Evaluating Deep Learning Architectures for Motion Classification in the Context of Lower Limb Exosekeletons

Bachelor's Thesis
of

Malte Voß

KIT Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
High Perfomance Humanoid Technologies Lab (H^2T)

Referee:	Prof. Dr.-Ing. Tamim Asfour
Advisors:	M.Sc. Isabel Patzer
	M.Sc. Christian R.G. Dreher

Duration: October 1st, 2019 — February 29th, 2020

Erklärung

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 28. Februar 2020

Malte Voß

Zusammenfassung

Augmentierende Exoskelette sind eine vielversprechende Technologie zur Unterstützung der menschlichen Leistungsfähigkeit. Anwendungen in industriellen, medizinischen und kommerziellen Bereichen sind möglich. Arbeitern kann beispielsweise große Belastung abgenommen werden, die Bewegungsfähigkeit von körperlich eingeschränkten Menschen verbessert werden oder Athleten beim Sport unterstützt werden. Eine genaue und intuitive Steuerung ist notwendig, um gute Benutzerfreundlichkeit zu gewährleisten, daher ist die Erkennung der Absichten hinsichtlich Bewegung des Benutzers entscheidend. In dieser Arbeit werden zwei Verfahren aus dem Bereich des Deep Learning – Convolutional Neural Networks (CNNs) und Netzwerke mit Long Short-Term Memory (LSTMs) – zur Klassifizierung menschlicher Bewegung verwendet und beurteilt. Menschliche Bewegung wird mit einem passiven Exoskelett der unteren Extremitäten aufgezeichnet, welches mit sieben 3D-Kraftsensoren und drei Trägheitsmesseinheiten ausgestattet ist. Die Klassifikation basiert auf einem Ansatz, bei dem Datenpunkte zu Zeitfenstern verkettet werden. Diese Zeitfenster dienen als zu klassifizierende Informationen. Der Auswertung liegt ein Datensatz zugrunde, der Bewegungsdaten von zehn Personen, die 14 alltägliche Bewegungen wie Gehen und Treppensteigen jeweils zehnmal ausgeführt haben, enthält.

Die Auswertung umfasst Klassifikatoren, die für eine Person optimiert sind (Single-Subject) und solche, die mit Daten mehrerer Personen trainiert wurden (Multi-Subject). Bei kleinen Fenstergrößen schneiden optimierte Modelle besser ab als die allgemeineren, jedoch schrumpft dieser Unterschied bei größeren Zeitfenstern. So hat das CNN beispielsweise eine durchschnittliche Klassifikationsgenauigkeit von 0,9680 im optimierten Fall, während diese für die allgemeinere Evaluation auf 0,9296 sinkt. Bei größeren Zeitfenstern liegen beide Werte über 0,99. Der LSTM-Ansatz übertrifft CNNs bei kleinen Fenstergrößen, so erreicht dieser eine Klassifikationsgenauigkeit von 0,9707 im Vergleich zu 0,9296 bei der Multi-Subject-Auswertung. Um die Generalisierungsfähigkeit zu testen, werden Modelle, die mit Daten mehrerer Probanden trainiert wurden, mit Daten eines zuvor nicht gesehenen Subjekts ausgewertet. Dabei wurde eine durchschnittliche Klassifikationsgenauigkeit von 0,9152 für CNNs und 0,9073 für LSTMs gemessen. Im Vergleich zu früheren Ansätzen zur Bewegungsklassifikation basierend auf Hidden-Markov-Modellen (Beil et al., 2018) für diese Daten konnte die Klassifikationsqualität verbessert werden, beispielsweise wurde die Klassifikationsgenauigkeit der Multi-Subject-Auswertung von 0,928 auf 0,999 erhöht.

Diese Bachelorarbeit ist wie folgt aufgebaut: Kapitel 1 motiviert zunächst Bewegungsklassifikation im Kontext von Exoskeletten. Kapitel 2 erläutert Grundlagen künstlicher neuronaler Netze und präsentiert übliche Architekturen. Darüber hinaus werden drei gängige Evaluationsarten für Klassifikation menschlicher Bewegungen eingeführt und Metriken zur Quantifizierung der Klassifikationsqualität vorgestellt. Kapitel 3 diskutiert ähnliche Arbeiten aus der Literatur und stellt verschiedene Sensorsysteme, Ansätze zur Bewegungsklassifikation und Evaluationsverfahren vor. Kapitel 4 behandelt das Design des verwendeten Exoskeletts, Modellierung der Sensordaten und Eigenschaften des benutzten Datensatzes. In Kapitel 5 werden zwei verschiedene Deep Learning Architekturen zur Bewegungsklassifikation vorgeschlagen. Ein CNN und ein

LSTM dienen als Klassifikatoren menschlicher Bewegungsdaten. Anschließend sind die Evaluationsergebnisse beider Modelle in Kapitel 6 dargestellt. Darüber hinaus werden die Ergebnisse mit bestehenden Arbeiten verglichen und die Möglichkeit der Echtzeit-Klassifizierung wird untersucht. In Kapitel 7 wird die Arbeit zusammengefasst und Ideen für die zukünftige Arbeiten gesammelt.

Abstract

Augmenting exoskeletons is a promising technology in assisting human performance. Applications in industrial, medical and commercial areas are feasible. For example, strain is put off a worker, mobility of physically limited people is increased or athletes are supported in their motions. Accurate and intuitive control is necessary to ensure ease of usage. Thus, recognition of the user's motion intentions is critical. In this thesis, two deep learning approaches – Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTM) – are evaluated for human motion classification. Human motion is recorded with a passive lower limb exoskeleton equipped with seven 3D-force sensors and three inertial measurement units. Classification is based on a sliding window approach, i. e., the most recent data points are concatenated and then classified. For the evaluation a data set containing motion data of ten subjects performing 14 motions ten times each is used.

The evaluation includes models that are optimized for one subject (Single-Subject) and models trained with data of multiple people (Multi-Subject). For small window sizes optimized models perform better than the more general models, but this difference shrinks for larger window sizes. For example, CNN achieves an average classification accuracy of 0.9680 in the optimized case, while for the more general evaluation this drops to 0.9296. With larger window sizes both values are over 0.99. The LSTM approach outperforms CNNs for small window sizes, e. g., they achieve a classification accuracy of 0.9707 compared to 0.9296 in Multi-Subject evaluation. To analyze generalization performance, models trained with data of multiple subjects are evaluated with data of an unseen subject. Here, an average classification accuracy of 0.9152 for CNNs and 0.9073 for LSTMs was assessed. In comparison to prior approaches for motion classification based on Hidden Markov Models (Beil et al., 2018) for this data, all evaluation scores could be improved, e. g., Multi-Subject classification accuracy increased from 0.928 to 0.999.

This thesis is organized as follows: Chapter 1 introduces and motivates motion classification in the context of exoskeletons. Chapter 2 provides fundamentals about neural networks and common architectures. In addition, three common evaluation types in the context of human motion classification are introduced and metrics for quantifying classification performance are presented. In Chapter 3, related research in the literature is discussed. Different sensor systems, motion classification approaches and evaluation methods are presented. Chapter 4 covers design of the used exoskeleton, modelling of sensor data and properties of the used data set. Two different deep learning architectures are proposed in Chapter 5. A Convolutional Neural Network and a Long Short-Term Memory model are employed as classifiers of human motion data. Then, evaluation results of both models are presented in Chapter 6. Furthermore, results are compared to existing work and online classification capability is investigated. In Chapter 7, the thesis is concluded and ideas for future work are proposed.

Table of Contents

Chapter 1.

Introduction

Exoskeletons are a promising technology in augmenting humans, as many real-world applications are possible: increasing mobility of elderly people (Figure 1.1(a)), assisting athletes in their sport (Figure 1.1(b)) or supporting workers in physically exhausting jobs like construction (Figure 1.1(c)).

While exoskeletons for rehabilitation and substitution have been in the focus of research, currently research is conducted in the field of exoskeletons for healthy subjects. The main goal of such wearable robots is to enhance the user's performance. Additionally, physical work bears the risk of injuring the worker due to high strain or bad body posture. This risk can be reduced, as forces are distributed to the exoskeleton and a good posture is promoted. Beil (2019) points out two main challenges of augmenting exoskeletons: an exoskeleton should assist the user in their motions but must not limit them. Thus, the structure of the exoskeleton must be *kinematically compatible* to the human body and a control system is necessary that recognizes the user's current motion and/or intention. Different use-cases offer opportunities for optimizing a control system. For example, if there is only one user for the system, its recognition unit can be optimized for that user. On the other hand, users may also want to use an exoskeleton spontaneously, resulting in a challenging task of recognizing motion of an unknown subject. Furthermore, sensors and their placement have to be chosen for recording human motion. Since motion recognition is based on these recordings, a sensor setup providing useful data is necessary.

The High Performance Humanoid Technologies Lab (H^2T) developed a passive lower limb exoskeleton to record human motion (Beil et al., 2018). Force sensors and inertial measurement units collect physical changes. To classify human motion, data points are concatenated to windows. In their work, Hidden Markov Models were employed to classify windows. This thesis extends their work on motion classification. Different deep learning architectures for motion classification are evaluated. Furthermore,

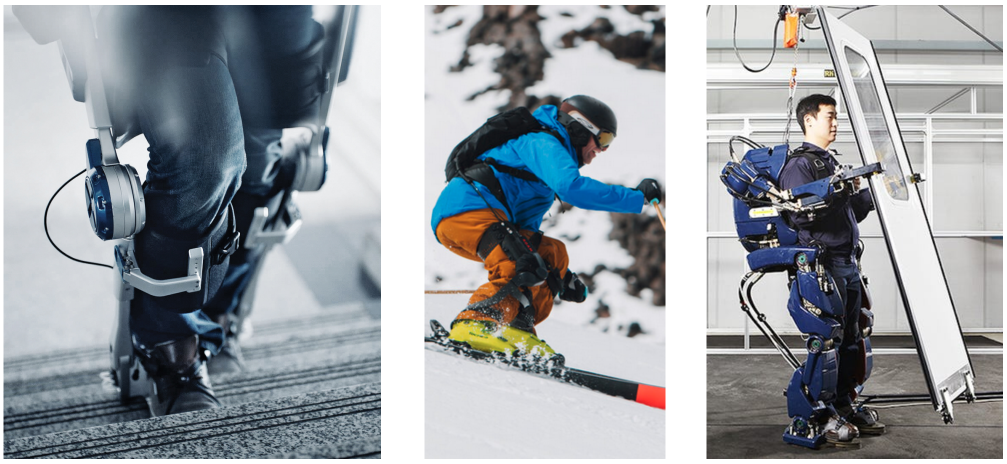


Figure 1.1.: Examples for augmenting exoskeletons. (a) is designed for people with limited muscular power and supports the lower body in locomotion. (b) is a commercially available exoskeleton assisting skiers. It reduces loads on the knee, thus protecting the knee and assisting muscles involved. (c) shows a full-body exoskeleton for industrial applications. Sources of pictures: (Hyundai Motor; Roam Robotics; Muoio).

constraints from the use-case are investigated. For example, a classifier should infer fast to enable online classification.

This thesis is organized as follows. Chapter 2 provides fundamentals about neural networks, quantifying quality of classification algorithms and evaluation types in the field of human motion classification. In Chapter 3, related approaches in the literature are discussed. Chapter 4 covers design of the used exoskeleton, modelling of sensor data and properties of the used data set. Two different deep learning models are employed for motion classification. They are specified in Chapter 5. Evaluation results are presented in Chapter 6. In Chapter 7, the thesis is concluded and ideas for future work are proposed.

Chapter 2.

Fundamentals

Section 2.1 starts with a brief introduction into artificial neural networks and relevant state-of-the-art network architectures. Then, used validation metrics are described in Section 2.2. In Section 2.3 different evaluation methods are presented.

2.1. Neural Networks

Artificial neural networks are a popular tool of machine learning. This section gives a short introduction to the topic. First, basic functionality and improvements are presented in Section 2.1.1. Then, Section 2.1.2 and Section 2.1.3 introduce special types of neural networks: Convolutional Neural Networks and Long Short-Term Memory. Lastly, basic hyperparameters of neural networks are presented in Section 2.1.4.

2.1.1. Basics

Artificial neural networks are inspired by biological nervous systems. The basic idea is to have multiple neurons connected like a directed graph. Biological signals are propagated from one neuron to its neighbors. One nerve cell can also receive multiple input signals that sum up in this case. In the biological neural cell, the input signal is only propagated further if a certain potential difference (approx. $-55mV$, called threshold potential) at its membrane is reached (Husar, 2020). This behavior can also be mathematically modeled: every artificial neural cell is described by its bias representing the threshold potential and an activation function for normalization. It should be noted, that this model is an overly simplified approximation of the physiological system (Wasserman and Schwartz, 1988). In former times, the sigmoid function $\sigma(t) = \frac{1}{1+e^{-t}}$ was used as a standard activation function. Currently most neural networks use the "rectified linear unit" $ReLU(t) = max(0,t)$. It is also proven that training networks with $ReLU$ activation functions is more successful than using sigmoid (Nair and Hinton, 2010). A network is not only parameterized by its neural cells but also by its connections. As mentioned above, a neural network in general can also be seen as a graph modeling connections (synapses) between the cells. The edges of this graph are weighted. These weights are seen as factors for the signal propagated through this edge. A cell z with incoming signals $x_i, i = 1, .., n$, weights $w_i, i = 1, .., n$ of the incoming edges bias b_z and activation function Φ propagates

$$x_z = \Phi(\sum_{i=1}^{n} w_i \dot{x}_i - b_z) \tag{2.1}$$

as its output. This describes the basic perceptron that was first simulated in 1957 (Rosenblatt, 1957). A single perceptron can only simulate a linear decision surface, e. g., the XOR-function can not be modeled with a single perceptron. This limitation is tackled with the concept of multilayer perceptrons (Leshno et al., 1993). An example of a multilayer perceptron is given in Figure 2.1.

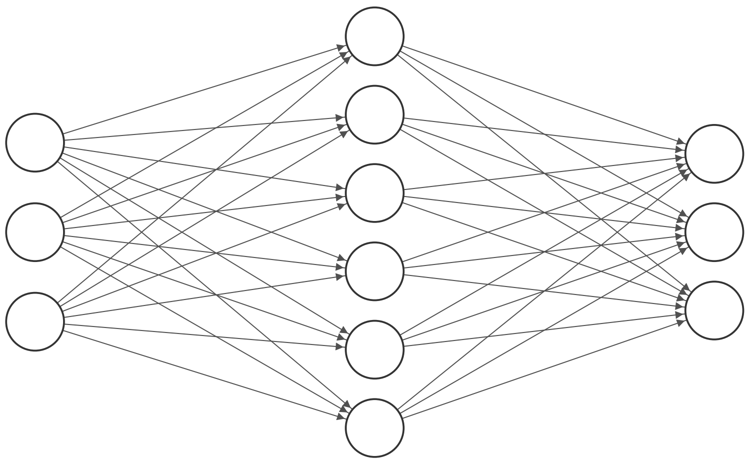


Figure 2.1.: Example of a multilayer perceptron with layers consisting of three, six and three neurons. The neurons of the first layer propagate the input signal. Neurons of other layers forward a signal based on Equation (2.1). Figure generated with *NN-SVG* (LeNail, 2019).

Most recently, deep learning made it practical to build large networks and usage of graphics cards sped training up by a lot. A comprehensive history of neural networks can be found in the literature (Schmidhuber, 2015). A textbook introduction to neural networks is given by Hertz (2018).

Real-world applications of neural networks are image recognition (Krizhevsky et al., 2012), machine translation (Thommes, 2017), medical diagnosis (Seetha and Raja, 2018), hydrology (Kratzert et al., 2019) and many more.

2.1.2. Convolutional Neural Networks

Convolutional Neural Network (CNN) is a class of artificial neural networks prominently used in computer vision tasks. In 2012, Krizhevsky et al. (2012) achieved best

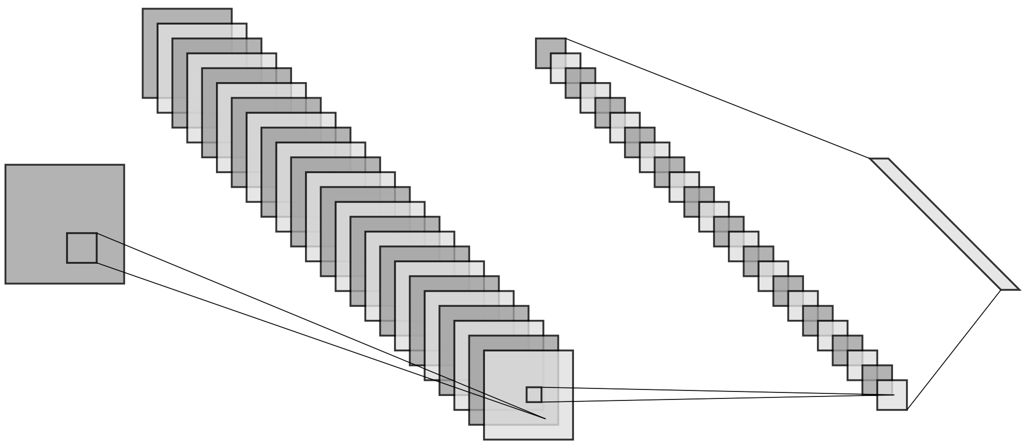


Figure 2.2.: Example of a Convolutional Neural Network . Multiple convolutions are performed on the input image (left), generating feature maps. Further reduction is done by a pooling layer, then the signal is transformed into a vector (right) by a fully connected layer. Figure generated with *NN-SVG* (LeNail, 2019).

results in the *ImageNet* classification benchmark. CNNs were first applied by LeCun et al. (1989).

For an comprehensive introduction see Goodfellow et al. (2016, chapter 9). A CNN employs the mathematical operation convolution to its input. Convolutions have many applications, ranging from signal analysis in electrical engineering to image manipulation in popular social networks like *Instagram*. A discrete convolution is parameterized by a convolution kernel. This kernel itself may be represented as a $\mathbb{R}^{n \times m}$-matrix. The contents of this kernel are called kernel weights. The main idea of CNNs is to learn these kernel weights via backpropagation. A convolutional layer can also perform multiple convolutions to its input, creating so called feature maps for every convolution. Commonly, a convolutional layer is followed by a pooling layer. A pooling layer reduces multiple outputs the previous layer to a single statistic, e. g., a 2×2 max-pooling layer reduces 4 pixels each to one with the new pixel being the maximum of the reduced 4 pixels. Pooling helps making processing more robust, as minor positional changes in the input are not forwarded by the pooling layer. An example CNN is given in Figure 2.2. Modern CNNs apply multiple convolutions to the input, e.g., Krizhevsky et al. (2012) employs 5 convolutional layers. The output of these layers can further be propagated to a classifying part of a network, e. g., dense layers. Other parameters like kernel size or stride are to be set by the network architect.

Since Krizhevsky et al. (2012) successfully employed CNNs for computer vision tasks, CNNs were applied for many more real-world problems. For example, face recognition, including face recognition of primates (Deb et al., 2018) and meteorology (Liu et al., 2016).

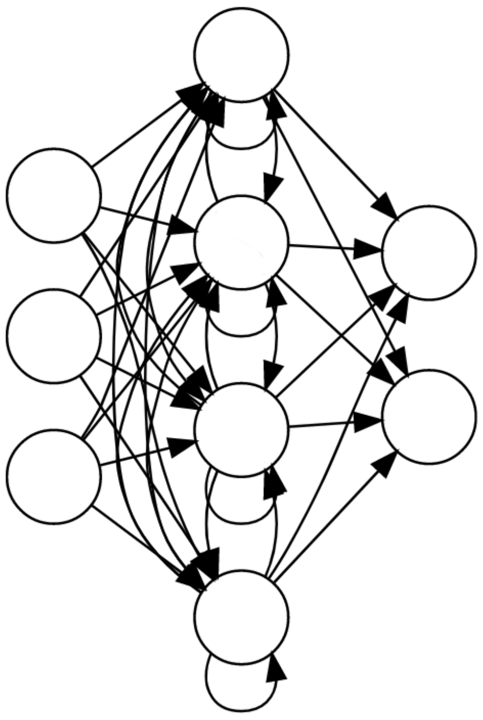


Figure 2.3.: Example of an recurrent neural network. Signals of the hidden layer are also forwarded to other neurons of that layer. Therefore, information persists in the network. Figure is conception of Graves (2012), changed for simplification.

2.1.3. Long Short-Term Memory

Long Short-Term Memory (LSTM) for artificial neural networks was first introduced in 1995 by Hochreiter and Schmidhuber (1995, 1997). Since then, many variations of this architecture have been proposed (Greff et al., 2015).

LSTM is a technique that enhances standard recurrent neural networks. A comprehensive introduction to recurrent neural networks is given by Graves (2012). The key idea of recurrent networks is that an input signal is not only propagated forwards from one layer to its successor but also backwards or to other neurons in the same layer. Thus, a neuron's output is not only forwarded but also persists in the networks internal state. When a new input is given to the network, its processing depends on the networks weights *and* its internal state. This concept is depicted in Figure 2.3. Recurrent neural networks are specialized for processing sequences (Goodfellow et al., 2016).

A known challenge of recurrent neural networks is the *vanishing gradient problem* (Hochreiter, 1991). It states that the influence of an input either changes exponentially – either in- or decreasing – while it is propagated in the network. LSTM tackles this challenge by employing a novel architecture: a LSTM block (unit) is used instead of a typical neuron that applies an activation function to the sum of its inputs. A LSTM block has three gates – input, output and forget – that combine the signal with the state of the block. With this technique, flow of the recurring signal can be controlled.

LSTM has been used to many real-world applications, e. g., speech recognition, handwriting recognition, reinforcement learning (Graves, 2012).

2.1.4. Hyperparameters of a Neural Network

The task of creating a suitable neural network for a specific task is challenging. Several parameters of a neural network have to be set before starting the training process.

First, a type of network has to be chosen for the specific application. See the article of Schmidhuber (2015) for an overview of different types. There are known conditions implying which type of network is suitable. E.g., in image processing CNNs are commonly used and LSTMs find their application in time-series analysis like speech recognition. These are just recommendations and there are several applications where a type of network is used in other contexts, e. g., `deepl.com` uses CNNs for machine translation (Thommes, 2017).

With a selected type of network there are many parameters to set. E.g., in a deep neural network consisting several dense (fully-connected) layers the number of neurons in each layer has to be set. Large layers bring the risk of over-fitting faster (Aggarwal, 2018) and need more computation power but small layers may not match the classification performance, as less features can be learned. As another example, in convolutional layers the convolution is to be parameterized: size of the kernel, stride and number of generated feature maps. For further information on setting the size of a neural network the recent work of Belkin et al. (2019) is recommended.

Furthermore, there are general parameters like the number of training epochs. With a large number of training epochs the model is likely over-fitting to the training data. With a small number of training epochs the model's learnable parameters are not adjusted enough to fit the task. In the learning process, more specifications have to be set. For the backpropagation algorithm a loss function has to be defined. Furthermore, a learning rate and an optimizer may be set. They influence the rate of change for learnable parameters. Adam (Kingma and Ba, 2015) and stochastic gradient descent are popular approaches for this issue.

The task of finding a good configuration of hyperparameters is known as hyperparameter tuning. Classification quality can be improved by hyperparameter tuning. Often, this is a manual task, but automated approaches are available (Bardenet et al., 2013).

2.2. Metrics

This section defines metrics used for measuring the quality of a classification. These metrics are already implemented by Pedregosa et al. (2011) in their *Python*-package *scikit-learn*. The package is used in the evaluation chapter to calculate classification quality scores.

In Sections 2.2.1 and 2.2.2, metrics for binary classification are introduced. Then, in Section 2.2.3, these definitions are used to generalize metrics for multiclass classification. Section 2.2.4 introduces confusion matrices as a compact overview of scores.

2.2.1. Accuracy

In binary classification, each sample is matched with one of two classes. Every sample has a true class that it belongs to. The classification then yields a prediction for the sample, i.e. the predicted class. Note, that in this chapter, classified and predicted are equivalent terms, describing the output of a classifier. One of the two classes is called the positive class, the other one is called negative class. The true and predicted classes can be positive or negative. This results in the definition of *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) and *False Negative* (FN) given in Table 2.1, where if true and predicted class match, the sample is correctly classified.

		true class	
		positive	negative
predicted class	positive	*True Positive* (TP)	*False Positive* (FP)
	negative	*False Negative* (FN)	*True Negative* (TN)

Table 2.1.: Definitions of *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) and *False Negative* (FN).

TP and TN are correctly classified samples. In statistics, FP is also known as *type I error* and FN is known as *type II error*. Furthermore, accuracy is then defined as a ratio of the number of correctly classified samples to the number of all samples in a set:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is the most used metric in the reviewed related work. It indicates best performance at its maximum value of 1. The following section defines more metrics used for measuring quality of machine learning approaches.

2.2.2. Precision, Recall, F_1-Score

While accuracy serves well as a quick and compact evaluation of a classifier, other metrics can give insight into the quality of the classification to reveal certain strengths and weaknesses. These are precision, recall and F_1-Score. They are defined as follows:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

A low precision indicates that many samples are incorrectly classified as positive labels. Recall (also known as sensitivity) can be interpreted as a classifier's ability to find all positive samples. As an example, consider a trivial classifier that always predicts the true label. Its recall is 1, as there are no FN. Its precision is the same as its accuracy and depends on the set in terms of the ratio of positive and negative samples. Lastly, the F_1-Score is defined as the harmonic mean of recall and precision.

All defined scores indicate best performance at their maximum value of 1. Each score should always be read in a context, e. g., in some applications a high recall is necessary, while other application may profit from a high precision at cost of recall. When detecting tumors (Seetha and Raja, 2018) in medical applications a high sensitivity is favorable over high precision, as not recognized tumors have severe consequences.

2.2.3. Multiclass Classification Metrics

The previously defined metrics are used for measuring performance of a binary classifier. The application of this thesis is multiclass classification. This means that there are multiple disjoint classes and each sample belongs to exactly one of the classes. This can also be generalized to multilabel classification, where each sample may belong to multiple classes.

To evaluate the performance of a multiclass classifier, the previously defined metrics can be used as follows: For every class the metric is calculated with that class being the positive class and all other classes combined being the negative class. Then these scores are averaged. The calculation of the average may be weighted. Depending on the weights properties of the classifier can be emphasized or suppressed.

Different averaging methods are discussed in this example: Given a sample with true labels y_{true} and predicted labels $y_{predicted}$ as follows.

$$y_{true} = [1,1,1,0,1,1,2,2,2,2,2,1,1] \tag{2.2}$$

$$y_{predicted} = [1,1,1,1,1,1,2,2,2,2,2,2,0] \tag{2.3}$$

Defined metrics are calculated for every class. Results are given in Table 2.2.

	Precision	Recall	F_1-Score	Support
0	0.00	0.00	0.00	1
1	0.83	0.71	0.77	7
2	0.83	1.00	0.91	5
micro avg	0.77	0.77	0.77	13
macro avg	0.56	0.57	0.56	13
weighted avg	0.77	0.77	0.76	13

Table 2.2.: Metrics for a multiclass classification example.

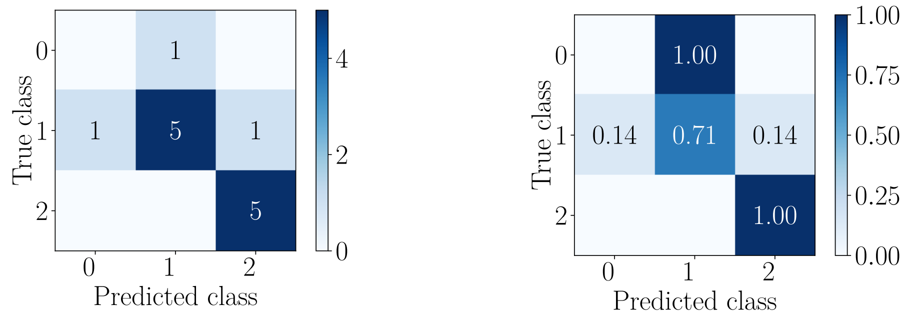
The support of each class is the number of occurrences of itself in y_{true}. To summarize metrics of every class, different approaches are eligible. Macro averaging is a naive approach, as every class is weighted with 1. If the classes are unbalanced this does not represent a classifiers true ability. Weighted averaging weights the score of every class with its support. This represents the distribution of classes better. Micro averaging calculates the scores based on all results. All counts of TP, TN are accumulated and scores are calculated with the formulas in Section 2.2.2. When applying micro averaging to precision and recall with all classes selected these scores degenerate to accuracy: The number of TP is calculated by counting the correctly classified samples. The remaining samples are incorrectly classified and are FP and FN simultaneously (i. e., a sample of class 1 is classified as 2 - it is a FN for class 1 and a FP for class 2). With these identities $FN = FP$ and $FN + TP$ being the number of all samples the formulas for precision and recall calculate the ratio of correctly labeled samples to all samples.

2.2.4. Confusion Matrix

To review classification performance for each class, confusion matrices are a popular presentation form. An element c_{ij} of the confusion matrix is the number of times samples of class i are classified as class j. All correct classifications are on the diagonal, incorrect ones are off the diagonal (Sammut and Webb, 2017). Furthermore, rows of a confusion matrix contain TP and FN of a class. Columns contain TP and FP of a class.

For better comparison normalization is performed. To normalize a confusion matrix, every row is normalized, i. e., dividing elements c_{ij} by the sum of row i. In a normalized confusion matrix, an element c_{ij} gives the ratio of how many samples of class i are classified as class j. Thus, values on the diagonal are the recall of a class.

In Figure 2.4, the absolute and normalized confusion matrix are given for the example of Section 2.2.3.



Absolute confusion matrix Normalized confusion matrix

Figure 2.4.: Confusion matrices for the example of Section 2.2.3.

2.3. Evaluation Types

This section explains different evaluation types commonly used in the context of human motion classification. First, in Section 2.3.1 stratified k-fold cross validation is explained and motivated. Sections 2.3.2 to 2.3.4 present different approaches of splitting the data set into training and testing set. Each evaluation type is motivated by possible real-world scenarios.

2.3.1. k-Fold-Cross Validation

Measuring the quality of a classifier can be done with different metrics (e. g., those explained in Section 2.2). The scores of those metrics assess quality of a classification approach. These results depend on the data used for evaluating a classifier. Thus, the *true* quality of a classifier cannot be calculated and must be approximated with data. As methods of machine learning are commonly applied to unseen data, this constraint should be taken into account when training a model. Therefore, a data set is split into a training and a testing set. The model is fit to the data in the training set and evaluated with data of the testing set. If the model performs significantly better on the training than the testing set, this is known as over-fitting (Aggarwal, 2018). An arbitrary split of the data set into training and testing might not be representative and calculating the classification error with this split yields a faulty estimation of the quality of a classifier. A popular method to estimate the classification error is k-fold cross validation (Sammut and Webb, 2017).

This approach splits the data set into k disjoint parts of approximately equal size. Then each part (fold) serves as the test set, while the other parts combined are the training set. An example for 3-fold cross validation is given in Table 2.3.

Round	Data Set			Metric Score
1	Train	Train	Test	a
2	Train	Test	Train	b
3	Test	Train	Train	c
Average				$(a+b+c)/3$

Table 2.3.: Example rounds of a 3-fold cross validation. The data set is split into 3 parts and each part serves as test set once.

k-fold cross validation yields k evaluations of classification quality. The estimation of the true error is the arithmetic mean of the errors in each fold.

This approach is enhanced by stratification: The data set is not arbitrarily split but the data is divided into folds that approximately have the same distribution of data. Stratification prevents folds from being unbalanced (e. g., in the worst case not containing data of one class).

The following sections present different evaluation types. In general, these evaluations describe a specific selection of data. They are based on k-fold cross validation, i. e., a classifier's quality is estimated with k-fold cross validation splitting selected data into training and testing set.

2.3.2. Single-Subject Evaluation

Single-Subject evaluation estimates the quality of a classifier for every subject in the data set individually: only the data of one subject is used for evaluating the quality of a classifier. In this manner, the classifier is evaluated for each subject. The classification error is then estimated as the average of these evaluation values.

This training technique may be applicable for exoskeletons that are worn by exactly one user. With this constraint it is possible to optimize a classifier specifically to the user. A downside of this approach is the requirement of recording user's motion data and training the classifier prior to using it.

In human motion analysis, trajectories of one motion type usually differ, i. e., a motion is not executed in the exact same way every time. This phenomenon is observable within repetitions performed by one subject (intra-subject variance), but also two persons typically perform the same motion differently, yielding inter-subject variance.

In Single-Subject evaluation, only intra-subject variance in motion execution is of interest.

2.3.3. Multi-Subject Evaluation

In Multi-Subject evaluation, motion data of multiple subjects is used as training and test set. In the context of lower limb exoskeletons, there are applications where a Multi-Subject evaluation is representative: e. g., an augmenting exoskeleton used by multiple people.

In this evaluation type, inter-subject variance is an additional challenge in comparison to Single-Subject evaluation (Section 2.3.2). A classifier trained within this evaluation is more general than one trained within Single-Subject evaluation as multiple subjects are expected to be classified accurately.

2.3.4. Leave-One-Subject-Out Validation

In Single-Subject and Multi-Subject evaluation (see Sections 2.3.2 to 2.3.3), data in the test set is expected to be similar to the one in the training set, as both are motion data from one subject, thus only intra-subject variance is the difference between a sample in the test set and one in the training set. In Leave-One-Subject-Out validation, the data of one person serves as the test set, while data of multiple other subjects serves as the training set. With this method the classifier is evaluated with data of an unseen subject. To classify this data, inter-subject variance is to be tackled. Leave-One-Subject-Out validation in the context of motion classification is a commonly used technique to test the generalization performance of a classifier.

This evaluation type is also supported by possible applications in motion classification: a trained classifier is deployed and classifies data from subjects that were not represented in the training process.

Chapter 3.

Related Work

This chapter presents approaches in the field of motion classification. Three different components of a concept are reviewed: System for recording data (Section 3.1), classification method (Section 3.2) and evaluation of the concept (Section 3.3). Section 3.4 summarizes and compares the related work.

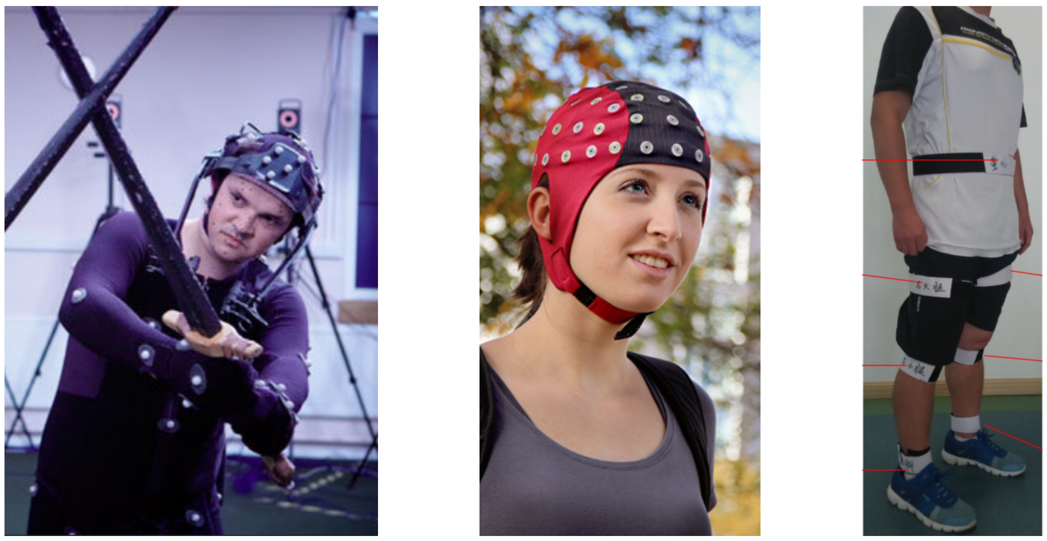
3.1. Sensor System

Different sensory systems are available to record data in the context of human motion analysis. This section presents the most common sensor setups. In Figure 1.1, examples of such systems are depicted.

3.1.1. Motion Capture and Cameras

Motion capture systems (e. g., Vicon, see Figure 3.1(a)) are a high quality tool in human motion analysis. Reflective markers are placed on a subjects body and their positions are traced with infrared cameras. This system is accurate for coarse movement but suffers from high cost and high set-up effort. Classification applied to motion capture data appears in Li et al. (2007); Lv and Nevatia (2006); Mandery et al. (2016). A cheaper system for motion capturing is vision-based. Here joint angles can be calculated from, e. g., pose detection (Moeslund et al., 2006). A similar approach can be found in Ratanamahatana and Keogh (2004).

These approaches are not suitable in the context of exoskeletons, as their usage is limited to the location of the sensory system. Therefore, literature based on data recorded with sensors mounted at the subject is surveyed hereafter.



(a) (b) (c)

Figure 3.1.: Examples for common sensor systems for motion capture.
Left: Vicon motion capture. Markers are placed on the subject, cameras trace their position. Picture from (Digital Arts).
Middle: Electroencephalogram. Electrodes placed at the head measure electric activity in the brain. Picture from (ANT Neuro).
Right: IMUs placed on the subject indicated with red lines. Picture is conception of Wang et al. (2015).

3.1.2. Electroencephalogram and Electromyogram

A popular concept of data collection in the context of motion classification is recording bioelectrical signals. Electroencephalography (EEG, e. g., Figure 3.1(b)) measures electric signals in the brain, while electromyography (EMG) measures electric signals at the muscles. This technique is very suitable when working with physically impaired subjects, as no physical motion itself is necessary for classifying the user's current motion intention. Classification is applied to such data in Villa-Parra et al. (2015); Huang et al. (2009); Yin et al. (2012). According to Beil (2019) these sensors require controlled environments and are complex to set up, as electrodes have to placed carefully. Therefore, they are not feasible for the usage with augmenting exoskeletons. Thus, in the context of augmenting exoskeletons, sensors that are mounted at the exoskeleton itself are preferable.

3.1.3. IMUs and Force Sensors

Motion changes the pose of a subject's limbs. This change can be measured with Inertial Measurement Units (IMUs). Possible features are orientation and acceleration of body parts. More features like joint angles can be derived. Kronenwett et al. (2017) used a single IMU attached to the foot to detect specific phases of the gait cycle and classifies running and walking. Um et al. (2017) classified gym exercises of many athletes recorded with a wearable forearm band containing an accelerometer and a gyroscope. Gong et al. (2018) attached two IMUs to the subjects' thighs and classify six locomotion types. Tunçel et al. (2009) used the data of two IMUs at an subjects leg to classify eight motions. Wang et al. (2015) attached seven IMUs to the subject and classify seven basic motions. In Figure 3.1(c), their sensor setup is depicted. Furthermore, interaction forces between the subject and the exoskeleton or environment can measured. Taborri et al. (2015) used IMUs and force sensors under the sole to classify the current phase of the gait cycle. The most closely related work is the one of Beil et al. (2018). They used data of three IMUs and seven force sensors attached to the exoskeleton to classify 13 common motions.

In addition, finding a minimal sensor setup is desirable to reduce cost and energy consumption of sensor systems (see, e. g., Daab, 2019; Patzer and Asfour, 2019). For further discussion on the topic, the survey of Novak and Riener (2015) is recommended.

3.2. Motion Classification Methods

Classifying data is a widespread task with many real-world applications. The field of machine learning offers many tools to solve classification tasks. In the context of motion classification, a comparison of several classifiers can be found in Tunçel et al. (2009). They apply Bayesian Decision Making, decision trees, least-square methods, k-nearest neighbor, dynamic time warping, support vector machines and artificial neural networks as classifiers for their data. As their work shows, many different techniques are eligible to tackle the classification task. In addition to the classification quality (see Section 2.2), other properties of a technique can be investigated, e. g., a short processing time to classify a data point is critical to do online classification. The following paragraph discusses different motion classification approaches.

In Kronenwett et al. (2017), a finite state machine was employed. States of this machine represent phases of locomotion. State transitions are triggered if recorded data exceeds manually defined thresholds. This technique is very efficient, as finite state machines are computationally cheap.

In Wang et al. (2015), a support vector machine performed the classification. Gong et al. (2018) employed an artificial neural network. Villa-Parra et al. (2015) compared these approaches by applying support vector machines and artificial neural networks. In their experiments, the artificial neural network showed a better performance than the support vector machine.

Beil et al. (2018) used Hidden Markov Models (HMMs). In their approach, each motion class corresponds to one HMM. To classify a sample, it is fed into every HMM. This yields a likelihood for every class and the class with highest likelihood is assigned to the sample. In Beil (2019) this approach competes with a final state machine and the author concluded that HMMs outperform final state machines. Their classification process is depicted and described in Figure 3.2.

Deep learning methods in the context of motion classification were also previously tested. For instance, Um et al. (2017) employed a Convolutional Neural Network (see Section 2.1.1). High computation power is required for training a deep neural network. After training, a model can be deployed to a mobile processing unit. Therefore, the

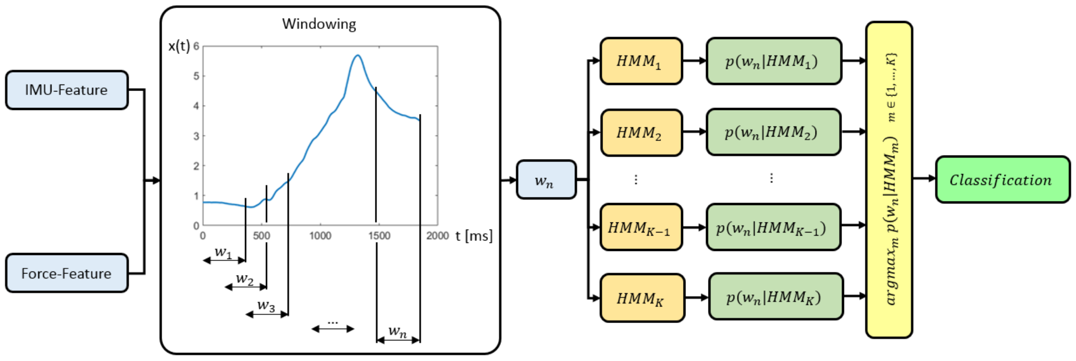


Figure 3.2.: Classification process with HMMs. In the training phase, one HMM is built for each motion class. Recorded features are concatenated and split into windows (left half). A window w_n is then processed by every HMM calculating probabilities $p(w_n|HMM_i)$, where i corresponds to a motion class. The window w_n is then classified as class j with $p(w_n|HMM_j) \geq p(w_n|HMM_i)$ for all i, i. e., argmax of probabilities. Figure is conception of Beil et al. (2018).

network structure and learned parameters (weights) have to be stored. The necessary storage capacity depends on the network size. Likewise, the computational complexity increases with the size. Thus, for online classification a powerful processing unit may be necessary.

3.3. Evaluation Methods

Evaluating a classification method requires a sample data set. Depending on the data set, different evaluation methods are available (e. g., see Section 2.3).

Furthermore, a data set consisting of recordings of more than one person are desirable to examine classification quality variance within subjects. This allows comparison between subjects. Moreover, more data points in the data set allow evaluation with more samples, thus giving a better approximation of the classification quality. Also, some classification tasks are harder than others, e. g., distinguishing similar classes is intuitively a harder task than classifying distinct classes. The number of classes can indicate the difficulty of a task.

Evaluation based on data of a single subject or where this is not specified is common in the literature (Tunçel et al., 2009; Wang et al., 2015; Kronenwett et al., 2017; Gong et al., 2018). Others gathered large data sets for their evaluation (Beil et al., 2018; Um et al., 2017). The latter also uses real-world data, while other data sets are built in lab conditions.

Before classifying data, it is usually converted into a uniform format. A popular approach to perform fast classification is called sliding window approach. Here, recorded data is split into windows of fixed size (Villa-Parra et al., 2015; Beil et al., 2018; Gong et al., 2018). Windows can be created at an arbitrary rate resulting in overlapping windows. Classification is then executed with such windows as data.

Others (Um et al., 2017; Wang et al., 2015) classify full repetitions of human motion. This procedure is not suitable in the context of augmenting exoskeletons, as online classification (classifying while the motion is performed) is necessary.

3.4. Classification Comparison

A compact comparison of approaches in the literature is given in Table 3.1. This list is limited to contributions where data is recorded with mechanical sensors, as these are considered suitable for recording data in the context of augmenting exoskeletons. Details about the presented literature can be found in the Sections 3.1 to 3.3.

Authors	Sensors	Method	Data Set	Window Size [ms]	Accuracy [%]
Tunçel et al. (2009)	2 Gyroscopes	Multiple methods	8 classes 1 subject	10000	94.2 to 98.2
Wang et al. (2015)	7 IMUs	SVM	7 classes n.a.	full motion	99.14
Um et al. (2017)	1 IMU	CNN	50 classes 1748 subjects	3920	83.34
Gong et al. (2018)	2 IMUs	ANN	6 classes 1 subject	250	97.79
Beil et al. (2018)	3 IMUs 7 Force Sensors	HMM	13 classes 10 subjects	300	92.80

Table 3.1.: Comparison of approaches in the literature. Different sensor systems are used for recording motion data, e. g., Um et al. (2017) uses one IMU, while Wang et al. (2015) mount seven IMUs at the subject. The third column lists several methods for motion classification. To test an approach, data sets of varying size are used. Small data sets may not be representative and thus, evaluation may not assess a classifiers *true* performance. The second to the last column shows the length of time windows, that serve as samples to classify. Large windows may not be suitable for online classification, which is desirable for augmenting exoskeletons. In the last column, classification accuracy of the approaches is given.

Approaches, where only data of a single subject is used, achieve best classification accuracy. This may be due to the fact, that their classifiers do not have to face the challenge of inter-subject motion variance. Best classification accuracy is achieved by Wang et al. (2015). In comparison to other approaches, they use a large amount of sensors, their data set has a small number of classes (motion types) and classifies data of full motion execution. This setting may not be applicable for real-world usage, as a low amount of sensors is desirable, many different human motions are possible and classification while motion is performed is required. Other approaches tackle some of these shortcomings, but classification accuracy drops.

Chapter 4.

Exoskeleton Design and Motion Data

This chapter describes the used exoskeleton from H^{2}T and how motion data was gathered.

First, the exoskeleton is described and used sensors and their placement are discussed in Section 4.1. Then, recorded and derived features are listed in Section 4.2. Finally, in Section 4.3 subjects and motions in the data set are presented.

4.1. Passive Lower Limb Exoskeleton

The exoskeleton used for this thesis was proposed by Beil et al. (2018). This passive lower limb exoskeleton for the left leg is depicted in Figure 4.1. This figure also shows the placement of attached sensors and the sensors itself. The exoskeleton consists of three frames made of soft aluminum, one for each thigh, shank and foot. They are connected with orthotic revolute joints (Otto Bock HealthCare; *17B47=20 / 17B57=20*) at the knee and ankle. To fixate the exoskeleton to the user, Velcro straps are used. The use of soft aluminum allows for small adjustment to the user over the otherwise rigid exoskeleton. There are three inertial measurement units (IMUs) placed at the upper leg, lower leg and the foot. The IMUs used are *3BNO055 IMU* by Robert Bosch GmbH. Furthermore, there are seven 3D-force sensors (Optoforce Ltd.; *OMD-30-SE-100N*) installed to Velcro straps measuring forces between the exoskeleton and the user's leg muscles.

4.2. Features

The following aspects are conception of Beil et al. (2018). The orientation and acceleration of every IMU is processed by a micro-controller (4SAM3X8E ARM Cortex-M3, Microchip Technology Inc.) at a frequency of 80Hz. The micro-controller stores orientation as quaternions and linear acceleration. Another representation of

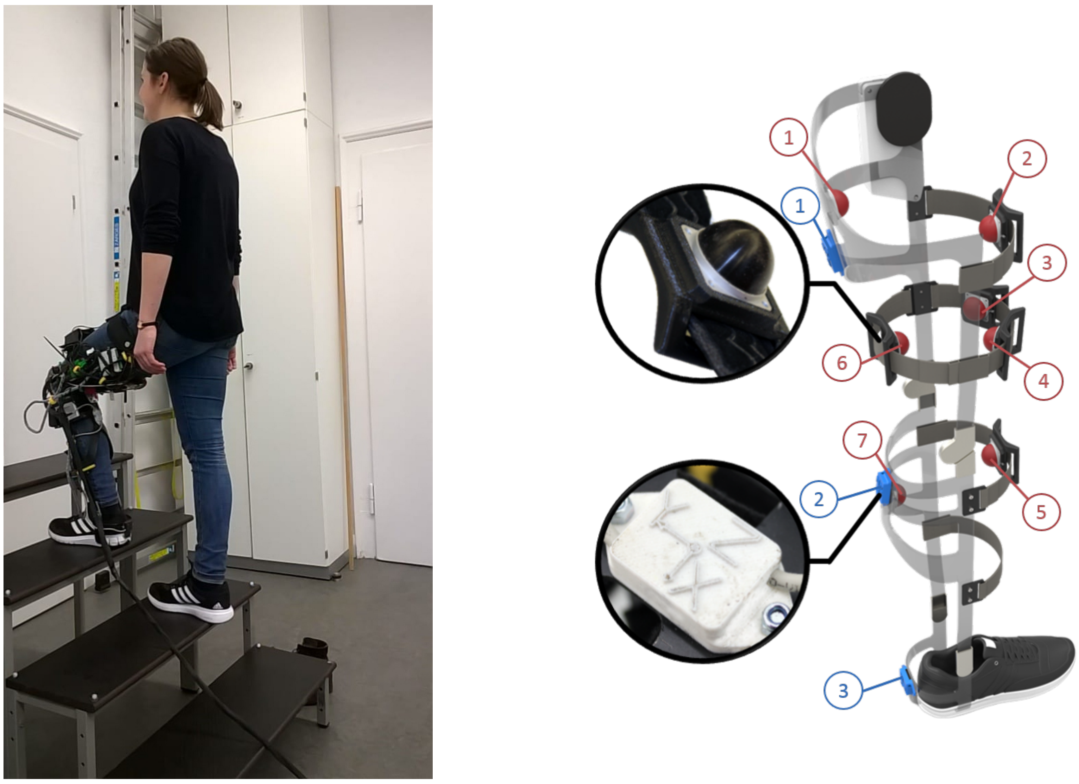


Figure 4.1.: Left: The passive lower limb exoskeleton in use. The exoskeleton is worn at the left leg. Electronics (sensors, processors) are powered by an external source. The exoskeleton offers stability due to its stiffness, but human motion is not assisted mechanically.
Right: Exoskeleton design and sensor placement. Force sensors are labeled in red, IMUs are labeled in blue. Force sensors are placed over big leg muscles in upper and lower leg. IMUs are placed at each main part of the leg, i. e., thigh, shank and foot.
Pictures are conception of Beil et al. (2018).

orientation in form of Roll-Pitch-Yaw Euler angles is calculated, as representation with quaternions is ambiguous. The force sensors capture the interaction forces between the exoskeleton and the subject. Two data acquisition units convert the raw analog data of the seven force sensors into a digital signal with a frequency of 100Hz. With this setup, signals are available at different frequencies. For further processing, signals of the IMU were interpolated to 100Hz and were merged with the signals of the force sensors. Differences between consecutive data points are calculated to normalize force values. Absolute force values may differ due to physical differences between subjects and different tightening of the Velcro straps. These differences serve as data for classification.

A common approach to classify sequences is called sliding window approach. Here consecutive data points are concatenated and form a window. These windows may also overlap. Window size can be chosen arbitrarily, and fixed or dynamic sizes are

	Average	Std-Dev
Age [a]	25.10	3.86
Height [m]	1.73	0.03
Weight [kg]	66.00	5.85
BMI [kg=m2]	22.05	1.54
UL Circumference [cm]	55.20	3.20
LL Circumference [cm]	36.75	1.31
UL Length [cm]	42.80	2.63
LL Length [cm]	41.10	2.30

Table 4.1.: Overview of subject attributes. This table was published by Beil et al. (2018).

also possible. Some classifiers can only process windows of fixed size, thus limiting the above decision.

4.3. Data

This section gives an overview of the data set that is the base of this thesis. The data set was recorded by Beil et al. (2018). Section 4.3.1 lists attributes of all subjects that participated in the motion recordings. Then a list of all motions is given in Section 4.3.2.

4.3.1. Subjects

The data set contains recordings of motions executed by 10 healthy subjects. The passive exoskeleton used in the recordings allows only small variance in the size of the users as it is hardly adaptable to the size of the wearer. An overview of all parameters is given in Table 4.1. A full list of subjects and their physical characteristics is given by Daab (2019). Note that in this thesis, IDs were changed to 1-10 for better readability. See Appendix A.1 for this mapping.

4.3.2. Motions

All subjects performed the following motions (Beil et al., 2018). This list is extended by the *stand* motion, as proposed in their outlook.

- Walking Forward (WF)
- Walking Backwards (WB)
- Turn Left (TL)
- Turn Right (TR)
- Sidesteps Right (SR)
- Sidesteps Left (SL)
- Going Upstairs (GU)
- Going Downstairs Backwards (GB)
- Going Downstairs (GD)
- Lift Object (LO)

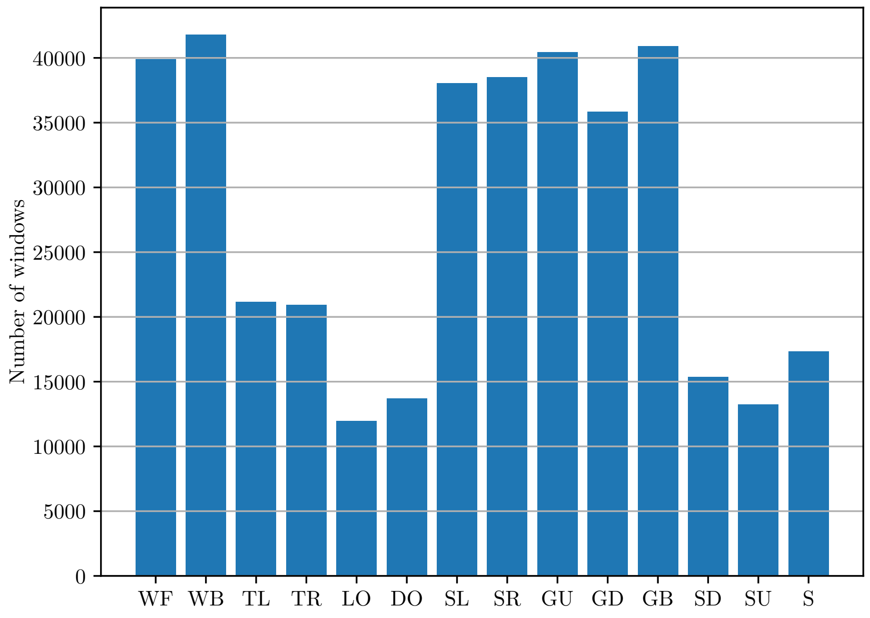


Figure 4.2.: Number of 300 ms windows of each motion type.

- Drop Object (DO)
- Stand Up (SU)
- Sit Down (SD)
- Stand (S)

Each motion was recorded ten times for every subject. The distribution of classes is given in Figure 4.2. On the one hand motions like GB take a long time to execute, as several steps are taken. On the other hand there are very short motions like SU or TL. Thus, the latter classes contain less windows and there is an imbalance of classes. This leads to challenges in calculating average scores of the classification. See Section 2.2 for further information. Thus, weighted averaging is suitable to calculate average scores.

Chapter 5.

Deep Learning Architectures

This chapter presents deep learning architectures employed in this thesis. Section 5.1 introduces a Convolutional Neural Network design, and Section 5.2 introduces a Long Short-Term Memory network design. The network designs were implemented with *TensorFlow* by Abadi et al. (2015).

5.1. Convolutional Neural Network Design

Um et al. (2017) employed a Convolutional Neural Network to classify exercise motion data. The network design used in this thesis is inspired by the work of Um et al. (2017). Thus, a similar design is employed. This basic architecture was also employed by Krizhevsky et al. (2012). Since the data set of H^2T serves more features than the one Um et al. (2017) used, an additional convolution was added to further reduce the data. Also more feature maps are generated in the convolutional layers to scale the network model to the data (see Section 4.2). The model is presented in Figure 5.1. The convolutional layers have 66, 64 and 62 feature maps. The output of the last layer is then flattened (i. e., a matrix input is reshaped to a vector output) and passed

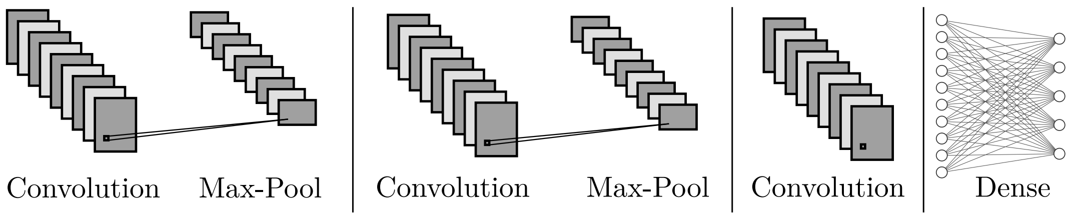


Figure 5.1.: The Convolutional Neural Network has three convolutional layers in the beginning and two dense layers for classification functionality. Figure was created with *NN-SVG* by LeNail (2019).

to the dense part of the architecture. First, a dense layer of 1024 neurons serves as a feature selector. The last layer is a dense layer of 14 neurons serving as the classifier. Its output is evaluated with a *softmax*-function to select the class.

The following parameters for the neural network are taken from Um et al. (2017). The Adam optimizer (Kingma and Ba, 2015) is used for optimization with a learning rate of 0.0005. Cross-entropy loss serves as the loss function in the training phase of the model. Dropout is a popular method to reduce over-fitting (Srivastava et al., 2014). It prevents propagation of signals in the network with a given probability. A probability of 0.5 is chosen for dropout.

5.2. Long Short-Term Memory Network Design

Basic concepts of LSTM are explained in Section 2.1.3. A simple architecture for classification purposes is proposed: a model consisting of just two layers. The first layer has 128 LSTM blocks (units), the second layer is a dense layer of 14 neurons serving as the output of the neural network. Classes of the classification task are matched with output neurons. The networks output is evaluated with a *softmax*-function.

Further parameters, like optimizer, learning rate, loss function and dropout probability, are set like for the CNN model (see Section 5.1).

Chapter 6.

Evaluation

This chapter presents quality of classification by the neural networks defined in Chapter 5. Sections 6.1 and 6.2 correspond to the models introduced in Sections 5.1 and 5.2. Each model is evaluated with the methods proposed in Section 2.3. An evaluation type selects specific data, then k-fold cross validation is run for the selected data. In this thesis, the number of folds k is set to 3 as proposed by Daab (2019). In the evaluations, an implementation of stratified k-fold cross validation by Pedregosa et al. (2011) is used. Motion classification quality is measured with the metrics defined in Section 2.2. Features used in evaluations are forces, linear acceleration and orientation (as Euler angles) if not specified otherwise. Section 6.1.4 and Section 6.2.4 present brief studies of hyperparameter tuning for the employed models. Graphics visualizing results of these evaluations are rendered with *matplotlib* by Hunter (2007). Section 6.3 compares classification results of this thesis to the ones of Beil et al. (2018) and Patzer and Asfour (2019). In Section 6.4, computational cost of classification with neural networks is discussed. Training runtime is presented in Section 6.5.

6.1. Convolutional Neural Network Evaluation

This section presents evaluation results for the Convolutional Neural Network architecture (see Section 5.1). Sections 6.1.1 to 6.1.3 correspond to evaluations presented in Sections 2.3.2 to 2.3.4. In Section 6.1.4, a small study for hyperparameter tuning is conducted.

6.1.1. Single-Subject Evaluation

A Single-Subject evaluation was conducted as explained in Section 2.3.2. Results of this evaluation with 300 ms windows are shown in Table 6.1. Detailed results for other window sizes are given in Appendix A.3.1. An overview of average metric scores with different window sizes is given in Table 6.2. Classification quality improves with larger window sizes. A reason for this relation may be the longer history of data points to analyse, thus motion specific events are more likely contained in a window.

Classification accuracy varies within subjects: When classifying 100 ms windows worst accuracy is at 0.931, while best accuracy is at 0.988. Intra-subject variances is considered to be the reason of these differences. Detailed evaluation results are given in Appendix A.3.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99970	0.99970	0.99970	0.99970
ID2	0.99978	0.99978	0.99978	0.99978
ID3	0.99965	0.99965	0.99965	0.99965
ID4	0.99973	0.99973	0.99973	0.99973
ID5	0.99937	0.99937	0.99937	0.99937
ID6	0.99966	0.99966	0.99966	0.99966
ID7	0.99946	0.99946	0.99946	0.99946
ID8	0.99985	0.99985	0.99985	0.99985
ID9	0.99996	0.99996	0.99996	0.99996
ID10	0.99979	0.99979	0.99979	0.99979
average	0.99969	0.99969	0.99969	0.99969

Table 6.1.: Single-Subject evaluation results of the CNN model with windows of 300 ms length.

Window Size [ms]	Accuracy	Recall	Precision	F_1-Score
100	0.9680	0.9680	0.9685	0.9681
200	0.9988	0.9988	0.9988	0.9988
300	0.9997	0.9997	0.9997	0.9997
400	0.9998	0.9998	0.9998	0.9998
500	0.9998	0.9998	0.9998	0.9998

Table 6.2.: Single-Subject evaluation average results of the CNN model with different window sizes.

6.1.2. Multi-Subject Evaluation

Evaluation results of Multi-Subject evaluation (see Section 2.3.3) for different window sizes are presented in Table 6.3. Classification quality benefits from larger window sizes. Figure 6.1 shows the confusion matrix of an evaluation with 100 ms windows. Confusion matrices are introduced in Section 2.2.4. Several often confused motions can be observed, for example, SD and DO is the pair of motions with highest confusion rate.

Window Size [ms]	Accuracy	Recall	Precision	F_1-Score
100	0.92962	0.92962	0.93014	0.92962
200	0.99364	0.99364	0.99368	0.99365
300	0.99878	0.99878	0.99879	0.99878
400	0.99961	0.99961	0.99961	0.99961
500	0.99990	0.99990	0.99990	0.99990

Table 6.3.: Multi-Subject evaluation average results of the CNN model with different window sizes.

6.1.3. Leave-One-Subject-Out Validation

This section presents results of Leave-One-Subject-Out validation (see Section 2.3.4). Table 6.4 contains results of two evaluations for different window sizes. Cross-validation accuracy of the training set (CV accuracy) and classification accuracy for the left out subject (LO accuracy) are given. As in previous evaluations, larger windows result in better classification accuracy. Inter-subject variance of LO accuracy is very notable. For 300 ms windows best LO accuracy is 0.96, while worst is at 0.82. A difference of 14 percentage points is large. It can be concluded that some subjects can be classified very well, while performance drops significantly for others. Classification results for other window sizes are given in Appendix A.4.1.

While previous evaluations were performed with the features Beil et al. (2018) used (force sensors, IMUs as linear acceleration and Euler angles), this evaluation had the orientation of the IMUs also given as quaternions. This results in better classification accuracy. Quaternions have a steady representation in contrast to Euler angles. Thus, interpolation works better with quaternions resulting in more accurate classification. For classification results for various feature combinations see Appendix A.2.

Compared to results of Single-Subject evaluation in Section 6.1.1, some similarities are observable. For example, most subjects, that achieve below average motion classification accuracy (see Table A.8), also achieve below average LO accuracy (see Table 6.4). This may be due to ambiguous motions, raising a challenge for a generalizing and an optimized model.

6.1.4. Hyperparameter Tuning

Finding a good set of hyperparameters is a difficult task. Methods for finding optimal hyperparameters are presented in literature (e. g., Bengio (2000) and Biedenkapp et al. (2017)). Hyperparameters of neural networks are presented in Section 2.1.4. These parameters were manually manipulated to see their effects and find a better configuration of the models presented in Chapter 5. Different modifications are investigated. They are presented in the following enumeration.

1. Larger dense layer. The second to the last layer is enlarged to have 2048 neurons.
2. Smaller dense layer. The second to the last layer has 512 neurons.
3. Smaller dense layer. The second to the last layer has 256 neurons.

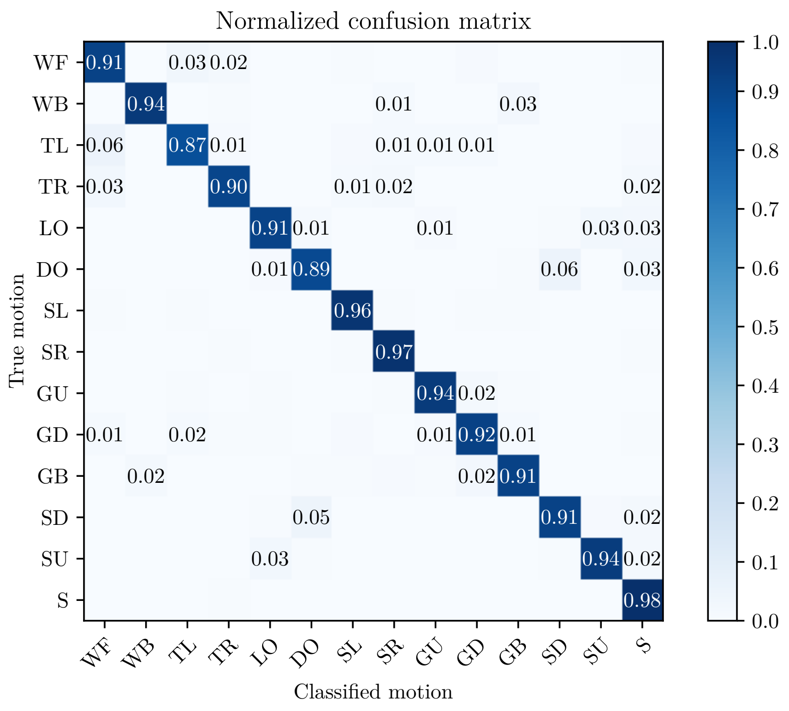


Figure 6.1.: Confusion matrix of Multi-Subject evaluation of the CNN model with 100 ms window size. Numbers smaller than 0.01 are hidden. Acronyms for motions are defined in Section 4.3.2. DO and SD as well as LO and SU are often confused. This can be explained as these pairs are similar motions. Furthermore, WB and GB are also often confused for the same reason. Turning left and right are distinguished well, but both are often confused with WF.

4. No Dropout. Dropout is completely removed from the model.
5. Vary Training epochs. A model is trained for 100 epochs and evaluated after every epoch.

Modified models are evaluated with Leave-One-Subject-Out evaluation as explained in Section 2.3.4 to test their generalization performance. All features are used as in Section 6.1.3. Leave-One-Subject-Out evaluation is fairly expensive, thus evaluations are conducted with one subject left out (ID4). In Section 6.1.3, an architecture is evaluated for ten subjects being left out once. Results are presented in Table 6.5.

LO accuracy is best for configuration 3. Cross-validation accuracy does not change a lot, and is best for the standard model. The LO accuracy oscillates a lot over epochs (see Figure 6.2). Therefore, having a large model suits well for tasks where validation data is similar to the training data (e. g., no inter-subject variance like in Single-Subject evaluation). A smaller number of neurons can reduce the risk of over-fitting to the training data and also reduces complexity of the model. Since LO accuracy is competitive when using a smaller model they are recommended when the

Subject	CV accuracy	LO accuracy
ID1	0.99005	0.91470
ID2	0.99101	0.94147
ID3	0.99192	0.84207
ID4	0.99305	0.84695
ID5	0.99213	0.88144
ID6	0.99130	0.90114
ID7	0.99199	0.81628
ID8	0.98966	0.88822
ID9	0.99102	0.88391
ID10	0.99242	0.78305
average	0.99145	0.86992
std-dev	0.00100	0.04527

Subject	CV accuracy	LO accuracy
ID1	0.99981	0.95715
ID2	0.99979	0.96420
ID3	0.99980	0.90282
ID4	0.99987	0.89712
ID5	0.99980	0.93584
ID6	0.99984	0.94466
ID7	0.99980	0.87592
ID8	0.99956	0.93056
ID9	0.99970	0.92094
ID10	0.99983	0.82321
average	0.99978	0.91524
std-dev	0.00009	0.04026

Table 6.4.: Leave-One-Subject-Out validation results of the CNN model for different window sizes. The second row contains accuracy of cross validation (CV), the third row contains accuracy of classifying the left out subject (LO). 100 ms on the left, 300 ms on the right. All features were used in this evaluation.

Modification	CV accuracy	LO accuracy
none	0.99988	0.89712
1	0.99986	0.89064
2	0.99977	0.89907
3	0.99968	0.90528
4	0.99937	0.87913
5	0.99798	0.89221

Table 6.5.: Results of hyperparameter tuning (Section 6.1.4). First row is the standard model (see Section 5.1) with no modification.

task is to classify an unseen subject as complexity of the model is reduced. Belkin et al. (2019) pointed out that (much) larger models can overcome the challenge of over-fitting. This can also be an interesting approach in the context of motion classification but will not be investigated in the scope of this thesis.

With the fourth modification (no dropout) the loss is minimized much faster as the network has access to the full data and is not obstructed by dropout. Generalization performance is significantly worse than without dropout, thus confirming the enhancement of dropout (Srivastava et al., 2014).

The last modification is evaluated only for one fold and is plotted in Figure 6.2. The training accuracy increases slower than the cross-validation accuracy as dropout obstructs classification in the training process. The accuracy when classifying motions of the left out subject oscillate around 0.88 after a 10 epochs already. While training and cross-validation accuracy still increase after that epoch, LO accuracy does not increase significantly and is also worse at some points. Thus, it can be concluded that longer training (i. e., more epochs) do not increase the generalization performance of the model in the context of classifying motions of a left out subject. A deeper analysis will be necessary to scientifically prove this statement.

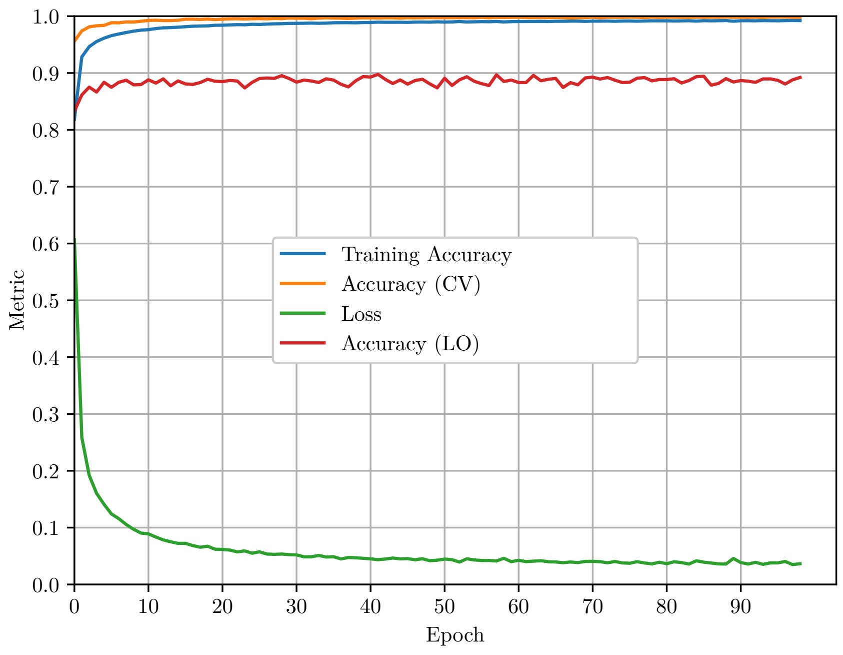


Figure 6.2.: Metrics of Leave-One-Subject-Out validation for ID4 over 100 epochs. Training loss is plotted in green, training accuracy in blue. In the training process weights of the network are adjusted to minimize the loss. Training and cross validation accuracy (orange) are almost constant after about 40 epochs. LO accuracy (red) oscillates around 0.88 after 10 epochs.

6.2. Long Short-Term Memory Network Evaluation

This section presents scores for the model presented in Section 5.2. Sections 6.2.1 to 6.2.3 correspond to the evaluations introduced in Sections 2.3.2 to 2.3.4. In Section 6.2.4, effects of changing hyperparameters of the model are observed.

6.2.1. Single-Subject Evaluation

Single-Subject evaluation is explained in Section 2.3.2. Results of this evaluation with 300 ms windows are shown in Table 6.6. Detailed results for other window sizes are given in Appendix A.3.2. An overview of average metric scores with different window sizes is given in Table 6.7. Like the CNN model, classification quality improves with larger window sizes. Classification accuracy varies within subjects: When classifying

100 ms windows worst accuracy is at 0.946, while best accuracy is at 0.992. Intra-subject variance is considered to be the reason of these differences.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99696	0.99696	0.99699	0.99696
ID2	0.99930	0.99930	0.99930	0.99930
ID3	0.99576	0.99576	0.99578	0.99576
ID4	0.99749	0.99749	0.99749	0.99749
ID5	0.99738	0.99738	0.99740	0.99738
ID6	0.99629	0.99629	0.99632	0.99629
ID7	0.99440	0.99440	0.99453	0.99441
ID8	0.99830	0.99830	0.99831	0.99830
ID9	0.99802	0.99802	0.99802	0.99802
ID10	0.99591	0.99591	0.99595	0.99592
average	0.99698	0.99698	0.99701	0.99698

Table 6.6.: Single-Subject evaluation results of the LSTM model with windows of 300 ms length.

Window Size [ms]	Accuracy	Recall	Precision	F_1-Score
100	0.97283	0.97283	0.97363	0.97298
200	0.99374	0.99374	0.99390	0.99377
300	0.99698	0.99698	0.99701	0.99698
400	0.99807	0.99807	0.99809	0.99807
500	0.99746	0.99746	0.99755	0.99745

Table 6.7.: Single-Subject evaluation average results of the LSTM model with different window sizes.

6.2.2. Multi-Subject Evaluation

Evaluation results of Multi-Subject evaluation (see Section 2.3.3) are presented in Table 6.8. It shows that classification quality benefits from larger window sizes. Figure 6.3 shows the confusion matrix of an evaluation with 100 ms windows. Similar to the confusion matrix of Section 6.1.2, pairs of motions that are often confused can be detected. As for the prior model, SU and LO as well as SD and DO are mistaken for one another more often than other pairs of motions. In addition, the same motions as in Section 6.1.2 are classified as S over 1%. This may be due to the fact that these motions result in a standing motion and are thus classified as S at the end of the recording.

6.2.3. Leave-One-Subject-Out Validation

This section overviews results of Leave-One-Subject-Out validation (see Section 2.3.4). Table 6.9 shows results for different window sizes. Like in the previous evaluations,

Window Size [ms]	Accuracy	Recall	Precision	F_1-Score
100	0.97066	0.97066	0.97094	0.97070
200	0.99422	0.99422	0.99424	0.99422
300	0.99765	0.99765	0.99766	0.99766
400	0.99894	0.99894	0.99894	0.99894
500	0.99895	0.99895	0.99896	0.99894

Table 6.8.: Multi-Subject evaluation average results of the LSTM model with different window sizes.

Subject	CV accuracy	LO accuracy	Subject	CV accuracy	LO accuracy
ID1	0.98429	0.92084	ID1	0.99808	0.95478
ID2	0.98322	0.92730	ID2	0.99778	0.94897
ID3	0.98353	0.85917	ID3	0.99860	0.89128
ID4	0.98635	0.84925	ID4	0.99842	0.89988
ID5	0.98491	0.88033	ID5	0.99871	0.93389
ID6	0.98447	0.89514	ID6	0.99868	0.93717
ID7	0.98555	0.82818	ID7	0.99873	0.89774
ID8	0.98475	0.87231	ID8	0.99801	0.89239
ID9	0.98492	0.87447	ID9	0.99784	0.89008
ID10	0.98623	0.79187	ID10	0.99883	0.82675
average	0.98482	0.86989	average	0.99837	0.90729
std-dev	0.00098	0.03877	std-dev	0.00038	0.03606

Table 6.9.: Leave-One-Subject-Out validation results of the LSTM model for different window sizes. 100 ms on the left, 300 ms on the right. The second row contains accuracy of cross validation (CV), the third row contains accuracy of classifying the left out subject (LO). All features were used in these evaluations.

classification accuracy also increases with larger window sizes. Accuracy increases for every subject, gains in accuracy differ a lot. While accuracy for ID7 increases by almost 7 percentage points, for ID8 this only increases by 2 percentage points. In addition, inter-subject variance is challenging in this evaluation. For 300 ms windows, worst LO accuracy is 0.83, while best is 0.95. A difference of 17% versus 5% incorrectly classified windows is observed. Thus, classifying motion of an unknown subject works fine for some subjects, but for others, the model approach has a bad performance. Classification results for other window sizes are given in Appendix A.4.2.

As stated in Section 6.1.3, an additional representation of the IMUs orientation is used in this evaluation.

Compared to Section 6.1.3, LO accuracy per subject is below or above average for both employed models. This may indicate that motions of some subjects are harder to classify, independent of the classifier.

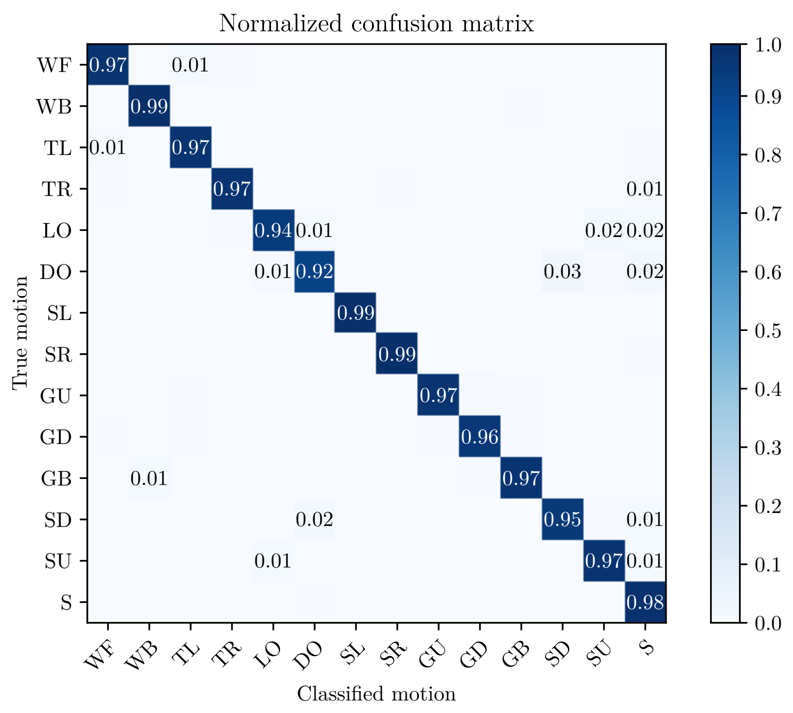


Figure 6.3.: Confusion matrix of Multi-Subject evaluation of the LSTM model with 100 ms window size. Numbers smaller than 0.01 are hidden. Acronyms for motions are defined in Section 4.3.2. Confusions are off the diagonal, SU and LO as well as SD and DO are the most confused classes due to their similarity.

6.2.4. Hyperparameter Tuning

In this subsection, the proposed LSTM model is varied to see how alterations of hyperparameters effect classification quality. As in Section 6.1.4, a Leave-One-Subject-Out validation (see Section 2.3.4) is conducted for ID4 and 300 ms windows. Tested models vary in the number of units in the LSTM layer. The number of units are set to 16, 32, 64 and 256. The standard model has 128 units in its LSTM layer. Classification accuracies for these modifications are given in Table 6.10.

For small numbers of units (i. e., 16, 32) cross-validation accuracy drops by at least 1%. LO accuracy is best for the largest network, but configurations with 64 and 128 units perform on a similar level.

It can be concluded, that larger networks perform better than smaller ones. Employing larger networks comes at cost of higher computational cost, see Section 6.4 for further discussion of this trade-off.

Number of units	CV accuracy	LO accuracy
16	0.9561	0.8469
32	0.9840	0.8856
64	0.9956	0.9006
128	0.9984	0.8999
256	0.9991	0.9051

Table 6.10.: Results of hyperparameter tuning (Section 6.2.4). Networks with a greater number of LSTM units achieve better classification accuracy.

6.3. Comparison

This section reviews results of previous sections and a comparison to the work of Beil et al. (2018) and Patzer and Asfour (2019) is given. An overview of average classification quality is given in Table 6.11. The first column lists different evaluation types and window sizes. Then accuracy values of these evaluations are listed for Convolutional Neural Networks, Long Short-Term Memory networks and Hidden Markov Models.

Evaluation Type	CNN	LSTM	HMM
SS 100 ms	0.9680	**0.9728**	0.9535
SS 200 ms	**0.9988**	0.9937	0.9739
SS 300 ms	**0.9997**	0.9970	0.9836
SS 400 ms	**0.9998**	0.9981	0.9895
SS 500 ms	**0.9998**	0.9975	0.9933
MS 100 ms	0.9296	**0.9707**	0.8292
MS 200 ms	0.9936	**0.9942**	0.8916
MS 300 ms	**0.9988**	0.9977	0.9280
MS 400 ms	**0.9996**	0.9989	0.9500
MS 500 ms	**0.9999**	0.9990	0.9646
LOSO 100 ms	0.8699	0.8699	n.a.
LOSO 200 ms	**0.8770**	0.8740	n.a.
LOSO 300 ms	**0.9152**	0.9073	0.8486
LOSO 400 ms	**0.9328**	0.9246	n.a.
LOSO 500 ms	**0.9415**	0.9400	n.a.

Table 6.11.: Comparison of classification techniques. Classification scores of Single-Subject evaluation (SS), Multi-Subject evaluation (MS) and Leave-One-Subject-Out validation (LOSO) are given as average accuracy. Hidden Markov Models (HMMs) were evaluated by Beil et al. (2018) and Patzer and Asfour (2019). Detailed evaluation results of Convolutional Neural Networks (CNN) and Long Short-Term Memory models (LSTM) are given in Section 6.1 and Section 6.2.

As stated in Section 6.1.3, motion classification accuracy is significantly better in Leave-One-Subject-Out validation when using all features as data for classification. Beil et al. (2018) did not consider quaternions for their classifiers. Table A.6 shows that LO accuracy drops to 0.8465. For classification performance with various feature combinations see Appendix A.2. For better comparability, Single- and Multi-Subject evaluation is run with the same data representation as in the previous work.

The proposed deep learning architectures of Chapter 5 perform consistently better than Hidden Markov Models. While for Single-Subject evaluation (SS) differences in motion classification accuracy are small, major improvements could be achieved for Multi-Subject evaluation (MS). In MS for small window sizes (100 ms), the LSTM model achieves best motion classification accuracy with 3% incorrectly classified windows, while CNNs and HMMs have an error of 7% and 17%.

In Leave-One-Subject-Out validation (LOSO), the proposed network architectures outperform the approach of classifying windows with HMMs. There, both artificial neural networks classify less than 10% of windows incorrectly, whereas HMMs do 15% wrong (300 ms windows). Furthermore, the proposed network models still perform better with window sizes of 100 ms than the HMM with 300 ms windows. For larger window sizes, classification accuracy is further improved, indicating a good generalization performance. This shows, that the models do not over-fit much to the subjects in the training set.

6.4. Computational Requirements

To classify motion online, a processing system at the exoskeleton is necessary. This constraint limits size and power of such systems. While deep learning models are usually trained on *very* powerful systems, many applications only provide a fraction of these resources to execute such a model for the user (Howard et al., 2017; Sandler et al., 2018). A similar challenge was approached by Hundhausen et al. (2019). In their proposed system, a Convolutional Neural Network is deployed on a micro-controller to classify images.

In this section, computational cost for classifying with the CNN model (Section 5.1) is presented. A complexity analysis for the LSTM model is not conducted in this thesis. To calculate the output of a neural network, an input signal is propagated forwards through the network (see Section 2.1.1). This is carried out by many Multiply-Accumulate (MAC) operations. Different types of layers process input signals differently and require diverse computational resources. A convolutional layer has the computational cost of (Howard et al., 2017):

$$K_1 \cdot K_2 \cdot C_i \cdot C_o \cdot F_1 \cdot F_2 \tag{6.1}$$

where the $K_1 \times K_2$ is the kernel size, C_i and C_o are the number of input and output channels, and $F_1 \times F_2$ is the size of the output feature map. Processing a dense layer can be realized as a matrix vector product with the layer's weights arranged in a matrix and the input as a vector. Thus, the computational cost of this is the product of count of input neurons and output neurons:

$$N_i \cdot N_o \tag{6.2}$$

In this analysis, other requirements like applying the activation function are neglected. Furthermore, the computational cost of other layers employed is relatively small in comparison to convolutional or dense layers. Therefore, their cost is not calculated, as runtime will only be affected marginally.

Table 6.12 gives an overview of computational complexity of the Convolutional Neural Network presented in Section 5.1 for classifying 300 ms windows (29 time steps). The convolutional layers demand most of necessary operations. The second to the last layer (dense) also requires a lot of MAC operations. Classifying one window costs over 25 million MAC operations.

Layer Type	Output Shape	Parameters	Operations
Convolution	$27 \times 37 \times 66$	660	593,406
MaxPooling2D	$27 \times 18 \times 66$	0	
Dropout	$27 \times 18 \times 66$	0	
Convolution	$25 \times 16 \times 64$	38,080	15,206,400
MaxPooling2	$25 \times 8 \times 64$	0	
Dropout	$25 \times 8 \times 64$	0	
Convolution	$23 \times 6 \times 62$	35,774	4,928,256
Dropout	$23 \times 6 \times 62$	0	
Flatten	8556	0	
Dense	512	4,381,184	4,380,672
Dropout	512	0	
Dense	14	7,182	7,168
Total		4,462,880	25,115,902

Table 6.12.: CNN model (see Section 5.1) size and computational cost. Number of parameters calculated with *TensorFlow* and operations calculated with formulas given in Section 6.4.

The neural network of Hundhausen et al. (2019) takes about 3.2 million MAC operations. This process has a runtime of493 ms with an on-device micro-processor (ARM Cortex-H7). Processing signals in a neural network can be further optimized, so the author's optimized version evaluates in 115ms. The proposed model of this thesis does not satisfy online classification requirements as a linear increase with the number of necessary MACs is expected, resulting in an optimized runtime of about 900 ms. Thus, either more powerful hardware or computationally simpler models are necessary.

The model was also run on a desktop CPU (Intel i7-4790K). There, classifying over 12,000 motion windows has a runtime of 4.47s equivalent to a runtime of less than1 ms per window. If installing a powerful system at the exoskeleton is not possible, computational cost of classifying with the neural network can be reduced. Simpler, smaller models (e. g., less convolutions, less feature maps, less neurons) require less

MAC operations as Equation (6.1) and Equation (6.2) easily show. Performance of smaller models is also evaluated in Section 6.1.4. In that evaluation, smaller models performed on a similar level. These models only differ in the size of the dense layer.

In addition, a models complexity depends on the window size, i. e., a model built for classifying 100 ms windows requires less MACs, as the input is smaller and less steps are needed to perform a convolution. Decreasing the window size also decreases the time to fill a window but comes at cost of worse classification accuracy (see Section 6.1.2).

For comparison a small model was created. It consists of a convolutional layer creating 16 feature maps, followed a max pooling layer. Their output is processed by a two-layer dense network of 32 and 14 neurons. Computational requirements are reduced a lot, but motion classification accuracy also drops, i. e., when classifying 300 ms windows left out accuracy for ID4 is 0.897 with the standard model. This drops to 0.789 with the small model. Moreover, cross-validation accuracy decreases from 0.9999 to 0.92. Having computationally simpler models may come at cost of less accurate classification. Finding a lightweight model for accurate online classification will be a task for future work.

A complexity analysis for the LSTM model is not conducted. In preliminary tests, however, a runtime similar to the CNN's was measured on the desktop PC. In general, inference with LSTMs is expensive. Approaches to reduce computational requirements of LSTMs can be found in the literature (Campos et al., 2017; Dennis et al., 2018).

6.5. Training Runtime

Training a neural network is a critical part of deep learning. A model is fit to the task in this process. The weights of the model are adjusted with backpropagation to minimize a loss function calculated based on training data (see Equation (2.1)). This procedure is computationally expensive. Since 2006, implementations of backpropagation for graphics cards are available. This enables significantly shorter runtime (Schmidhuber, 2015).

The proposed models of Chapter 5 are implemented with *TensorFlow* (Abadi et al., 2015) and are trained on a NVIDIA Titan X graphics card. Training a CNN model in Multi-Subject evaluation (see Section 2.3.3) takes about 18 min. For the LSTM model the algorithm has a runtime of about 7 min. Both models are trained with 300 ms second windows and two thirds of the data set serve as training data. As the LSTM model has much fewer parameters, the difference in runtime can be explained as fewer weights have to be fitted to the data. In comparison, Daab (2019) trained Hidden Markov Models with four fifth of the data set in about an hour on a powerful CPU (Intel i7-8700K). Thus, training the proposed artificial neural networks is substantially faster than the approach with HMMs.

Chapter 7.

Conclusion and Future Work

Within this thesis, two deep learning architectures were applied to the task of motion classification in the context of exoskeletons. Section 7.1 concludes results of this thesis. An outlook on future work is given in Section 7.2.

7.1. Conclusion

In this thesis, motion classification in the context of exoskeletons was processed by artificial neural networks.

The data set was recorded with a passive uni-lateral lower limb exoskeleton (Beil et al., 2018). It contains 14 common motion types performed by ten subjects. Sensors are attached to the exoskeleton. Three IMUs at upper leg, lower leg and foot measure orientation and linear acceleration. 7 3D-force sensors measure interaction forces between the exoskeleton and the wearer's muscles.

To classify motion, recorded data is concatenated to sliding windows. These windows are then fed into a classifier. In previous work with the data set, window sizes range from 100 ms to 500 ms (Beil et al., 2018). In this thesis, these window sizes are also used for classification.

Two artificial neural network architectures – CNN, which is adapted from Um et al. (2017), and LSTM – are proposed and their classification quality is measured with different evaluations.

Three different evaluations for different usage scenarios were conducted. First, in Single-Subject evaluation, a classifier is trained with data of one subject and performance is tested with other data of this subject. The real-world application can be an exoskeleton that is worn by only one person resulting in the opportunity to optimize the system for this person. Second, in Multi-Subject evaluation, data of multiple subjects serves as training data. Remaining data of these subjects is used for testing the classifier. This approach may be used when an exoskeleton is used by multiple known people. Third, Leave-One-Subject-Out validation shows generalization performance of a classifier. Here, a classifier is trained with data of multiple subjects. It is then tested with data of a subject not contained in the training set. Here, inter-subject motion variance is a big challenge.

For window sizes greater or equal to 200 ms Single-Subject and Multi-Subject evaluation accuracy is over 0.99. LSTM performs better than CNN when classifying 100 ms windows with a classification accuracy of 0.97 compared to 0.93. Compared to the HMM approach of Beil et al. (2018), motion classification accuracy was improved for all evaluations and all window sizes, e. g., in Multi-Subject evaluation for 100 ms windows, motion classification accuracy improved from 0.83 to 0.97. See Table 6.11 for a compact comparison.

7.2. Future Work

Reducing computational requirements of classification is challenging, as a classifier's architecture dictates its inference cost but also classification quality depends on the model layout. Approaches to reduce computational effort are, e. g., classifying smaller windows, employing smaller dense layers, creating less feature maps, employing smaller dense layers. While the former approaches were already applied within this thesis, further modification of the network model is still to be analysed.

In real-world applications, classification of motion transitions (i. e., changing from one motion to another) is critical. Sliding windows of a transition contain data points of the old motion and the new motion, but only the new motion type is necessary. This challenge is yet to be faced and motion data of transitions will be necessary.

Approaches to segment human motion can be found in the literature (see Lin et al., 2016, for an overview). Each segment represents a specific part of motion (e. g., gait cycle phase, motion primitive). With segmented data, specific events (e. g., heel strike) could be recognizable and segment lengths could be calculated, resulting in additional information, that might be useful for classification.

Single-Subject evaluation assessed best performance for the presented artificial neural network models. Thus, if viable, a profile system at the exoskeleton could achieve best performance for every user. Leave-One-Subject-Out validation revealed challenges in inter-subject generalization, as motion classification performance for unseen subjects drops. Novak and Riener (2015) recommend research on adaptive classifiers. Adaptive classifiers are pre-trained models used for motion classification of unseen subjects. The model is then optimized for the new subject during usage. This approach enables new users to use the exoskeleton without previously recording motion data, but also offers subject-specific optimization.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `tensorflow.org`. Accessed 2020-02-24. 23, 37

C. C. Aggarwal. *Neural networks and deep learning*. Springer, 2018. 7, 11

ANT Neuro. Ultra-mobile EEG & EMG recording platform. `https://www.ant-neuro.com/products/eego_sports`. Accessed 2020-02-24. 14

R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *International conference on machine learning*, pages 199–207, 2013. 7

J. Beil. *Kinematisch kompatible Gelenkmechanismen für Exoskelette der unteren Extremitäten*. PhD thesis, Karlsruher Institut für Technologie, 2019. 1, 14, 15

J. Beil, I. Ehrenberger, C. Scherer, C. Mandery, and T. Asfour. Human motion classification based on multi-modal sensor data for lower limb exoskeletons. In *International Conference on Intelligent Robots and Systems: Proceedings*, pages 5431–5436. IEEE, 2018. v, vii, 1, 15, 16, 17, 19, 20, 21, 25, 27, 34, 35, 39, 40

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. 7, 29

Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12 (8):1889–1900, 2000. 27

A. Biedenkapp, M. Lindauer, K. Eggensperger, F. Hutter, C. Fawcett, and H. Hoos. Efficient parameter importance analysis via ablation with surrogates. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 27

V. Campos, B. Jou, X. Gir'o i Nieto, J. Torres, and S. Chang. Skip RNN: learning to skip state updates in recurrent neural networks. *CoRR*, abs/1708.06834, 2017. 37

T. Daab. Systematische Exploration des Merkmalraumes für die Bewegungsklassifikation bei Exoskeletten der unteren Extremität. Bachelor's thesis, Karlsruher Institut für Technologie, 2019. 15, 21, 25, 37, 48

D. Deb, S. Wiper, A. Russo, S. Gong, Y. Shi, C. Tymoszek, and A. K. Jain. Face recognition: Primates in the wild. *CoRR*, abs/1804.08790, 2018. 5

D. K. Dennis, C. Pabbaraju, H. V. Simhadri, and P. Jain. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10976–10987. Curran Associates Inc., 2018. 37

Digital Arts. Vicon cara: the future of facial motion capture. `https://www.digitalartsonline.co.uk/news/motion-graphics/vicon-launches-cara-future-of-facial-motion-capture/`. Accessed 2020-02-24. 14

C. Gong, D. Xu, Z. Zhou, N. Vitiello, and Q. Wang. Real-time on-board recognition of locomotion modes for an active pelvis orthosis. In *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 346–350, 2018. 15, 16, 17

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`, Accessed 2020-02-24. 5, 6

A. Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012. 6, 7

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015. 6

J. A. Hertz. *Introduction to the theory of neural computation.* CRC Press, 2018. 4

S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1), 1991. 6

S. Hochreiter and J. Schmidhuber. Long short term memory. Technical Report FKI-207-95, Technische Universität München, 1995. 6

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–1780, 1997. 6

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 35

H. Huang, T. A. Kuiken, and R. D. Lipschutz. A strategy for identifying locomotion modes using surface electromyography. *IEEE Transactions on Biomedical Engineering*, 56(1):65–73, 2009. 14

F. Hundhausen, D. Megerle, and T. Asfour. Resource-aware object classification and segmentation for semi-autonomous grasping with prosthetic hands. In *International Conference on Humanoid Robots (Humanoids)*. IEEE/RAS, 2019. 35, 36

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 25

P. Husar. *Elektrische Biosignale in der Medizintechnik*. Springer eBooks. Springer Vieweg, Berlin, 2 edition, 2020. 3

Hyundai Motor. Hyundai showcases advanced wearable robots at 2017 geneva motor show. `https://www.hyundai.news/eu/brand/hyundai-motor-leads-personal-mobility-revolution-with-advanced-robots/`. Accessed 2020-02-18. 2

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations*, 2015. 7, 24

F. Kratzert, D. Klotz, J. Brandstetter, P. Hoedt, G. Nearing, and S. Hochreiter. Using LSTMs for climate change assessment studies on droughts and floods. *CoRR*, abs/1911.03941, 2019. 4

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 4, 5, 23

N. Kronenwett, J. Ruppelt, and G. Trommer. Motion monitoring based on a finite state machine for precise indoor localization. *Gyroscopy and Navigation*, 8:190–199, 2017. 14, 15, 16

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 5

A. LeNail. NN-SVG: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019. 4, 5, 23

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. 4

C. Li, P. R. Kulkarni, and B. Prabhakaran. Segmentation and recognition of motion capture data stream by classification. *Multimedia tools and applications*, 35(1): 55–70, 2007. 13

J. F. Lin, M. Karg, and D. Kulić. Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human-Machine Systems*, 46(3):325–339, 2016. 40

Y. Liu, E. Racah, Prabhat, J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. F. Wehner, and W. D. Collins. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *CoRR*, abs/1605.01156, 2016. 5

F. Lv and R. Nevatia. Recognition and segmentation of 3-D human action using HMM and multi-class adaboost. In *European conference on computer vision*, pages 359–372. Springer, 2006. 13

C. Mandery, M. Plappert, J. Borràs, and T. Asfour. Dimensionality reduction for whole-body human motion recognition. In *19th International Conference on Information Fusion (FUSION)*, pages 355–362, 2016. 13

T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006. 13

D. Muoio. Hyundai created its own 'iron man' exoskeleton suit. `https://www.businessinsider.com/hyundai-creates-iron-man-exoskeleton-photos-2016-5`. Accessed 2020-02-18. 2

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010. 4

D. Novak and R. Riener. A survey of sensor fusion methods in wearable robotics. *Robotics and Autonomous Systems*, 73:155–170, 2015. Wearable Robotics. 15, 40

I. Patzer and T. Asfour. Minimal sensor setup in lower limb exoskeletons for motion classification based on multi-modal sensor data. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 8158–8164. IEEE/RSJ, 2019. 15, 25, 34

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*, 12:2825–2830, 2011. 7, 25

C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 11–22. SIAM, 2004. 13

Roam Robotics. A robotic exoskeleton designed to enhance your skiing experience. `https://www.roamrobotics.com/ski`. Accessed 2020-02-19. 2

F. Rosenblatt. The perceptron, a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957. 4

C. Sammut and G. I. Webb, editors. *Encyclopedia of Machine Learning and Data Mining*. Springer US, Boston, MA, 2017. 10, 11

M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. 35

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. 4, 7, 37

J. Seetha and S. Raja. Brain tumor classification using convolutional neural networks. *Biomedical and Pharmacology Journal*, 11:1457–1461, 2018. 4, 9

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 24, 29

J. Taborri, E. Scalona, S. Rossi, E. Palermo, F. Patanè, and P. Cappa. Real-time gait detection based on hidden markov model: is it possible to avoid training procedure? In *International symposium on medical measurements and applications (MeMeA) proceedings*, pages 141–145. IEEE, 2015. 15

F. Thommes. DeepL stellt Googles Übersetzer in den Schatten, 2017. Accessed 2020-01-16. 4, 7

O. Tunçel, K. Altun, and B. Barshan. Classifying human leg motions with uniaxial piezoelectric gyroscopes. *Sensors*, 9:8508–8546, 2009. 15, 16, 17

T. T. Um, V. Babakeshizadeh, and D. Kulic. Exercise motion classification from large-scale wearable sensor data using convolutional neural networks. In *International Conference on Intelligent Robots and Systems*, pages 2385–2390. IEEE/RSJ, 2017. 14, 15, 16, 17, 23, 24, 39

A. Villa-Parra, D. Delisle-Rodríguez, A. López-Delis, T. Bastos-Filho, R. Sagaró, and A. Frizera-Neto. Towards a robotic knee exoskeleton control based on human motion intention through EEG and sEMG signals. *Procedia Manufacturing*, 3:1379–1386, 2015. 14, 15, 16

M. Wang, X. Wu, D. Liu, C. Wang, T. Zhang, and P. Wang. A human motion prediction algorithm for non-binding lower extremity exoskeleton. In *International Conference on Information and Automation*, pages 369–374. IEEE, 2015. 14, 15, 16, 17

P. D. Wasserman and T. Schwartz. Neural networks. II. What are they and why is everybody so interested in them now? *IEEE Expert*, 3:10–15, 1988. 3

Y. H. Yin, Y. J. Fan, and L. D. Xu. EMG and EPP-integrated human–machine interface between the paralyzed and rehabilitation exoskeleton. *IEEE Transactions on Information Technology in Biomedicine*, 16(4):542–549, 2012. 14

Appendix A.

Evaluation Results

This appendix contains detailed results from evaluations. This may serve as a basis for further research. First, the mapping of IDs for simplification is given. Then, detailed results are presented in the following sections.

A.1. ID Mapping

Table A.1 shows the mapping of simplifying IDs of users in the data set (see Section 4.3).

Actual ID	Simplified ID
ID674	ID1
ID917	ID2
ID1717	ID3
ID1718	ID4
ID1719	ID5
ID1720	ID6
ID1722	ID7
ID1723	ID8
ID1724	ID9
ID1725	ID10

Table A.1.: IDs of subjects are simplified.

A.2. Feature Reduction

Tables A.2 to A.7 show detailed evaluation results with Leave-One-Subject-Out validation (explained in Section 2.3.4) for various feature combinations. All evaluations

Subject	CV accuracy	LO accuracy
ID1	0.9987	0.7907
ID2	0.9991	0.7698
ID3	0.9986	0.6670
ID4	0.9987	0.6155
ID5	0.9989	0.6998
ID6	0.9983	0.7442
ID7	0.9989	0.7452
ID8	0.9993	0.6195
ID9	0.9985	0.6728
ID10	0.9990	0.4770
average	0.9988	0.6802
std-dev	0.0003	0.0885

Table A.2.: Force Sensors

Subject	CV accuracy	LO accuracy
ID1	0.9809	0.9407
ID2	0.9800	0.9523
ID3	0.9803	0.8992
ID4	0.9814	0.8716
ID5	0.9808	0.9053
ID6	0.9809	0.9124
ID7	0.9799	0.9248
ID8	0.9818	0.9157
ID9	0.9814	0.9035
ID10	0.9831	0.8643
average	0.9811	0.9090
std-dev	0.0009	0.0259

Table A.3.: Euler angles and linear acceleration.

were run with 300 ms windows. The CNN model as defined in Section 5.1 serves as classifier. A systematic feature space exploration for Hidden Markov Models was conducted by Daab (2019).

Subject	CV accuracy	LO accuracy
ID1	0.9961	0.9290
ID2	0.9965	0.9474
ID3	0.9964	0.9205
ID4	0.9967	0.9000
ID5	0.9964	0.9286
ID6	0.9963	0.9241
ID7	0.9961	0.8057
ID8	0.9963	0.8935
ID9	0.9960	0.8978
ID10	0.9979	0.7823
average	0.9965	0.8929
std-dev	0.0005	0.0522

Table A.4.: Quaternions

Subject	CV accuracy	LO accuracy
ID1	0.9998	0.9563
ID2	0.9997	0.9565
ID3	0.9998	0.8981
ID4	0.9998	0.9009
ID5	0.9998	0.9282
ID6	0.9999	0.9457
ID7	0.9998	0.8441
ID8	0.9998	0.9250
ID9	0.9998	0.9113
ID10	0.9998	0.8129
average	0.9998	0.9079
std-dev	0.0001	0.0448

Table A.5.: Force, Quaternions and Linear Acceleration

Subject	CV accuracy	LO accuracy
ID1	0.9994	0.9044
ID2	0.9994	0.9137
ID3	0.9995	0.8323
ID4	0.9995	0.8144
ID5	0.9994	0.8690
ID6	0.9990	0.8567
ID7	0.9992	0.8879
ID8	0.9995	0.8424
ID9	0.9994	0.8412
ID10	0.9995	0.7029
average	0.9994	0.8465
std-dev	0.0001	0.0567

Table A.6.: Force sensors, Euler angles, Linear Acceleration

Subject	CV accuracy	LO accuracy
ID1	0.9987	0.9348
ID2	0.9986	0.9478
ID3	0.9987	0.8773
ID4	0.9988	0.8846
ID5	0.9988	0.9241
ID6	0.9987	0.9381
ID7	0.9987	0.8530
ID8	0.9988	0.8944
ID9	0.9987	0.9085
ID10	0.9989	0.8003
average	0.9987	0.8963
std-dev	0.0001	0.0429

Table A.7.: All features used (i. e., Force sensors, Euler angles, Quaternions, Linear Acceleration)

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.9833	0.9833	0.9834	0.9833
ID2	0.9882	0.9882	0.9882	0.9881
ID3	0.9629	0.9629	0.9633	0.9629
ID4	0.9307	0.9307	0.9316	0.9308
ID5	0.9615	0.9615	0.9621	0.9616
ID6	0.9716	0.9716	0.9720	0.9716
ID7	0.9618	0.9618	0.9621	0.9618
ID8	0.9769	0.9769	0.9770	0.9769
ID9	0.9836	0.9836	0.9838	0.9837
ID10	0.9594	0.9594	0.9617	0.9599
average	0.9680	0.9680	0.9685	0.9681

Table A.8.: Single-Subject evaluation of the CNN model with 100 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.9999	0.9999	0.9999	0.9999
ID2	0.9995	0.9995	0.9995	0.9995
ID3	0.9990	0.9990	0.9990	0.9990
ID4	0.9972	0.9972	0.9972	0.9972
ID5	0.9979	0.9979	0.9980	0.9979
ID6	0.9985	0.9985	0.9986	0.9985
ID7	0.9982	0.9982	0.9982	0.9982
ID8	0.9990	0.9990	0.9990	0.9990
ID9	0.9998	0.9998	0.9998	0.9998
ID10	0.9984	0.9984	0.9984	0.9984
average	0.9988	0.9988	0.9988	0.9988

Table A.9.: Single-Subject evaluation of the CNN model with 200 ms window size.

A.3. Single-Subject Evaluation

A.3.1. Convolutional Neural Network

This subsection provides detailed Single-Subject evaluation results with different window sizes. The CNN model as defined in Section 5.1 serves as classifier. Tables A.8 to A.12 show scores of motion classification for window sizes from 100 ms to 500 ms.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99970	0.99970	0.99970	0.99970
ID2	0.99978	0.99978	0.99978	0.99978
ID3	0.99965	0.99965	0.99965	0.99965
ID4	0.99973	0.99973	0.99973	0.99973
ID5	0.99937	0.99937	0.99937	0.99937
ID6	0.99966	0.99966	0.99966	0.99966
ID7	0.99946	0.99946	0.99946	0.99946
ID8	0.99985	0.99985	0.99985	0.99985
ID9	0.99996	0.99996	0.99996	0.99996
ID10	0.99979	0.99979	0.99979	0.99979
average	0.99969	0.99969	0.99969	0.99969

Table A.10.: Single-Subject evaluation of the CNN model with 300 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99989	0.99989	0.99989	0.99989
ID2	0.99983	0.99983	0.99983	0.99983
ID3	0.99970	0.99970	0.99970	0.99970
ID4	0.99939	0.99939	0.99939	0.99939
ID5	0.99984	0.99984	0.99984	0.99984
ID6	0.99980	0.99980	0.99980	0.99980
ID7	0.99981	0.99981	0.99981	0.99981
ID8	0.99987	0.99987	0.99987	0.99987
ID9	0.99989	0.99989	0.99989	0.99989
ID10	0.99980	0.99980	0.99980	0.99980
average	0.99978	0.99978	0.99978	0.99978

Table A.11.: Single-Subject evaluation of the CNN model with 400 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99988	0.99988	0.99988	0.99988
ID2	0.99994	0.99994	0.99994	0.99994
ID3	0.99977	0.99977	0.99977	0.99977
ID4	0.99976	0.99976	0.99976	0.99976
ID5	0.99980	0.99980	0.99980	0.99980
ID6	0.99968	0.99968	0.99968	0.99968
ID7	0.99992	0.99992	0.99992	0.99992
ID8	0.99984	0.99984	0.99984	0.99984
ID9	0.99983	0.99983	0.99983	0.99983
ID10	0.99971	0.99971	0.99971	0.99971
average	0.99981	0.99981	0.99981	0.99981

Table A.12.: Single-Subject evaluation of the CNN model with 500 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.98218	0.98218	0.98247	0.98220
ID2	0.99214	0.99214	0.99217	0.99214
ID3	0.97726	0.97726	0.97788	0.97738
ID4	0.94618	0.94618	0.94752	0.94639
ID5	0.97042	0.97042	0.97150	0.97057
ID6	0.97010	0.97010	0.97095	0.97024
ID7	0.95987	0.95987	0.96099	0.96009
ID8	0.97948	0.97948	0.97974	0.97950
ID9	0.98300	0.98300	0.98314	0.98303
ID10	0.96762	0.96762	0.96992	0.96825
average	0.97283	0.97283	0.97363	0.97298

Table A.13.: Single-Subject evaluation of the LSTM model with 100 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99786	0.99786	0.99787	0.99786
ID2	0.99896	0.99896	0.99896	0.99896
ID3	0.99472	0.99472	0.99482	0.99473
ID4	0.98841	0.98841	0.98863	0.98845
ID5	0.99165	0.99165	0.99187	0.99168
ID6	0.99324	0.99324	0.99331	0.99326
ID7	0.99127	0.99127	0.99145	0.99130
ID8	0.99645	0.99645	0.99646	0.99645
ID9	0.99537	0.99537	0.99540	0.99538
ID10	0.98948	0.98948	0.99021	0.98963
average	0.99374	0.99374	0.99390	0.99377

Table A.14.: Single-Subject evaluation of the LSTM model with 200 ms window size.

A.3.2. Long Short-Term Memory

This subsection provides detailed Single-Subject evaluation results of the LSTM architecture defined in Section 5.2. Tables A.13 to A.17 show scores of motion classification for window sizes from 100 ms to 500 ms.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99696	0.99696	0.99699	0.99696
ID2	0.99930	0.99930	0.99930	0.99930
ID3	0.99576	0.99576	0.99578	0.99576
ID4	0.99749	0.99749	0.99749	0.99749
ID5	0.99738	0.99738	0.99740	0.99738
ID6	0.99629	0.99629	0.99632	0.99629
ID7	0.99440	0.99440	0.99453	0.99441
ID8	0.99830	0.99830	0.99831	0.99830
ID9	0.99802	0.99802	0.99802	0.99802
ID10	0.99591	0.99591	0.99595	0.99592
average	0.99698	0.99698	0.99701	0.99698

Table A.15.: Single-Subject evaluation of the LSTM model with 300 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99868	0.99868	0.99868	0.99868
ID2	0.99888	0.99888	0.99888	0.99888
ID3	0.99834	0.99834	0.99835	0.99834
ID4	0.99536	0.99536	0.99547	0.99537
ID5	0.99929	0.99929	0.99929	0.99929
ID6	0.99667	0.99667	0.99668	0.99667
ID7	0.99872	0.99872	0.99872	0.99872
ID8	0.99798	0.99798	0.99802	0.99798
ID9	0.99959	0.99959	0.99959	0.99959
ID10	0.99719	0.99719	0.99722	0.99719
average	0.99807	0.99807	0.99809	0.99807

Table A.16.: Single-Subject evaluation of the LSTM model with 400 ms window size.

Subject	Accuracy	Recall	Precision	F_1-Score
ID1	0.99927	0.99927	0.99927	0.99927
ID2	0.99904	0.99904	0.99904	0.99904
ID3	0.99629	0.99629	0.99638	0.99630
ID4	0.99387	0.99387	0.99448	0.99380
ID5	0.99875	0.99875	0.99876	0.99875
ID6	0.99873	0.99873	0.99874	0.99874
ID7	0.99306	0.99306	0.99316	0.99303
ID8	0.99869	0.99869	0.99869	0.99869
ID9	0.99855	0.99855	0.99858	0.99855
ID10	0.99836	0.99836	0.99837	0.99836
average	0.99746	0.99746	0.99755	0.99745

Table A.17.: Single-Subject evaluation of the LSTM model with 500 ms window size.

Subject	CV accuracy	LO accuracy
ID1	0.99005	0.91470
ID2	0.99101	0.94147
ID3	0.99192	0.84207
ID4	0.99305	0.84695
ID5	0.99213	0.88144
ID6	0.99130	0.90114
ID7	0.99199	0.81628
ID8	0.98966	0.88822
ID9	0.99102	0.88391
ID10	0.99242	0.78305
average	0.99145	0.86992
std-dev	0.00100	0.04527

Table A.18.: Leave-One-Subject-Out validation results for the CNN model with 100 ms window size.

A.4. Leave-One-Subject-Out Validation

This section provides additional evaluation results of Leave-One-Subject-Out validation for the proposed network architectures.

A.4.1. Convolutional Neural Network

Leave-One-Subject-Out validation results are presented in Tables A.18 to A.22. Each table corresponds to a window size in the range of 100 ms to 500 ms.

Subject	CV accuracy	LO accuracy
ID3	0.99951	0.87435
ID4	0.99951	0.86898
ID5	0.99934	0.90035
ID6	0.99947	0.91835
ID7	0.99932	0.84644
ID8	0.99949	0.90315
ID9	0.99912	0.90144
ID10	0.99953	0.80295
average	0.99941	0.87700
std-dev	0.00013	0.03540

Table A.19.: Leave-One-Subject-Out validation results for the CNN model with 200 ms window size. In this evaluation calculations for ID1 and ID2 failed and are therefore missing.

Subject	CV accuracy	LO accuracy
ID1	0.99981	0.95715
ID2	0.99979	0.96420
ID3	0.99980	0.90282
ID4	0.99987	0.89712
ID5	0.99980	0.93584
ID6	0.99984	0.94466
ID7	0.99980	0.87592
ID8	0.99956	0.93056
ID9	0.99970	0.92094
ID10	0.99983	0.82321
average	0.99978	0.91524
std-dev	0.00009	0.04026

Table A.20.: Leave-One-Subject-Out validation results for the CNN model with 300 ms window size.

Subject	CV accuracy	LO accuracy
ID1	0.99988	0.97370
ID2	0.99993	0.97020
ID3	0.99990	0.90991
ID4	0.99993	0.92470
ID5	0.99991	0.95256
ID6	0.99993	0.95855
ID7	0.99994	0.88989
ID8	0.99992	0.94846
ID9	0.99991	0.93561
ID10	0.99982	0.86455
average	0.99991	0.93281
std-dev	0.00003	0.03386

Table A.21.: Leave-One-Subject-Out validation results for the CNN model with 400 ms window size.

Subject	CV accuracy	LO accuracy
ID1	0.99995	0.98017
ID2	0.99992	0.97855
ID3	0.99994	0.91694
ID4	0.99995	0.93922
ID5	0.99993	0.95573
ID6	0.99993	0.96490
ID7	0.99992	0.90804
ID8	0.99997	0.95929
ID9	0.99991	0.93650
ID10	0.99995	0.87653
average	0.99994	0.94159
std-dev	0.00002	0.03148

Table A.22.: Leave-One-Subject-Out validation results for the CNN model with 500 ms window size.

Subject	CV accuracy	LO accuracy
ID1	0.98429	0.92084
ID2	0.98322	0.92730
ID3	0.98353	0.85917
ID4	0.98635	0.84925
ID5	0.98491	0.88033
ID6	0.98447	0.89514
ID7	0.98555	0.82818
ID8	0.98475	0.87231
ID9	0.98492	0.87447
ID10	0.98623	0.79187
average	0.98482	0.86989
std-dev	0.00098	0.03877

Table A.23.: Leave-One-Subject-Out validation results for 100 ms windows with the LSTM model.

A.4.2. Long Short-Term Memory

Tables A.23 to A.27 correspond to Leave-One-Subject-Out validations for window sizes ranging from 100 ms to 500 ms. The LSTM architecture serves as classifier.

Subject	CV accuracy	LO accuracy
ID1	0.99574	0.82282
ID2	0.99554	0.91778
ID3	0.99584	0.87869
ID4	0.99673	0.87578
ID5	0.99683	0.89109
ID6	0.99587	0.91608
ID7	0.99622	0.87405
ID8	0.99571	0.86403
ID9	0.99545	0.88682
ID10	0.99625	0.81257
average	0.99602	0.87397
std-dev	0.00045	0.03265

Table A.24.: Leave-One-Subject-Out validation results for 200 ms windows with the LSTM model.

Subject	CV accuracy	LO accuracy
ID1	0.99808	0.95478
ID2	0.99778	0.94897
ID3	0.99860	0.89128
ID4	0.99842	0.89988
ID5	0.99871	0.93389
ID6	0.99868	0.93717
ID7	0.99873	0.89774
ID8	0.99801	0.89239
ID9	0.99784	0.89008
ID10	0.99883	0.82675
average	0.99837	0.90729
std-dev	0.00038	0.03606

Table A.25.: Leave-One-Subject-Out validation results for 300 ms windows with the LSTM model.

Subject	CV accuracy	LO accuracy
ID1	0.99938	0.96734
ID2	0.99864	0.96274
ID3	0.99935	0.90137
ID4	0.99943	0.91651
ID5	0.99899	0.93822
ID6	0.99926	0.95927
ID7	0.99939	0.91587
ID8	0.99931	0.91763
ID9	0.99848	0.90176
ID10	0.99957	0.86517
average	0.99918	0.92459
std-dev	0.00034	0.03074

Table A.26.: Leave-One-Subject-Out validation results for 400 ms windows with the LSTM model.

Subject	CV accuracy	LO accuracy
ID1	0.99911	0.97378
ID2	0.99967	0.97845
ID3	0.99942	0.91229
ID4	0.99971	0.93284
ID5	0.99949	0.96237
ID6	0.99953	0.96865
ID7	0.99966	0.92402
ID8	0.99957	0.93546
ID9	0.99918	0.92407
ID10	0.99972	0.88783
average	0.99951	0.93998
std-dev	0.00020	0.02830

Table A.27.: Leave-One-Subject-Out validation results for 500 ms windows with the LSTM model.