier. Moreover, another test has been performed with the l2-norms of the sensors and the Root Mean Square (RMS) value of those l2-norms, which are more often used in HAR and AAR [16], [34], [41]. The RMS value shows the magnitude of the values, without taking the sign of the values into account. With taking the RMS value of the l2-norms, the classifier can compare the current l2-norm to the average magnitude of the l2-norms. The RMS value can be calculated with:

$$RMS = \sqrt{\frac{\sum_i x_i^2}{n}} \tag{3}$$

Here, x_i denotes each value in the window and n is the number of measurements. Thus, three input sets were tested in this experiment, which can be seen below in Tables 4, 5 and 6. The setup of these experiments will be described in Chapter 6.

Table 4: An overview of input set 1. Adapted from [12, p.4]

Feature	**Description**
Ax	Raw data from the accelerometer *x*-axis
Ay	Raw data from the accelerometer *y*-axis
Az	Raw data from the accelerometer *z*-axis
Gx	Raw data from the gyroscope *x*-axis
Gy	Raw data from the gyroscope *y*-axis
Gz	Raw data from the gyroscope *z*-axis
label	Label that belongs to each row's data
segment	Each activity has been segmented with a maximum length of 10s. Data within one segment is continuous. Segments have been numbered incrementally.
subject	Subject identifier

Table 5: An overview of input set 2. Adapted from [12, p.4]

Feature	Description
A3D	l2-norm (3D vector) of accelerometer axes
G3D	l2-norm (3D vector) of gyroscope axes
label	Label that belongs to each row's data
segment	Each activity has been segmented with a maximum length of 10s. Data within one segment is continuous. Segments have been numbered incrementally.
subject	Subject identifier

Table 6: An overview of input set 3. Adapted from [12, p.4]

Feature	Description
A3D	l2-norm (3D vector) of accelerometer axes
G3D	l2-norm (3D vector) of gyroscope axes
RMS of A3D	Root Mean Square value of the A3D signal
RMS of G3D	Root Mean Square value of the G3D signal
label	Label that belongs to each row's data
segment	Each activity has been segmented with a maximum length of 10s. Data within one segment is continuous. Segments have been numbered incrementally.
subject	Subject identifier

Shuffling and Reshaping After windowing and feature selection, the resulting windows were used as data instances for training and testing. Next to that, for two models (the LSTM and MLSTM-FCN) the number of windows has to be divisible by the batch size. Thus, for these models, the (training, testing and validation) data sets were all cropped to the maximum number of windows that is divisible by the batch size. This means that the last few samples in the dataframe were dropped. However, before performing this step, all windows from the datasets were shuffled, to ensure that those last few samples, that were removed, were random samples. For the NN no windows had to be removed, so only the windows in the training data were shuffled. All shuffling operations were performed with the `shuffle()` method from sklearn. For all classifiers the training data was shaped as M_{NxWxI}, in which M is the matrix, N the number of samples, W the window size and I the number of inputs. To fit the data into the NN, the data had to be reshaped into M_{NxWI}, in which M is the matrix, N the number of samples and WI the window size and number of inputs multiplied by each other. For the LSTM and FCNLSTM this operation was not needed, thus the data remained in its original shape.

Encoding labels In order to make the dataset more suited for most machine learning algorithms, the categorical labels had to be converted into numerical ones. However, as there is no ordinal relationship between the original categorical labels, one-hot encoding had to be applied to any numerical label representation to avoid the algorithm potentially trying to make use of any non-existent ordinal relationship. To do so, the various activity labels were first converted into numerical labels using the `LabelEncoder()` function provided within scikit-learn's preprocessing library. Following this, one-hot encoding was applied to the integer representation of the labels. The resulting encoded labels were added as an extra column to the dataframe.

4.6 Evaluation

As mentioned before, the testing phase was through leave one subject out validation. Next to that, the model was tested with the keras `predict()` method. Per horse, the experiment performance was saved in the database with the metrics explained in Chapter 4.6.1.

4.6.1 Evaluation methods

There are different evaluation methods for evaluating an algorithm. In this section, each of these methods will be discussed. First off, all results were gathered in a confusion matrix, where comparisons were made between the amount of right and wrong classifications. An example of a confusion matrix can be found in Table 7. In this confusion matrix, Positive represents an activity, for example "Walking". This means that Negative would represent all other classes, or "Not Walking". Thus, Predicted Positive would mean that the algorithm has predicted that the horse is performing the activity that corresponds with the Positive label. When the actual activity performed by the horse matches this prediction, this is called a True Positive (TP). When the algorithm predicts that the horse is not performing the activity (Predicted Negative), while the horse was actually performing the activity (Actual Positive), it is called a False Negative (FN). When the algorithm predicts that the horse is performing the activity (Predicted Positive), while the horse was actually not performing the activity (Actual Negative), it is called a False Positive (FP). Lastly, when the algorithm predicts that the horse is not performing the activity (Predicted Negative), and this prediction is correct (Actual Negative), it is called a True Negative (TN). The amount of correctly identified predictions can be found with TP+TN, while the amount of incorrectly identified predictions can be found with FN+FP.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Table 7: A confusion matrix showing the difference between classifications.

When evaluating the algorithm, these values (TP, FP, FN and TN) were used to calculate a performance score. Often-used performance scores are recall, precision, specificity, accuracy, balanced accuracy, F-score and MCC.

Recall The recall of an algorithm can be calculated with:

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

When the recall of an algorithm is high, many of the Actual Positive values have been classified correctly.

Precision The precision of an algorithm can be calculated with:

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

When the precision of an algorithm is high, many of the positive predictions were predicted correctly.

Specificity The specificity of an algorithm can be calculated with:

$$Specificity = \frac{TN}{TN+FP} \tag{6}$$

When the specificity of an algorithm is high, many of Actual Negative values have been classified correctly.

Accuracy The accuracy of an algorithm can be calculated with:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

The accuracy per experiment is calculated by taking the average of the accuracy of all activities. With a high accuracy, much more predictions are done correctly than incorrectly.

Balanced accuracy The balanced accuracy is an average of the recall of each activity. With a high balanced accuracy, much more predictions are done correctly than incorrectly for each class. The balanced accuracy score avoids a biased score towards the biggest classes. Thus, the balanced accuracy score gives a more complete view of the performance in imbalanced datasets [97].

F-score The F-score is an average of the precision and recall of an algorithm and can be calculated with:

$$F-score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

With a high F-score, many of the actual values have been classified correctly and many of the predictions were done correctly.

MCC The Matthews Correlation Coefficient (MCC) can be calculated with:

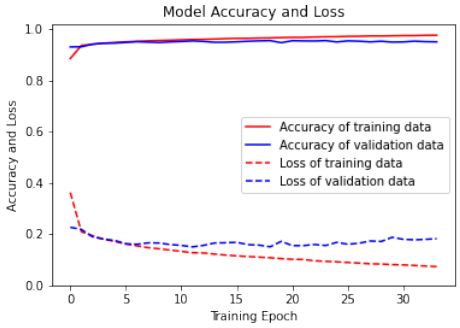
$$MCC = \frac{c * s - \sum_k^K p_k * t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) * (s^2 - \sum_k^K t_k^2)}} \tag{9}$$

Here, K = number of classes, t_k = number of times class k truly occurred, p_k = number of times class k was predicted, c = sum of all correctly predicted samples and s = total number of samples. The minimum value of the MCC score is between -1 and 0 and the maximum value is +1. Thus, it is scaled differently than the aforementioned metrics. When the MCC is at its minimum value (between -1 and 0), all predictions were done incorrectly. On the other hand, when the MCC is at its maximum value (+1), all predictions were done correctly. The MCC score is able to give a complete view of the performance on unbalanced data sets, unlike the accuracy and F-score [98].

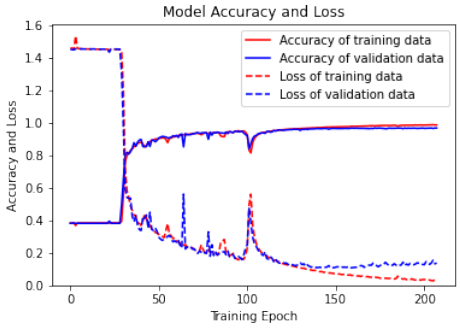
4.7 Hyperparameters

Batch size The reviewed state of the art papers differ from 20 to 1500 as the batch size [10], [34]–[36], [42], [71], although they are generally in the range 20 to 150. When using Tensorflow, it is best to use a batch size which is a power of two [99]. Due to time constraints, not multiple batch sizes were tested. Instead, the batch size was set at 256.

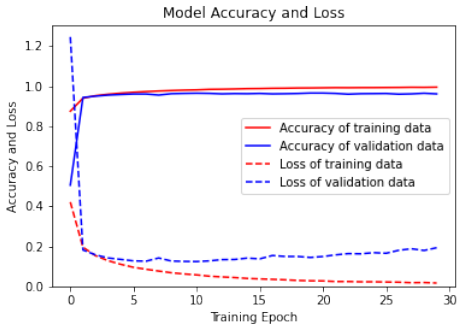
Number of epochs A training cycle consists of multiple epochs, which can differ a lot. From the reviewed state of the art, one study [87] only used two epochs, while another [79] trained for 1000 epochs. Each model was run for a maximum of 300 epochs, but to make sure the model does not overfit, an early stopping point has been implemented. The early stopping point was used to stop training at the point where the model was using the best weights, based on the validation set. This means that the validation performance score was the highest at the early stopping point. When continuing training, this point may be surpassed and the validation performance may decrease again. Thus, an early stopping point was introduced when the accuracy did not improve for ten epochs for the NN and the MLSTM-FCN, because these networks improved significantly at the start and soon reached its early stopping point, as can be seen in Figures 12a, and 12c. The early stopping point for the LSTM was set at 35 epochs, because the LSTM often needed ten to thirty epochs to vastly improve its learning, as can be seen in Figure 12b.



(a)



(b)



(c)

Figure 12: An example of a training cyclus for an (a) NN, (b) LSTM and (c) MLSTM-FCN.

Optimizer and learning rate The optimizer used in all models was Adam, which is efficient and a good fit for problems that have a lot of parameters or data [100]. The learning rate within this optimizer was set at 0.001 for the NN and LSTM, similar to Pucci et al. [79] and Elsworth and Güttel [92] and 0.0001 for the MLSTM-FCN, similar to Karim et al. [71].

4.8 Database

By the use of the Python package Peewee, a database was created to easily store experimental data. This database is connected to a virtual server. There are two tables, one for the experiment results per horse and one for the experiment results for a specific activity per horse. The experiment table consists of several columns, containing a summary of the results from all activities:

- Key: a unique key was generated per experiment with the uuid Python package.
- Username: The name of the person who performed this experiment.
- Horse: The name of the horse which was in the test dataset.
- Date: The date on which the experiment was conducted.
- Accuracy: The overall accuracy of this experiment.

- Balanced accuracy: The overall accuracy of this experiment, balanced on the amount of samples per class.
- F-score: The overall weighted F-score of this experiment.
- MCC: The overall MCC of this experiment.
- Confusion matrix: The confusion matrix of this experiment.
- Parameters: The parameters used for this experiment, including sliding window size, step distance, batch size, number of epochs, number of layers, number of cells per layer, dropout rate and learning rate.
- Description: A description of the specific settings used for this experiment. This can, for example, entail of the structure of the classifier. The description is added in text-format.

The activity table also has several columns, focusing on each activity within an experiment:

- Key: A unique key, corresponding to the experiment.
- Horse: The name of the horse which was in the test dataset for this activity.
- Activity: The name of this activity.
- Accuracy_activity: The accuracy of this activity.
- Recall_activity: The recall of this activity.
- Specificity: The specificity of this activity.
- Precision: The precision of this activity.
- TP: The number of times the system predicted Positive and the actual value was also Positive (True Positive).
- TN: The number of times the system predicted Negative and the actual value was also Negative (True Negative).
- FP: The number of times the system predicted Positive and the actual value was Negative (False Positive).
- FN: The number of times the system predicted Negative and the actual value was Positive (False Negative).

4.9 Tools

In this Graduation Project multiple tools were used, which will be described below.

4.9.1 ITC Geospatial Computing Portal

The ITC Geospatial Computing Portal from the University of Twente was used to run the code on [101]. This portal allows, among other programming languages, Python 3 code through the JupyterLab environment [102]. Apart from running the code, the CSV files containing the raw IMU data could also be stored on this server.

4.9.2 Programming language and packages

For this project Python 3 was used with a couple of packages.

pandas The Pandas package supports data analysis and data manipulation, allowing the user to retrieve and shape the data [103].

numpy The Numpy package allows mathematical calculations [104]. In this case, these calculations were used for conversions between number types (integer, float, etc.), retrieving maximum values and shaping arrays.

scikit-learn The scikit-learn (or sk-learn) library is made for predictive data analysis [97]. In this project, it was used to define the metrics and for preprocessing.

keras The keras package focuses on Deep Learning, including methods for implementing a Deep Learning algorithm [105].

peewee The peewee package was used to implement a database to store and retrieve the results of experiments [106].

seaborn Seaborn was used for data visualization [107]. In this case, seaborn was used to plot a confusion matrix.

Other packages Other packages used are scipy, glob, matplotlib, uuid, IPython, datetime, re and ast.

5 Specification

5.1 Requirements

In this section, requirements will be set for the AAR algorithms to answer the following sub-research question:

What are the requirements for an animal activity recognition algorithm that can classify multiple activities with an adequate performance?

A full overview of the requirements can be seen in Table 5.1. Since the pipeline is developed with other students, a requirement has been set on the use of Python as a programming language (requirement 1) so that each program uses the same language. Apart from that, Python is often used within data analysis, making it a very suitable language choice [108]. The data that will be used as an input into the algorithm is fully-labeled time-series data that has already been gathered. Thus, it is important that the algorithm is able to handle fully-labeled, time-series and offline data (requirements 2, 3 and 4). Next to that, the algorithm must be able to perform in a reasonable timeframe, in order to be able to run it multiple times and use it reasonably in the future (requirement 5). After performing the test and train phase, the algorithm must be able to be evaluated through a confusion matrix, and an overall accuracy, F-score and MCC (requirement 6). Out of the three (Leave One Out, Simple Split and Out-Of-Bag) most occuring evaluation methods, Leave One Out is most applicable and fair. Thus, the algorithm must be able to evaluate itself through the Leave One Out method (requirement 7).

It is of importance that there are at least a few different activities that can be recognized, because otherwise the data would still not be useful enough to analyze general behavior from the animals. The same dataset has previously been used in a study by Kamminga et al. [75] to classify activities with a simple classifier, where the denoted accuracy is 81% and the F-score 73%. Kamminga et al. used only data of the most annotated six horses performing six activities. Thus, a more advanced classifier should at least achieve an accuracy of 81% and an F-score of 73% on the classification of six activities by six horses (requirement 8).

5.2 Challenges

In Chapter 3.3, the limitations within the state of the art were discussed, of which some limitations are also challenges within this project. First, the imbalance of the dataset can be a challenge, because this can lead to a bias towards the bigger classes. Moreover, the classification of some of the activities is also a challenge when the activities are very similar. This may lead to misclassifications between the similar activities. Apart from that, some of the activities might also be performed at the same time, making it hard to label them under just one activity. Thus, the imbalance of the dataset, along with the similarities between activities are challenges for this project.

Requirements
1. The program must be written in Python.
2. The system must be able to handle fully-labeled data.
3. The system must be able to handle time-series data.
4. The system must be able to handle offline data.
5. The system must be able to classify the data in a reasonable timespan.
6. The system must be able to show and save the performance results of an experiment in a database, including the confusion matrix, overall accuracy, F-score and MCC.
7. The system must be able to evaluate itself with the Leave One Out evaluation method.
8. The pipeline must perform with a minimum accuracy of 81% and F-score of 73% on the classification of six activities by six horses.

Table 8: The requirements for a horse activity recognition algorithm.

The data used is orientation dependent, which adds an additional challenge for the classifier. Moreover, the heterogeneity of the animals is also relevant, since each animal generates different measurement results for each activity. This can be a challenge because the classifier needs to be able to generalize its results towards unseen subjects. In conclusion, the orientation dependence of the data, along with the heterogeneity of the subjects are challenges for this project.

There are many steps in an animal activity recognition pipeline which can be challenge for this project. First, choosing the right classifier for the data can be a challenge, since there are many different classifiers and each has its own advantages and disadvantages. Additionally, each classifier has its own hyperparameters which need to be tuned correctly for the classifier to perform optimally. Both choosing the classifier and its hyperparameters can be a challenge because of the variety of options available and its significant influence on the performance of the pipeline. Apart from that, there are different methods for splitting the data, windowing and feature selection. For example, the dataset needs to be split up between the training and testing data, for which a single split or a leave one subject out validation split method can be used. These choices can lead to different performances, where some methods give a more complete image of the performance of the classifiers than others. Thus, it can be a challenge to find the most optimal method for each of these steps in the pipeline. Last, the last step in the pipeline is the evaluation, for which different metrics can be used. Each metric has its disadvantages and none of these metrics can give a complete view of how well the classifier performed, which can be a challenge while evaluating the classifiers. Conclusively, many methods used in the pipeline can be considered a challenge in this project.

6 Evaluation

In this section, the evaluation of the NN, LSTM and MLSTM-FCN will be presented. First, an experiment is discussed, in which different input sets were inputted to the NN, LSTM and MLSTM-FCN to see whether a different input set has a positive influence on the performance of the classifier. The order of the experiments is explained in Figure 13. First, the batch size, window size, number of epochs and learning rate were set, as described in Chapter 4.5.1and Chapter 4.7. These are described as the fixed parameters in Figure 13. Afterwards, a grid search was performed for each of the input sets, which were described in Chapter 4.3. The hyperparameters used for each classifier can be seen in Table 10. Lastly, the performance of the three classifiers is compared, each run with its own optimal set of hyperparameters but with the same input set. With this, the main research question will be answered:

Which animal activity recognition algorithm performs the best on IMU horse data, and what aspects lead it to outperform other algorithms?

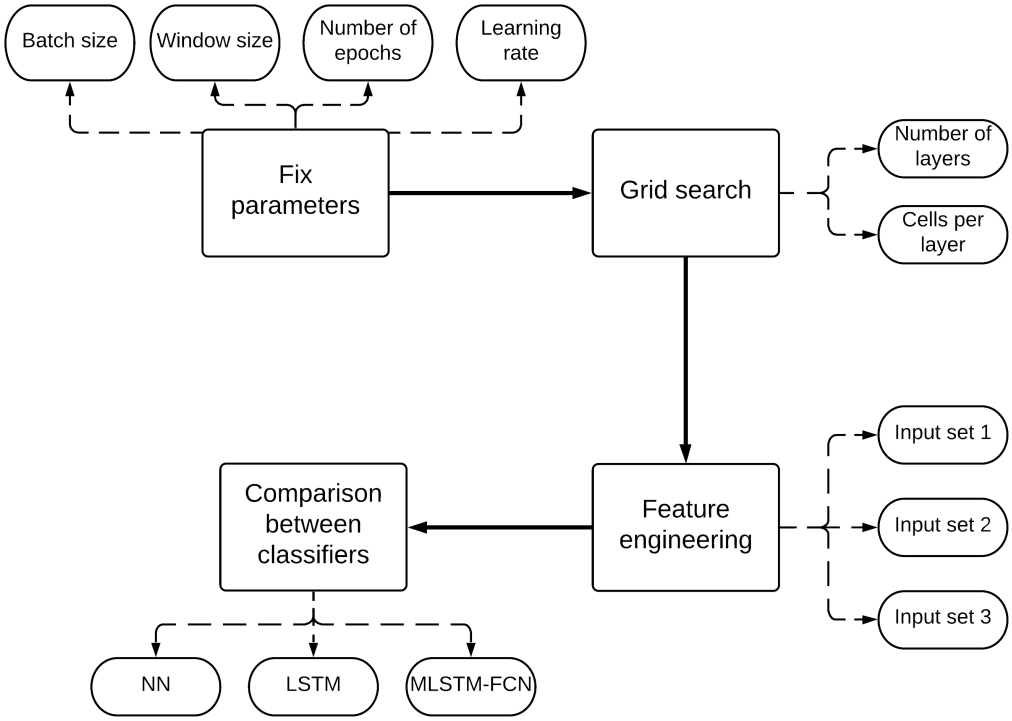


Figure 13: A diagram showing the steps in which the evaluation was performed.

Table 9: A brief overview of the experiments performed in Chapter 6.

Classifier	Hyperparameters
NN	Number of layers (1, 2) Number of cells per layer (100, 200)
LSTM	Number of layers (1, 2) Number of cells per layer (50, 200)
MLSTM-FCN	Number of cells in the LSTM layer (8, 50) Number of cells in the 1st & 3rd convolutional layer (128, 256) Number of cells in the 2nd convolutional layer (128, 256)

6.1 Feature engineering

In this subsection, the effect of the different input sets is shown, based on the performance of the NN, LSTM and MLSTM-FCN. The classifiers were trained on the three input sets described in Table 4, 5 and 6.

6.1.1 Neural Network

First, the hyperparameters were tuned for each model with a different input set. Generally, one or two hidden layers were used in a neural network according to the reviewed state-of-the-art studies [35], [70], [73]. Therefore, also networks with one and two layers were tested in this experiment. Moreover, the number of cells in those layers differs from 5 to 256 within the reviewed state of the art [35], [79]. Therefore, 100 and 200 cells per layer were tested.

First, the hyperparameters were tuned for the NN with input set 1 as input. Figure 14 shows that the models with 100 cells per layer yielded higher performance scores among all metrics than the models with 200 cells per layer. The performance scores for the networks with 100 cells per layer yielded similar performance scores and similar standard deviation. The configuration of 1 layer containing 100 cells was chosen as the configuration which was used to compare input set 1 with input set 2 and 3.

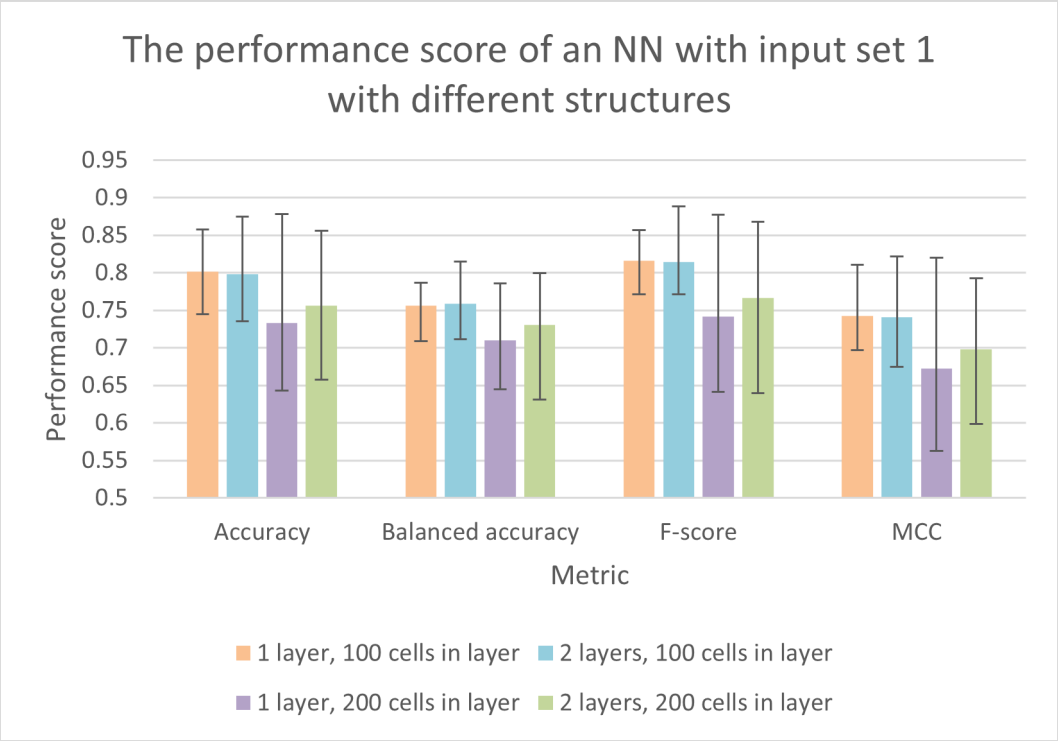


Figure 14: The performance score of an NN with input set 1 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Next, the hyperparameters were tuned for the NN with input set 2. Figure 15 shows that generally, the model seemed to yield similar results for all hyperparameters. However, the models with 100 cells per layer had a significantly higher standard deviation, where the reported accuracy differed between 55.4% and 87.6%. The performance scores for the networks with 200 cells per layer yielded similar performance scores and similar standard deviation. The configuration of 2 layers, with both layers containing 200 cells, was chosen as the configuration which was used to compare input set 2 with input set 1 and 3.

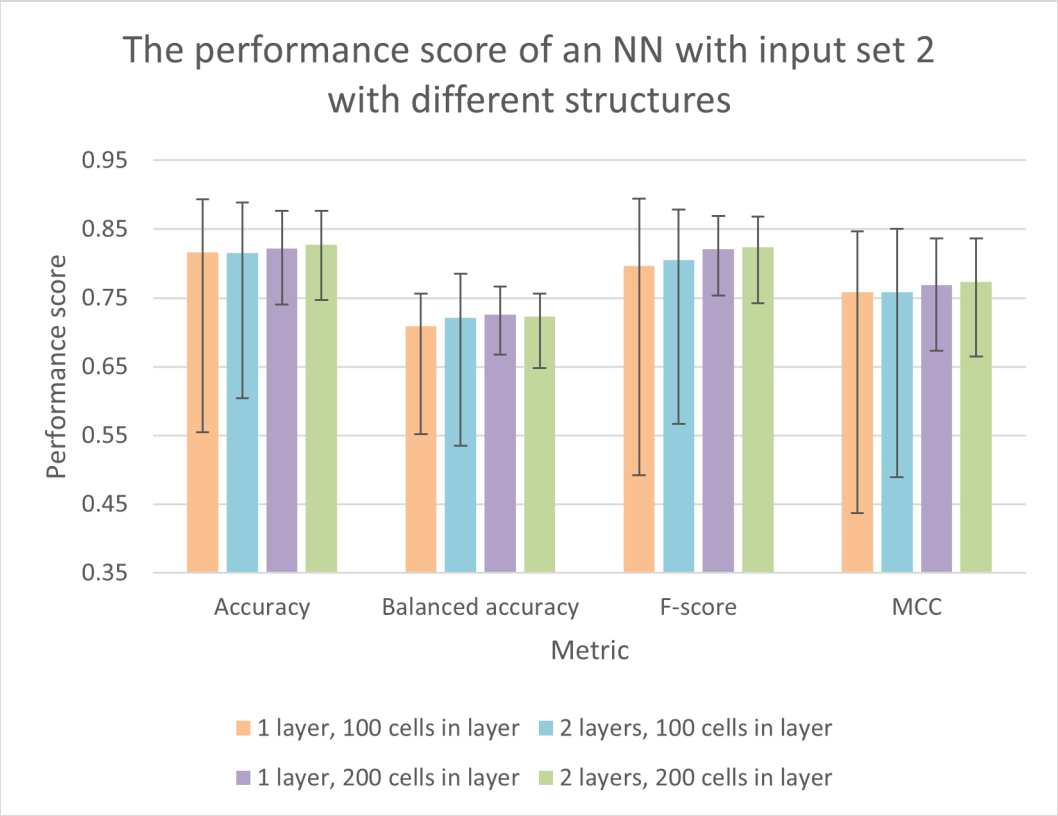


Figure 15: The performance score of an NN with input set 2 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Lastly, the hyperparameters were tuned for the NN with input set 3. Figure 16 shows that the NN with 200 cells per layer yielded higher performances than the models with 100 cells per layer. Moreover, the configuration with only 1 layer yielded slightly higher performance scores than the configuration with 2 layers. Thus, the configuration with 1 layer containing 200 cells was chosen as the configuration that was used to compare input set 3 with input set 1 and 2.

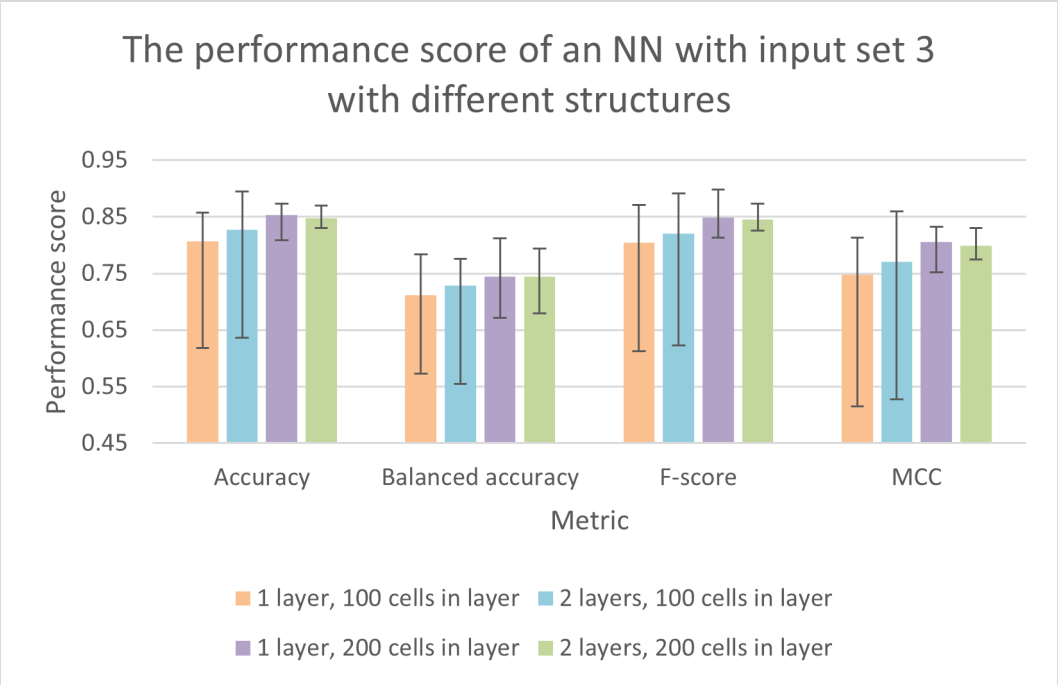


Figure 16: The performance score of an NN with input set 3 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

The performances from the best configurations were compared with each other, which can be seen in Figure 17. The NN with input set 3 yielded the highest accuracy (85.3%), F-score (84.9%) and MCC (0.806). However, the NN with input set 1 yielded a higher balanced accuracy score (75.7%) than the NN with input set 2 (72.3%) and input set 3 (74.5%). On the other hand, the NN with input set 2 yielded lower performance scores than the NN with input set 3 among all metrics. The NN with input set 1 yielded significantly lower performance scores than the NN with input set 3 on accuracy (80.2% against 85.3%), F-score (81.6% against 84.9%) and MCC (0.74 against 0.81). However, the NN with input set 1 yielded a higher performance score based on the balanced accuracy (75.7% against 74.5%). This can be explained with Figure 18, where the NN with input set 1 yielded significantly less False Negative misclassifications (2724 FN, 6407 FP) than the NN with input set 2 (3913 FN, 1870 FP) and 3 (3843 FN, 1246 FP) in the hardest class to classify (walking natural), which is also the class with the least amount of samples. On the other hand, the NN with input set 2 and 3 yielded significantly less False Positive misclassifications (1870 and 1246 respectively) than the NN with input set 1 (6407). Thus, the NN with input set 1 generally classified more samples as walking natural, and with this, decreased its accuracy, but increased its balanced accuracy. Overall, it seems that the NN with input set 3 scored the best performance results.

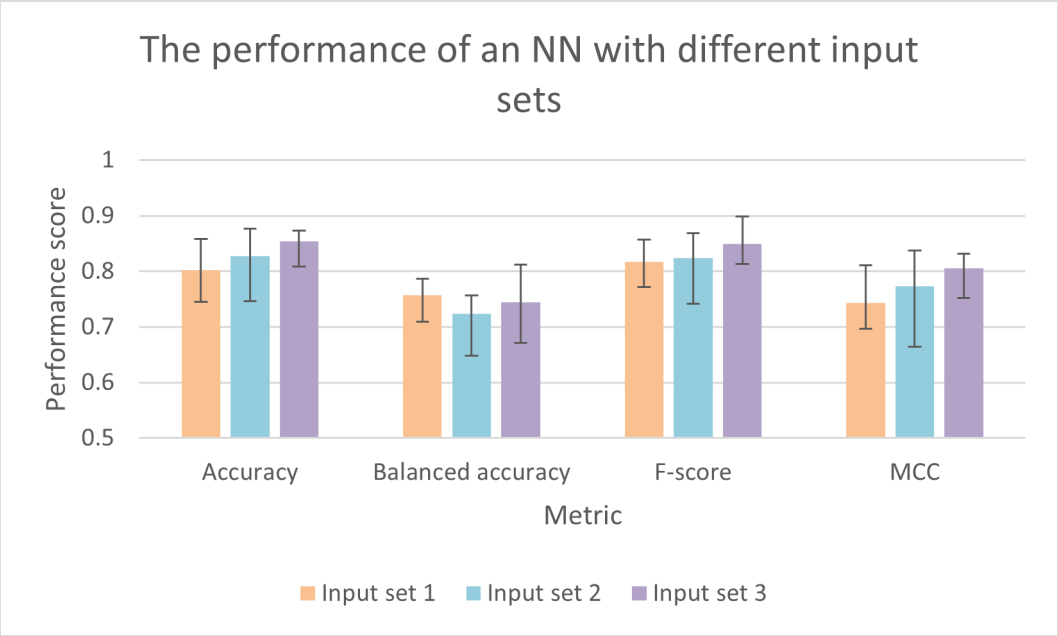
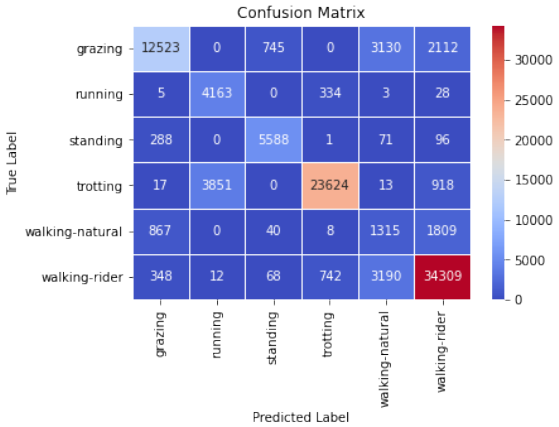
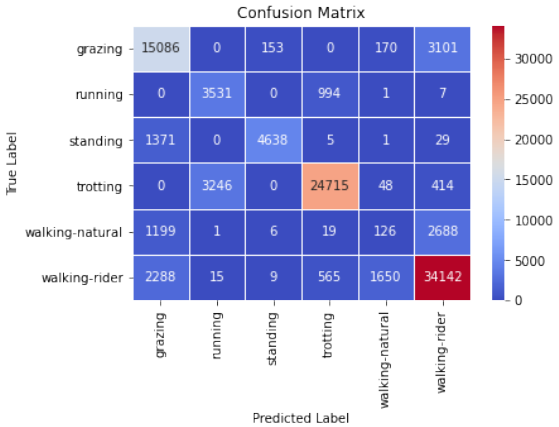


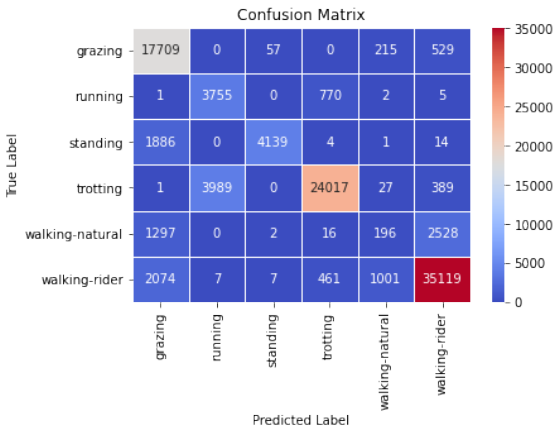
Figure 17: The performance score of an NN with different input sets. The error bars denote the standard deviation over the folds through leave one subject out validation.



(a)



(b)



(c)

Figure 18: The confusion matrix of an NN with input set (a) 1, (b) 2 and (c) 3.

6.1.2 Long Short-Term Memory

First, the hyperparameters were tuned for each model with a different input set. Generally, one or two LSTM layers are used in an LSTM network by the reviewed state-of-the-art [10], [34], [70], which was also the number of layers that was tested in this experiment. Moreover, for the state of the art LSTM models, often 32 to 50 cells per layer were used [10], [20], [34], [92]. Thus, 50 cells per layer were chosen. However, also more cells per layer were tested to see if this improves the performance. Therefore, 50 and 200 cells per layer were tested.

First, the hyperparameters were tuned for the LSTM with input set 1 as input. Figure 19 shows that the models with only 1 layer yielded the highest performance scores. Especially the model with 2 layers, containing 200 cells yielded low performance scores (62.8% accuracy, 56.4% balanced accuracy, 58.1% F-score and 0.487 MCC) with a high standard deviation. Overall, the model with 1 layer, containing 200 cells per layer, yielded the highest performance scores (79.7% accuracy, 79.2% balanced accuracy, 81.3% F-score and 0.763 MCC). Thus, the model with 1 layer, containing 200 cells per layer, was chosen as the configuration which was used to compare input set 1 with input set 2 and 3.

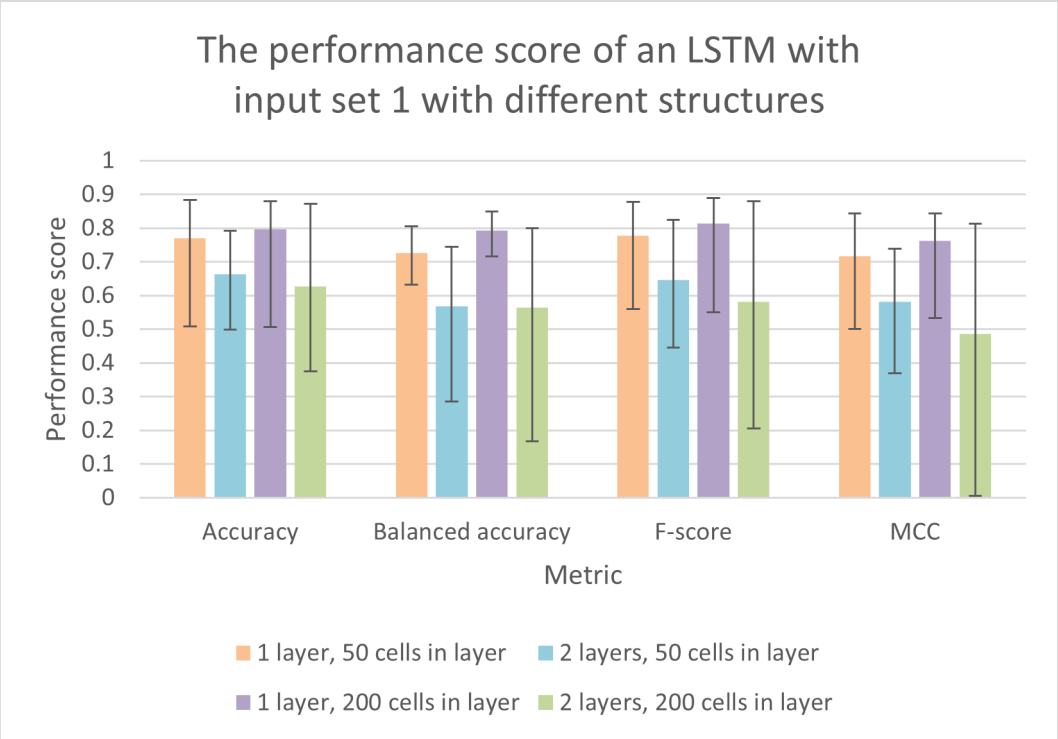


Figure 19: The performance score of an LSTM with input set 1 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Next, the hyperparameters were tuned for the LSTM with input set 2. Figure 20 shows that the configurations with 200 cells per layer yielded a better performance than the configurations with 100 cells per layer. Moreover, the LSTM with only one layer containing

50 cells had a significantly higher standard deviation than the other models. Both models containing 200 cells per layer yielded similar results with similar standard deviation rates. The configuration of 2 layers, with both layers containing 200 cells, was chosen as the configuration which was used to compare input set 2 with input set 1 and 3.

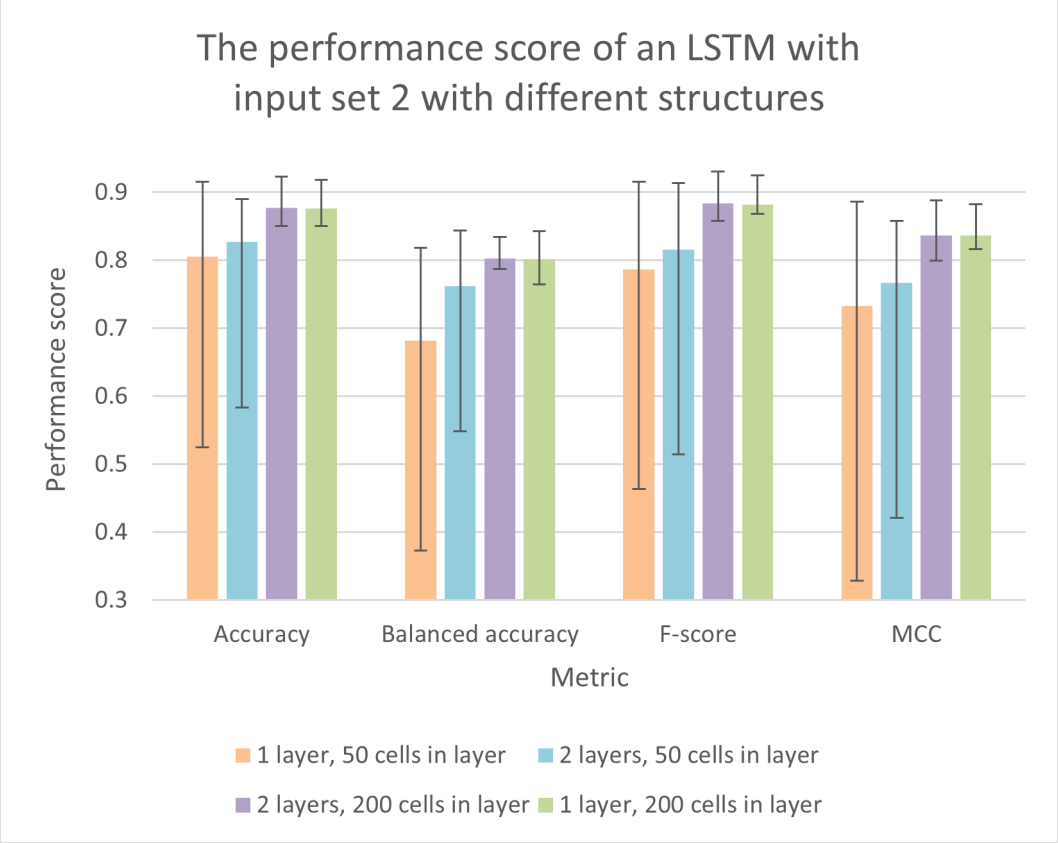


Figure 20: The performance score of an LSTM with input set 2 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Lastly, the hyperparameters were tuned for the LSTM with input set 3. Figure 21 shows that the LSTM with 2 layers, containing 50 cells, yielded a higher performance than the other models, based upon the accuracy, F-score and MCC. On the other hand, the model with 1 layer, containing 200 cells, yielded the highest performance based on the balanced accuracy score. However, these performances yielded quite similar scores based upon the balanced accuracy (79.2% against 80.0%). Thus, the configuration with 2 layers containing 50 cells was chosen as the configuration that was used to compare input set 3 with input set 1 and 2.

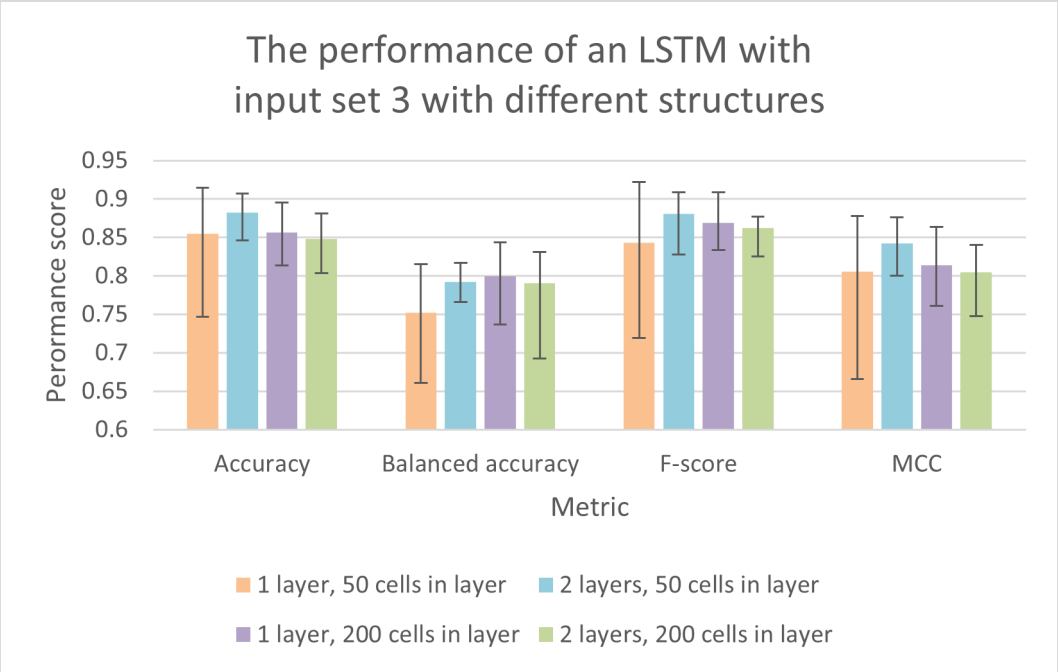


Figure 21: The performance score of an LSTM with input set 3 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

The performances from the best configurations are compared with each other, which can be seen in Figure 22. The LSTM with input set 2 and 3 yielded the highest accuracy (87.7% and 88.2% respectively), F-score (88.4% and 88.0% respectively) and MCC (0.837 and 0.843 respectively). The LSTM yielded a significantly lower accuracy (79.7%), F-score (81.3%) and MCC (0.763) with input set 1. Interestingly, the LSTM yielded a similar performance for input set 1 (79.2%), 2 (80.3%) and 3 (79.2%) when measuring the balanced accuracy. This difference can be explained with Figure 23, which shows that the LSTM could generally not classify the walking natural class well, yielding many misclassifications for all input sets (1957, 2795 and 3755). The LSTM with input set 1 had the least misclassifications in this class, although it had significantly more misclassifications among the classes grazing (7093) and walking rider (31578). This gave the LSTM with input set 1 a relatively low accuracy (79.7%), but a relatively high balanced accuracy (79.2%). Overall, it seems that the LSTM was best at accurately classifying all classes with input set 2 or 3.

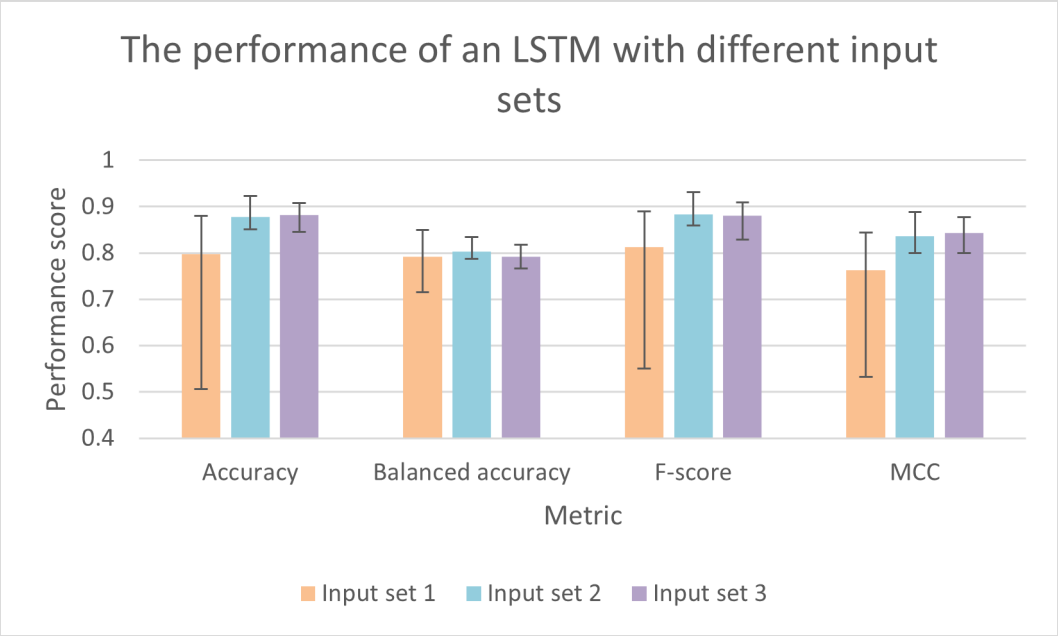
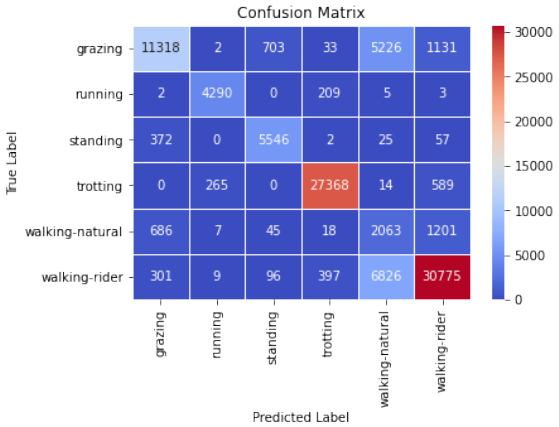
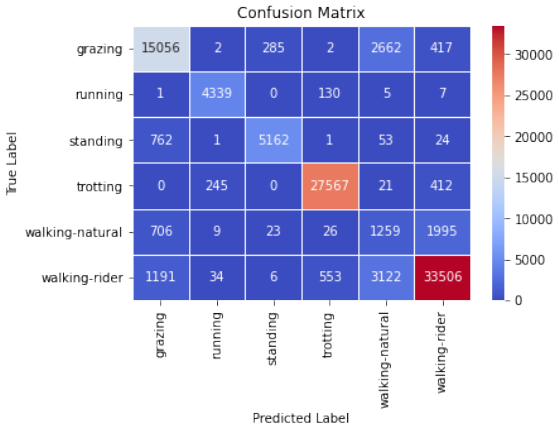


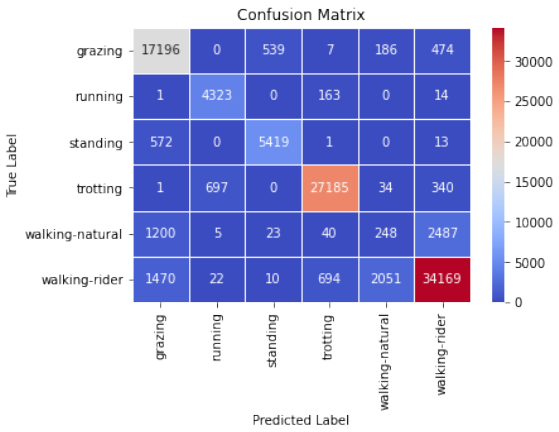
Figure 22: The performance score of an LSTM with different input sets. The error bars denote the standard deviation over the folds through leave one subject out validation.



(a)



(b)



(c)

Figure 23: The confusion matrix of an LSTM with input set (a) 1, (b) 2 and (c) 3.

6.1.3 Multivariate Long Short-Term Memory Fully Convolutional Network

The number of cells per LSTM layer were used from the study of Karim et al. [71], which was eight cells per LSTM layer. Next to that, a higher number (50) cells per LSTM layer was tested to see if this causes an increase in performance. Moreover, the number of cells in the convolutional layers was tested to see if this improved the performance. The first and third layer originally consisted of 128 cells. In this study, the model was tested once with 128 and again with 256 cells in the first and third layers. The second layer originally consisted of 256 cells and also for this layer, 128 and 256 cells were tested. All configurations that were tested can be found in Table 9.

First, the hyperparameters were tuned for the MLSTM-FCN with input set 1 as input. Figure 24 shows that the model with 8 cells in the LSTM layer and 128 cells in the convolutional layer and the model with 50 cells in the LSTM layer and 256 cells in the convolutional layers yielded higher performance scores among all metrics than the other models. Among those two best-performing models, the performance scores are similar. The configuration of 8 cells in the LSTM layer and 128 cells in the convolutional layers was chosen as the configuration which was used to compare input set 1 with input set 2 and 3.

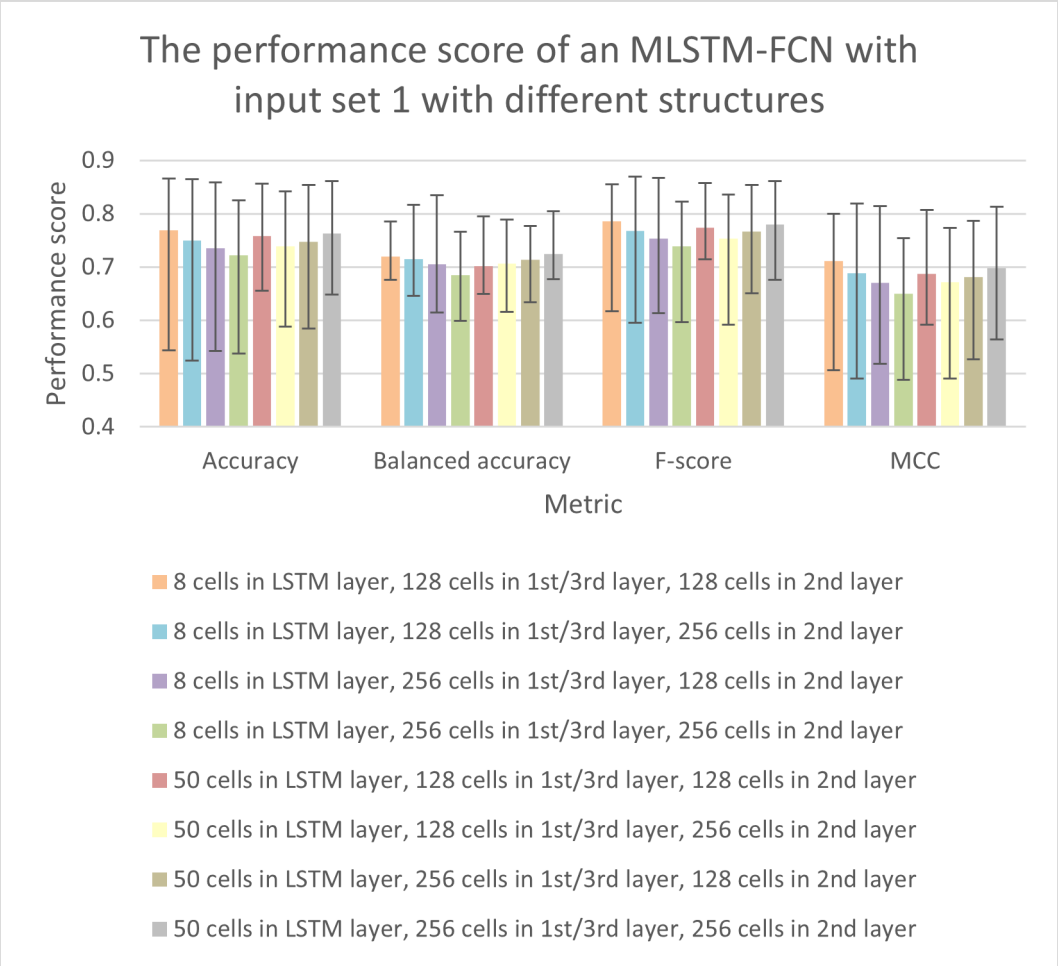


Figure 24: The performance score of an MLSTM-FCN with input set 1 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Next, the hyperparameters were tuned for the MLSTM-FCN with input set 2. Figure 25 shows that the configuration with 50 cells in the LSTM layer and 256 cells in the convolutional layers yielded the highest performance results. Additionally, the standard deviation was much higher (16 to 18%) for some of the other models than for the model with 50 cells in the LSTM layer and 256 cells in the convolutional layers (7%). The configuration of 50 cells in the LSTM layer and 256 cells in the convolutional layers was chosen as the configuration which was used to compare input set 2 with input set 1 and 3.

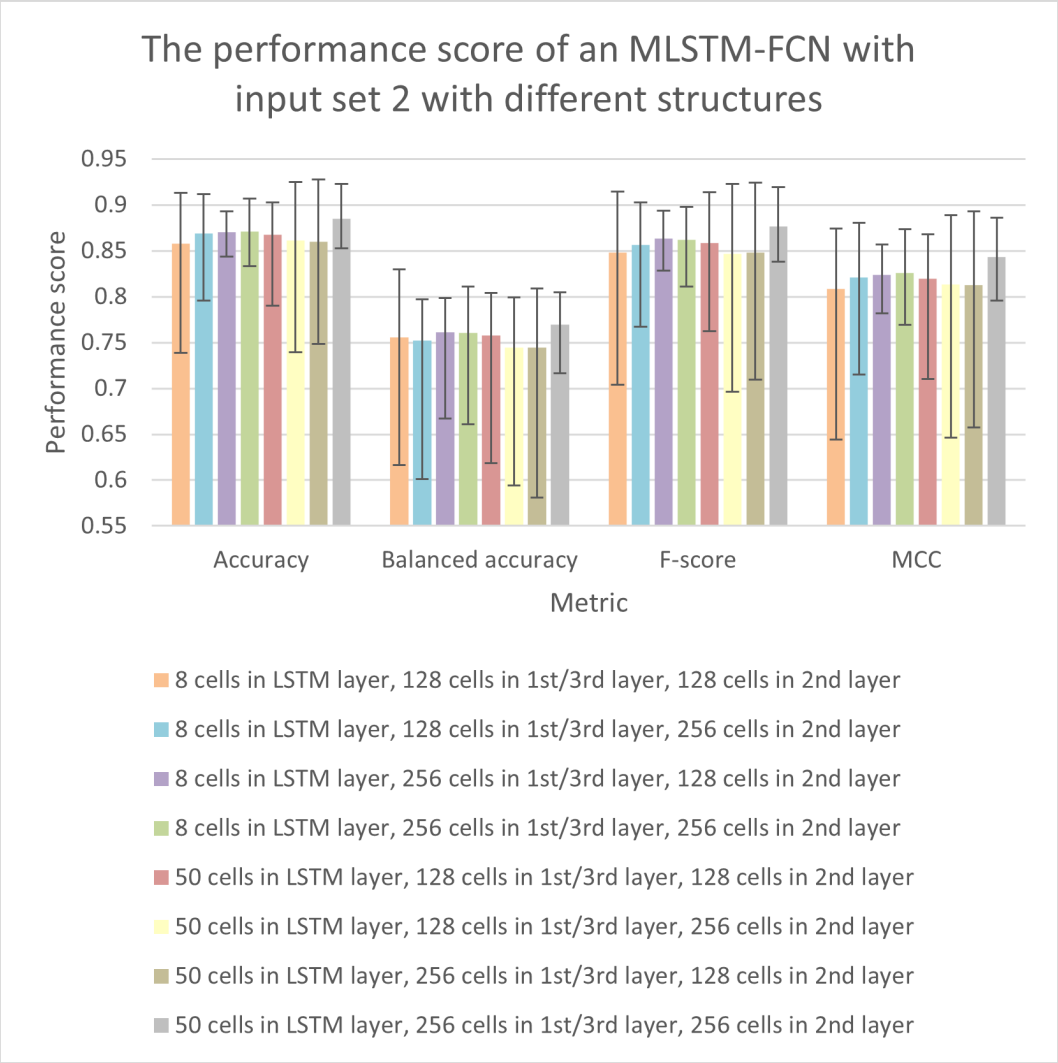


Figure 25: The performance score of an MLSTM-FCN with input set 2 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

Lastly, the hyperparameters were tuned for the MLSTM-FCN with input set 3. Figure 26 shows that the models all yielded quite similar performance results. Overall, the MLSTM-FCN with 50 cells in the LSTM layer and 256 cells in the convolutional layers, along with the model with 50 cells in the LSTM layer, 128 cells in the first and third convolutional layers and 256 cells in the second layer yielded the highest performance, with similar results. The configuration with 50 cells in the LSTM layer and 256 cells in the convolutional layers was chosen as the configuration that was used to compare input set 3 with input set 1 and 2.

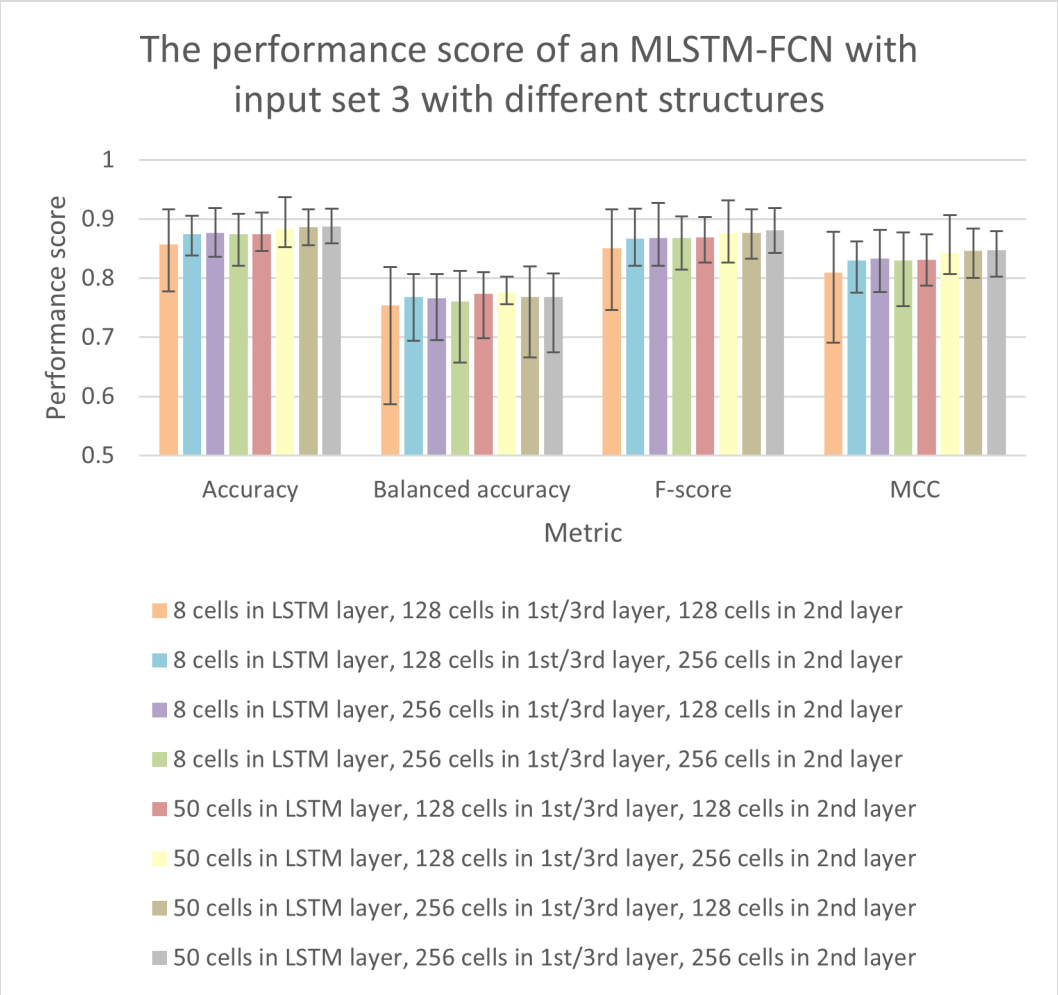


Figure 26: The performance score of an MLSTM-FCN with input set 3 with different structures. The error bars denote the standard deviation over the folds through leave one subject out validation.

The performances from the best configurations were compared with each other, which can be seen in Figure 27. The MLSTM-FCN with input set 1 yielded significantly lower performance scores than the MLSTM-FCN with input set 2 and 3. Next to that, the MLSTM-FCN with input set 2 and 3 yielded similar results, with input set 3 yielding a slightly higher standard deviation for the balanced accuracy (13.3%) than the MLSTM-FCN with input set 2 (8.8%). Conclusively, the MLSTM-FCN scored significantly higher performance scores when using input set 2 or 3 than using input set 1.

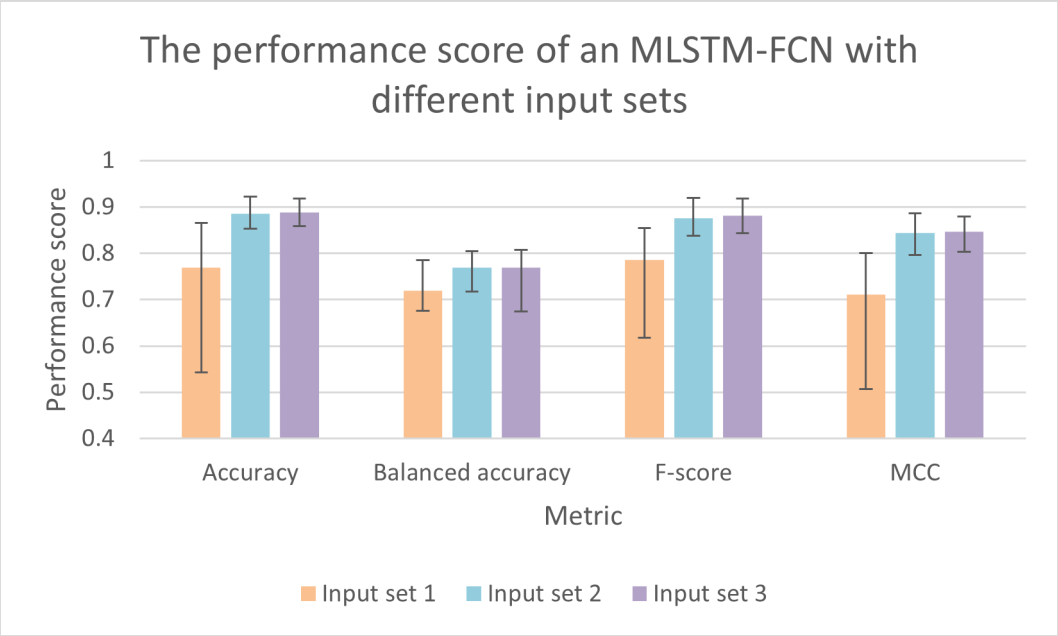


Figure 27: The performance score of an MLSTM-FCN with different input sets. The error bars denote the standard deviation over the folds through leave one subject out validation.

6.1.4 Conclusion

The influence of three input sets (1, 2 and 3) were tested with three classifiers (NN, LSTM and MLSTM-FCN). For the NN, input set 1 and input set 3 seemed to be the best input sets, where input set 3 yielded the highest performances when measuring the accuracy (80.2%, 82.8% and 85.3% for input sets 1, 2 and 3), F-score (81.6%, 82.3% and 84.9% for input sets 1, 2 and 3) and MCC (0.743, 0.773 and 0.806 for input sets 1, 2 and 3) and input set 1 yielding a higher performance score when measuring the balanced accuracy (75.7%, 72.3% and 74.5% for input sets 1, 2 and 3). For the LSTM, both input set 2 and 3 yielded higher performance results than input set 1 on all metrics. Among input set 2 and 3, the classifier yielded similar performances. The MLSTM-FCN yielded similar results as the LSTM, also yielding the highest results with input set 2 and 3. Overall, input set 1 generally yielded quite a low accuracy, F-score and MCC for all classifiers, but yielded a relatively high balanced accuracy score. However, for the LSTM and MLSTM-FCN, input sets 2 and 3 still yielded higher balanced accuracy scores.

Interestingly, it was expected that input set 1 would yield a lower performance on the NN, since this data is quite complex, and the model is quite simple. It is important to note that the NN yielded a lower accuracy with input set 1 than with the other input sets, it just yielded a higher balanced accuracy than with the other input sets. This is mainly because the NN with input set 1 yielded a significantly lower FN rate in standing (396) and walking natural (2724) but a significantly higher FN rate in grazing (4360) than the NN with input set 2 (standing: 1406, walking natural: 3913, grazing: 3424) and 3 (standing: 1905, walking natural: 3843,

grazing: 801). This difference might be because the standing and walking natural activities suffer less from being rotation dependent, which input set 1 is. On the other hand, the grazing activity might benefit from having orientation independent signals, as in input set 2 and 3. This is confirmed by the fact that the LSTM also yielded a higher FN rate from the grazing activity with input set 1 (7095) than 2 (3368) and 3 (1206) and a lower FN rate from the standing and walking natural activities with input set 1 (456 and 1957 respectively) than 2 (841 and 2759 respectively) and 3 (586 and 3755 respectively).

When evaluating input set 2 and 3, it was clear that the NN yielded a higher performance with input set 3 than with input set 2, thus benefiting from the added RMS values in this input set. This might be because the NN is not able to grasp the long-term dependencies of the data, such as the overall magnitude, very well. On the other hand, the LSTM and MLSTM-FCN might be able to calculate a feature, such as the RMS, by themselves, thus not yielding a significant difference in performance. Since input set 3 yielded high performance results for all classifiers, this input set is recommended to use for further AAR projects.

6.2 Comparison between classifiers

In this subsection, the performance results of the NN, LSTM and MLSTM-FCN are presented. Since input set 2 was used by Kamminga et al. [75] with the same dataset, input set 2 was also used in this experiment, so the results can be compared with the results of Kamminga et al. This input set is described in Table 5. The hyperparameter tuning for the three classifiers with input set 2 can be found in Chapter 6.1.1, 6.1.2 and 6.1.3. The optimal hyperparameters for each classifier, with input set 2, can be found in Table 10.

Table 10: The hyperparameters for which the highest performance is yielded with input set 2 by the NN, LSTM and MLSTM-FCN.

	NN	LSTM	MLSTM-FCN
Number of layers	2	2	-
Cells per layer	200 (Dense)	200 (LSTM)	8 (LSTM) 128 (Convolutional)

Please note: a dash (-) means this setting was not tested.

The performance results of the three classifiers with input set 2, along with the performance from the Naive Bayes (NB) classifier from Kamminga et al. [75] are shown in Figure 28. The balanced accuracy and the MCC score of the NB from Kamminga et al. are unknown. First, the NN, LSTM and MLSTM-FCN all outperformed the NB classifier Additionally, the NN was outperformed by the LSTM and MLSTM-FCN on all metrics. Moreover, the standard deviation was also bigger for the NN (11% to 17%) than for the LSTM (5% to 9%) and MLSTM-FCN (7% to 9%) for all metrics. Next to that, the LSTM and

MLSTM-FCN yielded similar performance results, where the MLSTM-FCN outperformed the LSTM based on accuracy (88.5% to 87.7%) and MCC (0.843 to 0.837), but the LSTM outperformed the MLSTM-FCN based on balanced accuracy (80.3% to 76.9%) and F-score (88.4% to 87.6%). As shown in Figure 29, the MLSTM-FCN had a lower False Negative misclassification rate for the walking with a rider (3322) and grazing (2166) activities than the LSTM for walking with a rider (4906) and grazing (3368). Since these two activities are also among two of the three biggest classes, this increased the accuracy of the MLSTM-FCN. On the other hand, the LSTM especially outperformed the MLSTM-FCN in terms of running (143 FN against 265 FN), walking natural (2759 FN against 3712 FN) and standing (841 FN against 1424 FN), which are the three smallest classes. Overall, it can be seen in Figure 30, that all classifiers yielded a bad performance in the walking natural activity. This is probably due to the fact that the walking natural activity is the smallest class, therefore the classifiers are biased towards the other classes. Apart from that, the activity is also very similar to the grazing and walking rider activities. Even though all classifiers performed poorly on the walking natural activity, the LSTM performed slightly better than the other classifiers on this class, yielding a recall of 25.7% on the walking natural activity, as opposed to 8.8% for the MLSTM-FCN and 9.0% for the NN.

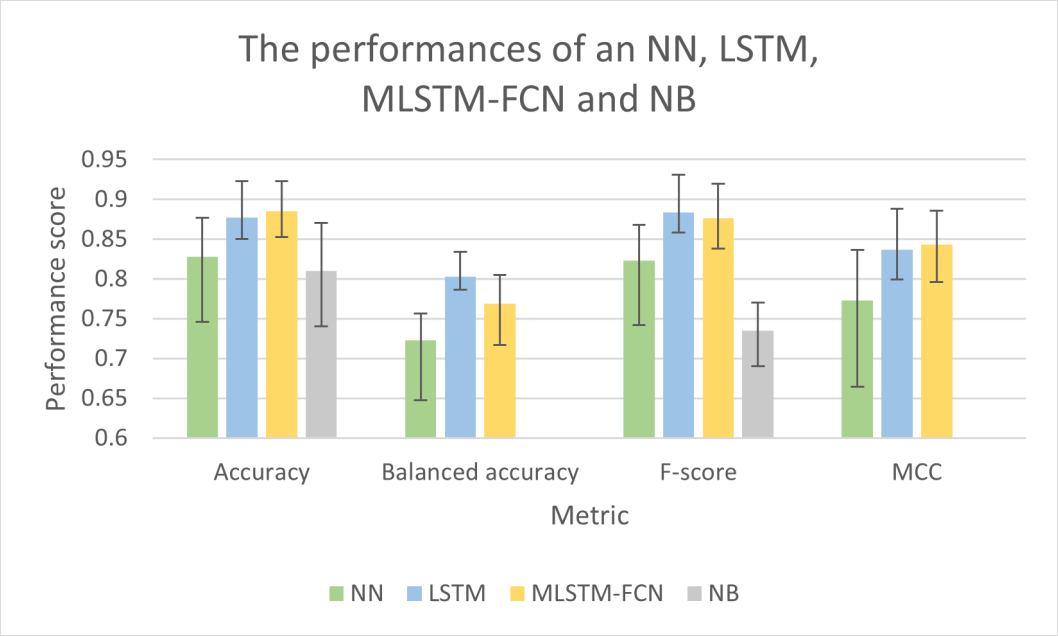
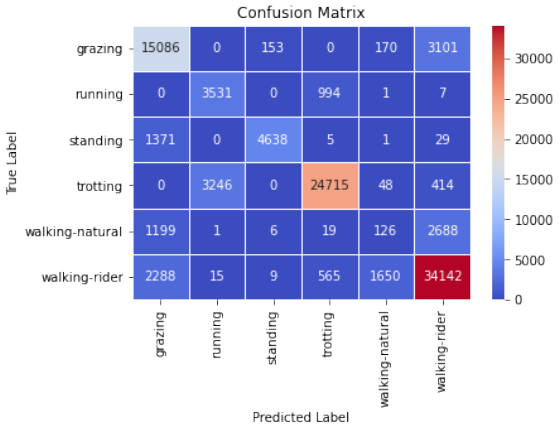
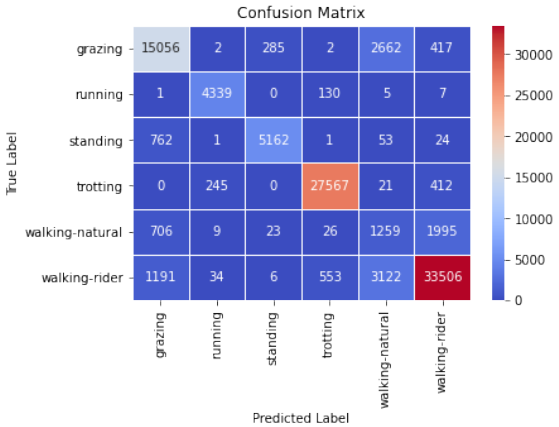


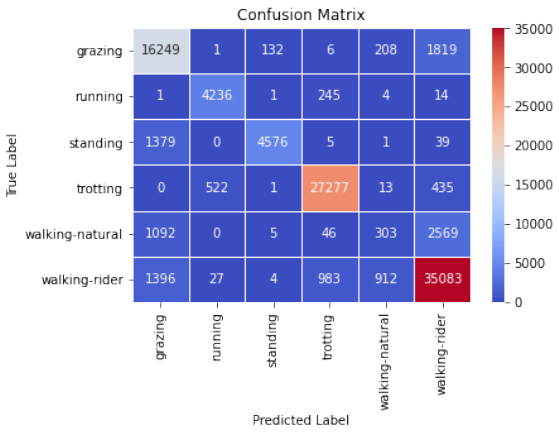
Figure 28: The performances of the NN, LSTM, MLSTM-FCN and NB classifiers trained on input set 2. The error bars denote the standard deviation over the folds through leave one subject out validation.



(a)

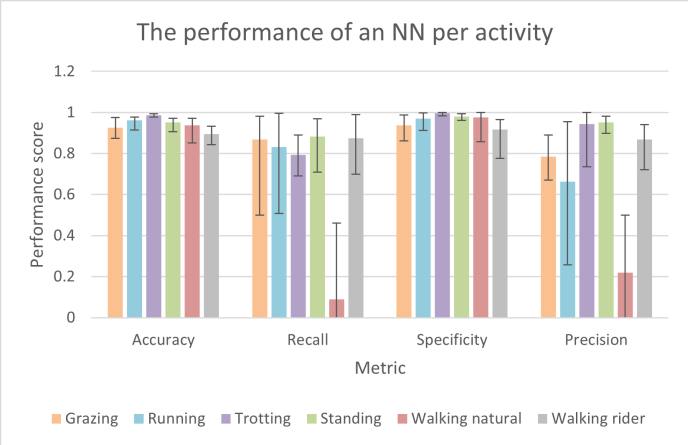


(b)

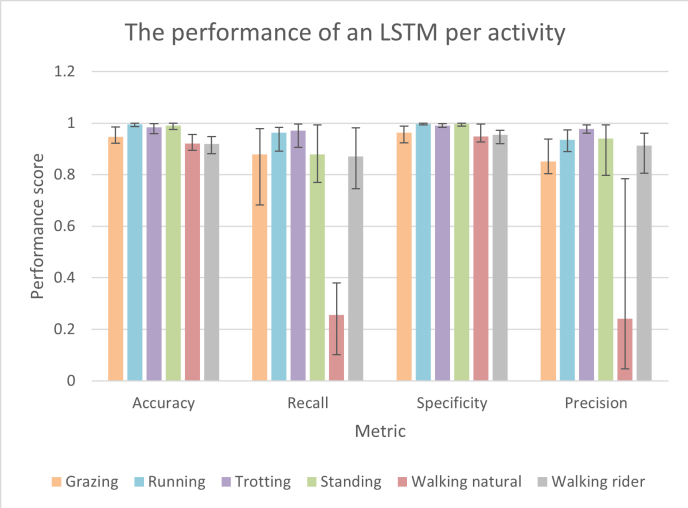


(c)

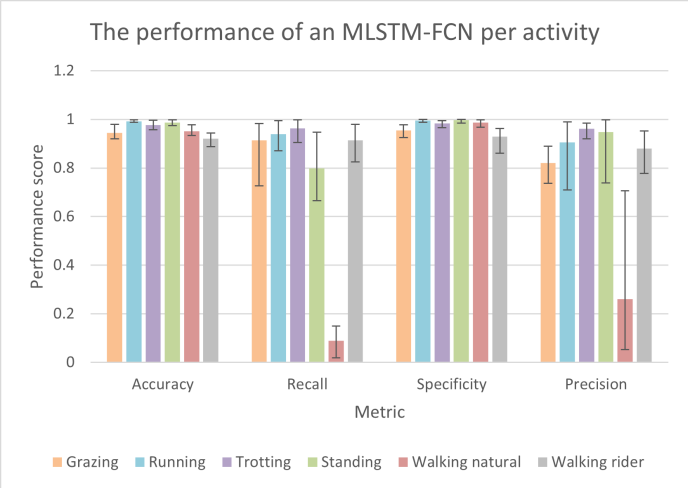
Figure 29: The confusion matrix of an (a) NN, (b) LSTM and (c) MLSTM-FCN trained on input set 2.



(a)



(b)



(c)

Figure 30: The performance scores per activity of an (a) NN, (b) LSTM and (c) MLSTM-FCN trained on input set 2. The error bars denote the standard deviation over the folds through leave one subject out validation.

Even though the LSTM yielded a higher performance than the MLSTM-FCN and NN based on its balanced accuracy and F-score, it is important to note that the LSTM took much more time to train, since it needed around 150 epochs to convert towards its highest performance and each training epoch took around 20 seconds. The NN and MLSTM-FCN, however, only needed around 40 epochs to convert towards its highest performance and each training epoch took around 10 seconds. Thus, overall, the NN and MLSTM-FCN trained 7 to 8 times faster than the LSTM. Even though this evaluation is not based upon the time-restraints, a classifier with a lower training time will be easier to experiment with and to eventually implement in an AAR system.

6.2.1 Possible causes for the difference in performance of the classifiers

Both the LSTM and MLSTM-FCN yielded higher performance scores than the NN. This might be due to the fact that the NN might have not been complex enough to grasp every aspect of the data, therefore yielding a lower performance than both the LSTM and MLSTM-FCN. The LSTM and MLSTM-FCN yielded similar results on the accuracy (88.5% to 87.7%), F-score (87.6% to 88.4%) and MCC (0.843 to 0.837) metrics, where the LSTM yielded higher scores on the smaller classes (walking natural and standing) and the MLSTM on the bigger classes (walking rider and grazing). The fact that the MLSTM-FCN was less able to classify the smaller classes might be attributed to the fact that it is a very complex model, which needs a lot of data. Therefore, there might have not been enough data for it to learn to its full extend. Moreover, since the accuracy was used as the metric during training, the model might have also simply focused more on yielding a higher accuracy, which might have caused a decrease in its balanced accuracy. On all models the validation accuracy was similar to the training accuracy and the testing accuracy was not significantly lower than the training accuracy, which shows that none of the models suffered from overfitting.

6.3 Requirement Evaluation

In Chapter 5 the requirements for the classifiers were stated, which will now be revisited. Firstly, the program is written in Python and is able to handle fully-labeled, offline time-series data, so requirements 1, 2, 3 and 4 are fulfilled. Next to that, it was possible to run the classifiers multiple times for different hyperparameters and input sets within the time span of this project. Therefore, also requirement 5 is fulfilled. However, it is important to note that the LSTM took much more time to train than the NN and MLSTM-FCN. Apart from that, as described in Chapter 4, the results, including a confusion matrix, accuracy, F-score and MCC, for each experiment are saved in a database, therefore fulfilling requirement 6. All experiments were performed with the Leave One Out evaluation method, thus also requirement 7 is fulfilled. Lastly, the performance goal of the classifier was to reach at least an accuracy of 81% and an F-score of 73% on the classification of six activities by six horses. This requirement was met by all models, with the NN (82.8% accuracy, 82.3% F-score)

slightly outperforming the classifier based on accuracy, but yielding a significantly higher F-score. Moreover, both the MLSTM-FCN and LSTM yielded significantly higher scores on both the accuracy, with 87.7% and 88.5% respectively, and the F-score, with 88.4% and 87.6% respectively. In conclusion, all requirements set in Chapter 5 were met.

6.4 Conclusion

Three input sets, which can be found in Table 4, 5 and 6, were used to see which input set yields the highest performance score from the classifiers. For each of the classifiers, with each input set, a grid search was performed to tune the hyperparameters. Since input set 1 is in a more complex data format than input set 2 and 3, it would be logical for the more complex models (LSTM and MLSTM-FCN) to perform better with input set 1 and the less complex model (NN) to perform better with input set 2 or 3. However, this was not the case, since the NN yielded the highest performance with input set 1 and 3 and the LSTM and MLSTM-FCN yielded the highest performance with input set 2 and 3. The LSTM and MLSTM-FCN probably yielded similar performances with input set 2 and 3 because they were able to extract the RMS feature by themselves. The NN, on the other hand, benefited from the added RMS values in input set 3. Overall, input set 3 yielded a high performance for all classifiers.

The main research question poses:

Which animal activity recognition algorithm performs the best on IMU horse data, and what aspects lead it to outperform other algorithms?

To answer this question, three classifiers (NN, LSTM and MLSTM-FCN) were compared based on their performance with input set 2. The LSTM and MLSTM-FCN yielded higher performance scores than the NN on all metrics. Among the LSTM and MLSTM-FCN, the LSTM yielded higher performance scores based upon the balanced accuracy (80.3% to 76.9%) and F-score (88.4% to 87.6%) and the MLSTM-FCN yielded higher performance scores based upon the accuracy (88.5% to 87.7%) and MCC (0.843 to 0.837). Moreover, it took the LSTM around 7 to 8 times longer to train than the MLSTM-FCN. The most confusion among activities, for all classifiers, comes from the grazing (high FP rate) and walking natural (high FN rate) activities. This confusion can mostly be attributed to the fact that the grazing, walking natural and walking rider activities are very similar. Next to that, the grazing activity is also the smallest class, which has caused a bias in all classifiers towards the other classes. Overall, both the LSTM and MLSTM-FCN yielded good performance results and are recommended to use in further AAR projects.

7 Conclusion

Within this report the performance of three classifiers have been evaluated on horse IMU data to answer the main research question:

Which animal activity recognition algorithm performs the best on IMU horse data, and what aspects lead it to outperform other algorithms?

However, before the main research question can be answered, the two sub-research questions will be answered. Firstly, the first sub-research question has been answered in Chapter 3:

Which animal and human activity recognition algorithms that have been developed in the last two years are the most promising to utilize for a horse activity recognition algorithm using IMU data?

From the state of the art it was concluded that the most recent studies most often use a CNN or LSTM, which have shown to be well-performing classifiers in these studies as well. Next to that, the CNN-LSTM has yielded a great performance in multiple studies, although it has not been used as often in the surveyed studies as the CNN and LSTM. Especially the MLSTM-FCN, a variation on the CNN-LSTM, yielded an outstanding performance. Next to that, the code of the MLSTM-FCN was published online, making it a suitable algorithm to use in this project. Other classifiers, such as NB, RF, HMM, SVM and KNN yielded varying results, with a high performance in some studies and lower performance in others. Next to that, some classifiers, such as CT, GBT, HMM and KNN, were not used by many of the surveyed studies, making it hard to evaluate their usefulness for this project. From this research it was concluded that the CNN, LSTM and MLSTM-FCN would be the most promising algorithms to use for a horse activity recognition algorithm. Eventually, an NN, LSTM and MLSTM-FCN were developed in Chapter 6. The second sub-research question has been answered in Chapter 5:

What are the requirements for an animal activity recognition algorithm that can classify multiple activities with an adequate performance?

The requirements that have been set up can be found in Table 5, which describe the required language, data structure, time constraints, functionality and performance. Next to that, the requirements have been revisited in Chapter 6, where it was concluded that all requirements were met. It is important to note that the LSTM needed a significantly longer training time than the NN and the MLSTM-FCN. Next to that, all classifiers improved upon the accuracy and F-score goals that were set in the requirements. Lastly, the main research question has been answered in Chapter 6:

Which animal activity recognition algorithms perform the best on IMU horse data, and what aspects lead it to outperform other algorithms?

As mentioned above, a NN, LSTM and MLSTM-FCN have been compared based on their performance on the horse IMU data set. This evaluation was performed through leave one subject out validation. From these tests, it was concluded that the LSTM (87.8% accuracy, 88.4% F-score) and MLSTM-FCN (88.5% accuracy, 87.6% F-score) yield a higher performance than the NN (82.8% accuracy, 82.3% F-score) on all metrics (accuracy, balanced accuracy, F-score and MCC). The LSTM and MLSTM-FCN probably outperform the NN because they are more complex and can thus understand the complex data better.

The MLSTM-FCN yielded higher a accuracy (88.5%) and MCC (0.843) than the LSTM (87.7% accuracy, 0.837 MCC), although the LSTM yielded a higher balanced accuracy score (80.3%) and F-score (88.4%) than the MLSTM-FCN (76.9% balanced accuracy, 87.6% F-score). This difference is mainly attributed to the fact that the MLSTM-FCN is better at classifying the classes with the most samples, therefore yielding a higher accuracy than the LSTM. On the other hand, the MLSTM-FCN is not as good at classifying the smallest classes, therefore yielding a lower balanced accuracy score than the LSTM. Thus, the LSTM is especially recommended when the classification of all classes is important, while the MLSTM-FCN is especially recommended when a high classification rate in general is more important. Moreover, the MLSTM-FCN is also recommended to use when there is limited training time, since it is 7 to 8 times quicker than the LSTM. Overall, it can be concluded that the LSTM and MLSTM-FCN performed best on the IMU horse dataset and are both recommended algorithms to use in future AAR projects.

8 Future Work

This study has shown the performances of an NN, LSTM and MLSTM-FCN, but the research on these classifiers could still be expanded upon. First, the CNN was not included in this study, despite being one of the most well-performing algorithms from the surveyed studies in the state of the art. Second, the validation data is now split from the training data with a Simple Split. However, the performance of the classifiers might increase if the Leave One Subject Out validation method was used here as well, since the model needs to be able to handle heterogeneity well during training. Next to that, the hyperparameter tuning could be performed with a smoother and faster optimization method, such as the Bayesian optimization method. This would also allow for different hyperparameters per fold of the leave one subject out validation, which could increase the performance of the classifiers. Moreover, the scaling of the test data was now performed with only information from the test dataset itself. However, in a future study, the information from the training data should be included to scale the test data.

Since the dataset that was used is now only labeled extensively enough on six activities, more data acquisition could give more information on the performance of the classifiers. For example, it would be useful to gather more data from the less-labeled activities to see how well the classifiers perform when they have to classify more classes, including classes that may yield quite similar data. Next to that, having more data could also increase the performance of more complex models, such as the MLSTM-FCN. Furthermore, the dataset is also quite imbalanced, which may decrease the performance of all classifiers. Thus, if more data was gathered, this could also solve the problems caused by the imbalance of the dataset. Moreover, in a future study, all other activities, which were not labeled extensively, should be added in the category "other", to see how well the classifier is able to recognize the difference between the six extensively labeled activities and the other activities.

Many different topics are still to be researched before the AAR pipeline can be used to monitor wildlife. First, the pipeline will need to be expanded, so that it can use unsupervised and active learning, which is currently being researched by other students. Moreover, since the AAR pipeline could be used for other animals as well in the future, it is important to test the classifier on other animals to see if it standardizes well over different types of animals. Next to that, since the classifier will run on a collar, the resource restrictions need to be evaluated in a future study. Even though the LSTM yields a high performance, it also requires high computational time and power. Despite the MLSTM-FCN being almost 7 to 8 times quicker than the LSTM, the MLSTM-FCN might require more memory than the LSTM. Thus, it is probably not able to operate with a low memory and low impact on the battery of the tracking device. Therefore, in a future study it should be investigated whether an LSTM and MLSTM-FCN would be able to run based with a low memory and low impact on the battery of a tracking device. If this is not possible, solutions should be offered to use the LSTM or the MLSTM-FCN in a new pipeline. For example, this state of the art classifier

can be used to label a large amount of data, which can then be used as input into a classifier which requires less resources.

9 References

[1] I. M. Gren, T. Häggmark-Svensson, H. Andersson, G. Jansson, and A. Jägerbrand, "Using traffic data to estimate wildlife populations," *Journal of Bioeconomics*, vol. 18, no. 1, pp. 17–31, 2016, ISSN: 15736989. DOI: 10.1007/s10818-015-9209-0.

[2] T. Grandin, *Improving animal welfare: A practical approach*, 3rd, Temple Grandin, Ed. CABI, 2015, p. 356, ISBN: 9781780644677.

[3] C. C. Wilmers, B. Nickel, C. M. Bryce, J. A. Smith, R. E. Wheat, V. Yovovich, and M. Hebblewhite, "The golden age of bio-logging: How animal-borne sensors are advancing the frontiers of ecology," *Ecology*, vol. 96, no. 7, pp. 1741–1753, Jul. 2015, ISSN: 00129658. DOI: 10.1890/14-1401.1.

[4] J. Mench, "Why It Is Important to Understand Animal Behavior," *ILAR Journal*, vol. 39, no. 1, pp. 20–26, Jan. 1998, ISSN: 1084-2020. DOI: 10.1093/ilar.39.1.20.

[5] A. Eerdekens, M. Deruyck, J. Fontaine, L. Martens, E. D. Poorter, and W. Joseph, "Automatic equine activity detection by convolutional neural networks using accelerometer data," *Computers and Electronics in Agriculture*, vol. 168, Jan. 2020, ISSN: 01681699. DOI: 10.1016/j.compag.2019.105139.

[6] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," in *Proceedings First International Workshop on Human Behavior Understanding (in conjunction with the 20th Int. Conf. on Pattern Recognition)*, A. A. Salah, T. Gevers, N. Sebe, and A. Vinciarelli, Eds., Istanbul: Springer, Aug. 2010, pp. 38–51.

[7] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, D. R. Millán, and D. Roggen, "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, 2013. DOI: 10.1016/j.patrec.2012.12.014.

[8] A. Reiss and D. Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring," in *The 16th IEEE International Symposium on Wearable Computers*, 2012.

[9] C. Krupitzer, T. Sztyler, J. Edinger, M. Breitbach, H. Stuckenschmidt, and C. Becker, "Hips Do Lie! A Position-Aware Mobile Fall Detection System," in *IEEE International Conference on Pervasive Computing and Communications*, IEEE, Aug. 2018, ISBN: 9781538632246. DOI: 10.1109/PERCOM.2018.8444583.

[10] S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong, "Sensor Data Acquisition and Multimodal Sensor Fusion for Human Activity Recognition Using Deep Learning," *Sensors (Switzerland)*, vol. 19, no. 7, p. 1716, Apr. 2019, ISSN: 1424-8220. DOI: 10.3390/s19071716.

[11] J. W. Kamminga, D. V. Le, J. P. Meijers, H. Bisby, N. Meratnia, and P. J. M. Havinga, "Robust Sensor-Orientation-Independent Feature Selection for Animal Activity Recognition on Collar Tags," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 1, p. 27, Mar. 2018. DOI: 10.1145/3191747.

[12] J. W. Kamminga, L. M. Janßen, N. Meratnia, and P. J. M. Havinga, "Horsing Around-A Dataset Comprising Horse Movement," Tech. Rep. 4, Sep. 2019, p. 131. DOI: 10.3390/data4040131.

[13] N. Kleanthous, A. Hussain, W. Khan, J. Sneddon, and A. Mason, "Feature Extraction and Random Forest to Identify Sheep Behavior from Accelerometer Data," in *Intelligent Computing Methodologies*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 408–419, ISBN: 9783030607951. DOI: 10.1007/978-3-030-60796-8{_}35.

[14] D. Micucci, M. Mobilio, and P. Napoletano, "UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones," *Applied Sciences*, vol. 7, no. 10, p. 1101, Oct. 2017, ISSN: 2076-3417. DOI: 10.3390/app7101101.

[15] W. Gomaa, R. Elbasiony, and S. Ashry, "ADL Classification based on autocorrelation function of inertial signals," in *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017*, vol. 2017-Decem, IEEE, 2017, pp. 833–837, ISBN: 9781538614174. DOI: 10.1109/ICMLA.2017.00-53.

[16] K. Açıcı, Ç. B. Erdaş, T. Aşuroğlu, and H. Oğul, "HANDY: A Benchmark Dataset for Context-Awareness via Wrist-Worn Motion Sensors," *Data*, vol. 3, no. 3, p. 24, Jun. 2018, ISSN: 2306-5729. DOI: 10.3390/data3030024.

[17] R. Mutegeki and D. S. Han, "A CNN-LSTM Approach to Human Activity Recognition," in *2020 International Conference on Artificial Intelligence in Information and Communication*, IEEE, Feb. 2020, ISBN: 9781728149851. DOI: 10.1109/ICAIIC48513.2020.9065078.

[18] S. Ashry, T. Ogawa, and W. Gomaa, "CHARM-Deep: Continuous Human Activity Recognition Model Based on Deep Neural Network Using IMU Sensors of Smartwatch," *IEEE Sensors Journal*, vol. 20, no. 15, pp. 8757–8770, Aug. 2020, ISSN: 15581748. DOI: 10.1109/JSEN.2020.2985374.

[19] D. Anguita, A. Ghio, L. Oneto, P. Xavier, and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," in *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, 2013.

[20] W. Qi, H. Su, C. Yang, G. Ferrigno, E. De Momi, and A. Aliverti, "A Fast and Robust Deep Convolutional Neural Networks for Complex Human Activity Recognition Using Smartphone," *Sensors*, vol. 19, no. 17, p. 3731, Aug. 2019, ISSN: 1424-8220. DOI: 10.3390/s19173731.

[21] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, R. J. Brachman and T. Dietterich, Eds. Madison: Morgan & Claypool, 2009, pp. 9–11, ISBN: 9781598295481.

[22] J. Kamminga, "Hiding in the deep : online animal activity recognition using motion sensors and machine learning," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, Oct. 2020, ISBN: 9789036550550. DOI: 10.3990/1.9789036550550.

[23] I. N. da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, *Artificial Neural Networks: A Practical Course*. Springer International Publishing Switzerland, 2017, p. 24, ISBN: 978-3-319-43162-8.

[24] X. Ying, "An Overview of Overfitting and its Solutions," in *Journal of Physics: Conference Series*, vol. 1168, Institute of Physics Publishing, 2019. DOI: 10.1088/1742-6596/1168/2/022022.

[25] M. Mubasir, *Don't Overfit! II — How to avoid Overfitting in your Machine Learning and Deep Learning Models*, Jul. 2020. [Online]. Available: https://towardsdatascience.com/dont-overfit-ii-how-to-avoid-overfitting-in-your-machine-learning-and-deep-learning-models-2ff903f4b36a.

[26] S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, C. Campbell, T. R. Golub, and J. P. Mesirov, "Estimating Dataset Size Requirements for Classifying DNA Microarray Data," *Journal of Computational Biology*, vol. 10, no. 2, pp. 119–142, 2003.

[27] V. H. Tam, S. Kabbara, R. F. Yeh, and R. H. Leary, "Impact of Sample Size on the Performance of Multiple-Model Pharmacokinetic Simulations Downloaded from," *Antimicrobial Agents and Chemotherapy*, vol. 50, no. 11, pp. 3950–3952, Nov. 2006. DOI: 10.1128/AAC.00337-06.

[28] M. Kim, C. Y. Jeong, and H. C. Shin, "Activity Recognition using Fully Convolutional Network from Smartphone Accelerometer," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju: IEEE, Nov. 2018, pp. 1482–1484, ISBN: 9781538650400. DOI: 10.1109/ICTC.2018.8539419.

[29] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, pp. 103–134, 2000, ISSN: 08856125. DOI: 10.1023/a:1007692713085.

[30] H. M. Kalayeh and D. A. Landgrebe, "Predicting the Required Number of Training Samples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 6, pp. 664–667, Nov. 1983, ISSN: 01628828. DOI: 10.1109/TPAMI.1983.4767459.

[31] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, Nov. 2013. DOI: 10.1016/j.ins.2013.07.007.

[32] E. Casella, A. R. Khamesi, and S. Silvestri, "Smartwatch Application for Horse Gaits Activity Recognition," in *International Conference on Smart Computing*, Washington: IEEE, Jun. 2019, pp. 409–416.

[33] X. Zhou, W. Liang, K. I. Wang, H. Wang, L. T. Yang, and Q. Jin, "Deep-Learning-Enhanced Human Activity Recognition for Internet of Healthcare Things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6429–6438, Jul. 2020, ISSN: 23274662. DOI: 10.1109/JIOT.2020.2985082.

[34] L. K. Wang and R. Y. Liu, "Human Activity Recognition Based on Wearable Sensor Using Hierarchical Deep LSTM Networks," *Circuits, Systems, and Signal Processing*, vol. 39, no. 2, pp. 837–856, Feb. 2020, ISSN: 15315878. DOI: 10.1007/s00034-019-01116-y.

[35] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, "Deep Learning Models for Real-time Human Activity Recognition with Smartphones," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 743–755, Apr. 2020, ISSN: 15728153. DOI: 10.1007/s11036-019-01445-x.

[36] S. Ashry, W. Gomaa, M. G. Abdu-Aguye, and N. El-Borae, "Improved IMU-based Human Activity Recognition using Hierarchical HMM Dissimilarity," *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics*, pp. 702–709, 2020. DOI: 10.5220/0009886607020709.

[37] V. Sturm, D. Efrosinin, N. Efrosinina, L. Roland, M. Iwersen, M. Drillich, and W. Auer, "A Chaos Theoretic Approach to Animal Activity Recognition," *Journal of Mathematical Sciences*, vol. 237, no. 5, pp. 730–743, Mar. 2019, ISSN: 15738795. DOI: 10.1007/s10958-019-04199-9.

[38] E. Bocaj, D. Uzunidis, P. Kasnesis, and C. Z. Patrikakis, "On the Benefits of Deep Convolutional Neural Networks on Animal Activity Recognition," in *International Conference on Smart Systems and Technologies*, Osijek: IEEE, Oct. 2020, pp. 83–88, ISBN: 9781728197593. DOI: 10.1109/SST49455.2020.9263702.

[39] N. Pirinen, M. Pastell, A. Mykkänen, C. McGowan, and H. Hyytiäinen, "Validation of a tail-mounted triaxial accelerometer for measuring foals' lying and motor behavior," *Journal of Veterinary Behavior*, vol. 39, pp. 14–20, Sep. 2020, ISSN: 15587878. DOI: 10.1016/j.jveb.2020.06.004.

[40] X. Zhang and J. Zhang, "Subject independent human activity recognition with foot IMU data," in *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN*, IEEE, Dec. 2019, pp. 240–246, ISBN: 9781728152127. DOI: 10.1109/MSN48538.2019.00054.

[41] L. Chen, S. Fan, V. Kumar, and Y. Jia, "A method of human activity recognition in transitional period," *Information*, vol. 11, no. 9, p. 416, 2020, ISSN: 20782489. DOI: 10.3390/INFO11090416.

[42] M. Gil-Martín, R. San-Segundo, F. Fernández-Martínez, and R. de Córdoba, "Human activity recognition adapted to the type of movement," *Computers and Electrical Engineering*, vol. 88, 2020, ISSN: 00457906. DOI: 10.1016/j.compeleceng.2020.106822.

[43] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, Jan. 2014. DOI: 10.1145/2499621.

[44] *FEATURE | meaning in the Cambridge English Dictionary*. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/feature.

[45] P. Padmanabhan and S. Puthusserypady, "Nonlinear analysis of EMG signals - a chaotic approach," in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Francisco: IEEE, Sep. 2004.

[46] J. P. Connell, A. DiMercurio, and D. Corbetta, "Dynamic Systems Theory," in *Encyclopedia of Animal Cognition and Behavior*, Springer International Publishing, 2017. DOI: 10.1007/978-3-319-47829-6{_}1594-1.

[47] R. Kohavi, B. Becker, and D. Sommerreld, "Improving Simple Bayes," in *The 9th European Conference on Machine Learning*, Poster Papers, 1997, pp. 78–87.

[48] G. I. Webb, E. Keogh, R. Miikkulainen, and M. Sebag, "Naïve Bayes," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Springer US, 2010, pp. 713–714. DOI: 10.1007/978-0-387-30164-8{_}576.

[49] K.-M. Osei-Bryson, "Assessing Cluster Quality using Multiple Measures - A Decision Tree Based Approach," in *The next wave in computing, optimization, and decision technologies*, B. Golden, S. Raghavan, and E. Wasil, Eds., New York: Springer, 2005, ch. 6, pp. 371–384.

[50] G. Louppe, "Understanding Random Forests: From Theory to Practice," Ph.D. dissertation, Université de Liège, Jul. 2014.

[51] Y. Zhao and Y. Zhang, "Comparison of decision tree methods for finding active objects," *Advances in Space Research*, vol. 41, no. 12, pp. 1955–1959, 2008, ISSN: 02731177. DOI: 10.1016/j.asr.2007.07.020.

[52] J. S. Rao and W. J. E. Potts, "Visualizing Bagged Decision Trees," in *The Third International Conference on Knowledge Discovery and Data Mining*, California: Association for the Advancement of Artificial Intelligence, Aug. 1997, pp. 243–246.

[53] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" In *Lecture Notes in Computer Science*, vol. 7376, Springer, Berlin, Heidelberg, 2012, pp. 154–168, ISBN: 9783642315367. DOI: 10.1007/978-3-642-31537-4{_}13.

[54] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, J. Pepper, S. Weiss, and P. Hauke, Eds. New York: Apress Media, 2015, ISBN: 978-1-4302-5990-9.

[55] M. A. Ladds, A. P. Thompson, D. J. Slip, D. P. Hocking, and R. G. Harcourt, "Seeing It All: Evaluating Supervised Machine Learning Methods for the Classification of Diverse Otariid Behaviours," *PLOS ONE*, vol. 11, K. Yoda, Ed., Dec. 2016. DOI: 10.1371/journal.pone.0166898.

[56] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng, "Stochastic Gradient Boosted Distributed Decision Trees," in *Proceeding of the 18th ACM conference on Information and knowledge management*, New York: ACM, Nov. 2009, pp. 2061–2064.

[57] D. Jurafsky and J. H. Martin, "Hidden Markov Models," in *Speech And Language Processing: An Introduction to Natural Language Processing , Computational Linguistics, and Speech Recognition*, First Edit, Financial Times Prentice Hall, 2020, ch. A, ISBN: 978-0131873216.

[58] Kantardzic Mehmed, "Data Mining: Concepts, Models, Methods, and Algorithms," in *Data Mining: Concepts, Models, Methods, and Algorithms*, Second, New Jersey: John Wiley & Sons, Inc., Hoboken, 2011, ch. 4, pp. 105–108, ISBN: 978-0-470-89045-5.

[59] K. O'shea and R. Nash, "An Introduction to Convolutional Neural Networks," Tech. Rep., 2015.

[60] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach," in *Procedia Computer Science*, vol. 132, Elsevier B.V., Jan. 2018, pp. 679–688. DOI: 10.1016/j.procs.2018.05.069.

[61] L. Zhang and F. Xiang, "Relation classification via BiLSTM-CNN," in *International Conference on Data Mining and Big Data*, vol. 10943 LNCS, Shanghai: Springer, Jun. 2018, pp. 373–382, ISBN: 9783319938028. DOI: 10.1007/978-3-319-93803-5{_}35.

[62] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018. DOI: 10.1016/j.eswa.2017.11.028.

[63] A. Ruiz-Garcia, J. Schmidhuber, V. Palade, C. C. Took, and D. Mandic, "Deep neural network representation and Generative Adversarial Learning," *Neural Networks*, vol. 139, pp. 199–200, 2021. DOI: 10.1016/j.neunet.2021.03.009.

[64] H. Iba and N. Noman, *Deep Neural Evolution: Deep Learning with Evolutionary Computation*, T. Bäck and L. Kari, Eds. Singapore: Springer, 2020, p. 54.

[65] R. Zhu, X. Tu, and J. X. Huang, "Deep learning on information retrieval and its applications," in *Deep Learning for Data Analytics*, Elsevier, 2020, ch. 7, pp. 125–153. DOI: 10.1016/b978-0-12-819764-6.00008-9.

[66] M. Bianchini, M. Maggini, F. Scarselli, and L. C. Jain, *Innovations in Neural Information Paradigms and Applications*. Warsaw: Springer, 2009, vol. 247, pp. 238–244, ISBN: 978-3-642-04002-3.

[67] X. Yuan, L. Li, and Y. Wang, "Nonlinear Dynamic Soft Sensor Modeling with Supervised Long Short-Term Memory Network," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, Feb. 2019, ISSN: 19410050. DOI: 10.1109/TII.2019.2902129.

[68] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The Performance of LSTM and BiLSTM in Forecasting Time Series," in *IEEE International Conference on Big Data*, Los Angeles: IEEE, 2019, pp. 3285–3292. DOI: 10.1109/BigData47090.2019.9005997.

[69] A. Ayman, O. Attalah, and H. Shaban, "An efficient human activity recognition framework based on wearable imu wrist sensors," in *IST 2019 - IEEE International Conference on Imaging Systems and Techniques, Proceedings*, IEEE, Dec. 2019, ISBN: 9781728138688. DOI: 10.1109/IST48021.2019.9010115.

[70] F. M. S. Bragança, S. Broomé, M. Rhodin, S. Björnsdóttir, V. Gunnarsson, J. P. Voskamp, E. Persson-Sjodin, W. Back, G. Lindgren, M. Novoa-Bravo, C. Roepstorff, B. J. Van Der Zwaag, P. R. Van Weeren, and E. Hernlund, "Improving gait classification in horses by using inertial measurement unit (IMU) generated data and machine learning," *Scientific Reports*, vol. 10, Oct. 2020. DOI: 10.1038/s41598-020-73215-9.

[71] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Networks*, vol. 116, pp. 237–245, Aug. 2019. DOI: 10.1016/j.neunet.2019.04.014.

[72] M. G. Conners, T. Michelot, E. I. Heywood, R. A. Orben, R. A. Phillips, A. L. Vyssotski, S. A. Shaffer, and L. H. Thorne, "Hidden Markov models identify major movement modes in accelerometer and magnetometer data from four albatross species," *Movement Ecology*, vol. 9, no. 7, Feb. 2021. DOI: 10.1186/s40462-021-00243-z.

[73] R. Arablouei, L. Currie, B. Kusy, A. Ingham, P. L. Greenwood, and G. Bishop-Hurley, "In-situ classification of cattle behavior using accelerometry data," *Computers and Electronics in Agriculture*, vol. 183, p. 106 045, Apr. 2021. DOI: 10.1016/j.compag.2021.106045.

[74] R. T. Van Den Berg, "Designing a small and low-energy wild life tag for parakeets within an urban environment capable of tracking and online activity recognition," University of Twente, Enschede, Tech. Rep., Aug. 2019.

[75] J. W. Kamminga, N. Meratnia, and P. J. M. Havinga, "Dataset: Horse Movement Data and Analysis of its Potential for Activity Recognition," in *The 17th ACM Conference on Embedded Networked Sensor Systems*, New York: Association for Computing Machinery, Nov. 2019, pp. 22–25. DOI: 10.1145/3359427.3361908.

[76] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition," in *SenSys 2015 - Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, Seoul: Association for Computing Machinery, Inc, 2015, pp. 127–140, ISBN: 9781450336314. DOI: 10.1145/2809695.2809718.

[77] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2019. [Online]. Available: https://archive.ics.uci.edu/ml.

[78] M. Zhang and A. A. Sawchuk, "USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, New York: ACM, Sep. 2012, pp. 1036–1043, ISBN: 9781450312240.

[79] R. Pucci, A. Micheli, S. Chessa, and J. Hunter, "Machine learning approaches for identifying prey handling activity in otariid pinnipeds," Tech. Rep., 2020.

[80] L. Al-Frady and A. Al-Taei, "Wrapper Filter Approach for Accelerometer-Based Human Activity Recognition," *Pattern Recognition and Image Analysis*, vol. 30, pp. 757–764, Jan. 2020. DOI: 10.1134/S1054661820040033.

[81] R. Mojarad, F. Attal, A. Chibani, and Y. Amirat, "Automatic Classification Error Detection and Correction for Robust Human Activity Recognition," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2208–2215, Apr. 2020, ISSN: 23773766. DOI: 10.1109/LRA.2020.2970667.

[82] R. K. Priyadarshini, A. B. Banu, and T. Nagamani, "Gradient Boosted Decision Tree based Classification for Recognizing Human Behavior," in *International Conference on Advances in Computing and Communication Engineering*, IEEE, 2019, pp. 1–4. DOI: 10.1109/ICACCE46606.2019.9080014.

[83] S. Bosch, F. Serra Bragança, M. Marin-Perianu, R. Marin-Perianu, B. J. van der Zwaag, J. Voskamp, W. Back, R. Van Weeren, and P. Havinga, "EquiMoves: A wireless networked inertial measurement system for objective examination of horse gait," *Sensors*, vol. 18, no. 3, p. 850, Mar. 2018. DOI: 10.3390/s18030850.

[84] L. B. St George, T. Spoormakers, F. M. Serra Bragança, P. R. van Weeren, and S. J. Hobbs, "The use of surface electromyography for quantification of changes in biceps femoris and triceps brachii muscle activity during induced forelimb and hindlimb lameness," *Equine veterinary journal*, vol. 51, no. S53, p. 28, 2019, ISSN: 0425-1644. DOI: 10.1111/evj.48{_}13152.

[85] B. Bruno, F. Mastrogiovanni, and A. Sgorbissa, "A Public Domain Dataset for ADL Recognition Using Wrist-placed Accelerometers," in *Robot and Human Interactive Communication*, Edinburgh: IEEE, Aug. 2014.

[86] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, Mar. 2011, ISSN: 1931-0145. DOI: 10.1145/1964897.1964918. [Online]. Available: https://dl.acm.org/doi/10.1145/1964897.1964918.

[87] D. Xhemali, C. J. Hinde, and R. G. Stone, "Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages," *IJCSI International Journal of Computer Science Issues*, vol. 4, no. 1, pp. 16–23, 2009, ISSN: 1694-0784.

[88] S. D. Jadhav and H. P. Channe, "Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques," *International Journal of Science and Research*, vol. 5, no. 1, pp. 1842–1845, Jan. 2013, ISSN: 2319-7064.

[89] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, G. Tröster, P. Lukowicz, G. Pirkl, D. Bannach, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, and J. d. R. Millàn, "Collecting complex activity data sets in highly rich networked sensor environments," in *Seventh International Conference on Networked Sensing Systems*, Kassel, 2010.

[90] A. Ignatov, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks," *Applied Soft Computing*, vol. 62, pp. 915–922, Jan. 2018. DOI: 10.1016/j.asoc.2017.09.027.

[91] *Human Activity Recognition (HAR) Tutorial with Keras and Core ML*, Aug. 2018. [Online]. Available: https://towardsdatascience.com/human-activity-recognition-har-tutorial-with-keras-and-core-ml-part-1-8c05e365dfa0.

[92] S. Elsworth and S. Güttel, "Time Series Forecasting Using LSTM Networks: A Symbolic Approach," University of Manchester, Tech. Rep., Mar. 2020.

[93] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun. 2014.

[94] K. G. Keegan, P. F. Pai, D. A. Wilson, and B. K. Smith, "Signal decomposition method of evaluating head movement to measure induced forelimb lameness in horses trotting on a treadmill," *Equine Veterinary Journal*, vol. 33, pp. 446–451, 2001. DOI: 10.2746/042516401776254781.

[95] C. Peham, M. Scheidl, and T. Licka, "A method of signal processing in motion analysis of the trotting horse," *Journal of Biomechanics*, vol. 29, no. 8, pp. 1111–1114, Aug. 1996. DOI: 10.1016/0021-9290(95)00179-4.

[96] J. Kamminga, P. Havinga, N. Meratnia, and V. L. Duc, *Towards Deep Unsupervised Representation Learning from Accelerometer Time Series for Animal Activity Recognition*, Aug. 2020.

[97] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[98] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, Jan. 2020, ISSN: 14712164. DOI: 10.1186/s12864-019-6413-7.

[99] *CIFAR-10 Classification using Intel Optimization for TensorFlow*, May 2018. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/cifar-10-classification-using-intel-optimization-for-tensorflow.html.

[100] S. Bock and M. Weis, "A Proof of Local Convergence for the Adam Optimizer," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2019-July, IEEE, Jul. 2019, ISBN: 978-1-7281-1985-4. DOI: 10.1109/IJCNN.2019.8852239.

[101] *Geospatial Computing Portal*, 2020. [Online]. Available: https://crib.utwente.nl/.

[102] *Project Jupyter*, Apr. 2021. [Online]. Available: https://jupyter.org/.

[103] *pandas - Python Data Analysis Library*, Apr. 2021. [Online]. Available: https://pandas.pydata.org/.

[104] *NumPy*, 2020. [Online]. Available: https://numpy.org/.

[105] *Keras: the Python deep learning API*. [Online]. Available: https://keras.io/.

[106] C. Leifer, *peewee*. [Online]. Available: http://docs.peewee-orm.com/.

[107] M. Waskom, *seaborn: statistical data visualization*, 2020. [Online]. Available: https://seaborn.pydata.org/.

[108] *Applications for Python*. [Online]. Available: https://www.python.org/about/apps/.