universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Using neural networks for functional gait disorder classification based on ground reaction force measurements

verfasst von / submitted by

Thomas Spendlhofer Bakk. BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2020 / Vienna, 2020

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 826
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Sportwissenschaft
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Arnold Baca
Mitbetreut von / Co-Supervisor:	

Contents

Abstract

Quantification of Ground Reaction Force (GRF) measurements combined with a simple visual inspection is a commonly used approach for conducting gait analysis in clinical practice. Nonetheless, it is not a trivial task, requiring an experienced clinician to draw valid conclusions about the impairment. Several attempts have been made in order to automate that process, but very few focussed solely on data obtained by measurements conducted using force plates. The most exhaustive research in the field so far, tries to map gait impairments to the location (the joint) of the corresponding injury using the five main classes: *Healthy - Hip - Knee - Ankle - Calcaneus*. This thesis tries to expand on that approach by employing different versions of neural networks for the classification task. Even though the proposed methods specialize in recognizing temporal dependencies and relationships between the individual components of GRF-measurements, experiments revealed that the increase in accuracy is minimal when compared to much simpler architectures. The best result achieved scored 73.97% accuracy on a randomly sampled validation-set outperforming the established benchmark on a Support Vector Machine (SVM) by 10%. However, high variability within classes and individuals make GRF-classification a challenging problem and significantly lower accuracies where found for a larger test-set (61.91%), raising doubts about the generalizability and robustness of the implemented classifiers.

1. Introduction

Normal walking is one of the most central human physical activities and can be described as a continuous interchange between mobility and stability (Lakany, 2008). In order to exhibit a natural gait pattern, the human body requires free passive mobility and appropriate muscle action as basic preliminaries. Any restrictions in either the timing or intensity of the muscle activity or limitations in the normal free mobility of a joint will result in an abnormal gait. Persistent abnormalities due to impairments in either physical or neurological functions (such as injuries, diseases, pain, or problems of motor control) are often referred to as pathological gait patterns. Usually, such inadequacies are (at least partially) compensated by the human body, which tries to retain a certain amount of functionality by exerting additional effort. However, as reported by Lakany (2008), these compensation mechanisms might result in joint strain, muscle overuse, lack of muscle growth or soft tissue contraction. Under such circumstances, clinical intervention becomes inevitable and gait analysis is commonly used for the assessment of motion disorders (Baker, 2013).

In order to identify gait impairments, the human walking pattern needs to be recognized and classified according to the clinical status of the analysed motor function (Figueiredo et al., 2018), requiring a definition of "normal/abnormal" gait. A widely adapted procedure in medical science to enable such differential diagnoses is the establishment of a so-called reference range,

described by a number of values that are close enough to a previously defined average, which has been calculated from a large sample of measurements across different persons. However, as Kirtley (2006) reports, those values are difficult to interpret in the case of gait analysis, because human walking patterns are known to have a large biological variability. Thus, normative gait ranges are often restricted to using only ± 1 SD (standard deviation), in contrast to other medical diagnostics where ± 2 or even 3 SD are required to be classified as "abnormal" (e.g. blood tests).

To complicate the task even further, the number of causes which can induce gait abnormalities is large and each one of them might lead to another variation of measurements, making the definition of healthy gait even more complex. Fortunately, while the list of reasons that impair a patient's ability to walk might be endless, the abnormalities imposed on the mechanics of walking can be described by just four functional categories (Perry, 1992):

- *Deformity*: The passive mobility is restricted by tissues, preventing the patient from attaining normal postures and ranges of motion used in walking.
- *Muscle Weakness*: The patient is not able to exhibit sufficient muscle strength to meet the demands of normal gait.
- *Sensory Loss*: Proprioceptive impairments prevent the patient from knowing the position of the hip, knee or foot and the type of contact with the ground.
- *Pain*: Musculoskeletal pain often caused by excessive tissue tension, commonly related to trauma or arthritis. Can lead to deformity and muscle weakness.

In this context, various approaches have been applied to the task of distinguishing between healthy and impaired gait patterns, relying on temporal-spatial parameters (e,g, the gait cycle), gait kinematics (e.g. positions of body segments) or kinetics (e.g. ground reaction forces), contributing to an ever increasing toolchain available to gait laboratories and clinicians (Kirtley, 2006).

Even though the methods available in clinical gait analysis are becoming more numerous and refined, the recordings obtained by such measurements usually consist of a huge amount of data that is characterised by high-dimensionality, temporal dependences, strong variability and non-linear relationships and correlations within itself (Chau, 2001a). Therefore, data interpretation is not an easy task and for valid conclusions to be drawn, the assessment of an experienced clinician is required (Slijepcevic et al., 2018a).

1.1. Motivation

In recent years, the increasing success of machine learning has inspired several attempts to automatize this process (see Chau (2001a,b) for a basic overview of methods and Figueiredo et al.

(2018) for approaches based on machine learning) in order to support clinicians in their diagnosis. According to Figueiredo et al. (2018), an automated classification of gait patterns would provide several advantages by enabling the following:

1. Easy comparison to healthy gait, thus providing a quantitative and non-invasive option to assess locomotive functions.
2. Personalization of gait training by adjusting assistance according to the recognized movement disorder.
3. Planning and coordinating of future treatment according to the subject's needs and gait impairment(s).
4. Measuring and describing the progress of gait treatment by comparing gait rehabilitation at baseline and follow-up sessions.
5. Providing an objective, quick and cost-efficient support for clinical gait analysis.

With the rapid advancement and development of machine learning techniques for signal recognition, where new technologies are introduced every year, it is not surprising that the number of different methods employed and tested in the context of gait analysis is considerably high. Attempts have been made to apply neural networks (Lozano-Ortiz et al., 2010; Zeng et al., 2016; Vieira et al., 2015), support vector machines (SVMs) (Wu et al., 2007; Wu and Wang, 2008; Levinger et al., 2009), nearest neighbour classifiers (Mezghani et al., 2008; Alaqtash et al., 2011) and even various clustering approaches (k-means, hierarchical, etc.) (Ferrarin et al., 2012) to the task, typically reporting moderate to high accuracy when distinguishing between different patient groups or pathologies (Lozano-Ortiz et al., 2010; Wu et al., 2007; Levinger et al., 2009; Alaqtash et al., 2011; Soares et al., 2016). However, along with the diversity in techniques, the use-cases also differ distinctively from each other, trying to distinguish between characteristics such as the affected/non-affected limb in hemiplegic patients (Williams et al., 2015), transfemoral amputation (Soares et al., 2016), lower limb fractures (Muniz and Nadal, 2009) and distinction of subjects with neurological disorders (Zeng et al., 2016; Alaqtash et al., 2011). In spite of the promising results obtained by these researchers, they are often trying to differentiate between two highly dissimilar gait patterns, giving little indication on the performance of the employed method on a different classification task.

The significance for clinical practice of these studies is further limited by the fact that they often include both, kinetic and kinematic measurements of the subject, derived from a 3-dimensional gait analysis. While the knowledge obtained is highly valuable for scientific research and applications, the inclusion of kinematic data is often considered problematic in a clinical setting. In addition to the force platforms used for measuring ground reaction forces (GRFs), the recording of kinematic data requires a 3-dimensional gait analysis system, thus introducing the following drawbacks (Slijepcevic et al., 2018a) to the measuring process:

- The recording process is relatively time consuming when compared to simple GRF measurements.
- High acquisition and maintenance costs for the recording device.
- Requirement of highly specialized/trained staff for conducting the analysis.

Therefore, such devices are hardly used in clinical practice. According to Slijepcevic et al. (2018a) a commonly used approach for gait diagnosis in clinical settings is the combination of a simple visual inspection (or 2D video recording) with the quantification of GRFs. Despite the prevalence of such methods, relatively few attempts have been made to use only GRF data for automated classification of gait patterns (Soares et al., 2016; Goh et al., 2014; Slijepcevic et al., 2017, 2018a,b, 2020) and to the author's best knowledge, there exists no automatic approach that is currently used in clinical praxis. Considering the overall high amount of research in that area, it is slightly surprising that no attempt has been made yet to put those techniques into practical use. However, closer inspection reveals some shortcomings in either the results obtained or the research method applied, preventing them from being deployed in clinical gait analysis. The main arguments can be summarized as follows:

- *Generalization*: Most research has not been verified on large datasets, thus it remains unclear whether the obtained results can be generalized to a broad population. For example the dataset classified by Soares et al. (2016), consists only of data from 20 healthy persons and 12 subjects with transfemoral amputation, which is usually not enough to train a robust and reliable classifier that is suited for complex real-world scenarios.
- *Cost/Time*: The method used is either too time-consuming or too cost-intensive to be used in clinical practice (e.g. utilization of a 3-dimensional gait analysis system).
- *Accuracy*: The classification accuracy is not high or not reliable enough to justify the usage of the method. For example, Slijepcevic et al. (2018b) report an accuracy of 67.8% when trying to identify the injured body region associated with a gait disorder (i.e. one of *normal* - *hip* - *knee* - *calcaneus*), which is quite far from the usual requirements in medical settings (often demanding an accuracy of 90% or higher).
- *Usefulness*: The information provided does not contribute to the diagnosis, i.e. the clinician is able to draw the same conclusion with a few simple questions or observations.

Even if just one of these disadvantages is applicable for any automated tool, it will be sufficient to prevent it from being used for actual diagnosis. Therefore, it is important to take all of those issues into account when trying to develop a method that can be used outside of research laboratories. The research conducted in this thesis attempts to provide a contribution to the field of automatic gait classification, by trying to improve on accuracy achieved by previous research (Slijepcevic et al., 2018b, 2020), in the hope of advancing it towards the long term goal of mak-

ing it suitable for clinical practice. Aiming for easy adaptability into such a setting, the issues raised above are addressed in the following way:

- *Generalization*: Usage of the large *GaitRec* dataset (2,295 patients), provided by Horsak et al. (2020), that is publicly available, facilitating reproducibility and further research. The data is already split into a *training* and *testing* part, benefiting reliability and helping to build a classifier that can be assumed to be applicable to a broader populace reasonably well.
- *Cost/Time*: Measurements are conducted solely by the usage of force plates, which are commonly available in clinical practice.
- *Accuracy*: Maximization of accuracy by implementing different neural network architectures that have been shown to achieve good performance on GRF-measurements (Alharthi and Ozanyan, 2019) and various other time-series data (Fawaz et al., 2019a).
- *Usefulness*: By associating gait abnormalities with body locations in a similar fashion to Slijepcevic et al. (2018a, 2020), the output of the classification provides information that could be used for detecting arthropathies or diseases at an early stage as well as indicating secondary disorders.

Condensing these concepts down to their very core, the ultimate goal of this research is the development of an automated gait analysis tool, purely based on quantification of GRF-measurements. It should be able to identify the associated location of an impairment for any gait disorder (or none, in the case of normal gait) according to the following five classes: *healthy* - *hip* - *knee* - *ankle* - *calcaneus*, providing the most likely class (possibly paired with the probability of correctness) as its output. The classifier needs to be both, accurate and reliable in order to provide high-quality support for clinical diagnosis by indicating the injured/affected body location(s).

1.2. Applications

Due to the very natures of such a classifier, its main area of application would of course be as an assistive tool in clinical gait analysis. However, its usage is not restricted to a clinical setting. In combination with wearable devices for measuring GRFs, it could become a valuable method for planning, controlling and evaluating rehabilitation progress, providing new opportunities for optimization of the whole process. The implications of such an approach are purely speculative, but it is not implausible that an enhanced procedure might lead to more focussed exercising, possibly enabling a faster recovery. Thus, the proposed assistive tool could be beneficial in the following settings:

- *Diagnostics*: The primary of advantage of such a classifier is the non-invasive identification of injured body regions, possible indicating impairments/diseases (such as arthropathies)

early in their development. In addition, it might be able to provide valuable information about secondary injuries which may be easily overlooked during clinical examination by a physician (Slijepcevic et al., 2018a)

- *Rehabilitation*: In a clinical setting the assistive tool could be used for evaluation of the applied treatment by comparing the initial measurements (i.e. after admission) to those of follow-up sessions, enabling personalization of the rehabilitation process by planning and coordinating future treatments accordingly. Wearable devices would further extend the application to individual training-sessions, providing immediate feedback about exercises (e.g. on the quality of execution). Such information could be used to make the rehabilitation progress visible to the patient, possibly increasing their motivation. Accompanied by proper instructions, this method could also be employed for home training outside of rehabilitation centers.

1.3. Research Question

The goal of this thesis is to expand on the research previously conducted by Slijepcevic et al. (2017, 2018a,b, 2020), trying to improve the performance of their classifier by using different neural network architectures such as convolutional neural networks (CNNs) and long short-term memory (LSTM). While LSTMs have been built to model temporal relations and are thus inherently qualified to classify time-series (such as GRF-measurements), CNNs have their roots in the task of image classification, where they have been able to deliver outstanding results Krizhevsky et al. (2017). However, CNNs have recently been introduced with great success to the task of time-series classification (see Zhao et al. (2017) for the basic reasoning behind this achievement) and many different adaptions have been tested since (Fawaz et al. (2019a) provide a good and recent overview).

However, since it remains unclear how well those methods can be adapted to the task of GRF-classification, the main purpose of this research is to implement those architectures and compare their accuracies for the classification task proposed by Slijepcevic et al. (2018a, 2020) on the *GaitRec* dataset provided by Horsak et al. (2020). In other words, the networks should be able to separate the GRF-measurements according to the classification task described earlier, associating gait disorders with the location of the impairment (i.e. one of the five classes: *healthy* - *hip* - *knee* - *ankle* - *calcaneus*). Since the data being used has only been published recently, this work has the additional objective of providing a first baseline for that dataset. Furthermore an attempt is made to provide some guidelines and promising approaches for future research by exploring a few of the implications that data selection and preprocessing might have on the classification result. In summary, this thesis is trying to answer (or at least gain some insights) about the following questions:

- Which neural network architecture is best suited for the task of GRF-classification (i.e. achieves the highest accuracy)?
- Does the input format (i.e. preprocessing and transformations applied to the data) influence the performance?
- Which procedure is best suited for aggregating the information provided by the individual *trials* (i.e. multiple measurements) recorded during a single gait session, in combination with neural networks?
- How much data is actually necessary for an accurate classification (i.e. how many data points are needed, is the combination of data from both legs beneficial)?
- What insights can be gained about the problem itself (e.g. are all classes equally difficult to identify)?
- Can the results of Slijepcevic et al. (2018a), suggesting that an equal number of samples for all classes is more important than a large amount of data, be verified on the implemented approaches?

1.4. Research Approach

In an attempt to answer the questions raised in the previous section, this thesis follows a strictly empirical procedure. First, the existing literature dealing with automated classification of GRF-measurements is analyzed and evaluated, identifying the various methods that have been proposed for the task. Due to the relatively low number of publications employing neural networks to GRF-classification, this is followed by an extensive literature review of such approaches that have been investigated for the larger (and thus less specific) problem of time-series classification. It is revealed that a large amount of neural network architectures have been proposed and evaluated for the more general case, achieving high accuracy on similar tasks. However, no universal solution has been found yet, as the performance of those methods depends on both the data and the classification task. Hence, a selection of networks that have been shown to achieve high accuracy on various time-series (Fawaz et al., 2019a,b) are implemented and evaluated on the *GaitRec* dataset.

In addition to these (complex) architectures, a number of simpler networks, each employing different feature extraction methods, are assessed as well, in order to gain insights about the amount and quality of the information contained within GRF-measurements. Experiments are conducted to first determine the parameters of each architecture, before their overall performance is assessed (and compared to previous approaches) on the aforementioned dataset. Apart from determining the best model (among the ones implemented), this procedure is able to reveal the contribution of each network component, indicating promising approaches for future

research.

Since, no publications have been found employing complex neural networks (e.g. CNNs) to the given classification task (Slijepcevic et al. (2018a) only evaluated a simple multi-layer perceptron), it can not be presumed that the pre- and post-processing methods used in other studies are necessarily the best choice for such models. Therefore, after identification of the best architectures, the implications of such methods on the performance are investigated as well. Again empirical verification is employed by conducting a direct comparison of the observed changes in accuracy, enabling the determination of the best set of data transformations. After completion of those experiments, all of the questions raised in Section 1.3 can be addressed, satisfying the purpose of this work.

1.5. Structure of this Thesis

Due to the high amount of different experiments conducted and their dependences among each other, it is important to clearly outline the structure being followed in this thesis. The first chapter will introduce the current state-of-the-art of GRF-based classification by examining previously conducted research and its implications on the proposed approach, divided into three main parts. While Section 2.1 will provide an overview of research on GRF-based automated gait classification in general, Section 2.2 is restricted to publications dealing with the exact same problem as this thesis. Finally, Section 2.3 summarizes the conclusions of earlier attempts and how CNNs might be employed to overcome their limitations.

Chapter 3 will establish the necessary background on the topic of time-series classification (Section 3.1), followed by an explanation of the fundamental neural network types used in this research, detailed in Sections 3.2- 3.5. Due to its high importance for the proposed methods, this is followed by Chapter 4, dedicated to the *GaitRec* dataset, detailing its composition and fundamental properties. Furthermore, an explanation of the different components constituting a single GRF-measurement will be provided, describing the recording procedure and additional processing steps.

An overview of the general workflow followed in the experiments is given in Chapter 5. The data selection procedure and its possible implications on performance will be discussed in Section 5.1, followed by an explanation of the applied pre-processing methods in Section 5.2. Due to the high accuracy achieved by CNNs on image classifications tasks, several proposed transformation approaches used to convert time-series to images, are introduced in Section 5.3, outlining the necessary calculations and their possible benefits to the classification. A description of the evaluated neural network architectures and their parameters is given in Section 5.4, structured in an incremental fashion, starting from a basic MLP (Section 5.4.1) and ending with state of the art image-classification models such as Google's *Inception v3* in Section 5.4.3, employed for

the transformed time-series. This chapter is then completed by an analysis of post-processing methods in Section 5.5, that can be applied to boost the accuracy of the prediction *after* the actual classification.

Details on the experimental setup will be provided in Chapter 6, describing the computational environment and parameters for data selection and pre-processing. This is followed by the general evaluation procedures in Section 6.2, outlining how the best settings have been obtained for each neural networks before explaining the fine-tuning process. Additional insights into the comparison process used for evaluating pre- and post-processing methods will be presented in Section 6.2.4, followed by as similar discussion for the effects of data selection and class aggregation (Section 6.3).

The experimental results are reported in Chapter 7, concluded by an analysis and discussion of the outcome. The main findings will then be summarized in Chapter 8, drawing conclusions from the experiments and considering their implications for future research.

2. Related Work

This chapter is intended to provide an overview of previous research in the field of automated GRF-classification. Since a wide variety of approaches has been employed, the existing literature can be roughly split into three parts according to their influence on this research. The first section will provide an outline of methods used to classify GRF-measurements for a variety of tasks such as malfunction or activity recognition, highlighting their general idea and achieved results. The second part is dedicated to research that has been conducted on a subset of the data that was later published under the name *GaitRec* (Horsak et al., 2020), trying to solve the same classification problem as this thesis (i.e. mapping gait disorders to the associated joint). The last section presents a summary of the previous research, discussing areas that could benefit from employing more complex neural networks, concluding by reporting the results of the only publication that could be found investigating such architectures for GRF-data.

2.1. GRF-based classification

One of the first works concentrating solely on GRF for gait classification was published by Köhle and Merkl (1996) and uses self-organizing maps (SOMs) for mapping gait patterns to distinct body parts. Their research was one of the first focusing on automatic classification and demonstrated that the SOM was not only able to distinguish between malfunctions in the left and right motor system, but could also represent the location of the actual impairment (e.g. pelvis, knee, ankle) to a large degree. They further refined their research (Köhle and Merkl,

2000) by using radial basis function (RBF) networks and report an accuracy of 76% on a dataset consisting of 487 persons and 15 associated body regions. While this is an impressive result, unfortunately no details on the composition of the dataset are disclosed. This is problematic because according to Köhle and Merkl (2000), some malfunction classes still show a rather low recognition accuracy, and we do not know how many of those were included in the data (i.e. the high accuracy could be caused by a large number of well separable injuries). Analyzing their work is being made more complicated by the fact that they included patients with a prosthesis as well as other injuries which might contribute to the high accuracy achieved, as Soares et al. (2016) show that distinguishing between patients using a prosthesis and a healthy control group can be done reasonably well.

Further research has been conducted by Goh et al. (2014), who tried to identify different activities (e.g. walking, jogging, running) based on GRF measured by an instrumented treadmill. They implemented an artificial neural network (ANN) approach, which was able to deliver an accuracy of 72% on the task. A notable contribution is their observation that distinguishing between patterns using a single stride results in a lower accuracy than using multiple ones. This suggests that the individual variation between two steps is considerably high and should be accounted for in any classification approach (e.g. by averaging across multiple steps).

The previously mentioned study by Soares et al. (2016) puts its focus on GRF measurements of patients walking with a prosthesis and reveals that statistically significant differences in many portions of the waveforms exist when compared to able-bodied participants. The existence of such differences implies that implementing an automated approach for distinguishing between those classes should be possible, as a well designed classifier would be able to take advantage of the distinct parts in each waveform. However, this has only been verified for patients using a prosthesis and does not necessarily generalize to patients suffering from an injury (e.g. a fractured bone).

2.2. Approaches based on *GaitRec*-data

Slijepcevic et al. (2017) provide the most exhaustive study so far, including the GRF measurements of 910 persons in total. They classified the subjects (being of various physical composition and gender) manually into five different impairment classes (*healthy* - *hip* - *knee* - *ankle* - *calcaneus*) resulting in 182 patients per class. Within a single class (i.e. a specific joint), multiple disorders (associated with the joint) are contained, such as ligament ruptures, fractures or joint replacement surgery. The dataset used in their research actually forms a subset of the more exhaustive one that was later published as *GaitRec* by Horsak et al. (2020) and will be used as part of this thesis. Because of the high relevance of this dataset for the conducted research, a detailed explanation of the recording and pre-preprocessing steps is given in Section 4 and will

be omitted here, since the procedure is basically identical.

Because GRF-measurements are usually taken at a high sampling frequency, the output obtained from force plates or similar devices is of high dimensionality as well. Therefore, it is a common approach to reduce the dimensions of the signals, for example by expressing them as discrete gait parameters such as local minima/maxima or time-distance parameters of the waveform (Alaqtash et al., 2011). Conducting a principal component analysis (PCA) is an alternative method producing a similar effect and, although it originated from a different field, research has demonstrated that is is able to provide a suitable input for GRF-classification tasks (Soares et al., 2016; Muniz and Nadal, 2009). Consequently, the first step in the evaluation process applied by Slijepcevic et al. (2017) investigated the effect of different PCA-based representations on GRF-data. The main goal of performing a PCA is to transform the input into a representation of lower dimensionality, while preserving as much variance as possible. Several approaches, such as running PCA on the concatenated raw signals (early fusion) versus conducting PCA for each signal separately before concatenating (late fusion) or PCA of commonly used discrete parameters, are possible and have been evaluated. The authors report that late fusion outperformed early fusion and that the usage of COP signals was able to improve accuracy, while the inclusion of the unaffected limb's GRF had no impact on the result.

Building on the knowledge obtained from this first approach, Slijepcevic et al. (2018a) continued to evaluate the different signal representations as well as various machine learning methods. For this purpose, three different parametrizations were used:

- 52 discrete and time-discrete parameters extracted from all five GRF input signals.
- PCA obtained from the raw waveforms using the best setting from Slijepcevic et al. (2017).
- PCA of the z-standardized and min-max normalized discrete and time-discrete parameters.

All of these representations of the input signals have been classified using Support Vector Machines (SVMs) with linear and radial basis function (RBF) kernels as well as a k-nearest neighbour (k-NN) classifier (using grid search over various values of k) and a multi-layer perceptron (MLP) with different numbers and sizes of hidden layers. In conclusion, the results for the MLP and k-NN classifiers were all outperformed by the SVM, achieving an accuracy of 54.3% in combination with the PCA obtained from the raw signals, the discrete parameters took second place (46.8%), while the PCA of discrete parameters scored the lowest result of 45.6%. Even though the reported accuracy of the MLP (52.5%) suggests low performance, its will be re-evaluated as a part of this research to provide a baseline for the more complex approaches. Additionally, because Slijepcevic et al. (2018a) restricted their further experiments to the SVM (and managed to increase its accuracy to 59.5% on a balanced dataset), the MLP has never been employed in a setting where the data from both legs was combined or with the raw waveform as an input.

The data of the unaffected leg was also used in the next publication by Slijepcevic et al. (2018b), in the form of a delta signal representing the difference between the affected and unaffected limb. Additionally, they experimented with adding the first derivatives of the raw waveforms as a possible source of information, testing various combinations of input signals. By combining the GRFs of the affected side together with their first derivates and the delta signals as training data for a SVM, they managed to achieve their best result so far, obtaining an accuracy of 67.8% on the classification task *healthy* - *hip* - *knee* - *calcaneus* (note that the *ankle* class was omitted from this study).

In their latest study (Slijepcevic et al., 2020), they returned to the original problem using five classes, investigating the impact of different aggregation methods of individual *trials* recorded during the same gait analysis session. According to their results, the highest accuracy has been found using the same combination of signals mentioned above, coupled with a majority voting based approach, where the class for each *trial* is predicted individually. The final classification result is then determined by calculating the statistical mode of all measurements recorded during the same session. Using this setting, they report an accuracy of exactly **62%**, which is the highest score published for this classification task and the benchmark for evaluating the results of this thesis. It should also be noted that this result was achieved using all components recorded during a GRF-measurement, whereas experiments conducted with different subsets have been found to be less accurate.

2.3. Towards an CNN-based classification

Since this research tries to solve the same classification problem, it relies heavily on the groundwork provided by Slijepcevic et al. (2017, 2018a,b, 2020), trying to improve accuracy even further by adapting current state-of-the-art machine learning methods for the task at hand. Therefore, the main insights and problems reported in their research will be briefly summarized, highlighting some shortcomings that could potentially be taken advantage of by the neural network architectures investigated in the later chapters of this thesis.

- So far, no automated approach has surpassed 62% accuracy on the given task, although higher values are reported for different GRF-based classification problems.
- There is high degree of individuality within GRF-measurements, resulting in a high standard deviation for all signals in a single class.
- Using all available data (i.e. *trials*) for training combined with majority voting seems to have an advantage over other aggregation methods.
- Some classes are harder to distinguish than others, indicated by the better result of the study where only four classes were used.

- Including all components obtained during a GRF-recording facilitates higher accuracy (i.e. there is viable information in each signal).
- Taking a representation of the whole waveform into account (PCA) works better than a set of manually selected discrete parameters.
- Including the data from the unaffected leg helps to yield better classification results.
- Temporal relations that can be assumed to exist within the signals have not been accounted for.
- Relationships between the different GRF signals have been disregarded as well, due to the nature the employed classifier (SVM).

The last two points are the main focus of this research, trying to take advantage of temporal dependencies and inter-signal correlations to achieve higher accuracy. To develop a better understanding on how these areas can be improved, it is necessary to take a look on how a SVM (and a MLP for that matter) are trying to classify the provided data. Both approaches regard the data as a vector with a predefined length, where each coordinate corresponds to a different dimension in a multi-dimensional space. Thus, they examine the signal as a whole and the notion of time is lost. However, that does not mean that there is no time-related information left in the signal. An important characteristic of data produced by sequential measurements (like the one obtained by force plates), is that the passing of time enforces an inherent ordering on the data (Last et al., 2004). More precisely, this implies that each dimension (i.e. each position in the vector) corresponds to a certain time-step and the information that the second position has been recorded after the first one is inherently encoded into the signal. This information is always present within the data, but what is lost when classifying with an SVM (and to a lesser degree with an MLP) is the ability to draw conclusions from the ordering in which certain events (consisting of a sequence of values) occur within the signal. For example, it might be important if a specific event occurs *before* or *after* another and additional information could be gained for increasing accuracy, if this is accounted for. An easily understandable example has been illustrated in Figure 1, displaying two identical signals with one of them shifted along the temporal axis.

Both a SVM and a MLP would look at the values in each dimension (in this case time) separately and observe that there is a clear difference between the two signals (e.g. $X[11] = 1$ while $Y[11] = 8$). But if the notion of time is taken into account, both signals actually represent the same phenomenon, Y just occurs 4 seconds after X but the pattern (first a small peak, then a big peak) remains unchanged. In gait analysis this is usually accounted for by normalizing the time to 100% stance (which would place the peaks roughly around the same time-step), but for a disorder classification, this might not be good enough. Consider the case of a ruptured ligament and suppose that the resulting instability will be visible by a shift in the centre of pressure (COP)

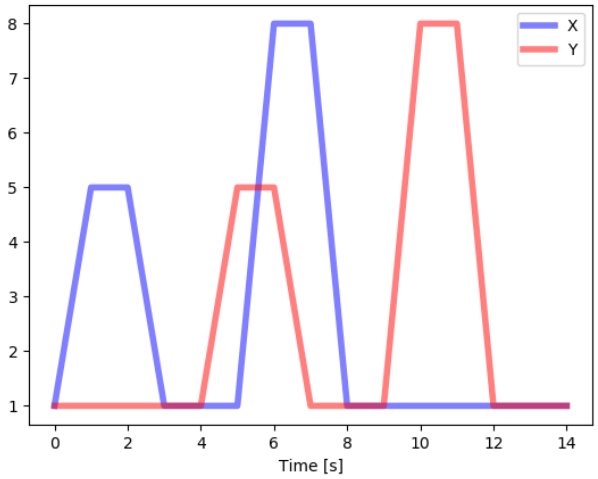

Figure 1: Demonstrating the importance of time preservation. Both X (blue) and Y (red) consist of the same values, but Y has been shifted by 4 seconds.

during walking (this is just a simple example to illustrate the idea, not an observation made on the actual data). Depending on their muscular strength, some patients might be able to partially compensate the instability by exerting additional muscular effort during the initial phase of the step, resulting in the same shift happening at a different point in time. Thus, even if the actual signals look different, they should be classified into the same group, as they are caused by the same injury.

The loss of information between GRF components when using a SVM/MLP is easier to illustrate. Because the input for those types of machine-learning techniques has to be a 1-dimensional vector, the simplest method is to concatenate all signals together to form a single, long input vector. Another option would be to classify each signal separately and then combine the output of these classifications into the final result. It should be clear, however, that by applying any of these methods, all correlations between signals are lost. Since GRF measurements consist of several components, they do not only contain information about each single waveform separately, but additionally contain information about interrelations between those signals (Valdés-Sosa et al., 2006). In other words, the relationship between different signals might also provide important clues for an accurate classification. As a simple example, it could be relevant whether two signals are close together or not, independently of the actual values (i.e. no matter if they meet at a peak or a valley).

The idea to investigate the importance of temporal and inter-signal relationships of GRF measurements has been explored by Alharthi and Ozanyan (2019), trying to classify patients with Parkinson's disease according to the severity level. Although the classification task and data used are quite different, the key points of their research will be briefly summarized here to justify the general idea and establish some of the methods examined in this thesis (such as CNNs

and LSTMs) as viable approaches. The authors used a dataset containing samples of 166 person, where only the vertical ground reaction force (F_v) was recorded using eight sensors placed underneath each foot. Additionally the sum of all eight tensors of each food was added to the inputs, resulting in 18 signals total. Three different classifications approaches were implemented and evaluated on this data: *(a)* a 1-dimensional CNN, *(b)* a 2 dimensional CNN and *(c)* a long short-term memory (LSTM), representing the same major architectures that will be investigated in this research (a detailed explanation of their main concepts is given in Chapter 3). All implemented networks were fairly complex consisting of four *(a)*, seven *(b)* and three layers *(c)* respectively, all of them outperforming earlier classification approaches using the same dataset. While a traditional input layout was used for *(a)* and *(c)* in the form of a $time\text{-}steps \times signals$ matrix, a different approach was chosen for the 2-dimensional network. The transformation applied fuses the input into a spatio-temporal 3D matrix (Costilla-Reyes et al., 2018), by arranging the signals at each time-step in a similar fashion to a digital image, enabling the convolution to analyze the relations between signals in a more fine grained manner than the 1-dimensional case.

Previous attempts established the baselines of 92.7% or 84.48% using a SVM (Abdulhay et al., 2018; Wu et al., 2017) and 88.89% when employing an MLP (Ertuğrul et al., 2016) on the same task, while the 1-dimensional CNN was able to achieve an accuracy of 95%. Both, the 2-dimensional CNN and the LSTM scored slightly higher, reaching 96%, the highest reported value so far. It is noteworthy though, that the training process for the LSTM required considerably more time, taking almost 90 minutes compared to approximately 10 minutes for the two CNNs. Even though the question remains of whether or not those improvements can the transferred to the task of classifying gait impairments, their results demonstrate that a significant boost in performance can be achieved when employing those methods to GRF-data, due to their abilities of exploiting inter- and intra-signal relationships.

3. Background

Despite the fact that high accuracies have been reported for the machine learning approaches implemented in this research, good performance is not guaranteed by simply inputting the data into the network and hoping for the best. A number of parameters has to be carefully selected and adjusted according to the data and classification task. Therefore, a fundamental understanding of different data representations and the way how machine-learning algorithms are trying to exploit those to group similar items together (i.e. classification) is required in order to enhance accuracy. Understanding this process is not only useful for determining the best layout of the data, but also for reasoning about which architectures to evaluate, as some machine-learning techniques are better suited for a certain task than others.

The current chapter will provide basic information about the problem of time-series classifica-

tion (TSC), since GRF-measurements can be regarded as a special case of a multivariate time-series. This implies that they do not only share common characteristics, but also that knowledge obtained about the more general problem can be transferred to the specific instance. This enables the investigation of machine-learning techniques that have been successfully applied to tasks that are somewhat similar to GRF-classification, taking advantage of a wider field of scientific research.

Additionally, the basic principles of neural networks will be explained in an incremental fashion, starting from the MLP as the most basic form over recurrent networks such as the LSTM up until current state-of-the-art deep learning approaches. This serves the purpose of establishing a common background knowledge that will be used in the later parts of this thesis, where the design of different architectures will be discussed and evaluated. As the concepts introduced in this chapter constitute the core of this research, a fundamental understanding about their mechanics is required. Apart from a detailed explanation, the emphasis in this section will be put on highlighting the potential usefulness of each type of model for enhancing accuracy on the classification task.

3.1. Time-series classification (TSC)

As previously mentioned, GRF-measurements can be considered as special instance of a multivariate time-series. By their very nature they consist of sampled data points, observed over time from a continuous, real-valued process (Längkvist et al., 2014). If just one input signal is measured in this way, a univariate time series is created, but in case of GRF-data, several signals are captured at the same moment, consisting of three force components and the two directions of movement for the center of pressure (COP), creating a multivariate time-series composed out of five separate signals. According to Fawaz et al. (2019a) time-series can be formally defined as follows:

- A **univariate** time-series $X = [x_0, x_1, \ldots, x_t]$ represents real values in an *ordered* set, where the number of real values T is defined by the length of X (i.e. $T = t + 1$).
- An M-dimensional **multivariate** time-series $X = [X^0, X^1, \ldots, X^M]$ is built from M different univariate time-series where $X^i \in \mathbb{R}^T$.

A labelled dataset of GRF measurements $D = (X_0, Y_0), (X_1, Y_1), \ldots, (X_N, Y_N)$ can thus be defined as a collection of pairs (X_i, Y_i), where X_i is a multivariate time-series (consisting of 5 signals) and Y_i is the corresponding class (i.e. impairment) label.

The ultimate goal in a time-series classification problem consists of training a classifier on the dataset D, which is able to predict the corresponding label to any given series of values provided as an input. However, as Gamboa (2017) points out, care has to be taken when analysing time-

series, because they often contain temporal dependencies, meaning that two otherwise identical points of time could actually belong to different classes, depending on previous/future values of the series. Therefore, feature selection is one of the most important steps when trying to build a classifier for time-series and many handcrafted feature extraction schemes have been developed across different applications (Cui et al., 2016). Such feature extraction methods can range from simple statistics (e.g. mean, variance) to more complex features from spectral or fluctuation analysis.

As those algorithms are prone to being highly application dependent, a common standard within the TSC research community is to evaluate them against the UCR time-series archive (Dau et al., 2018), which is considered a gold standard (Bagnall et al., 2016). By now this dataset consists of 85 subsets across different fields, allowing for a better comparison of different algorithms. Traditionally, one of the most popular methods used on that dataset is a nearest neighbour classifier. This algorithm basically works on the assumption that the class of a given series is the same as the one of its closest neighbour (according to some distance measurement). In other words, given an unlabelled sample, the algorithm identifies the (already classified) instance with the minimal distance (i.e. the nearest neighbour) and assigns the same class to the new sample. A popular variant with increased robustness against outliers is the k-nearest neighbour algorithm, where the classification result is chosen based on the majority class of the k closest instances.

In a comparative study of distance functions, Lines and Bagnall (2014) showed that the Dynamic Time Warping (DTW) distance is able to deliver the best results in this setting and still provides a very strong baseline in the field (Bagnall et al., 2016). Furthermore, the experiments by Lines and Bagnall (2014) revealed that an ensemble of individual nearest neighbour classifiers (with different distance functions) is able to yield a significantly more accurate result than its individual components. In the most exhaustive study on the UCR dataset so far, Bagnall et al. (2016) verified the results of 18 different algorithms, revealing ensemble classifiers as the current state-of-the-art when dealing with TSC problems, as the seven most accurate algorithms all consisted of various ensembles.

The top accuracies across all datasets were achieved by COTE (Collective of Transformation Ensembles) (Bagnall et al., 2015), an algorithm that uses a combination of classifiers in the time, autocorrelation, power spectrum and shapelet domain. In total, COTE makes use of 35 different classifiers, using a weighted vote system to determine the final category for each sample. Lines et al. (2018) later expanded on that system by introducing a hierarchical vote system and two new classifiers (including two additional signal transformations), making the new algorithm (referred to as HIVE-COTE) even more accurate.

However, there are some important points to consider before applying those algorithms to a specific problem domain:

- The increased accuracy comes at the cost of additional computations and as Fawaz et al. (2019a) points out, training 37 classifiers as well as cross-validating all hyper-parameters in combination with the time complexity needed for some of the input transformations, increases computational requirements tremendously. In some situations, training such a classifier can even become prohibitively expensive (Lucas et al., 2019).
- Those algorithms generally represent a result that is best on *average*, across all datasets. Taking a closer look at the results obtained by Bagnall et al. (2016), reveals that there are significant differences between the various problem types (such as signals obtained by electric devices, spectrographs, motion capture and sensor readings) and the length of the time series seems to be a decisive factor as well. Therefore, those algorithms will work well with high probability when applied to a specific problem, but there is no guarantee that they will achieve the highest accuracy for a given dataset.
- All of those algorithms have only been validated for univariate time-series. Scientific results for multivariate time-series are rare and research seems to be still in its early stages (Fawaz et al., 2019a). As of now, all signals of a multivariate time-series would have to be concatenated together in order to use these classifiers, which might induce information loss (depending on the problem domain). Nonetheless, this technique seems to be commonly applied when dealing with multivariate time-series (Zheng et al., 2016; Liu et al., 2019; Slijepcevic et al., 2017).

In order to alleviate the issue of excessive data preprocessing and feature engineering, several studies (Zheng et al., 2016; Cui et al., 2016; Wang et al., 2016) have been investigating convolutional neural networks (CNNs) for an end-to-end time series classification. Unlike the classifiers above, which are based on a number of selected features (35 of them in the case of COTE), feature selection and classification are learned jointly by a CNN, requiring no further processing of the input data (Hatami et al., 2017). In other words, the CNN is able to learn and select the best "view" on the input data by itself, making further transformations unnecessary. The general concepts and underlying architecture of a CNN that enable this process, will be the topic of Section 3.3.

3.2. Multi-layer Perceptron (MLP)

Neural networks, as the name suggests, have been designed to simulate the learning process of a human brain (although in a very simplified manner) relying on neurons as its basic units. The neuron can be either active (i.e. *firing*), transmitting a message to all neurons it is connected to, or inactive, meaning it does not communicate. The state of a neuron is decided by the activation function applied to the weighted sum of all of its inputs, basically deciding whether the neuron is *firing* or not. If a neuron has the inputs $x_1, x_2, \ldots, x_n$ associated with the weights $w_1, w_2, \ldots w_n$

and the activation function f, then its output y is decided by:

$$y = f(\sum_{i=1}^{n} w_i x_i) \tag{1}$$

The actual learning process taking place within a neuron is quite simple. By adapting the weights in a certain way, the neuron can be conditioned to react only to a certain type of input, for example enabling it to recognize a single class of a classification problem. However, when the input data is not linearly separable, a special layout of such neurons is required to accomplish that task. Such a layout, which is commonly referred to as a multi-layer perceptron (MLP) (Marsland, 2009) is illustrated in Figure 2. It consists at least of an input layer, a hidden layer (enabling the non-linear classification) and an output layer providing the final result, however, no upper limit on the number of hidden layers applies, (apart from the available computing resources). Because each neuron within a layer is connected to all the neurons of neighbouring layer, this kind of architecture is often referred to as a *fully-connected* or *dense* layer. Because the former is ambiguously used only the terms MLP and *dense* layer will be used in this thesis to denote this type of neural network.

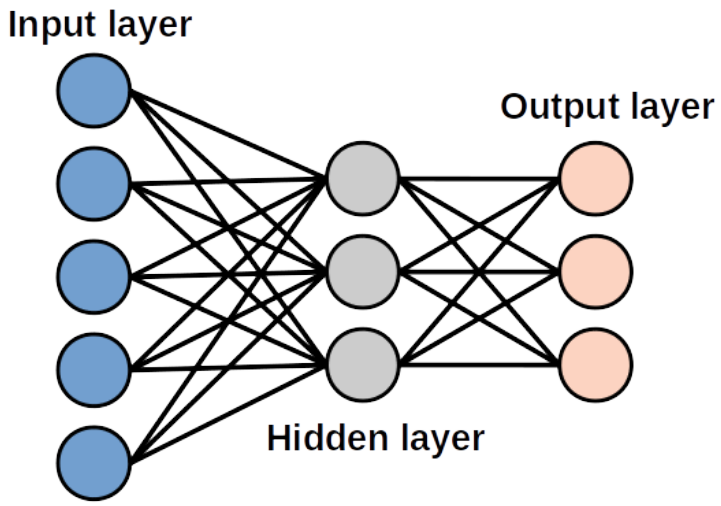

Figure 2: The multi-layer perceptron consisting of multiple layers of connected neurons, where each neuron of one layer is connected to all the neurons of its neighbouring layers. Adapted from Marsland (2009)

Due to its basic form, the MLP still retains some advantages over the more complex forms of neural networks. As a consequence of its simplicity, the implementation is very straightforward and it does not need as many computing resources, resulting in a training process that can be completed reasonably fast. On the other hand, it is not able to model temporal relations or interactions between signals, resulting in (possibly) less accuracy. Therefore, contrasting its performance against more advanced architectures specialized in extracting such relationships, additionally serves the purpose of gauging how much information is actually contained between

the five components of the GRF-data as well as within their temporal dependences.

3.3. Convolutional Neural Networks (CNNs)

CNNs are a variation of neural networks commonly used in image processing tasks in order to reduce the number of parameters to be learned by the network (Gamboa, 2017). They represent one of the most well-known variants of neural networks, because they incorporate feature learning and classification task into one unified network architecture and have been successfully applied to image, speech and text recognition (Liu et al., 2019). Their ability to identify relevant features for the classification by themselves, without the need of finding the best representation beforehand, basically eliminates the need for significant domain expertise and manual feature extraction (Karim et al., 2019).

In contrast to an MLP, the neurons in the hidden layer are only connected to some of the input layers and weights are shared across neurons within a group. By using this kind of architecture, each group of neurons calculates their own convolution of the image, commonly called a *feature*. Such a convolution can be seen as a sliding window moving over the time series, applying a filter at each step. This process is illustrated in Figure 3. In general, as Fawaz et al. (2019a) describe it, such a filter can be viewed as a non-linear transformation of the time series, resulting in a "processed" version of the signal. To describe it in a more formal way, the convolution at each time stamp t is given by the following equation (Fawaz et al., 2019a):

$$C_t = f(\omega * X_{t-\frac{l}{2},t+\frac{l}{2}} + b) | \forall t \in [1, T] \tag{2}$$

In the above formula, C denotes the result of a convolution ($*$ is the dot product), which applies a filter ω of length l and a bias parameter b to a time-series X of length T. To calculate the final output of each neuron, the activation function f is applied to the result of the convolution. The output (of a single filter) can then be considered as another univariate time-series and using several different filters will result in a multivariate time-series with a dimensionality equal to the number of filters applied.

To be more specific, this process describes a 1-dimensional convolution, which is commonly used for TSC problems. In contrast to the 2-dimensional convolution used in image classification tasks (where the sliding window moves along the *width* and *height* of the image), the height of the window (i.e. the number of signals in a multivariate time-series) remains fixed in the 1D case and the window only moves along the temporal axis. Operating in this setting, the convolution considers all signal at once, providing the additional advantage of accounting for the relationship between signals in multivariate time-series. Note that in the univariate case, the window would just include one signal and the convolution would be exclusively done in time domain.

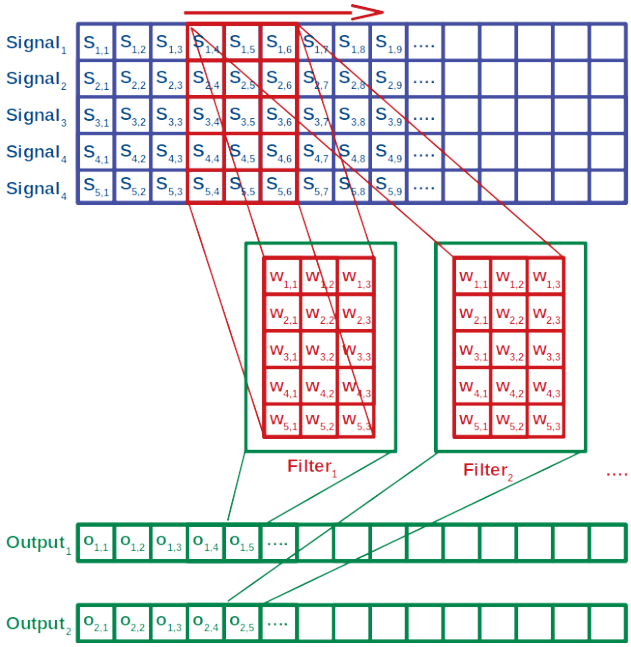

Figure 3: 1-dimensional convolution of a multivariate time series. The sliding window (red) moves along the time-axis convoluting all signals together at each step. Every filter has its own set of weights, applying a unique transformation at each time-step t. In the picture, time steps t_1, t_2, t_3, t_4 and t_5 have already been processed (with t_5 being displayed). Subsequently, the sliding window will move one position to the right and the convolution will be applied to the signal at t_6, determining the output at o_6 for all filters.

One of the first studies to use this kind of architecture for TSC was conducted by Zheng et al. (2016). In their work, they split multivariate time-series into separate signals, perform a 1D convolution on each of them and finally combine the results into a MLP to obtain the final output. On the tested datasets, this approach outperformed both nearest neighbour methods (as discussed earlier) and conventional MLPs by 5-10% in all cases. Further research has been done by Cui et al. (2016), who developed a so-called multi-scale convolutional neural network (MCNN) trying to move towards a more general architecture (that works well on different time-series) by adding a *transformation* stage before the actual convolution. This stage includes downsampling and filtering of the data in order to obtain features that are more robust (e.g. by removing high-frequency perturbations and random noise) and identify suitable scales (i.e. a representation where the pattern becomes recognizable). Their work is interesting because they evaluated their architecture on the UCR dataset, comparing it to the previously mentioned ensemble classifiers (i.e. the current state-of-the-art in TSC). While they could establish that their approach works better than an ordinary CNN, it was still less accurate than COTE (although not significantly). Similar results were later obtained by Zhao et al. (2017), for a CNN consisting of multiple convolutional layers, evaluated on a subset of the UCR data.

3.4. Long Short-Term Memory (LSTM)

Recurrent neural networks (RNNs) have been designed with the goal of extracting temporal information from the input sequence for either classification or prediction, as they both constitute a similar problem (Hüsken and Stagge, 2003). The basic idea is to use the output of the current time step as an input to the next time-step, enabling the network to make decisions based on previous states (see Figure 4 for an illustration).

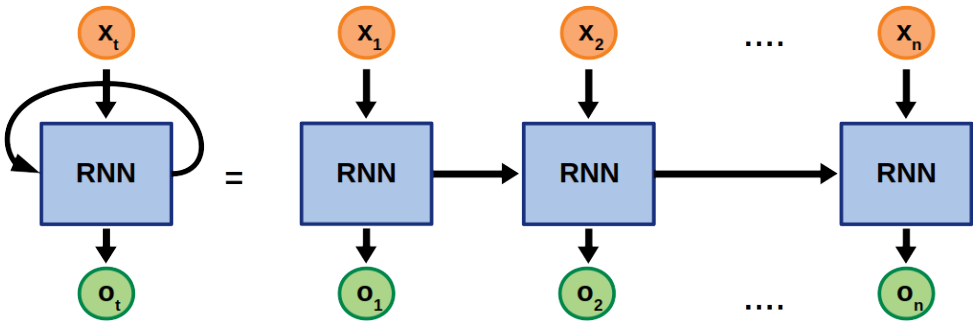

Figure 4: An unrolled recurrent neural network. The output of the previous time step is used as an input to the current one, enabling the network to learn temporal relations.

Since it is a fundamental property of time-series, that a point x_i is influenced by all its predecessors $\{x_{i-1}, x_{i-2}, \dots\}$, several studies (Hüsken and Stagge, 2003; Malhotra et al., 2017; Che et al., 2018) have investigated recurrent neural networks for time-series classification. However, while being a simple and powerful model in principle, RNNs face a number of issues that severely limit their usefulness (Pascanu et al., 2013). Most prominent among them are the vanishing and exploding gradient problems described in Bengio et al. (1994). These terms refer to the large increase/decrease that can be observed in the norm of the gradient in the back-propagated error during training of a RNN. Such phenomena are caused by the long term components of the signal, which can grow/decay exponentially more than the short term components (Pascanu et al., 2013). In the case of exploding gradients, this may lead to oscillating weights, while the impact of the long term components will be reduced to zero for the vanishing gradient problem, making it impossible to learn temporal correlations. Both versions make proper training of the network impossible, preventing this approach from being used in many applications.

In order to address this issue, Hochreiter and Schmidhuber (1997) introduced the long short-term memory (LSTM) using gating functions into the internal state of their network, making the error back-propagation constant, thus avoiding the aforementioned problems. This enabled the LSTM to be trained more effectively than RNNs, becoming one of the most widely adopted recurrent networks (DiPietro and Hager, 2020). According to Karim et al. (2018a) a LSTM does not only maintain a hidden vector H for storing information from previous states, but also a memory vector M, which is responsible for controlling the output and updating H. Perhaps a

little more understandable, Alharthi and Ozanyan (2019) describe each neuron (referred to as a *block*) of an LSTM as consisting of three gates:

- *Forget Gate*: This gate decides which information to maintain within each block.
- *Input Gate*: Responsible for controlling the data flow into the block
- *Output Gate*: Takes care of the output based on the computed activation of the block.

Strangely enough, even though they have been built for temporal classifications, studies evaluating only RNN architectures (without added convolution) on the UCR dataset are quite rare, one exception being Smirnov and Nguifo (2018), comparing traditional RNNs against the LSTM and a convolutional approach developed by Wang et al. (2016) (which will be discussed later in this thesis). They conclude that among all RNNs tested, the LSTM achieves the best classification results (significantly better than its competitors) and is outperformed only by the CNN, which has been established as a strong baseline on the UCR data (Wang et al., 2016). Judging from these results, it can be assumed that a combination of CNN and LSTM might be able to provide an ideal solution to the TSC problem. This idea has been investigated by Karim et al. (2018a), proposing an architecture that applies an LSTM in parallel to a CNN to extract both long and short term dependencies from the data, by concatenating the output from both networks. Their experiments show that this approach is able to outperform most state-of-the-art models for TSC, while requiring minimal pre-processing of the data and upholding the ability of the network to perform the feature extraction on its own (Karim et al., 2019).

3.5. Deep Neural Networks

Recent algorithmic advances and availability of more and faster hardware have enabled scientists to build and train deeper neural networks, obtained by stacking multiple (convolutional) layers on top of each other. This has been deemed too difficult and inefficient (Bengio et al., 1994) in the past, but started to change with the training method proposed by Hinton et al. (2006), which allowed for fast learning of so called Deep Believe Networks. Soon, this technique, now commonly referred to as *Deep Learning* (LeCun et al., 2015) was applied to various types of networks with great success. As explained by Gamboa (2017), latest deep architectures use several modules (each consisting of multiple layers) that are trained separately and stacked together afterwards, so that the output of one module becomes the input of the next.

Wang et al. (2016) established a first benchmark for TSC using deep neural networks by comparing three different types of architectures (see Figure 5) :

- *MLP*: This architecture consists of several hidden layers of MLPs, each consisting of 500 neurons and using a rectified linear unit (ReLU) to calculate the activation of the neuron. The last layer uses a softmax activation and consists of as many neurons as there

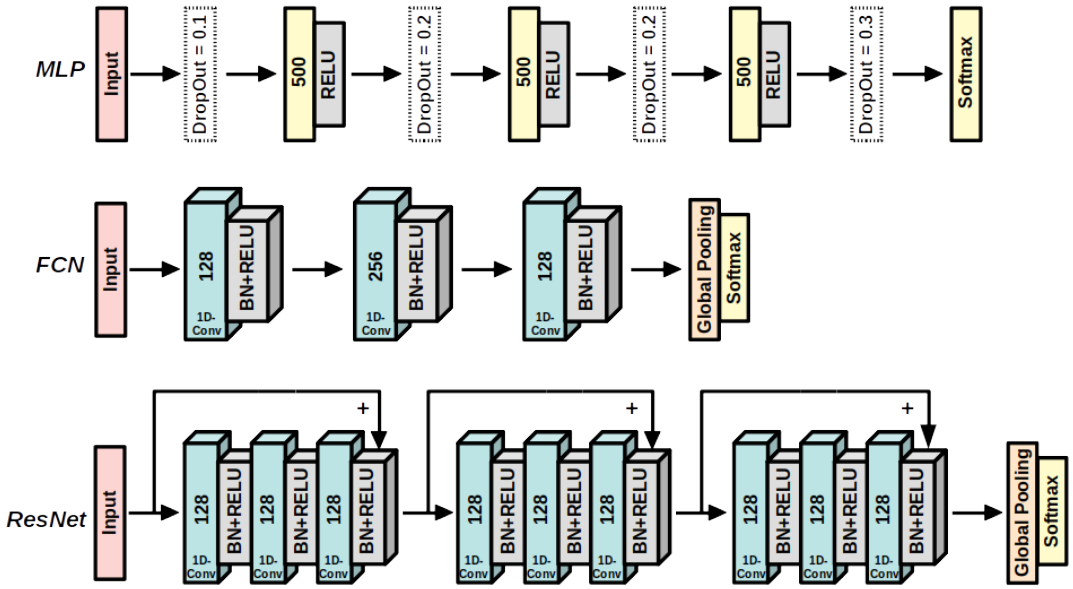

Figure 5: Deep architectures as proposed by Wang et al. (2016). The models are ordered by increasing number of layers. For the MLP the numbers within the solid rectangles denote the amount of neurons in each layer, whereas for the other architectures, the number within a box represents the number of filters used in the convolution.

are classes within the time-series, because it is responsible for providing the final output (i.e. the classification). The *DropOut*-layers (Srivastava et al., 2014) in between are used to guarantee a better generalizability of the networks, by randomly ignoring the denoted percentage of neurons from the previous layer.

- *FCN*: Their so called fully-convolutional network consists of three 1-dimensional convolutions followed by a batch-normalization (Ioffe and Szegedy, 2015) and an activation function. Each convolution operates with a different sliding window (decreasing in size from 8 over 5 to 3) and applies a different number of filters (128, 256, 128). The output of the last convolution is then passed through a global *Average Pooling*-Layer to reduce dimensionality (by taking the average along each feature axis) before the final output is produced in a similar fashion to the MLP.
- *ResNet*: The residual network is the "deepest" architecture employed, consisting of three modules, where each module is basically a full FCN (without input and output layer). A shortcut connection is added between those modules (also called "residual blocks" - hence the name) to enable the gradient flow directly through the bottom layers.

In their paper, Wang et al. (2016) then continue to validate those models against a subset of the UCR data, comparing them against established nearest neighbour methods such as COTE. They conclude that both FCN and ResNet are able to classify time-series with good accuracy, without a statistical significant difference in performance to COTE. Despite this, they conclude that the FCN yields the best results overall (giving the best accuracy in 18 out of the 44 tested

datasets, five more than the second place) and is able to generalize quite well, even though the training accuracies are close to 100%. ResNet, while achieving comparable performance, has a more limited generalizability, thus scoring slightly lower. The authors attribute this to the lack of complex patterns within the time-series of the UTC data and suggest using such a structure only for larger or more complex signals. Finally, the results for the MLP architecture are significantly worse, giving the lowest score across all compared models.

Those experiments have been later repeated in a similar, but more exhaustive study by Fawaz et al. (2019a). Their work compared the architectures above against other approaches (e.g. a pure convolutional approach called Multi-scale Convolutional Neural Network (MCNN) developed by Cui et al. (2016) and another deep architecture called Time Le-Net, originally proposed by Guennec et al. (2016), to name the more prominent ones). They used the full set of data available in the UCR archive (containing 85 different TSC tasks) for their experiments, on which ResNet was revealed to outperform all other approaches significantly by achieving the best accuracy on 50 datasets. It should be noted though, that the FCN took second place with 36 wins (if two or more architectures achieve the same highest accuracy on a dataset, all of them are denoted as winners), followed by a large gap to the third place with merely 17 wins.

3.5.1. Deep learning for multivariate time-series

Multi-variate time-series classification represents a special sub-category of TSC, characterized by additional interactions and co-movements within a group of signals (Liu et al., 2019). Although several specialized architectures have been developed taking this source of information into consideration (Zheng et al., 2016; Zhao et al., 2017), in principle it is possible to extend all approaches mentioned in the previous section for the multivariate case. After their analysis for univariate instances, Fawaz et al. (2019a) continued to compare those specialized architectures against the extended versions of FCN and ResNet on a dataset of 12 multivariate time-series. Their results were surprising, revealing that both FCN and ResNet outperformed their more specialized counterparts, making them better suited for the task at hand. Since then, several architectures have been proposed to better exploit the aforementioned characteristics, most of them based on the following ideas:

- *Separate convolution across domains*: This technique tries to extract temporal features and inter-signal relationships by applying convolutions in each domain individually. Usually the temporal convolution is performed first (one each signal) before their results are combined in a convolution across signals. This order is not a requirement however, and different versions have been applied as well (e.g. by Alharthi and Ozanyan (2019)).
- *Dilated Convolution*: In this case, the convolution is not applied to consecutive time-steps, but instead skips some values in between. Using a sliding window of $length = 3$

and a dilatation rate of two, this would mean that in the first step (i.e. the sliding window is located at the beginning of the time series), the convolution is applied to x_0, x_2 and x_4 instead of x_0, x_1 and x_2, skipping one value in between. For a dilatation rate of 3, two values would be skipped and so on.

An example of such an architecture was proposed by Liu et al. (2019), with the network dividing the convolutions into a univariate and multivariate stage (the core concept of which is depicted in Figure 6). First, the signals are arranged into a 3-dimensional tensor (with the last dimension being time), before (based on the concept explained above), features are extracted from each signal using a 1-dimensional convolution along the temporal axis. After this, 2-dimensional convolutions of various sizes are applied in order to extract information about the interaction across signals. Different sizes might be able to find different characteristics from the data, with large-scale filters more likely to detect symmetric properties and small scale filters being able to discover particular characteristics (Liu et al., 2019).

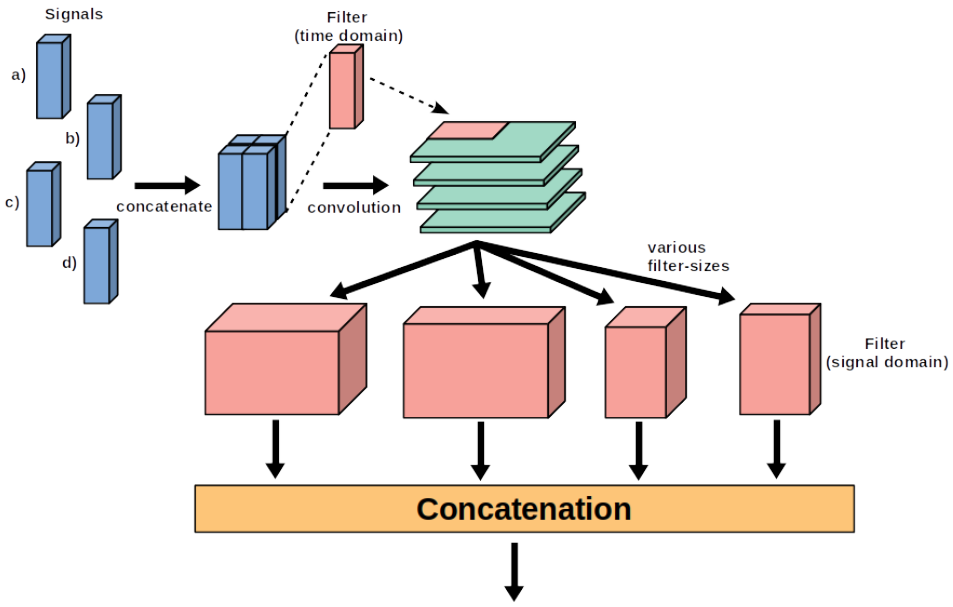

Figure 6: Core concept of the network proposed by Liu et al. (2019) (slightly simplified version). The signals are arranged into a 3-dimensional tensor before applying a convolution across the temporal axis. Afterwards various 2-dimensional convolutions are applied to extract different features about the relationship of the data. The results of this stage are then combined for further processing.

The concept of dilated convolution was implemented by Yazdanbakhsh and Dick (2019) in an attempt to develop a network for recognizing human physical activity (e.g. walking, jogging, standing) based on data obtained from an accelerometer. It consists of multiple 2-dimensional convolutions with every second layer using the dilated variant. The simple version consists of six layers using a dilatation rate of two (only in the time-domain). Their experiments show that this model is able to deliver accuracies on-par with established classifiers using hand-crated

features to classify the same datasets.

One of the most recent approaches to the problem (somewhat similar to Liu et al. (2019), but using a deeper architecture), was proposed by Fawaz et al. (2019b). Their architecture is heavily inspired by the idea of the so-called *Inception*-network, originally introduced to the field of image classification by Szegedy et al. (2014). The core feature of an *Inception*-layer is the application of several convolutions (with different filter sizes) at the same time, followed by a concatenation of the results (see "Concatenation"-step in Figure 6). The resulting architecture, referred to as *InceptionTime* by Fawaz et al. (2019b) is quite complex and contains several concepts that need to be explained:

- *Bottleneck*-layer: This layer uses a sliding window of length 1 and applies a number of filters m where m is considered to be smaller than the number of signals within the multivariate time-series. Thus, this layer performs a dimensionality reduction to reduce the input size (note that if $m = 1$ the result is basically an univariate time-series).
- *Max-Pooling*: An operation taking only the maximum value from a sliding window, thus reducing the output size.
- *Residual Connections*: Shortcut connections between modules of a network as explained earlier on the example of the ResNet-architecture.

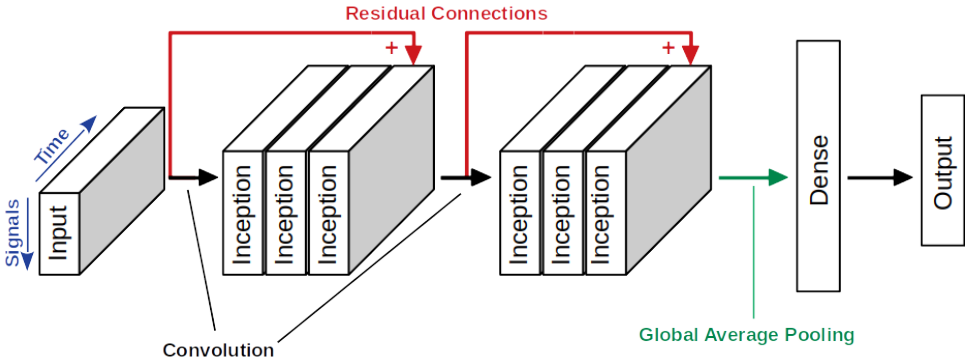

Figure 7: Overall architecture of *InceptionTime* according to Fawaz et al. (2019b). The whole network consists of two inception modules that are connected by residual connections. Global average pooling is applied to the final output of the inception-layers, before the final classification is created by a MLP.

The overall architecture of the network can be seen in Figure 7, while the details of a single *Inception*-Layer are illustrated in Figure 8. Three of these layers are stacked on top of each other to form a single inception module. Furthermore, the authors noticed that a single *Inception*-network exhibits a high standard deviation in accuracy, which they attribute to the random weight initialization and the stochastic optimization process itself. To mitigate this effect, they adopt an ensemble of networks (similar to how COTE uses an ensemble of nearest neighbour classifiers) and define the final output as the average across all individual models. The number

of networks to be used was determined empirically on the UCR dataset, concluding that there are no further significant improvements when using more than 5 of them together. Thus the model is computationally intensive, but, according to their results, still an order of magnitude faster than HIVE-COTE and slightly more accurate, outperforming all other CNN approaches significantly (Fawaz et al., 2019a,b).

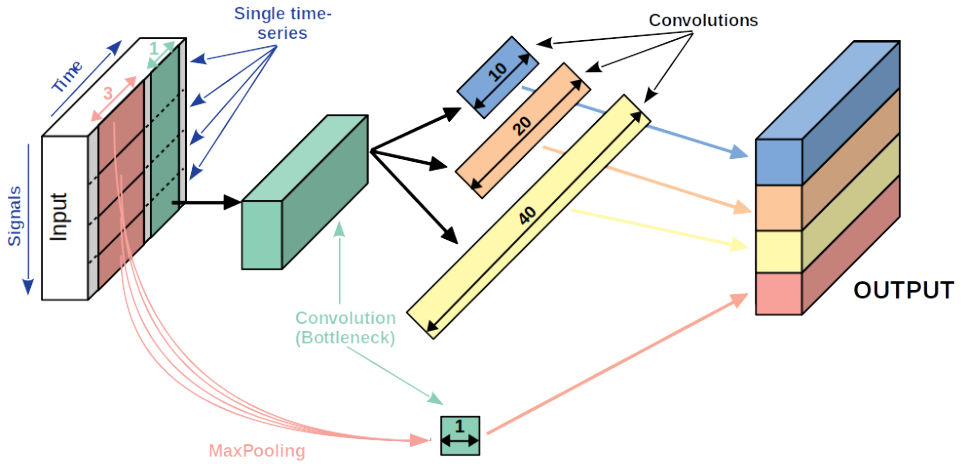

Figure 8: Detailed information about the *Inception*-operation used by to Fawaz et al. (2019b). First, the *Bottleneck*-layer is applied, that is a convolutional layer with a sliding window of size 1 and m filters (in the illustration $m = 1$). Afterwards, multiple convolutions of different lengths (10, 20, 40) are executed, and their outputs are concatenated. Additionally, to increase the robustness against small perturbations a MaxPooling-operation (on each signal separately, with a window size of 3) followed by a *Bottleneck*-layer is added to the final output. All parallel convolutions use a set of 32 filters resulting in a dimensionality of $4 \times 32 = 128$ for the output.

From the contents explained in this chapter, it should be clear that there is an abundance of CNN approaches that could be used for time-series classification and the state-of-the-art changes rapidly (Gamboa, 2017; Fawaz et al., 2019a)). Additionally, from the experiments conducted on the UCR-dataset, it becomes obvious that determining the best architecture for a single classification problem is a difficult undertaking. Because most networks introduced in this chapter have been designed to work best on average, there is always a possibility that they might fail at a specific task. This underlines the importance of a comparative study of methods for the GRF-classification problem, as it is almost impossible to identify the most suitable model based on research with different datasets.

4. The *GaitRec*-dataset

As this thesis has been designed for the purpose of extending the research conducted by Slijepcevic et al. (2017, 2018a,b, 2020), by trying a different approach to solve the GRF-classification

problem, it would be natural to evaluate it against the same data for easy comparison. However, the authors have since extended their dataset, publishing a new, publicly available version called *GaitRec* in Horsak et al. (2020), including more individual samples. This was motivated by the fact that it is difficult to obtain annotated large-scale datasets (as required by many supervised learning applications such as neural networks) and the authors express their hope that it will contribute to the development of assistive machine learning techniques for gait analysis (Horsak et al., 2020). Therefore, in order to support those ideals, the decision was made to switch to that new dataset, even though it might reduce comparability to the previous studies. On the other hand, using publicly available data does not only help to make this research more reproducible but additionally provides a (first) benchmark for any further experiments or studies using this database.

As described in Horsak et al. (2020), the dataset is already split into three parts:

- An unbalanced training set consisting of 52,745 trials, in the following denoted as *TRAIN*. The term *unbalanced* refers to the fact the number of samples belonging to each of the different classes varies, resulting in an unevenly distribution. Patients with ankle injures are the most frequent, making up 29% of the data, while only a limited number of healthy (i.e. no disorder) patients are available (11%). The percentages for the other classes are located somewhere in between those two.
- A balanced training set consisting of 6,308 trials, in the following denoted as *TRAIN-BALANCED*, where the number of patients belonging to each class is equal.
- A test set that should only be used for evaluation of the results (i.e. the classifier), in the following denoted as *TEST*, containing 22,987 trials. It is important to note that this dataset is also unbalanced, again limited by the number of measurements for healthy persons.

According to Horsak et al. (2020), the data provided is derived from an existing clinical gait database from an Austrian rehabilitation centre and was recorded between 2007 and 2018 during clinical practice, following the procedures of the rehabilitation centre. The data has been obtained from a total of 2,295 persons of various age (mean: 41.5, standard deviation: 12.1) and body mass (mean: 83.6, standard deviation: 17.3), with a male to female ratio of approximately $3:1$. The following section will give a brief summary of the recording process, with additional details available from Horsak et al. (2020).

4.1. Data recording

Measurements were taken by two force plates (Kistler, Type 9281B12, Winterthur, CH), placed in a consecutive order on an approximately 10m long walkway. One or several measurement

sessions (over the course of the rehabilitation process) were performed by each patient, with each session consisting of several recordings of two consecutive steps. Such bi-lateral measurements were taken by asking the participant to walk unassisted (and without walking aid) at a self-selected speed across a walkway with two embedded force plates until a certain number of valid recordings (usually 10) were taken. Such a valid recording, referred to as a *trial*, was defined by a clean foot strike on each force plate while exhibiting a natural walking pattern. Five signals were recorded in total, consisting of three ground reaction force components (vertical, medio-lateral, and anterior-posterior forces) and the two directions of movement for the centre of pressure (COP) (see Figure 9 for example waveforms of all five signals).

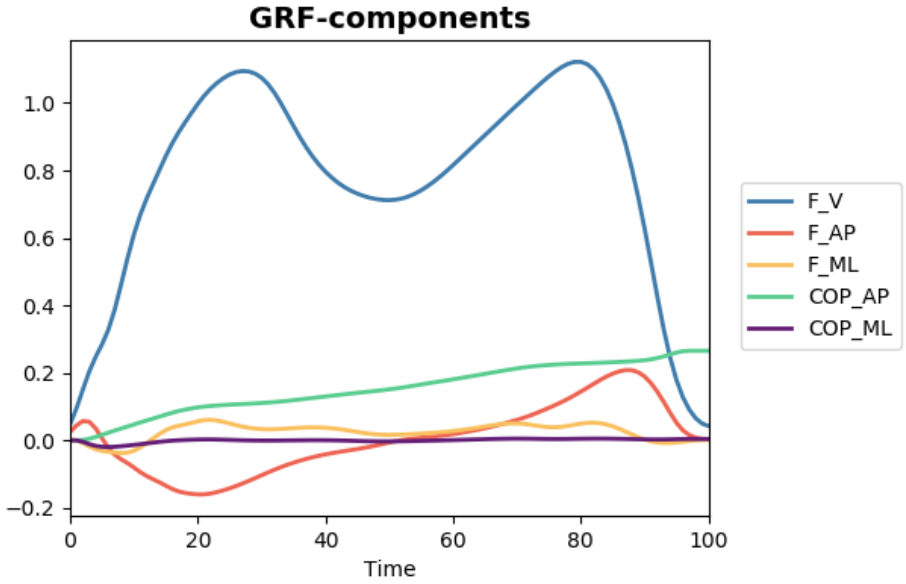

Figure 9: An example recording of all 5 components (processed version, see summary on page 34) taken from the dataset provided by Horsak et al. (2020). The data represents a single measurement taken from a healthy person (i.e. their shapes are representative for normal gait). F denotes the force components (V = vertical, AP = anterior-posterior, ML = medio-lateral) and COP the centre of pressure

Those five components were then transformed into digital signals using a sampling rate of 2000Hz and a 12-bit analogue-digital converter (DT3010, Data Translation Incorporation, Marlboro, MA, USA) with a signal input range of $\pm$10V. The available data was downsampled to 250Hz and a threshold of 35N was applied to all force components, with the COP being calculated afterwards. The data obtained in this fashion (denoted by the keyword *raw*, as no further processing was applied) is available as part of the *GaitRec* archive (Horsak et al., 2020).

The other part of the dataset was refined further, calculating the COP only when the vertical force reached 80N. Furthermore the medio-lateral coordinates were centered around the mean,

while the anterior-posterior ones were centered at zero (resulting in a normalization to % stance). All signals where then filtered by a 2nd order low-pass butterworth filter with a cut-off frequency of 20 Hz and temporally aligned to start with the initial contact and end with the toe off before time-normalizing them to 100% stance (i.e. 101 time-steps) by re-sampling. In order to provide comparability across individual subjects, the amplitudes of the force components were normalized to be expressed as multiples of the body weight by dividing them by the product of body mass times acceleration due to gravity. Additionally, Horsak et al. (2020) excluded outliers as well as sessions with less than three trials from the dataset.

The dataset was manually annotated by a well-experienced physical therapist, based on the available medical diagnosis of each patient, featuring one of five main categories (depending on the location of the injury). The following list provides an overview along with the most common injuries for each class:

- *Hip*: Fractures of pelvis/tight, luxation of the hip joint, coxarthrosis, total hip replacement
- *Knee*: Fractures of patella/femur/tibia, rupture of the cruciate/collateral ligaments or menisci, total knee replacement
- *Ankle*: Fractures of the malleoli/talus/tibia/lower leg, ruptures of the ligaments or Achilles tendon
- *Calcaneus*: Calcaneus fracture, ankle fusion surgery
- *Healthy control*: No injuries

Note that a more detailed annotation based on these injuries is available, but was not used in this research due to the focus on detecting the associated joint for a gait impairment, rather than the specific type of the disorder. The key points of the processed version of the dataset can thus be summarized as follows:

- Each measurement consists of five components: vertical, anterior-posterior and medio-lateral force components (F_v, F_ap, F_ml) and the COP movement in anterior-posterior and medio-lateral direction (COP_ap, COP_ml).
- All signals were temporally aligned (to 100% stance) and re-sampled to 101 points (i.e. time-steps).
- Measurements were taken for the right and left leg of two consecutive steps, resulting in data being available for the *affected* and *unaffected* leg (although, in some cases both legs are injured and this separation does not apply).
- A single recording consists of several *Trials* (i.e. multiple measurements) conducted in immediate succession, resulting in several instances of the same problem.

5. Methods

When developing a strategy to determine the best possible combination of neural network type and pre-/post-processing methods, it is important to take the characteristics of the the problem that should be solved into consideration. Most of the network architectures examined in this research, have been proposed for a task different then GRF-classification, while the majority of the data preparation steps are problem specific and have been evaluated on GRF-data before. This discrepancy makes a structured approach an absolute necessity in order to verify the impact of each step in the classification process on the final accuracy. The proposed method splits the workflow into several distinct parts, enabling the isolated evaluation of each component in order to find a good solution for the associated task. This procedure is illustrated in Figure 10 and the sections within this chapter are structured according to this general outline. Starting from the beginning of the graph (at the data selection step), the methods used in each component will be introduced and explained, detailing their influence on the classification process.

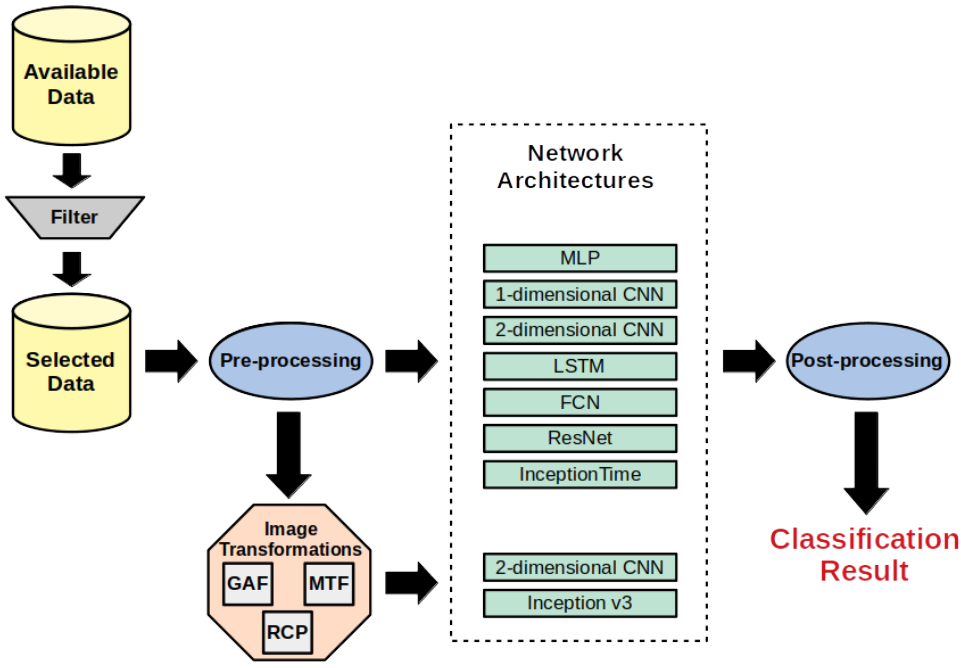

Figure 10: An outline of the general workflow for the classification of GRF-measurements used in this thesis. Each depicted component can be evaluated individually in order to estimate its influence on the achieved accuracy, revealing the combination(s) with best performance on the task.

5.1. Data Selection

As mentioned in Chapter 4, the *GaitRec* dataset used in this research consists of the three subsets *TEST*, *TRAIN* and *TRAIN-BALANCED*. While the latter has been prepared in a fashion that

removes possible biases from the data, the first two have not received the same treatment. Therefore, a dedicated data selection step is required, filtering out samples that could have a negative influence on accuracy. The following variables from the dataset have the potential of biasing the results if maintained during training and/or testing of the classifier:

- *Walking speed*: While measurements for patients were generally taken at a self-selected walking speed, this is not the case for the healthy control group. Their recordings are available at three different speeds (slow, fast and self-selected).
- *Orthopaedic shoes/insoles*: Some of the participants used such orthopaedic aids, known to have an impact on the walking patterns, resulting in possible distortions of the respective waveforms.
- *Readmissions*: Monitoring the individual progress of each patients, typically resulted in several measurements taken during the rehabilitation process, corresponding to different levels of severity for each injury. Therefore, the inclusion of such samples might blur the boundaries between the classes, as they are expected to become more similar to those of healthy persons.
- *Injuries on both legs*: Since Slijepcevic et al. (2018b) demonstrated that using the data of the *unaffected* side provides a valuable contribution to the classification accuracy, lower performance can be expected for this group, because that additional information is not available/different for such patients.

Furthermore, only one of the three datasets features an equal distribution of samples for all available classes. This is problematic, because imbalanced data is known to bias machine-learning algorithms towards the majority group (Krawczyk, 2016). Due to this behaviour, the classifier is more likely to perform well if trained on a balanced dataset, a result which has been confirmed by Slijepcevic et al. (2018a) for GRF-measurements. In addition, the biases mentioned earlier have been removed from the *TRAIN-BALANCED* dataset, meaning that it does not include different walking speeds, readmissions, patients wearing orthopaedic shoes/insoles or with injuries on both legs. However, those samples still exist within the *TEST* and *TRAIN* sets and care has to be taken when evaluating the classifier.

5.2. Pre-processing Methods

In the context of this thesis, pre-processing refers to four distinct steps conducted in order to prepare the data for being processed by a neural network. In principle, all of the steps listed in this section are entirely optional, but often result in better performance when applied.

5.2.1. Normalization

Because the activation of a single neuron within a neural network is computed as a weighted sum of the inputs (see Section 3.2), different data ranges can introduce a bias within the classification (Nayak et al., 2014). In a typical machine learning application, this is prevented by applying some form of normalization to the data, before forwarding it to the neural network. Even though the GRF-measurements used in this research have already been normalized with respect to the individual (i.e. expressed as multiples of the body weight), the signals still differ considerably in their range, with the vertical component typically exceeding the others by a factor of 5 (or larger, see Figure 9), suggesting that additional normalization is needed. Since the choice of normalization method has been shown to have an influence on the quality of the classification for time-series data Bhanja and Das (2018), the following two common techniques are investigated:

- **Min-max** normalization: The values are rescaled to be within the range $[a, b]$ by applying Formula 3:

$$\tilde{X} = a + \frac{(X - min(X)(b - a)}{max(X) - min(X)} \tag{3}$$

- **Z-score** normalization: The values are normalized to zero mean and unit variance, using the Formula 4 with μ as the mean and σ as the standard deviation:

$$\tilde{X} = \frac{X - \mu}{\sigma} \tag{4}$$

5.2.2. Resampling

Since GRF-measurements are produced by discretization of a continuous process, the question of how to determine the ideal sampling-rate in order to be able to accurately represent the original signal, arises. Because the sampling rate is directly proportional to the dimensionality of the data, it is also connected to computational efficiency. Higher-dimensional data demands more hardware resources for processing, resulting in a trade-off between speed and accuracy of the signal representation. Unfortunately, an accurate representation does not necessarily correlate with an accurate classification, because the information contained within the signal can be highly redundant. An example for different re-samplings of the vertical component of a GRF-measurement is illustrated in Figure 11, where a re-sampling $\tilde{X_k}$ of the original series X is defined as:

$$\tilde{X_k} = \{x_{0k}, x_{1k}, x_{2k}, \ldots, x_{\lfloor \frac{100}{k} \rfloor k}\} \tag{5}$$

In other words, to prepare a concrete example, the series $\tilde{X_2}$ consists of every second value contained in the original data. The main purpose of this process is to investigate the amount of data that needs to be maintained in order to obtain an accurate classification. Or, to be more

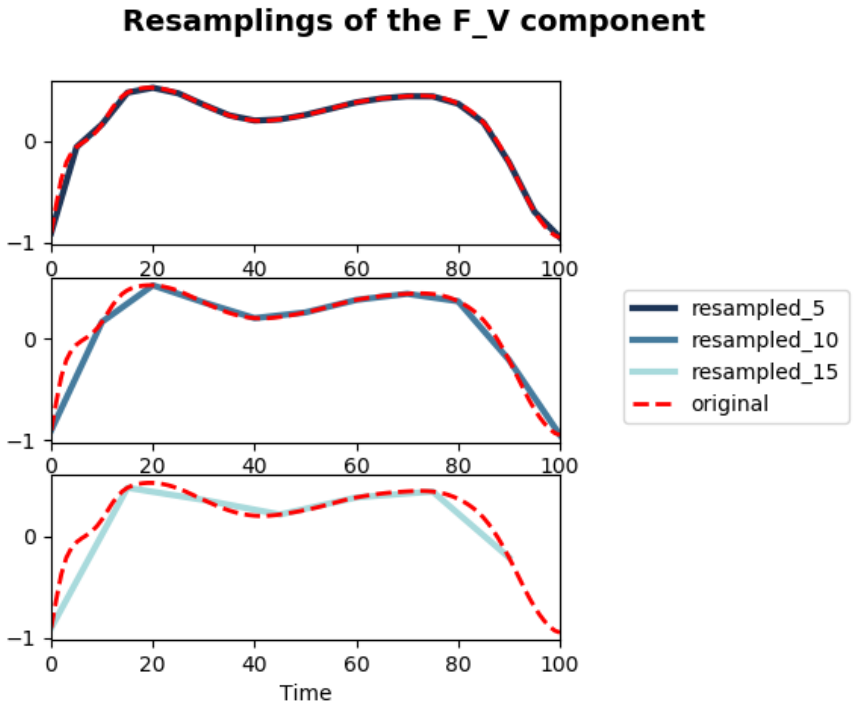

Figure 11: Comparison of the original data (101 points) and different resampling intervals. $resampling\_5$ indicates that only every 5th data-point is used to represent the signal. It can be observed that, while the amount of information lost increases with the length of the interval, the general outline of the curve is maintained, even if only 7 points remain.

specific, the task of determining the maximum k for all possible re-samplings $\tilde{X}_k$, so that the classification accuracy is not significantly decreased when compared to the original series X.

5.2.3. Aggregation of trials

Another factor well known to influence the accuracy of machine learning methods is the amount of data available. It has been stated by Roh et al. (2019), that not having enough training data is one of the major problems in this field. Determining the right amount of data, however, is not a trivial task and depends on many factors such as the classification problem itself, the complexity/dimensionality of the data and of course the neural network architecture. Fortunately, the dataset used in this research provides an easy way of obtaining more training-data by making use of individual trials.

As stated earlier, it is a common procedure to record several trials during a single gait analysis session. Several studies (Christian et al., 2016; Eskofier et al., 2013; Soares et al., 2016) propose calculating the mean waveform across all trials before further classification, to produce a more robust representation. Because the neural network tries to learn distinctive features for each class, this is a promising approach, helping to reduce the variability within the data. Nonetheless,

each trial represents a valid measurement and could be used individually if more training data is required. This would increase the number of available samples almost by a factor of 10 without the need to artificially create more samples by augmenting the data, possibly resulting in a more reliable classifier since all actual measurements are accounted for.

5.2.4. Ordering

The general form of representation for GRF-measurements is a $time\text{-}steps \times components$ matrix (or $time\text{-}steps \times 2 \cdot components$, if the data from both legs is used). As this study focusses primarily on utilizing data from both the affected and the unaffected side, the available matrix has a dimensionality of 101×10. When inputting this data into a CNN, the last dimension is generally referred to as *channels* and in the case of a 1-dimensional network, all of them are used in a single convolution operation. Since the network is able to learn and adapt the weights that are applied to each channel, the order of the components within the last dimension is irrelevant (i.e. the same weights are learned for a signal, independent of its position). This is an important characteristic since there is no inherent ordering (except time) between the components in a GRF-recording.

However, when using a 2-dimensional CNN, the sliding window does not only move along the temporal axis, but along the component axis as well. Hence, depending on the size of the window, the order becomes relevant, as it determines the selection of signals considered in the convolution. There is a wide range of possibilities on how to arrange the signals, but usually rectangular shapes are preferred (because otherwise the data would have to be padded with zeros). Since the number of available signals for the GRF-data is rather limited, the number of possible data layouts is restricted as well. For example, for the spatio-temporal 3D matrix proposed by Costilla-Reyes et al. (2018), the signals can only be arranged as either a 2×5 or 5×2 matrix, with the time-steps added as the last dimension. The following list outlines the possible layouts, alongside their characteristics and possible advantages for the classification:

- $101 \times 10 \times 1$: This format is the most flexible one, as any number of signals can be selected by adapting the window size.
- $101 \times 1 \times 10$: Using this layout should achieve similar results to the 1-dimensional case, as the height of the window needs to be equal to 1.
- $2 \times 5 \times 101$: This method, that encodes the time as the channels, was successfully applied to GRF-data by Alharthi and Ozanyan (2019) (although they had more signals available). Note that the order of the first two dimensions does not matter, as the same effect can be obtained by exchanging the *height* and *width* of the sliding window.
- $101 \times 2 \times 5$: The force components are still encoded as the channels, but they are split

into the affected and unaffected side (i.e. the legs can be considered separately or in combination).

- $101 \times 5 \times 2$: Similar to above but enables comparisons between legs for each individual signal).

5.3. Image Transformations

In order to benefit from the high accuracy that CNNs have been able to achieve on image classifications tasks for TSC purposes, the idea of transforming a time-series into an image was proposed (Gamboa, 2017). One of the first approaches was presented by Wang and Oates (2015a), but multiple different methods were developed since (Elias et al., 2015; Hatami et al., 2017; Garcia et al., 2020). The key idea behind this principle is simple: A time-series does not only contain information in time domain, but also, for example in frequency domain. If both domains are presented simultaneously, it can be visually represented as an image (Verstraete et al., 2017). Spectograms, commonly used to visualize audio signals are a good example of such a process. As Garcia et al. (2020) states, a good image encoding should be able to extract relevant patterns for the classification, while at the same time being resistant against noise and small perturbations. As such, it should be obvious that different encodings work well for different types of time-series. In the following three (four to be exact, but two of them are very similar) image transformations will be introduced that have demonstrated good classification accuracy on multiply datasets of the UCR archive (Wang and Oates, 2015a,b; Hatami et al., 2017) and therefore seemed promising to deliver good representations for GRF-measurements.

5.3.1. Gramian Angular Fields (GAFs)

The idea of Gramian Angular Fields has first been published by Wang and Oates (2015a), who later refined their idea further (Wang and Oates, 2015b) by introducing both the Gramian Angular Summation Field (GASF) and the Gramian Angular Difference Field (GADF) for TSC tasks. Both use the idea of a polar coordinates based representation to visualize the time-series by introducing the following transformations for a time series $X = \{x_0, x_1, \ldots, X_n\}$ for n real-valued observations:

- Rescale X so that all values are in the interval $[-1, 1]$ or $[0, 1]$ by

$$\tilde{x}_t = \frac{(x_t - max(X) + (x_t - min(X))}{max(X) - min(X)} \quad \text{or} \tag{6}$$

$$\tilde{x}_t = \frac{(x_t - min(X)}{max(X) - min(X)} \quad \text{respectively.} \tag{7}$$

- Transform into polar coordinates (ϕ, r) as follows:

$$\phi = arccos(\tilde{x}_t) \text{ with } -1 \leq \tilde{x}_t \leq 1, \tilde{x}_t \in \tilde{X} \text{ and } r = \frac{t}{N}, t \in \mathbb{N} \tag{8}$$

  where t is the timestamp and N is a constant factor to regularize the span of the polar coordinate system.

- Consider the trigonometric sum/difference between the points to identify temporal correlations:

$$GASF = [cos(\phi_i + \phi_j)] \text{ and} \tag{9}$$

$$GADF = [sin(\phi_i - \phi_j)] \tag{10}$$

- This creates the following matrix (only the GASF example is illustrated here, but the GADF is obtained in a similar fashion):

$$GASF = \begin{bmatrix} cos(\phi_1 + \phi_1) & \dots & cos(\phi_1 + \phi_n) \\ \vdots & \ddots & \vdots \\ cos(\phi_n + \phi_1) & \dots & cos(\phi_n + \phi_n) \end{bmatrix}$$

The whole process has been illustrated in Figure 12. Note that the different methods of rescaling will lead to different angular bounds. The interval $[0, 1]$ corresponds to the angular bounds $[0, \frac{\pi}{2}]$, while the values in the interval $[-1, 1]$ are bounded by $[0, \pi]$, providing different information granularity. For a detailed discussion on how to interpret the resulting images please refer to Wang and Oates (2015b), for the purpose of GRF-classification it is sufficient to know that temporal dependencies are preserved (since the time increases from top-left to bottom-right) and temporal correlations are contained because $GAF_{i,j||i-j|=k}$ represents the relative correlation by superposition/difference of directions of the considered time interval k.

On a final note, it should be mentioned that originally, only the GASF was used for image classification (Wang and Oates, 2015a), while the GADF was introduced later (Wang and Oates, 2015b) to be used in combination for enhancing accuracy. While some improvements were made in Wang and Oates (2015b) by using both transformations, it has to be remarked that accuracy was only increased slightly for most datasets. This suggests that the features extracted by GASF and GADF respectively are largely similar and the information gain by combining them is small. However, this was not investigated any further by Wang and Oates (2015b), making it impossible to deduce any further differences between those methods (e.g. if one of them is superior to the other).

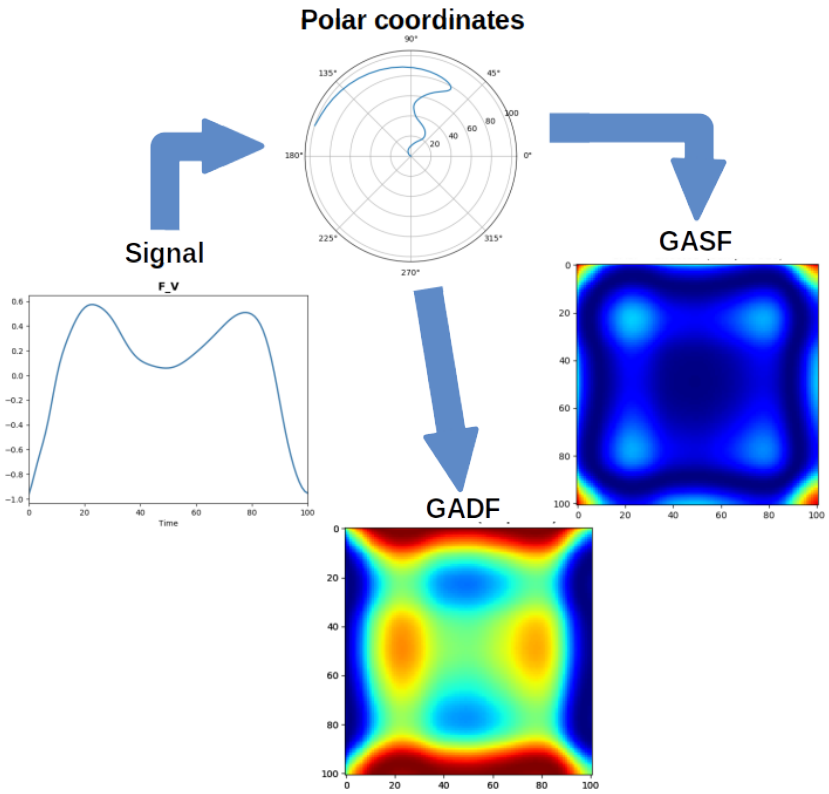

Figure 12: Converting a time-series into a Gramian Angular Field on the example of the vertical force component (f_v). In the first step, the time-series is transformed into the polar coordinate system and then the GASF and GADF are obtained by summation and subtraction respectively.

5.3.2. Markov Transition Field (MTF)

The Markov Transition Field is another idea introduced by Wang and Oates (2015a). It is obtained by representing the Markov transition probabilities sequentially in order to preserve information about time. Taking a time series X, the first step in this encoding consists of identifying a number of quantile bins Q and assigning each x_t to its corresponding bin q_j where $j \in [1, Q]$. Next, a $Q \times Q$ adjacency matrix W is created by counting the number of transitions along the time axis (in the manner of a first order Markov chain). In other words, $w_{i,j}$ gives the number of occurrences of the following event: *Point p_t from quantile q_i is followed by point p_{t+1} from quantile q_j*. The final Markov transition matrix is then created through normalization by $\sum_{i=1}^{Q} w_{i,j} = 1$.

From the Markov transition matrix, the $n \times n$ MTF matrix is created by spreading out the Matrix

W along the time time domain, resulting in:

$$MTF = \begin{bmatrix} w_{i,j|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{i,j|x_1 \in q_i, x_n \in q_j} \\ w_{i,j|x_2 \in q_i, x_1 \in q_j} & \cdots & w_{i,j|x_2 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{i,j|x_n \in q_i, x_1 \in q_j} & \cdots & w_{i,j|x_n \in q_n, x_n \in q_j} \end{bmatrix}$$

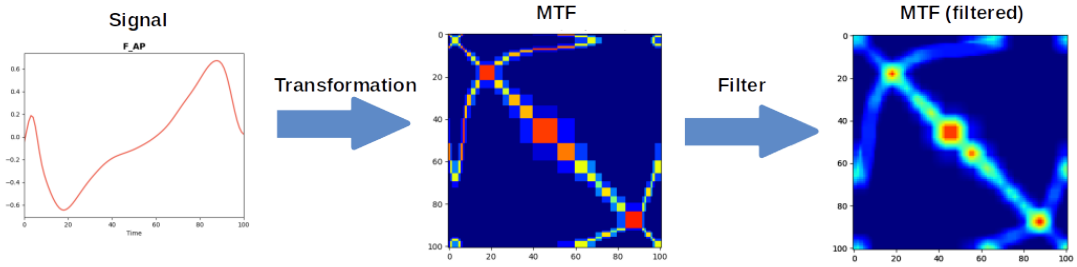

Figure 13: Converting a time-series into a Markov Transition Field on the example of the anterior-posterior force component (f_ap), using 20 quantile bins. After the MTF has been created, a 7×7 blurring filter is applied in order to make the edges smoother and obtain the final image.

Thus the value of a $MTF_{i,j}$ denotes the likelihood (as calculated in W) of a transition between the bin in which x_i is placed and the bin of value x_j. As such, the MTF matrix encodes the multi-span transition probabilities found in the time-series X and $M_{i,j||i-j|=k}$ gives the likelihood of a transition between points with the time interval k. The authors further suggest to apply a blurring filter to each non-overlapping $m \times m$ patch of the resulting image to make the size more manageable. For filtered or low-frequency time-series, this has the additional advantage of spreading out the information. This effect can be observed in Figure 13, illustrating the transformation process.

On the datasets tested by Wang and Oates (2015a), the classification based on MTF images generally achieves a better accuracy than the GASF variant, but unlike the example of GASF and GADF discussed earlier, a combination of both inputs enhances accuracy significantly. It seems like both variants extract different features from the time-series, that are able to complement each other when used together. Therefore the authors suggest to always use a combination of GASF and MTF (or GASF, GADF and MTF in their later paper) for image based TSC using CNN, by encoding the combined information into a multi-channel image (similar to how in a RGB image each pixel is represented by three values, storing the red, green and blue colour components separately, they propose a format consisting of a GADF, GASF and MTF value).

5.3.3. Recurrence Plot (RCP)

According to Hatami et al. (2017), time-series are characterized by a distinct recurrent behaviour (e.g. periodicities and irregular cyclicities). Typically, time-series are generated by dynamic nonlinear systems or stochastic processes, where the recurrence of states is a common phenomenon. Recurrence plots, introduced by Eckmann et al. (1987), take an existing time-series and transform it into a matrix of recurrences. This transformation is able to reveal the points in which some trajectories return to a previous state (Garcia et al., 2020). To obtain a recurrence plot from a time-series, the following steps are described by Eckmann et al. (1987) to be followed in practice:

- An embedding dimension d needs to be chosen to construct the d-dimensional orbit of x_t.
- Next, a time-delay embedding (τ) needs to be set to create the phase space trajectory.
- More specifically a state S_i is defined by $S_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(d-1)\tau})$

With those variables defined, each pixel in the recurrence plot is given by the following formula (Hatami et al., 2017):

$$RCP_{i,j} = \theta(\epsilon - ||S_i - S_j||), \quad S \in \mathbb{R}^m, \quad i, j = 1 \ldots K \tag{11}$$

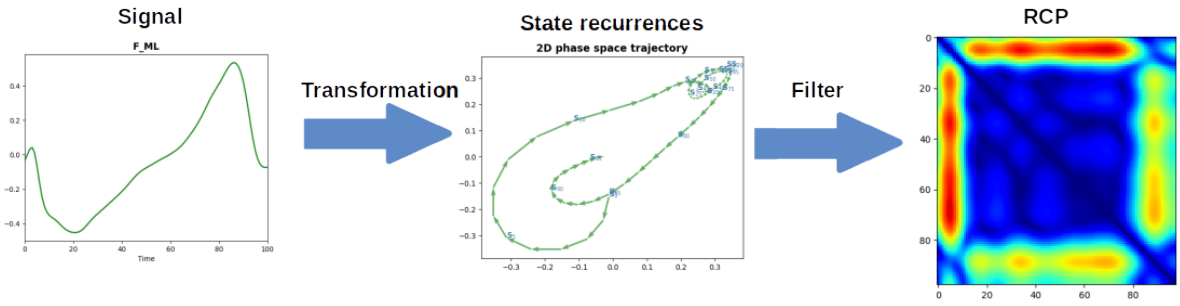

Figure 14: Creating the 2-dimensional phase space with the time-delay embedding $\tau = 3$ on the example of the medio-lateral force component (f_ml). The distance between those states is then used for creating the recurrence plot (according to formula 8). To make the image visually recognizable and avoid clustering, only every 5$^{\text{th}}$ state has been labelled in the middle plot.

In this equation, K is the number of (recurrent) states S that are considered, m is the dimensionality of each state and $||\cdot||$ is a norm (typically the euclidean norm is used). The parameter ϵ denotes a threshold distance and $\theta()$ is the Heaviside function, and their application will result in a binary image, containing both *texture* (dots, lines) and *typology* information (e.g. drifts periodicity, homogeneity). which can be used to analyse the time-series (Eckmann et al., 1987). Note, however, that the binarization used in this process results in the loss of information about the

time-series. Therefore, this step is commonly skipped in automated TSC using CNNs (Hatami et al., 2017; Garcia et al., 2020) resulting in the usage of the simplified formula:

$$RCP_{i,j} = ||S_i - S_j|| \tag{12}$$

Figure 14 illustrates the steps to be followed in order to obtain a recurrence plot using the medio-lateral force component as an example. Hatami et al. (2017) was one of the first to use the combination of RCP and CNN for time-series classification, revealing that this approach performs better than the combination of GAF and MTF proposed by Wang and Oates (2015a) (and other CNN architectures) on a certain subset of the UCR data.

All of the image transformations introduced above have the common disadvantage that they have been developed exclusively with univariate time-series in mind. To adapt them for the multivariate case, careful consideration is necessary because the standard technique of simply creating one long vector by concatenation is likely to fail. This process is expected to introduce jump discontinuities at the fusion points, resulting in artefacts that might dominate the resulting image. An example for such a behaviour is depicted in Figure 15, where the square-like patterns and the overall cross-shape are entirely due to jump discontinuities.

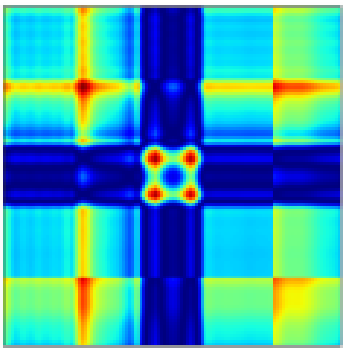

Figure 15: Result of concatenating all five GRF-measurements before the image transformation. Depicted is the resulting GASF for a randomly selected healthy patient.

While stacking images for each signal on top of each other (in a similar fashion to the approach used by Wang and Oates (2015b) to combine GASF, GADF and MTF) is one possibility, this will result in pictures with at least 5 channels (or even more if different transformations are combined) resulting in higher computation time. In order to alleviate that concern, the signals of a single measurement can be fused together in a continuous fashion by the following method:

Let x_{100} denote the last value of signal X and y_0 be the first value of the next signal Y. When concatenating X and Y the following transformation is applied to Y:

$$y_i = y_i + (x_{100} - y_0), i = 0, 1, \ldots, 100 \tag{13}$$

5.4. Neural Network Architectures

Because this thesis is of a comparative nature, its goal is to evaluate a number of different models on the *GaitRec*-dataset, identifying the ones with highest performance. For this purpose, the most successful networks, developed for time-series classification and evaluated on the UCR dataset (see Section 3.5, that have shown high accuracy for multivariate time-series have been selected. The following architectures will be implemented and compared:

1. **FCN**: As developed by Wang et al. (2016), making use of 3 convolutional layers.
2. **ResNet**: A residual network using the FCN as a building block for a deeper architecture. Proposed alongside the FCN by Wang et al. (2016)
3. **InceptionTime**: The most recent and complex variant with the best performance on the UCR dataset. Implemented as described byFawaz et al. (2019b) but without building an ensemble.

The ensemble-based approach has been omitted because it would introduce a bias into the final comparison. Since it can be assumed that all architectures are equally sensible to the random initialization and stochastic procedures, using an ensemble for just a single network would provide an unfair advantage. Since it was computationally infeasible to conduct such a broad comparison with ensembles for each type of network, it was decided to just use the basic form of each model.

Apart from this already established approaches, an attempt is made to build a comparable solution out of the basic components introduced in Chapter 3, expanding on those concepts by stacking more (or different) layers on top of each other, finding the most suitable combination for GRF-classification. On top of this structural changes, each type of layer additionally features a certain number of parameters that have to be determined as well. Naturally, this leads to a high diversity in the networks providing many options for the task. The network components and parameters implemented and compared in this thesis, are introduced in the following sections.

But before delving deeper into the details, an import concept when assessing the performance of such networks needs to be established. The term *overfitting* is used to describe a phenomenon commonly observed when training neural networks, where the model starts to recognize characteristics of the individual samples instead of generalizable features representative for a certain class. This results in a high accuracy on the training-data that is not transferable to other datasets (e.g. the validation- or test-set). During the training process this is indicated by an increase in the loss-function for the validation-set, while the one for the train-set keeps decreasing (see Figure 19 & 21 in Section 7 for a reference). Because this behaviour is generally undesirable, neural networks aim to reduce the influence of *overfitting* as much as possible, by making use of some

specialized techniques detailed in the next section.

5.4.1. Basic Architectures

The most basic architecture evaluated in this research is an MLP (see Section 3.2 for an overview). Due to its low complexity, the number of modifications and parameters is quite limited. The following options are explored for the MLP:

- *Layers*: The number of hidden layers within the network.
- *Neurons*: The number of neurons used in each hidden layer. The value can differ between layers within the same network.
- *Batch-normalization*: This concept has been introduced by Ioffe and Szegedy (2015), in order to optimize the learning process within a network. It is usually added after each layer to re-normalize (by re-centering and re-scaling the data to zero mean and unit variance) the output according to the currently processed batch of samples. In theory, this enables faster and easier learning for the next layer, as the variability of its input is reduced.
- *Dropout*: A method that has been proposed by Srivastava et al. (2014) to prevent *overfitting* of the network, by randomly "dropping out" (temporarily removing a neuron from the network together with all its connections) a certain percentage (i.e. the so-called *dropout-rate*) of the nodes.

Because most models implemented in this thesis use some kind of MLP for the final classification (usually with a single hidden layer), the parameters given above are used for those architectures as well, with the dropout-rate being independent from dropout that might have been applied to previous layers. Consequently, the number of parameters is increased for those networks, adding to their perceived complexity. In the case of convolutional layers (as used by the CNN) the additional variables given below need to be specified:

- *Layers*: The number of convolutional layers within the network.
- *Kernel-size*: The size of the sliding window used in each layer. Can be either 1- or 2-dimensional, depending on the type of the convolution.
- *#Filters*: The number of filters (i.e. different views on the data) applied in each layer.
- *Pooling*: A feature reduction layer applied after the convolution. It takes a number of values within a sliding window (defined by *pool-size*) and aggregates them depending on the type of the pooling operation. Max-pooling (i.e. taking the maximum value) and average-pooling (taking the mean) are investigated in this research.
- *Separable convolution*: Only applicable for 1-dimensional CNNs. The convolution is divided into one operation across time-domain (applied first, for each component individ-

ually), followed by a convolution across signal-domain.

- *Strided convolution*: The convolution is not applied at every point, but instead skips some values. For example, when using a stride of three, the convolution is applied to the first, forth, seventh (and so on) value.
- *Dilated convolution*: The sliding window is dilated, applying the convolution to non-consecutive values (e.g. every second value would be considered for a dilation-rate of 2).
- *Skip-connections*: Concatenating the features extracted by each layer to the original input before passing them on to the MLP (i.e. both, the original data as well as the new representations are considered for classification).
- *Batch-normalization & Dropout*: Same as for the MLP, with the dropout for CNN and MLP operating independently.

Furthermore, it was investigated whether an approach similar to Liu et al. (2019), basically consisting of two convolutions, one for each dimension (i.e. time and signal) of the data , would work well with the given data. For this purpose, a two-layered 2-dimensional CNN was employed, restricting one of the dimensions of the kernel in each layer to one (i.e. $(x, 1)$ in the first and $(1, y)$ in the second layer). Thus, a convolution along the temporal axis is computed first, followed by a convolution in signal domain in the next layer. Apart from those restrictions, all other parameters are applied as listed above, for determining the most accurate network. This approach will be denoted as **2DCNN-1DKernels** in the remainder of this thesis.

Performance of these models is compared to the two convolutional architectures proposed for GRF-measurements by Alharthi and Ozanyan (2019):

- **GRF-1D**: This network consists of four convolutional layers using a kernel-size of 2, doubling the number of filters in each layer starting from 12 (i.e. 12, 24, 48 and 96). Each layer is followed by max-pooling with a pool-size of 2, and a dropout of 50% after the final pooling operation. The output is then analysed by an MLP featuring 50 neurons and a dropout rate of 20%
- **GRF-2D**: Three 2-dimensional convolutions (each with a kernel-size of (2, 2) are applied using 12, 24 and 48 filters, each layer followed by average pooling with a pool-size of (2, 2). This block is being followed by a dropout of 50% and an MLP consisting of 100 neurons.

It is important to note that the 2-dimensional network employed by Alharthi and Ozanyan (2019) uses a spatial encoding of the signals, by arranging them in a 2-dimensional form, encoding the time-steps as the channels of the series. Because, only a limited number of signals are available in the *GaitRec* dataset, two variants of the network were implemented. In the first one,

the network is not modified, giving up on their special data arrangement (i.e. using the same $time\text{-}steps \times signals$ representation as the other architectures). The second variant, referred to as **GRF-2D-modified** reorders the data into a tensor of shape $2 \times 5 \times 101$, with the first dimension corresponding to the leg, the second to the signals and the last to the time-steps. The network was adapted to use only two layers (instead of three) with kernel-sizes of (2, 1) and (1, 2) respectively, while (all other settings remained the same.

5.4.2. LSTM Architectures

An LSTM layer can be applied with or without and added MLP, adding the following parameters to the ones previously introduced:

- *Layers*: The number of LSTM layers within the network.
- *Units*: The number of memory units used in each LSTM layer. Note that the length of the time-series puts an upper limit on this value because it is not useful to try to remember more time-steps than present in the series.
- *Dropout*: Same as for the MLP, but operating independently.
- *Recurrent Dropout*: The same concept of dropout, but being applied to the recurrent connections within the LSTM.

Again, the performance of these networks is contrasted to a model successfully used by Alharthi and Ozanyan (2019) to classify GRF-measurements:

- **GRF-LSTM**: This architecture uses two stacked LSTM layers, the first one consisting of 100 and the second one of 40 units, employing both dropout and recurrent dropout of 20% in each layer. This is followed by batch-normalization and a dropout of 50% before employing an MLP with 20 neurons.

In addition to such basic LSTM architectures, different combinations of LSTM and CNN are explored as well, as they might be able to yield better accuracy (see Section 3.4). In such a case, all of the parameters listed previously were investigated in the following two settings:

1. *Serial architecture*: In this setting, CNN and LSTM are applied in serial, with the LSTM layer following after the convolutional layers. In other words, the LSTM is used to analyse the temporal dependencies between the additional features extracted by the CNN and might thus learn a better representation of the whole series.
2. *Parallel architecture*: CNN and LSTM are employed independently, concatenating their outputs before the final classification (done by an MLP). The MLP is than able to consider the features learned by both types of architectures, possibly increasing accuracy even further.

Finally, the combination of LSTM and CNN proposed by Karim et al. (2018a) is investigated as well, due to the high accuracy it achieved on the UCR dataset (outperforming several pure CNN approaches) and a specific adaption for multivariate time-series (Karim et al., 2018b) being available. Their architecture is based on the convolutional network developed by Wang et al. (2016) (being explained in Section 3.5) used for extracting short-term dependencies and an LSTM applied in parallel to examine the long-term features. Similar to the concept above, their results are concatenated and classified by an MLP afterwards. Despite the high accuracy achieved, the usage of the LSTM in their model remains heavily disputed, because it is preceded by a dimensional shuffle (basically transposing the time-series), applying the LSTM to the signal dimension instead of along the temporal axis. As the functionality of the LSTM in such a situation is highly obscure, its usefulness was investigated in a follow-up paper Karim et al. (2019), revealing that the network including the dimensional shuffle performs better than without. While this provides some insights on the performance of the approach, the actual reasoning remains unclear. The authors speculate that this is due to the LSTM (without dimensional shuffle) and CNN part extracting the same information form the signal, thus not being able to benefit from each other.

In a later study (Karim et al., 2018b), this architecture was extended for multivariate time-series by adding a *squeeze-and-excite* block (Hu et al., 2018) after the first two convolutional layers. According to the authors, this is essential to enhance performance for multivariate time-series, because it adaptively recalibrates the learned feature maps. As such, the whole process can be considered as form of a self-attention mechanism applied to the output of the preceding layers. The block's main functionality consists of capturing the dependencies across signals, trying to identify and put special attention to the more important ones. In other words, it is a more sophisticated attempt of extracting inter-correlations between signals, that outperformed their previous approach on the tested multivariate datasets.

While evaluating this kind of architecture, in the following referred to as **LSTM-FCN**, seems promising due to its high performance on time-series data, the obscurity remaining about the actual contributions of the different components is slightly concerning. Therefore, several types of this network are examined by enabling/disabling the dimensional shuffle and/or the *squeeze-and-excite* extensions.

5.4.3. Image Architectures

Because the form of the input is fundamentally different after an image transformation is applied to a time series, special architectures are required in order to classify them. Wang et al. (2016) proposed the usage of tiled convolutional networks while Hatami et al. (2017) used a simple two-layered 2-dimensional CNN for the same task, achieving comparable results. Ac-

cording to Wang et al. (2016), tiled convolution has been adapted because of its ability to learn invariant features and the reduced requirement for labelled training data. However, the dataset chosen for this research provides a sufficient amount of labelled instances and the invariances learned (predominantly scale and rotational, see (Ngiam et al., 2010) for more details) are most likely irrelevant for the given data. Taking a look at the examples given in Section 5.3, it can be observed that the features are distinctly different from traditional image classification (e.g. no objects and edges) making the benefit of more complex feature extraction techniques questionable. Therefore, the network described by Hatami et al. (2017), will be the primary reference for this research:

- **IMG**: A two-layered 2D-CNN using a kernel-size of (3, 3) with 32 filters followed by max-pooling with a pool-size of (2, 2) and dropout (25%) in each layer. The output is then processed by a MLP with 128 neurons and dropout of 50% before the final *output*-layer.

This network has been originally developed for univariate time-series, but can be easily adapted to the multivariate case by increasing the number of *channels*, depending on the signals and number of image transformations used for the classification. Even though it would be possible to apply this prodecure to other well established image-recognition CNNs too, those networks are usually built with 3-channel images (i.e. RGB-format) in mind and would have to be re-trained from scratch to adapt to a different input format, making their effectiveness questionable (because they have been designed and evaluated for a different task).

In order to avoid those problems and leverage state-of-the-art image classification networks, Karimi-Bidhendi et al. (2018) propose a different technique for multivariate time-series. Their method consists of creating a $n \times n$ transformed image (more specifically, their research is devoted to the GADF-transformation) for each signal k, concatenating them vertically (or horizontally) to create a single image with $height = kn$ and $width = n$ (for the vertically concatenated case). They continue by taking a pre-trained version of a well established image classification network (Google's Inception v3 to be exact), removing the final output layer, using it solely for feature extraction. This new set of features (in this case a vector of length 2048) is then used as an input to a classical MLP, with three hidden layers, making this part the only one that is actually trained. Final testing on the UCR archive (for the univariate case) and several multivariate series reveals that their architecture shows incredibly performance (considering the results obtained by Wang and Oates (2015a), where GAF and MTF had to be combined to be competitive) outperforming all ensemble-classifiers and evaluated neural networks.

The main characteristics of this approach, in the following referred to as **IMG-Inception** when applied to GRF-measurements are:

- Each signal is transformed to an image separately, but the outputs are arranged to form

one single, larger image (i.e. images are placed right next to each other).

- Features are extracted by passing the image through the Inception v3 network (with the final classification layer removed). The model is already pre-trained (i.e. the weights are already set).
- Because the Inception v3 network is trained on RGB-images, the input to that architecture must have three channels, requiring a conversion of the image to RGB based on a pre-defined mapping.
- An MLP with three hidden layers (800, 400 and 100 neurons), each followed by batch-normalization and a dropout of 50%, is used to classify the output obtained from Inception v3.

5.5. Post-processing Methods

The aggregation of trails has already been mentioned in Section 5.2.3, but there exist further methods besides using the mean of each waveform to combine their information. Slijepcevic et al. (2020) discuss a so called *late fusion* approach (the name is derived from the fact that the aggregation takes place **after** the actual classification, in contrast to *early fusion* approaches such as taking the mean waveform), based on majority voting. This idea is based on the observation that there is a high variability within the individual trials and since the mean is known to be sensible to outliers, such measurements might lead to a distortion of the results. Therefore, Slijepcevic et al. (2020) suggest to use the classifier to predict each individual trial instead, combining the resulting class labels into a single final one by calculating the statistical mode. Their results indicate that this approach is somewhat superior to *early fusion*, achieving an accuracy of 61% (compared to 58.9% for the mean waveform) on a SVM.

According to their reasoning, this difference is due to the partial removal of available input information at an early stage (when *early fusion* is applied) resulting in a less robust classifier. Whether or not this hypothesis is applicable to neural networks as well, is investigated as a part of this thesis by the methods discussed in Section 5.2.3, while this part is solely dedicated to examining the effects of majority voting itself. The method proposed Slijepcevic et al. (2020) only considers trials with a likelihood of at least 40% for one of the five classed when producing the final label, thus reducing the negative influence of ambiguous trials. As such, it implements a very simply weighting procedure, differentiating between trustworthy and untrustworthy trials. Therefore, the benefits of majority voting can not be simply limited to the training process, but might be comparable to an ensemble-based approach, using slightly different inputs instead of different classifiers.

6. Experimental Setup

All experiments were conducted in the programming language Python (version 3.7.7) using the *Keras* functional API available in TensorFlow 2 for implementing the different architectures. More specifically, TensorFlow 2.2.0 was used, built with support for NVIDIA®CUDA (version 10.0.130) in order to use the GPU (graphics processor unit) for faster neural network calculations via the *cuDNN* library. The codebase created to support this research is publicly available and can be freely downloaded from *github*[1] for easy reproducibility. Additionally, the best configurations (and weights) for each evaluated model have been saved and are available as part of the same archive. In order to make the experiments as transparent as possible, (hyper-)parameter searches were performed using the *Weights & Biases API*[2], providing a graphical visualization of the results for easy comprehension. The majority of this searches (referred to as a *sweep* by the API) is publicly available from the project page[3], offering an opportunity to investigate the accumulated data.

Training and testing of the neural networks was conducted on a server provided by Media Computing Research Group from the *FH St. Pölten* using a x86 64-bit architecture powered by an Intel®i7-6900K CPU. Additionally, the system was outfitted with a NVIDIA®GeForce GTX 1080 graphics card as an accelerator, that has been used for the majority of the neural network calculations. All results reported have been obtained using data available as part of the *GaitRec* dataset (Horsak et al., 2020), which is freely available for further research.

6.1. Data preparation

From the two different versions of the data available in the *GaitRec* archive, only the processed dataset (i.e. not the one denoted by *raw*, see Chapter 4) was used in all experiments. This choice was made because the normalization applied aims to remove differences between individuals, where otherwise it would be difficult to compare two waveforms based on their numerical values. Thus, the neural network does not need to take such differences into account and can focus solely on the task of differentiating between classes.

However, due to some characteristics of the dataset and/or the evaluated classifier, some additional preprocessing steps were necessary. The standard procedure outlined below has been followed in all experiments unless it is **explicitly** noted otherwise:

1. Removal of all data associated with other walking speeds than self-selected. Patients that were not wearing normal shoes or walking barefoot were excluded from the data

[1] `https://github.com/delta-leader/GRF-classification`
[2] `https://www.wandb.com`
[3] `https://wandb.ai/delta-leader/ground-reaction-force-classification`

(i.e. patients using orthopedic shoes or insoles were removed, including those where the footwear was not reported). Only initial measurements were taken into account, resulting in exclusion of all but the first measurements per patient/admission.

2. If the *affected* side was not indicated in the dataset (e.g. healthy persons, patients with injuries in both legs), it was determined randomly for each session. Once its role was established, it was maintained during all experiments.
3. All trials for a single gait analysis session were aggregated in an *early fusion*-approach by calculating the mean waveform of each signal.
4. The resulting data was normalized to be within the range of $[-1, 1]$, for each signal individually. The normalization was performed globally using the minimum and maximum value of a given dataset in a component-wise fashion by applying Formula 3. However, since the *TEST*-set should not be touched until the final evaluation (Horsak et al., 2020), it can not be used to determine the parameters of the normalization. Therefore the *TEST* was normalized according to the parameters obtained from the training-set (usually the *TRAIN-BALANCED* dataset). This process can potentially result in values outside of the specified range, causing problems for certain methods (e.g. GAF-transformation). In such a case, the values were clipped to fit the expected range after normalization.
5. The force components were represented in the resulting tensor in the following order: F_v, F_ap, F_ml, COP_ap and COP_ml.
6. If concatenation of the signals was required (e.g. for the MLP), it was done in a continuous fashion by applying Formula 13.
7. The data for the *affected* and *unaffected* side was processed individually and then concatenated along the last dimension of the output tensor, doubling the dimensionality along that axis.

6.2. Evaluation

As mentioned in Section 4 the *GaitRec*-data is already split into a *TEST*, *TRAIN* and *TRAIN-BALANCED* set, where the last one is a subset of *TRAIN* with an equal amount of samples for each class. After applying the standard preprocessing procedure, the following numbers are obtained for each dataset:

1. *TEST*: 688 recorded sessions (containing the average signals of 5,408 trials)
2. *TRAIN*: 1,234 recorded sessions (containing the average signals of 10,386 trials)
3. *TRAIN-BALANCED*: 730 recorded sessions (containing the average signals of 6,400 trials)

To assess the performance of a model when determining its best parameters without using the *TEST* set, the data was split further into a train- and validation-set with the latter containing 20% of the data originally available in the *TRAIN-BALANCED* set. All experiments to verify specific architectures and fine-tuning models by determining their hyper-parameters were conducted on this split of the data. Therefore 484 sessions were available for training and 146 sessions were contained in the validation-set. Unless it served the purpose of answering a specific research question (i.e. Does the inclusion of patients wearing orthopaedic shoes reduce classification accuracy?) all models were trained on this training/validation-split and exceptions will be denoted **explicitly**.

All models were trained and evaluated using the following settings:

- *Loss-function*: Categorical crossentropy was applied, given by the following function:

$$Loss = -\sum_{i=1}^{\#classes} y_i \cdot log(\tilde{y}_i) \tag{14}$$

  Where $\tilde{y}_i$ is the i-th scalar value in the model output and y_i is the corresponding target value.

- *Metrics*: Accuracy, meaning the performance is given by the number of correct predictions divided by the number of samples.

- *Optimizer*: Adam optimizer (see Kingma and Ba (2017) for reference), defined by the parameters $learning\text{-}rate$, β_1, β_2, $epsilon$ and $amsgrad$.

Reproducibility of the experiments was ensured by setting the random number generator to the same value before training each neural network and only deterministic operations were used by setting the environmental variable *TF_DETERMINISTIC_OPS*. If, for some reason, such operations were not available for a specific architecture, the result is reported as the average of five runs.

6.2.1. Evaluation of individual architectures

The typical evaluation pipeline for a specific architecture consisted of the following three stages:

1. Determination of the optimal parameters for an architecture. This was only applied for the models designed in this thesis and omitted for pre-defined networks such as **FCN**, **ResNet** and **InceptionTime**.
2. Selection of the best hyper-parameters according to the procedure described in Section 6.2.3.
3. Evaluation of the model on the *TEST* set using the ideal settings determined in the steps above.

Step 1& 2 were performed using the training/validation split obtained from the *TRAIN-BALANCED* dataset, by selecting the best model according to the highest accuracy it achieved on the validation set. The training process in step 1 (prior to hyper-parameter determination) was standardized across architectures to guarantee comparability, using the following parameters:

- *Batch-size*: 32, defining the number of samples seen before the weights are updated.
- *Epochs*: 100, the number of iterations over the whole dataset (i.e. each sample is processed 100 times).
- *Optimizer*: Adam optimizer with a learning-rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-07}$ and $amsgrad = False$.

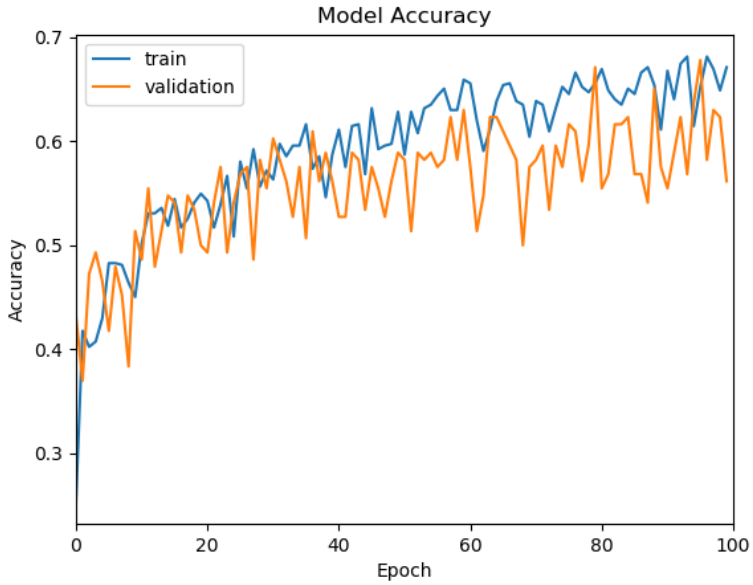

Figure 16: Accuracy during training for train- and validation-set. It can be observed that alterations in the train-set usually induce even larger changes in the validation-set, resulting in strong fluctuations. Therefore, the best accuracy on the validation-set is usually achieved before the training finishes (i.e. before 100 epochs).

Note that the models were compared according to the highest accuracy achieved in the validation-set, which usually happened before 100 epochs were reached (see Figure 16). In order to determine the optimal parameters for each of the architectures introduced in Section 5.4.1 & 5.4.2 two types of searches across the parameter-space were performed:

1. *Grid-search*: In a first step, the basic settings were determined by trying different combinations of layers and neurons, kernel-sizes or units for MLP, CNN and LSTM respectively.
2. *Bayesian-optimization*: A search across all parameters was performed using the Bayesian optimization process described by Snoek et al. (2012) and implemented in *Weights & Biases*.

The specific settings for the grid-searches executed for each architecture are summarized in the list below:

- **MLP**: $layers = \{1, 2, 3\}$, $neurons = \{10, 20, \dots 500\}$
- **1DCNN**: $layers = \{1, 2\}$, $kernel\text{-}size = \{2, 3, 5, 7, 9, 11, 15, 21, 33\}$, $\#filters = \{8, 16, 32, 64, 128, 256, 512, 1024\}$
- **2DCNN**: $layers = \{1, 2\}$, $\#filters = \{8, 16, 32, 64, 128, 256, 512, 1024\}$ $kernel\text{-}size = (x, y)$ with $x \in \{2, 3, 5, 7, 9, 11, 15, 21, 33\}$ and $y \in \{2, 3, 4, 5, 6, 7, 8, 9\}$
- **LSTM**: $layers = \{1, 2\}$, $units = \{10, 20, \dots 100\}$

For the searches employing the Bayesian-optimization process, the following values were set (depending on the architecture, see 5.4.1):

- $layers = \{1, 2\}$
- $neurons = [20; 300]$
- $batch\text{-}normalization = \{True, False\}$
- $dropout = [0.0, 0.5]$
- 1D: $kernel\text{-}size = [2; 20]$
- 2D: $kernel\text{-}size = (x, y)$ with $x \in [2; 20]$ and $y[2; 10]$
- CNN: $\#filters = \{8, 16, 32, 64, 128, 256, 512, 1024\}$
- CNN: $pool\text{-}type = \{Max, Avg, None\}$
- 1D-CNN: $pool\text{-}size = [2; 5]$
- 2D-CNN: $pool\text{-}size = (x, y)$ with $x \in [2; 5]$ and $y[2; 4]$
- 1D-CNN: $separable\text{-}convolution = \{True, False\}$
- CNN: $skip\text{-}connections = \{True, False\}$
- 1D-CNN: $stride = [1; 5]$
- 2D-CNN: $stride = (x, 1)$ with $x \in [2; 5]$
- 1D-CNN: $dilation = [1, 20]$
- 1D-CNN: $dilation = (x, y)$ with $x \in [1; 20]$and $y[1; 3]$
- LSTM: $units = [20; 100]$
- LSTM+CNN: $mode = \{parallel, serial\}$
- LSTM-FCN: $dimensional\text{-}shuffle = \{True, False\}$
- LSTM-FCN: $squeeze\text{-}and\text{-}excite = \{True, False\}$

Note that in the specifications given above $dilation$ and $stride$ are mutually exclusive, therefore, in case of the 1D-CNN, if $dilation \neq 1$ then $stride = 1$ is implied and the other way around. Furthermore, the input for all models using 2-dimensional convolution was provided in the $101 \times 10 \times 1$ format for the reasons given in Section 5.2.4, sometimes requiring slight adaptions of the values listed above. For example, to assure that the combination of stride and pool-size did not exceed 10 in the second dimension. Additionally, when evaluating the **2DCNN-1DKernels** architecture the the pool-size was restrict in the same way as the kernel-size (i.e. to (x, 1) in the first and (1, y) in the second layer) to reflect the intention of the network.

Overall, applying this procedure resulted in an exhaustive parameter search, training and testing more than 1,000 models per architecture. The performance of each model was evaluated on the validation-set, generally selecting the model with the highest accuracy for further comparison on the *TEST* set.

6.2.2. Evaluation of image transformations

The calculations performed when transforming a time-series into an image require a number of parameters to be determined beforehand. First of all, there is the problem of aggregating the different components. While concatenation would work (a method to avoid the resulting jump discontinuities is described by Formula 13), it was not investigated any further in this research, due to the expected bias introduced to the transformation. Therefore, each signal was transformed separately for the purpose of this thesis.

Furthermore, the ideal settings for each conversion were established, either analytically or empirically. The corresponding free variables to each transformation are listed below:

- *GAF*: The only adjustable parameter is the data range (i.e. either $[-1, 1]$ or $[0, 1]$). Since this only affects the granularity of the information, not much difference is expected for the GRF-data.
- *MTF*: The optimal number of quantile bins ($bins$) and the size of the blurring kernel ($blur\text{-}size$) need to be determined.
- *RCP*: Both, the embedding dimension d, as well as the time-delay τ, need to be chosen. But, in contrary to the *MTF*, some methods are available for estimating these parameters (refer to Wallot and Mønster (2018) or Kantz and Schreiber (2003) for a detailed explanation). The recommended approach is based on the Average Mutual Information (AMI) function and the False Nearest Neighbour (FNN) function, but can only be applied to univariate time-series. Therefore a grid search across the results of the estimations for each signal was performed to determine the optimal values.

The optimal values for those parameters were determined empirically, using the **IMG** network

in order to determine how much information (with respect to the classification task) was maintained in the resulting image. Each transformation was considered individually and the parameters yielding the highest accuracy on the validation-set were chosen to be used for the corresponding transformation in the remainder of this research. The optimal values together with a brief description of how they have been obtained are reported in the following list:

- *GAF*: Better performance was observed using a data range from $[0, 1]$, resulting in an accuracy increase of approximately 3% and 7% for GADF and GASF respectively when compared to a range of $[-1, 1]$.
- *MTF*: Different values for the number of quantile bins have been tested, starting from 10 and increasing by five up to a value of 50, with the highest score being obtained by using 25 bins. Adding an additional blurring filter to the process, lead to an accuracy enhancement of +3% with a blurring kernel of shape (2, 2). Further quadratic kernels were tested (up to a value of 10), but no higher increases have been observed.
- *RCP*: Using the AMI and FNN functions, the search space for the embedding dimension was determined to be $d = \{2, 3, 4\}$ while a delay of $\tau = \{2, 3, 4, 5, 7, 8\}$ was calculated. With an euclidean distance metric employed, the best combination of $d = 2$ and $\tau = 3$ was found on the validation set. It is noteworthy that re-normalization to an interval of $[0, 1]$ led to a slight boost of +0.60%. This is important for analysing the combination of images, as it guarantees that the results of all transformations are within the same range.

Due to the fact that Wang and Oates (2015a,b) report better performance if different image transformations are combined, that area was investigated as well. Due to the slightly smaller size of the RCP (98×98 - when using the settings determined above, compared to 101×101 for the other images), all images have been resized to 98×98 when used in combination. Furthermore, to assure that each image is considered equally important by the classifier, the RCP was normalized to $[0, 1]$, to match the range of the other formats.

When deployed on the **IMG** architecture, all images were concatenated along the last dimension, resulting in a tensor of shape $98 \times 98 \times 5$ (or $98 \times 98 \times 10$ if the data from both legs is used) when using the RCP. Different transformations were concatenated along this dimensions as well, producing a $98 \times 98 \times 20$ tensor if two images are combined. In contrast, the **IMG-Inception** architecture requires a spatial layout, where the signals were arranged along the horizontal axis, while the data for the different sides and transformations was aligned horizontally, giving a shape of 392×490 for the combination of two transformations. An example of such a layout using all four available images is depicted in Figure 17

For the feature extraction via the Inception v3 network as employed by the **IMG-Inception** architecture the pre-trained version of that model (using the *imagenet*-data) provided by *Keras*, was used. However, since that network requires its input to consist of three channels (i.e. a

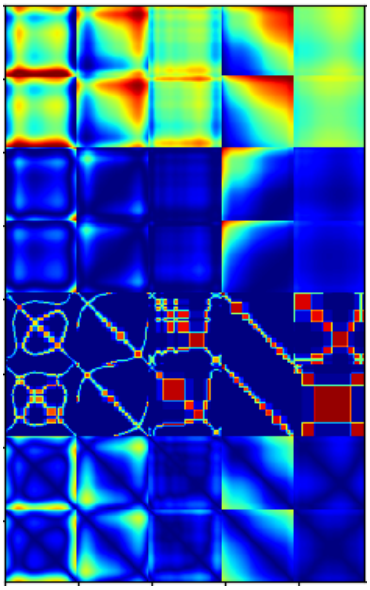

Figure 17: Data layout when using all four image transformations as well as signals from the affected and unaffected side as input to the **Inception-IMG** architecture. The resulting image has a width of 490 and a height of 784 pixels.

RGB-image) an RGB-conversion has to be applied. Because the details for this process are omitted from the original paper (Karimi-Bidhendi et al., 2018), the mapping was done via the *jet*-colormap available from the *matplotlib* library, producing similar output to the one depicted in the actual publication.

6.2.3. Fine-tuning of architectures

Before evaluating a specific model on the *TRAIN*-set, an attempt was made to optimize its performance by determining its ideal hyper-parameters (e.g. learning-rate, batch-size, epochs, etc.). For this purpose, the same Bayesian optimization process as described in Section 6.2.1 was employed, searching the hyper-parameter space according to the following settings (using the *adam*-optimizer as specified in Section 6.2):

- $batch\text{-}size = [8; 512]$
- $learning\text{-}rate = [0.0001, 0.01]$
- $beta_1 = [0.5, 0.99]$
- $beta_2 = [0.6, 0.999]$
- $amsgrad = \{True, False\}$

Both networks used for classifying the time-series transformed to images (i.e. **IMG** and **IMG-Inception** were excluded from this process, reporting their results using the standard procedure detailed in Section 6.2. This has been necessary because the models are evaluated on different

input combinations and fine-tuning it to one transformation (e.g. GADF) was revealed to be deteriorating to the other representations. For example, a decrease of accuracy by 50% was observed if the ideal hyper-parameters obtained from optimization using GADF images were employed with the input being changed to GASF instead. Since the basic settings perform comparatively well (the accuracy increase on the GADF was only 3%), no further attempts on fine-tuning have been made, because the impact on other transformations is impossible to predict. In order to compensate for this lack of fine-tuning, at least partially, both models were trained for a longer period of time, using 150 epochs instead of the default 100.

A comprehensive overview of the hyper-parameters used for each architecture is provided in Table 1, with the numeric values being rounded to four decimal digits. Note that two sets of values are provided for the **FCN** architecture. The *original* set of hyper-parameters corresponds to those used by Wang et al. (2016) in their evaluation on the UCR dataset, while the *tuned* values are the result of searching the hyper-parameter space based on GRF input data. This has been done in order to provide a contrast against the other networks that have been excessively fine-tuned on the GRF-measurements, indicating whether or not there are any issues with generalizability.

Architecture	**Adam-optimizer**					**Train-parameters**	
	learning-rate	β_1	β_2	ϵ	*amsgrad*	*batch-size*	*epochs*
MLP30	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
MLP90	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
MLP2	0.0019	0.9346	0.7534	$1e^{-07}$	*False*	92	293
MLP-D	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
1DCNN-strided	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
1DCNN-dilated	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
2DCNN-dilated	0.001	0.9	0.999	$1e^{-07}$	*False*	32	100
2DCNN-1DKernels	0.0046	0.7660	0.8063	$1e^{-07}$	*True*	130	181
LSTM	0.0012	0.7892	0.8694	$9.0589e^{-07}$	*False*	357	111
FCN (tuned)	0.0056	0.7027	0.9612	$1e^{-08}$	*True*	23	222
FCN (original)	0.001	0.9	0.999	$1e^{-08}$	*False*	32	100
ResNet	0.0045	0.6774	0.8609	$1e^{-08}$	*True*	18	210
InceptionTime	0.0063	0.817	0.6258	$1e^{-08}$	*False*	38	136
IMG	0.001	0.9	0.999	$1e^{-08}$	*False*	32	100

Table 1: Hyper-parameters for all models participating in the final comparison. Note that the **FCN** is twice, once for the standard and once for the fine-tuned version. Parameters were rounded to 4 decimal digits.

6.2.4. Pre- and post-processing methods

Unfortunately, due to the nature of those procedures restricting some of them to certain architectures, different evaluation settings are needed as well. While the proposed methods for nor-

malization and aggregation can, in principle, be applied in any setting, care has to be taken when assessing the effects of re-sampling or ordering. Since re-sampling reduced the number of data-points available, the parameters for networks such as CNNs or LSTMs, that are inherently depended on the dimensionality, would have to be adapted as well. The ordering, on the other hand, only has an impact on models using 2-dimensional convolution (see Section 5.2.4) and can be ignored for all other architectures. Therefore, the evaluation procedure differs depending on the investigate method.

For assessing the influence of different **orderings** (i.e. input formats), an adapted **2DCNN-1DKernels** architecture was employed to assess each of the different input layouts. In other words, a two-layered network was used, with the convolution being restricted to time domain in the first and to signal domain in the second layer (with the exception of the $101 \times 1 \times 10$ format, where a convolution along the second dimension is meaningless and only a single layer was deployed). A restricted version of the parameters search described in Section 6.2.1 was conducted for each layout, without considering *stride* or *dilation* and limiting pooling to dimensions with a size greater than two. The impact of the input formats is then assessed by a comparison of the highest accuracy achieved on the validation-set.

Reducing the number of data points via **re-sampling** is expected to have a significant impact on most of the architectures that have been optimized for a signal length of 101 time-steps. Therefore, the effects of this method are investigated using the MLP (because none of its parameters depend on a certain layout of the data), and the unoptimized **FCN**. This network has been selected to represent the convolutional approaches because it has been designed with the UCR dataset in mind, therefore it should be somewhat robust against inputs of various lengths in its unmodified form. The classification accuracy of the re-sampled GRF-measurements is contrasted against the performance of the original one on the validation-set, in order to assess how much information is maintained within the signals.

For studying the effects of **normalization** and **aggregation of trials**, all networks studied in this research are employed, with the exception of the **IMG** architecture when investigating normalization methods. Since the image transformations employ some kind of normalization by themselves, either directly (GASF, GADF) or indirectly (MTF, RCP), they should be robust against different input ranges, and will not be examined in this context. The different methods are compared with respect to their training-, validation- and test-accuracy contrasting **min-max** and **z-score** normalization against the original data (i.e. as provided in the *GaitRec* archive). In order to enable a fair comparison, the *TEST* data was normalized using the parameters obtained from the *TRAIN-BALANCED* dataset for each of the methods respectively.

The different ways of aggregating the individual trials recorded during the same gait analysis session were analysed on the same three sets. However, a re-training of the models is required when using all trials, resulting in two networks for each architecture and two input formats de-

noted by *mean* for the mean waveform and *allTrials* for the individual trials. The evaluation is performed in two steps. First the the accuracy for the models trained with *allTrials* is calculated and compared with respect to the two input formats (since the models trained on the mean waveform have already been evaluated when comparing architectures). This is a pure pre-processing step, with the accuracies reported for each individual sample that is provided to the classifier. In the second step, the influence of post-processing is determined by utilizing those models to predict input that has not been aggregated yet and performing majority voting to obtain the final label for each session.

The voting process itself is similar to the one applied by Slijepcevic et al. (2020), considering only trials with a likelihood of at least 40% for one of the five classes during the procedure, in order to reduce the negative influence of ambiguous trials. However in rare cases (for certain architectures) no such trial could be found for some specific sessions. In such situations, the threshold was reduced to 20%, basically considering all trials for the process. The statistical mode was calculated for all trials fulfilling that precondition, thus producing the final class label. If two (or more) classes had the exact same number of votes, the final decision was made based on the class probability (i.e. the class with the highest probability across all valid trials was selected). It is important to note that during this process, the validation- and test-set were normalized according to the parameters the model has been trained on (i.e. either the mean waveform or all trials), because otherwise performance would be degraded significantly.

6.3. Effects of filtering and class aggregation

As part of the investigations conducted to provide more insights into the problem of classifying gait disorders according to body locations itself, the distinctiveness for each class is explored. For a "good" classification, the overlap between classes should be small, as this indicates that they can be separated reasonably well. This class analysis is conducted via a confusion matrix, where each row represents the predicted class labels, while the column indicates the actual class. Thus, sensitivity, specificity and precision can be calculated, providing an in depth view of the classifier.

Additionally, the confusion matrix obtained from using the data of both legs is contrasted to the one from just the affected leg. It has been attested by Slijepcevic et al. (2018a,b), that accuracy can be improved by combining that data and this procedure promises to provide a more detailed anlaysis of the advantages. For example, as of now, it is still unclear whether or not all classes are able to benefit equally from the reference provided by the unaffected leg. Such insights might be useful when trying to build a better classifier, revealing information about the data itself. Since the results obtained by Slijepcevic et al. (2018a,b) indicate that some impairment locations might be harder to distinguish from each other than from other classes due to their

similarity, such differences should be investigated.

As another consequence of their research, inter-class discrepancies could also be examined by combining classes that are hard to separate into a single, broader class. Their results indicate that considerably higher accuracy could be achieved by that procedure, due to obtaining more distinct categories. Two artificial classes are created in order to assess the impact of such an approach, each of them fusing two of the main injury locations together:

- *Upper Leg (UL)*: Combines the *hip* and *knee* class to form a single entity, representing the upper joints (i.e. above the shank).
- *Lower Leg (LL)*: Union of the classes located below the shank, consisting of *ankle* and *calcaneus*.

For those experiments, the architecture achieving the highest accuracy on the newly proposed classification problem (i.e. *Healthy - UL - LW*) is selected by adding up the amount of correct predicted for their contained sub-classes. This architecture is then retrained on the three class problem with the same data, revealing whether or not those two problems are equally difficult to solve. A good classifier might be able to take advantage of the fused classes, extracting better features and therefore increasing accuracy.

As far as the filtering of the data is concerned, experiments were restricted to the best performing neural networks on the *TEST* set, because re-training on the whole *TRAIN* set is required. The *TRAIN-BALANCED* set can not be used because the included samples do not feature the examined characteristics as mentioned in Section 5.1. For each of the three presumed biases (i.e. orthopaedic shoes/insoles, readmissions and injuries in both legs), two models were trained, including and excluding the respective characteristics. These models were then evaluated on the corresponding filtered/unfiltered versions of the test-set in order to assess their impact. While being rather limited and not nearly enough to fully understand the expected biases, this approach should be sufficient to determine if it is possible to make better use of the available data. Since the accuracy of neural networks typically increased if more data is provided, and the *GaitRec* archive contains much more data than currently used for training, this seems worthwhile to investigate, possibly revealing future directions of research.

7. Results & Discussion

The results of all experiments are presented and discussed in this chapter. Analysis is generally conducted on the independent *TEST* set and the validation-set, with the exception of attempts conducted to select the optimal parameters for a certain neural network architecture. As the usage of the *TEST* set for determining the ideal settings would inevitably introduce a bias to the result (possible hurting their generalizability), such experiments were limited to the validation-

set and therefore only those values can be reported. Due to the large number of architectures (and parameters for each) investigated in this research, comparison on the *TEST* set was limited to only a selected few representatives for each type of network, in order to maintain computational feasibility. For easier understanding and to provide a comprehensive structure to this chapter, those results will be presented first, following the general outline below:

- Comparison of different network parameters for the architectures as explained in Section 6.2.1
- Comparison of image transformations
- Comparison of network architectures
- Comparison of pre-processing methods
- Influence of post-processing via majority voting
- Influence of data selection
- Insights about class separability

7.1. Neural network parameters

This section is dedicated to the results obtained when testing different parameters for the architectures described in Sections 5.4.1 & 5.4.2 and structured according to the four neural network concepts MLP, 1-dimensional CNN, 2-dimensional CNN and LSTM. The best settings for each type of architecture are determined and appropriate candidates for further comparison on the *TEST* set are selected by evaluating their accuracy on the validation-set.

7.1.1. MLP

An overview of the results for the grid search performed across the number of neuron in an MLP with a single hidden layer is reported in Table 2 (restricted to the most relevant outcomes). The highest accuracy of **67.81%**) has been achieved using 90 and 440 neurons respectively, with the lower number being preferable due to the reduced computational effort. Starting from 30 neurons the network achieves a stable performance of about 65% fluctuating by approximately $\pm 2\%$, revealing no further correlations between accuracy and the number of neurons.

The influence of all other investigated parameters on the performance of the MLP, as revealed by the experiments, are summarized below:

- *Hidden Layers*: Generally, increasing the number of hidden layers did not yield a higher accuracy. However, a single two-layered model (out of 361 tested ones) was able to beat the threshold of 67.81%, achieving **69.18%** accuracy on the validation set.

# Neurons	Best Validation-Accuracy
20	61.07%
30	65.07%
50	63.01%
60	65.07%
70	67.12%
90	**67.81%**
110	65.75%
130	66.44%
150	65.75%
170	65.75%
190	67.12%
200	63.70%
⋮	⋮

Table 2: Highest accuracy achieved on the validation set when training a MLP with a varying number of neurons in the hidden layer for 100 epochs.

- *Batch-Normalization*: Adding a batch-normalization layer after each hidden layer had a positive effect on accuracy.
- *Dropout*: While dropout often improved accuracy, it did not necessarily lead to a better performance, obtaining mixed results across the experiments. Results seem to suggest that smaller dropout-rates are favourable, but even with higher values competitive accuracy has been observed.

Overall, the best performance on the validation-set across all evaluated settings has been achieved by a single-layer MLP using 253 neurons with batch-normalization and dropout (ca. 25%), resulting in an accuracy value of **71.92%** after training for 79 epochs. However, that particular model exhibits strong fluctuations in its loss function for the validation-set (see Figure 18), suggesting that maybe it just "happened" to come across a good solution for the current input. This seems to be a general disadvantage of most architectures including dropout, hinting at the existence of many local optima within the optimization space.

Four candidates were selected for further comparison on the validation set, featuring the following parameters/characteristics:

- **MLP30**: A single layer MLP using just 30 neurons, representing the minimal approach that achieves good results.
- **MLP90**: An MLP using 90 neurons, representing the best accuracy achieved with by a single layer architecture without dropout.
- **MLP2**: The only two-layered architecture achieving better performance than the single-

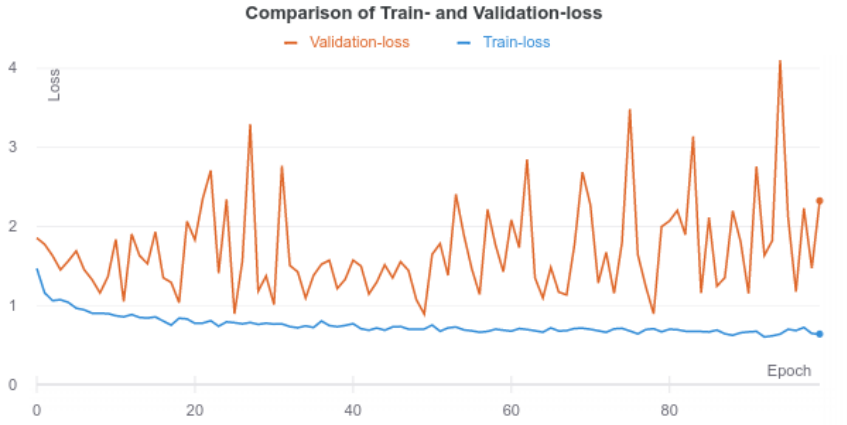

Figure 18: Loss accumulated during training of a single layer architecture with 253 neurons and dropout (ca. 25%). It can be observed that while the loss for the training-data decreases steadily, there is a high fluctuation in the validation-loss, making it likely that the model just "guessed" a good solution instead of actually learning it.

layered ones, using 50 and 80 neurons, without additional modifications (i.e. the best model without dropout).

- **MLP-D**: The best model using dropout, consisting of 253 neurons, batch-normalization and dropout with a rate of roughly 24.68%.

Hyper-parameter searches have been performed for the **MLP-D** and **MLP2** only, in order to contrast their performance against the non-tuned variants **MLP30** and **MLP90**. Thus, by evaluating them on the test-set it becomes possible to gauge the influence of fine-tuning on the generalization ability of the MLP. By selecting the optimal hyper-parameters, accuracy of the **MLP2** could be improved by about +1% while no further increase was observed for the **MLP-D**, indicating that it already discovered an optimal solution.

Finally, it should be mentioned that (in contrast to the methods that will be discussed later in this chapter), the MLP hardly *overfits* the training-data. As can be seen from Figure 19, displaying the loss and accuracy for the **MLP2** network, both curves seem to flatten around the same time, with the difference in loss being negligible. In other words, even though a longer training period induces a lower training-loss, the amount is insignificant and does not improve accuracy any further (on neither the training- nor the validation-data). In fact, for the **MLP2** variant of this network, the accuracy on the validation-data generally remains within a 5-10% distance of the training-accuracy.

7.1.2. 1-dimensional convolution (1D-CNN)

Figure 20 depicts a comprehensive summary of the grid search performed for a 1D-CNN with a single convolutional layer. Generally, this architecture does not seem to perform much better

Figure 19: Comparison of loss and accuracy for both the training- and validation-set obtained from using the **MLP2** architecture and training for a period of 100 epochs. Above: Loss. Below: Accuracy.

than the single-layer MLP, but for certain combinations of kernel-sizes and number of filters, a higher accuracy can be observed. The best result is achieved by using 128 filters and a kernel-size of 3, scoring **71.23%** accuracy, after executing training for 24 epochs. It should be mentioned that the 1D-CNN is very prone to *overfitting*, reaching close to 100% accuracy on the training-data after 60 iterations, with the loss function for the validation-set actually starting to increase again from around the 20^{th} epoch. This phenomenon is illustrated in Figure 21.

Further experiments revealed the following consequences of parameter modifications:

- *Additional 1D-CNN-layers*: No improvements (with one exception) were made when increasing the number of layers.
- *Batch-Normalization*: Applying Batch-Normalization after the convolution seems to reduce the best accuracy achieved by a network.
- *Dropout*: Results for applying dropout (both after the convolution and the MLP) remain mixed. If anything experiments indicate that lower values work better.
- *Kernel-sizes*: Smaller kernels seem to achieve better results (except for dilated convolu-

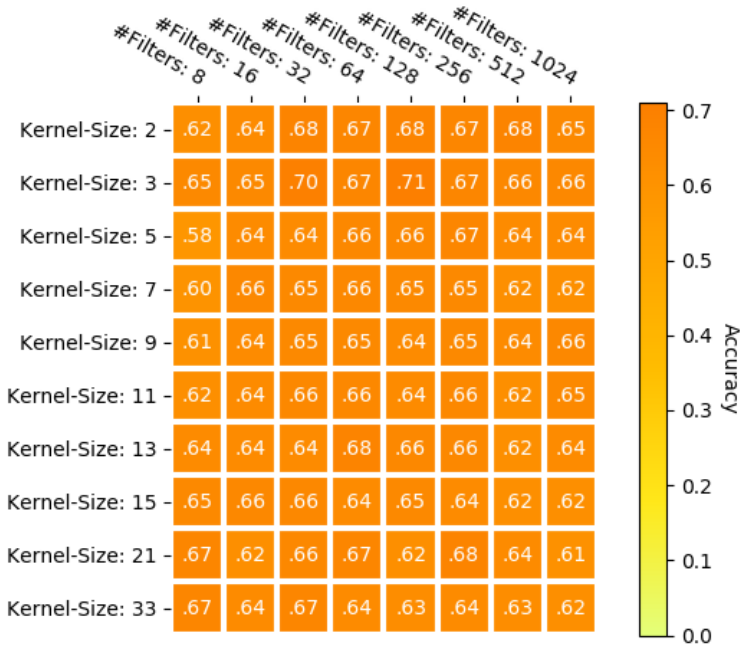

Figure 20: Overview of the best accuracy achieved on the validation set using a network consisting of a single 1-dimensional convolutional layer followed by a MLP with 90 neurons. Showing the results for various kernel-sizes and numbers of filters applied.

tion, where the opposite is the case).

- *#Filters*: Less filters appear to work better in the first layer, while the number can be increased for successive ones.
- *Neurons in the hidden layer*: Using a higher number of neurons tends to increase accuracy up to a certain degree.
- *Pooling*: Did not have much of an impact on single-layer networks, but increased accuracy for two or more layers, independent of the pool-type (i.e. for both max- and average-pooling) and pool-size.
- *Separable convolution*: Mixed results have been observed, but effects, if they exist are small, and the total impact seems to be negligible.
- *Strided convolution*: Helped to increase accuracy, especially when used in combination with *skip-connections*.
- *Dilated convolution*: Enhanced accuracy for most cases, but results indicate that the dilation-rate should not be too high.
- *Skip-connections*: Had a positive effect on all networks except for the two-layered dilated variant.

Overall, the two best performing architectures are the following, scoring **73.97%** accuracy on

Figure 21: Comparison of loss and accuracy for both the training- and validation-set obtained from using a 1-dimensional CNN layer ($kernel\text{-}size = 3$, $\#filters = 128$) followed by an MLP-layer consisting of 90 neurons. Above: Loss. Below: Accuracy.

the validation set each:

- **1DCNN-strided**: A single convolutional layer with a kernel-size of 5, 175 filters and using a stride of 5. Max-pooling is applied after the convolution, using a pool-size of 4, followed by a dropout with a rate of ca. 38.78%. The result is combined with the original data using skip-connections before inputting it into an MLP consisting of 185 neurons and dropout of about 31.55%.
- **1DCNN-dilated**: The only two-layered network that performed better than all dilated single-layer versions. The first layer uses a kernel-size of 18, 131 filters and a dilation-rate of 12, while the second one has a kernel of size 6, 31 filters and a dilation-rate of 14. Both layers are followed by a max-pooling operation (with pool-size = 3) and a dropout of 14.20%. The result is then analysed by an MLP using just 46 neurons and a high dropout-rate of 43.70%.

The common variable in both networks is a high dropout after the MLP, which prevents them

from *overfitting* the training data (see Figure 21), achieving their highest accuracy after 93 and 88 epochs respectively. In addition, both networks seem to benefit from analysing long-term patterns within the time-series, extracted by using either a large stride or a high dilation-rate, as well as a significant feature reduction by the max-pooling layer(s).

Finally, it should be reported that the hyper-parameter search performed for those two architectures, did not yield any gains in accuracy. This might signify that instead of determining the best architecture in general, the models were tuned to the specific set of learning parameters used in the experiments. However, as can be observed in Figures 21 & 19, the validation-accuracy generally tends to fluctuate in a range of $\pm 5\%$, making it unlikely that other architectures outperform the selected ones due to a superior learning ability. This seems to be especially true when compared to the performance of the **GRF-1D** model employed by Alharthi and Ozanyan (2019), which did not exceed an accuracy of 67.12%, even after extensive fine-tuning.

7.1.3. 2-dimensional convolution (2D-CNN)

Overall, the achieved results of the 2D-CNN resemble those of the 1-dimensional case, being very vulnerable to *overfitting* as well. Because of the large search area, only the highest accuracies are reported for this network. However, to get a little more context for judging the general performance of this model (especially in comparison to the 1D-CNN), the results for all experiments using a kernel-size of 11 along the temporal axis are depicted in Figure 22. Of the 384 tested variants (using only one convolutional layer), only two settings have been able to outperform the top result achieved by a standard 1D-CNN by a margin of about +0.6%. These two architectures use a a kernel-size of (2, 2) and 16 filters or (11, 6) and 8 filters respectively, resulting in an accuracy of **71.92**%.

The search across the parameter space generally revealed the same correlations and relationships as for the 1-dimensional variant with a few noteworthy details:

- While adding a second convolutional layer did not improve accuracy for normal and strided convolution, it did generally yield better results for the dilated case.
- The best results were achieved on a single-layer network with strided convolution (**73.97%**) using a kernel-size of 9 in the signal dimension, which is almost identical to a 1-dimensional convolution. Incidentally, the top four strided networks all used a kernel-size of either 8, 9 or 10 as their second dimension, suggesting that there is no advantage to a 2-dimensional convolution.
- **73.97%** accuracy was also achieved by a two layer dilated network using a kernel-size of (14, 5) in the first and (3, 3) in the second layer.
- The observed influences of batch-normalization, #filters, neurons, pooling and skip-connections

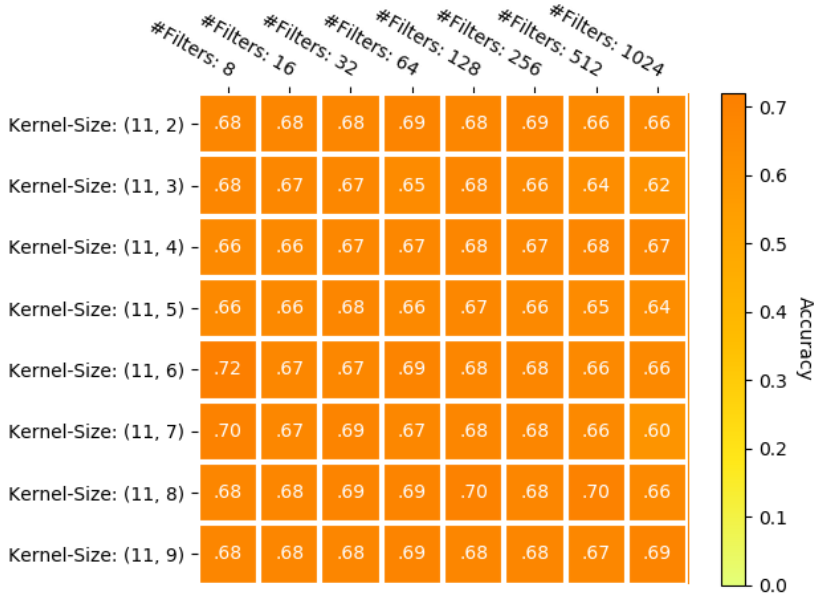

Figure 22: Example of the grid search performed to determine the best accuracy on the validation-set for a network consisting of a single 2-dimensional convolutional layer followed by a MLP with 90 neurons. Showing the results for various kernel-sizes (first dimension fixed to 11) and numbers of filters applied.

were identical to the 1D-CNN.

- *kernel-sizes*: Mixed results were observed for the first dimension (corresponding to time), with the strided variant tending to favour larger values, while the dilated version preferred smaller ones. For the second dimension (i.e. signal-domain) both of them benefited from larger kernels.

Beyond the basic variant, experiments conducted on other architectures relying on 2-dimensional convolutions yielded the following results:

- **2DCNN-1DKernels**: Generally working well, the highest obtained accuracies for this approach are slightly lower than the ones reported above, with a maximal value of 72.60%. However, fine-tuning this network by selecting the ideal hyper-parameters put it one equal footing with the others, scoring **73.97%** as well.
- **GRF-2D**: Using the unmodified network from Alharthi and Ozanyan (2019) resulted in poor accuracy never reaching more than 65.58% accuracy on the validation-set.
- **GRF-2D-modified**: Better results were obtained by the modified variant using the proposed spatial encoding of the signals, with a maximum accuracy of 71.23% being observed.

Results seem to indicate that the 2D-CNN cannot outperform its 1-dimensional counterpart, achieving identical highest accuracy values. Furthermore, judging by the good performance of

the **GRF-2D-modified** model, the input layout seems to be of secondary importance, but this will be investigated in more detail in Section 7.4. Since the best strided variants of this architecture are very similar to the 1D-CNN, they were not considered any further because almost identical results are expected. Thus the candidates for further testing are:

- **2DCNN-dilated**: The network consists of a (14, 5) kernel using a dilation-rate of (18, 1) and 92 filters in the first layer and a kernel of size (3, 3) with a dilation-rate of (9, 2) and 134 filters in the second layer. Max-pooling, with a pool-size of (2, 3) and a dropout of ca. 30.03% was applied after each convolutional layer, followed by an MLP with 154 neurons an a dropout of approximately 27.66%.
- **2DCNN-1DKernels**: Composed of two layers with a kernel-size of (4 ,1), using 63 filters and (1, 4), using 20 filters respectively, this model employs batch-normalization, max-pooling of shape (5, 2) and a dropout of about 20.37% after each layer. The outputs of all layers are then concatenated by skip-connections before analysing them with an MLP (162 neurons) followed by batch-normalization and a dropout of 19.47%.

As a last note, similar to the 1-dimensional networks, no further performance increase was observed when attempting to fine-tune the hyper-parameters for the **2DCNN-dilated** model. This might imply the existence of a threshold within the given dataset, possibly indicating that either *(a)* an optimal solution has been obtained that cannot be improved any further or *(b)* the CNN is not able to extract the needed information from the data. In any case, it can be observed that several architectures are able to achieve identical accuracies close to 74%, but none of them are able to actually surpass that value.

7.1.4. Long short-term memory (LSTM)

The results for the grid search performed to determine the optimal number of memory units for a single-layered LSTM are presented in Table 3.

The best performance of **65.07%** is achieved by an LSTM using 50 units, with most of the other observed values being either 2% or 4% lower. Using an additional MLP layer after the LSTM seemed to have an negative impact on the performance, while using dropout was generally beneficial, with a lower dropout-rate being preferable. By stacking two LSTM layers on top of each other a slight accuracy improvement of +1% was observed and the **GRF-LSTM** model proposed by Alharthi and Ozanyan (2019) was able to outperform all other pure LSTM approaches by scoring **69.86%** on the given task. The superiority of this architecture could be due to the employed batch-normalization layer, which has not been investigated for the other networks, but more research would be necessary to support that theory.

Extending the LSTM by adding convolutional layers generally yielded better performance for all

# Units	Best Validation-Accuracy
10	52.05%
20	60.27%
30	61.64%
40	61.64%
50	**65.07%**
60	61.64%
70	63.70%
80	61.64%
90	63.70%
100	63.01%

Table 3: Highest accuracy achieved on the validation set when training a LSTM with a varying number of memory units for 100 epochs.

types of convolution and in both parallel and serial settings. Most interestingly, while the dilated convolution seemed to work better in a serial architecture the opposite could be observed for the strided variant. However, the highest accuracy for both types was obtained in a parallel setting scoring up to 71.23% (strided) and 72.60% (dilated). Since those results are actually lower than the ones obtained by the 1D-CNN itself, it can be concluded that the combination of CNN and LSTM ist not able to enhance each others performance. It does seem like the features extracted by these two architectures are highly redundant, making the combined approach ineffective on the given dataset.

This hypothesis is further strengthened by the result obtained for the **LSTM-FCN** architecture (Karim et al., 2018a,b). The maximum accuracy observed when testing different parameters for their model was 71.23%, with experiments revealing that neither the dimensional shuffle nor the *squeeze-and-excite* blocks were actually beneficial for the performance. On the contrary, a negative impact on accuracy for both can be reported when classifying GRF-measurements.

Since the LSTM seemed to be unable to enhance accuracy in combination with a CNN, only one candidate, representing a pure LSTM architecture was selected for further comparison:

- **LSTM**: As slightly modified version of the architecture specified by Alharthi and Ozanyan (2019), using the same general layout but with an adapted dropout-rate of ca. 0.44 instead of 0.5 after batch-normalization and before the MLP, resulting in an accuracy of **69.86%** after training for 79 epochs (with fine-tuned hyper-parameters).

7.2. Image Transformations

The performance of the different image transformations (and combinations thereof) for both the **IMG** and **IMG-Inception** network can be seen in Table 4. The highest accuracy on the

validation-set is achieved by the GADF transformation, scoring **69.18%**, followed by the RCP with 67.12%. All other image transformations and most interestingly also the different combinations perform considerably worse. Especially the MTF does not seem to be able to extract much useful information for the classification task, achieving only 58.22% accuracy. Furthermore, in contrast to the findings of Wang and Oates (2015a,b), the different formats do not seem to complement each other, with the accuracy of the combinations never outperforming the highest accuracy of its individual components. Either the contained information is largely redundant or the high dimensionality due to stacking signals and formats (20 channels for a combination of two image transformations) severely limits the learning ability of the network.

Images	Training-Accuracy		Validation-Accuracy	
	IMG	IMG-Inception	IMG	IMG-Inception
GADF	97.95%	99.14%	**69.18%**	49.32%
GASF	90.41%	97.95%	63.70%	48.63%
MTF	99.32%	98.29%	58.22%	45.89%
RCP	97.78%	98.63%	67.12%	52.74%
GADF+GASF	98.04%	98.80%	65.07%	50.00%
GADF+MTF	99.14%	98.97%	56.16%	49.32%
GADF+RCP	97.26%	98.97%	68.49%	**53.42%**
GASF+MTF	95.89%	98.29%	55.48%	44.52%
GASF+RCP	87.16%	99.32%	63.70%	54.11%
MTF+RCP	99.32%	99.49%	56.16%	52.74%
GADF+GASF+MTF	98.12%	99.14%	56.16%	50.68%
GADF+GASF+RCP	86.47%	98.97%	65.75%	50.68%
GADF+MTF+RCP	96.58%	98.97%	58.22%	52.74%
GASF+MTF+RCP	97.09%	98.46%	54.11%	50.68%
GADF+GASF+MTF+RCP	96.23%	99.14%	60.27%	50.00%

Table 4: Showing the performance of different combinations of image representations. Results are obtained by training the **IMG** and **IMG-Inception** models for 150 epochs using the best parameters for each image transformation.

The **IMG-Inception** network proposed by Karimi-Bidhendi et al. (2018) seems to support the redundant information theory, with only two combination (GASF+RCP and GADF+RCP) outperforming the best individual image transformation (RCP - 52.74%) by approximately 1-2%. In general, the accuracies achieved by this network are low, rarely scoring more than 50%. This is most likely due to the Inception v3 network being pre-trained on conventional images, learning to extract features such as forms or edges. However, as can be seen in Figure 17, such characteristics are rarely found in the transformed images, resulting in low performance.

By tuning the hyper-parameters to specific image-transformations the performance of both networks could be increased, for example achieving 71.23% accuracy with the combination of

GADF and GASF on the **IMG** network and 57.53% when tuning the **IMG-Incpetion** network to only GADF images. However, there is no indication that these transformations will be able to outperform the results obtained by much simpler networks like the **MLP-D** used on the original input.

7.3. Architecture comparison

The performance achieved by all examined architectures on the independent test-set will be disclosed in this section. Due to the fact that the test-set is not balanced, the results might be difficult to interpret as just the total accuracy can be misleading and it is therefore important to disclose the exact class composition of the *TEST* set. After preparation of the data according to the procedure described in Section6.1, the test-set used in this research consists of 688 samples, featuring the following numbers per class:

- *Healthy*: 80 samples
- *Hip*: 162 samples
- *Knee*: 192 samples
- *Ankle*: 172 samples
- *Calcaneus*: 82 samples

A comprehensive comparison of the performance by all tested architectures is presented in Table 5, listing the accuracies for the training-, validation- and the independent test-set. The highest value on the test-set is achieved by the **2DCNN-dilated** architecture, scoring **60.3%** accuracy, followed by the**1DCNN-strided** (58.28%) and **MLP2** (57.56%) architectures. It seems like the performance on the validation-set is not a good indication for the independent test-set, generally overestimating the accuracy by 10-15%.

Judging from these results, there is no benefit in adding additional layers when classifying GRF-measurements, as neither of the more complex networks (**FCN**, **ResNet** or **InceptionTime**) manages to outperform the single-layered **MLP30** using just 30 neurons for the classification. On the contrary, the complex variants seem to suffer from massive *overfitting*, reaching 100% accuracy on the train-set, without a corresponding increase in validation-accuracy. **ResNet** seems to be most affected by this problem, as its score is the lowest among all non-image based architectures. For the image transformations, the results from the previous section can be validated, with the GADF achieving the highest and the MTF the lowest accuracy. The ranking for the GASF and RCP is reversed on the *TEST* and validation set, but they remain within a range of 1% from each other.

Because the *TEST* dataset is not balanced, it is difficult to infer knowledge about the individual

Architecture	TRAIN-BALANCED		TEST
	Train-accuracy	*Validation-accuracy*	*Test-accuracy*
MLP30	66.44%	65.07%	56.54%
MLP90	66.61%	67.81%	55.96%
MLP2	76.71%	70.55%	57.56%
MLP-D	77.40%	71.92%	56.25%
1DCNN-strided	77.40%	73.97%	58.28%
1DCNN-dilated	78.08%	73.97%	57.85%
2DCNN-dilated	94.18%	73.97%	**60.03%**
2DCNN-1DKernels	82.36%	73.97%	56.54%
LSTM	75.51%	69.86%	53.92%
FCN (tuned)	100%	71.92%	54.07%
FCN (original)	99.83%	64.38%	55.67%
ResNet	100%	70.55%	49.27%
InceptionTime	100%	69.86%	54.51%
IMG (GADF)	97.95%	69.18%	55.81%
IMG (GASF)	90.41%	63.70%	52.18%
IMG (MTF)	98.63%	56.85%	44.48%
IMG (RCP)	97.95%	64.38%	51.45%

Table 5: Performance of models being trained on the *TRAIN-BALANCED* dataset when evaluated on the test-set. In case of the **FCN**, results are reported for the original and fine-tuned version and the **IMG** network lists the accuracies for all individual transformations.

classes from the just the overall accuracy alone. Therefore, the results obtained on this dataset have been displayed in Table 6 in more detail. The *ankle* class seems to be the most difficult to recognize with only a single model scoring more than 50% accuracy, followed by the *knee* class, where five networks are able to pass that threshold. Accuracy is quite high for the *healthy* and *calcaneus* class, which is probably the reason for the high disparity between the results for the validation- and test-set. Because those classes are under-represented in the *TEST* data, overall accuracy is reduced for all classifiers.

When ranking the models by the score they achieve on each class (i.e. the best scoring model for a class receives rank one, the second best rank two and so on), the **1DCNN-strided** architecture is the overall winner, closely followed by the two **dilated** variants. Surprisingly the **IMG** achieves the forth place when using the GADF as input, followed by a larger gap to the different types of **MLP** architectures. In this kind of ranking procedure, the complex networks are all outperformed by the simple architectures with the **InceptionTime**, **FCN** and **ResNet** coming in last (in that order). The considerable performance differences on the various classes between the models suggest than an ensemble of different architectures might be beneficial for the classification. Especially the (comparatively) high accuracy achieved by the **IMG** network (in combination with GADF) on the *ankle* class is remarkable and could be useful to all other

Architecture	**TEST-set** (models trained on TRAIN-BALANCED)					
	Healthy (n=80)	*Hip (n=162)*	*Knee (n=192)*	*Ankle (n=172)*	*Calcaneus (n=82)*	*Overall (n=688)*
MLP30	85.00%	62.35%	46.35%	49.42%	56.10%	56.54%
MLP90	85.00%	57.41%	58.85%	31.40%	69.51%	55.96%
MLP2	85.00%	51.85%	**60.94%**	40.70%	69.51%	57.56%
MLP-D	56.25%	69.75%	46.88%	44.77%	75.61%	56.25%
1DCNN-strided	**92.50%**	64.81%	45.83%	40.12%	79.27%	57.56%
1DCNN-dilated	88.75%	66.05%	41.67%	48.26%	69.51%	57.85%
2DCNN-dilated	81.25%	61.73%	52.60%	47.67%	79.27%	**60.03%**
2DCNN-1DKernels	70.00%	55.56%	50.52%	47.09%	79.27%	56.54%
LSTM	77.50%	62.96%	39.06%	39.53%	78.05%	53.92%
FCN (tuned)	78.75%	56.17%	43.75%	45.35%	68.29%	54.07%
FCN (original)	76.25%	59.26%	56.25%	33.14%	74.39%	55.67%
ResNet	66.25%	**74.69%**	26.04%	34.88%	67.07%	49.27%
InceptionTime	82.50%	58.64%	39.58%	43.02%	78.05%	54.51%
IMG (GADF)	86.25%	60.49%	28.13%	**56.98%**	79.27%	55.81%
IMG (GASF)	75.00%	56.79%	47.92%	33.72%	69.51%	52.18%
IMG (MTF)	87.50%	40.12%	34.38%	37.79%	48.78%	44.48%
IMG (RCP)	82.50%	59.88%	45.31%	21.51%	**81.71%**	51.45%

Table 6: Performance of models being trained on the *TRAIN-BALANCED* dataset when evaluated on the test-set. Accuracies for each individual class and the entire dataset are provided, with the highest values highlighted in bold.

methods.

Robustness was verified for the classifier with the highest accuracy on the test-set, performing ten train- and test-runs with the **2DCNN-dilated** architecture, averaging its performance on the test set. The mean accuracy obtained is slightly lower than the one given above, scoring 59.61% with the lowest and highest values being 56.98% and 62.35% respectively. A standard deviation of 1.79% was calculated as a consequence, revealing that there is no statistical significance between the top performing networks.

Finally, to investigate the influence of more training data being available (even though the classes are not balanced), Table 7 illustrates the obtained results for the same networks as above, but this time trained on all available data (i.e. the entire *TRAIN*-dataset with the exception of the validation-set). The validation-set is exactly the same for both attempts (except for an adjusted normalization process to match the training-data) and the selection of the best model (i.e. when to stop training) is based on the highest accuracy achieved on that part of the data.

Surprisingly, using all available data (i.e. the entire *TRAIN* set) did only benefit one of the MLP networks, while it generally resulted in higher accuracy for all convolutional models. Especially **ResNet** was able to increase its performance on three of the classes resulting in an overal im-

Architecture	**TEST-set** (models trained on TRAIN-set)					
	Healthy (n=80)	*Hip* (n=162)	*Knee* (n=192)	*Ankle* (n=172)	*Calcaneus* (n=82)	*Overall* (n=688)
MLP30	70.00%	51.23%	44.27%	**68.02%**	45.12%	54.94%
MLP90	73.75%	33.33%	53.65%	67.44%	51.44%	54.51%
MLP2	85.00%	35.80%	60.94%	42.44%	**79.27%**	55.38%
MLP-D	88.75%	47.53%	**66.15%**	57.56%	56.10%	**61.05%**
1DCNN-strided	77.50%	53.09%	62.50%	52.33%	62.20%	59.45%
1DCNN-dilated	77.50%	**66.67%**	51.56%	50.00%	73.17%	60.32%
2DCNN-dilated	81.25%	59.26%	54.69%	51.74%	70.73%	60.03%
2DCNN-1DKernels	73.75%	49.38%	59.38%	64.53%	52.44%	59.16%
LSTM	75.00%	54.32%	50.00%	53.49%	54.88%	55.38%
FCN (tuned)	92.50%	59.26%	50.52%	57.56%	51.22%	59.30%
FCN (original)	85.00%	61.11%	42.19%	51.74%	73.17%	57.70%
ResNet	85.00%	50.00%	60.42%	40.12%	65.85%	56.40%
InceptionTime	52.50%	29.63%	60.94%	47.67%	75.61%	51.02%
IMG (GADF)	86.25%	56.79%	31.77%	61.63%	67.07%	55.67%
IMG (GASF)	86.25%	35.80%	59.38%	58.14%	37.80%	54.07%
IMG (MTF)	**93.75%**	25.31%	54.17%	31.40%	65.85%	46.37%
IMG (RCP)	91.25%	59.88%	46.35%	59.88%	26.83%	53.20%

Table 7: Showing the accuracies achieved by the models when being trained on the entire *TRAIN* set and evaluated on the test-set. Both, the overall score as well as the accuracies for each class are listed, with the highest values highlighted in bold.

provement of almost +7%. However, the results also reveal one of the big disadvantages when training on an unbalanced dataset: Having more data available for some classes, causes the network to predict those more often, based on their higher probability of being correct. This effect can be observed for most of the models when using the *TRAIN* dataset, resulting in a reduced accuracy for the under-represented classes (in this case *healthy* and *calcaneus*) while performance is improved for the others, simply because they are predicted more often. Most interestingly, this tendency does not affect all architectures to an equal degree, with the **IMG** in combination with the MTF actually being able to improve its accuracy for both of those two classes, while some of the others managed to increase at least one of them. More research is necessary in order to understand this behaviour and its implications for the classification task. In the meantime, the following conclusions can be drawn based on the conducted experiments:

1. An even larger dataset or at least an increased number of samples for the under-represented classes would be beneficial. This would allow for the creation of a larger (balanced) validation-set while still having enough data available for training, helping the obtained results to generalize better on the *TEST* set. Additionally, as indicated by the results obtained on the *TRAIN* set, some classifiers might be able to perform better if more data

can be used for training.

2. Small differences in accuracy on the validation-set are unlikely to transfer to the *TEST* set and it is hard to predict which architecture will perform best.

3. It seems like the convolution is able to extract some additional features from the data, with the CNNs generally achieving higher results than the MLPs. Their impact remains small, however, only improving accuracy slightly when compared to other architectures.

7.4. Influence of pre-processing

The results for the different preprocessing methods are presented in this section, starting with the influence of normalization on the proposed classifiers. The accuracies achieved by the various architectures with either **min-max** or **z-score** normalization applied are contrasted against just the original data (i.e. no normalization) in Table 8. It should be remarked however, that with the exception of the **FCN (original)** network, all models have been fine-tuned with an input format using **min-max** normalization and are therefore biased to prefer this kind of input. Furthermore, the **IMG** network was not evaluated for this pre-processing mode, because most of the image transformations enforce some king of normalization anyway, resulting in a low influence of the method.

Architecture	Training-Accuracy			Validation-Accuracy			Test-Accuracy		
	Original	*Min-max*	*Z-score*	*Original*	*Min-max*	*Z-score*	*Original*	*Min-max*	*Z-score*
MLP30	55.82%	66.44%	72.43%	54.79%	65.07%	67.12%	38.37%	56.54%	54.51%
MLP90	60.27%	66.61%	76.37%	56.85%	67.81%	67.81%	49.13%	55.96%	55.38%
MLP2	66.61%	76.71%	69.01%	60.27%	70.55%	65.75%	46.37%	57.56%	54.07%
MLP-D	76.88%	77.40%	77.05%	63.01%	71.92%	67.81%	59.16%	56.25%	52.03%
1DCNN-strided	59.93%	77.40%	87.33%	65.75%	73.97%	68.49%	51.60%	58.28%	57.56%
1DCNN-dilated	57.19%	78.08%	92.47%	59.59%	73.97%	67.12%	48.40%	57.85%	54.51%
2DCNN-dilated	69.69%	94.18%	99.66%	62.33%	73.97%	64.38%	57.70%	60.03%	55.96%
2DCNN-1DKernels	64.55%	82.36%	85.10%	62.33%	73.97%	69.18%	51.74%	56.54%	55.23%
LSTM	65.92%	75.51%	87.16%	56.16%	69.86%	59.59%	52.33%	53.92%	43.46%
FCN (tuned)	100%	100%	100%	63.01%	71.92%	64.38%	46.95%	54.07%	48.26%
FCN (original)	99.32%	99.83%	99.83%	57.53%	64.38%	65.75%	44.04%	55.57%	54.51%
ResNet	100%	100%	100%	63.70%	70.55%	66.44%	50.58%	49.27%	53.34%
InceptionTime	100%	100%	100%	65.07%	69.86%	65.07%	52.47%	54.51%	52.33%

Table 8: Comparison of accuracies for training-, validation- and test-set for different normalization methods and architectures. "Original", denotes that no normalization has been applied at all, while the name of the normalization method is given for the other column headers.

It can be observed that applying no normalization is clearly the worst choice, especially for the less complex networks, resulting in a lower performance. Both **min-max** and **z-score** seem to be good choices for most of the architectures, with the exception of the **LSTM**, where **z-score** is outperformed by the original input on the test-set. As indicated by the almost identical accuracy

on the **FCN (original)** model for both normalization methods, the differences on other networks are most likely due to the bias mentioned earlier. This seems to suggest that both of them work equally well for the given dataset, if the parameters of the models are tuned accordingly.

As far as the *input layout* is concerned, the effects seem to be negligible as well. As explained in Section6.2.4 this has been evaluated on a modified version of the **2DCNN-1DKernels** architecture, since the order of the input only affects 2D-CNNs. For an $101 \times 10 \times 1$ input tensor, the achieved performance is already documented in Section 7.1.3, scoring 72.60% (without hyper-parameter tuning). An almost identical result can be accomplished using the $101 \times 1 \times 10$ and $2 \times 5 \times 101$ formats, while accuracy decreases slightly (to 71.92%) in case of the input being provided as a $101 \times 2 \times 5$ tensor. The most favourable data shape seems to be $101 \times 5 \times 2$ where several models have been able to achieve a top accuracy of 73.97%, suggesting that the differences between the affected and unaffected leg play an important role during the classification process. However, the observed disparity between formats remains small and will most likely vanish when fine-tuning the hyper-parameters of each model. Nonetheless, there are three important observations to be made:

1. The $101 \times 5 \times 2$ layout, facilitating comparisons between legs achieves the highest accuracy, suggesting that this is an important source of information for the network.
2. Even if all time-steps are convoluted in a single operation (as it is the case for the $2 \times 5 \times 101$ format), the accuracy remains high. This seems to indicate that the amount of redundancy within the data is high, especially for the temporal relationships, which seem to be insignificant for the classification.
3. The 2D-CNN is able to perform well independent of the input ordering, indicating that the influence of inter-signal dependencies is small as well, because the amount of additional information is low.

Those assumptions have been verified in the next step, looking at the impact of *re-sampling* on the data. Experiments were limited to the best performing **MLP** architectures and the unmodified **FCN**, due to the expected robustness against input of varying length and the high amount of re-training required for this evaluation. However, since the sole purpose of this endeavour is to determine how much information remains within the re-sampled measurements, this approach should be more than sufficient. An overview of the obtained results is provided in Table 9.

Surprisingly, even though the highest accuracy was achieved using all available data-points (i.e. resampling of 1), the amount of redundant data seems to be considerably high. The values suggest that only 10% of the data are actually needed to achieve good accuracy, scoring only 3-5% lower for the MLPs and achieving an identical result for the FCN. In order to verify this hypothesis, an attempt has been made to fine tune the parameters of **MLP-D** on the input-series $\tilde{X_{10}}$ (i.e. 11 time-steps), looking for a score that is identical or close to the 71.92% achieved on the

Architecture	Sampling interval									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
MLP2	70.55%	67.81%	65.75%	65.75%	69.18%	65.07%	65.75%	65.75%	65.75%	65.07%
MLP-D	71.92%	68.49%	69.86%	67.81%	68.49%	67.81%	67.81%	68.49%	67.81%	69.18%
FCN (original)	64.38%	63.70%	63.70%	64.38%	64.38%	60.96%	59.59%	62.33%	64.38%	64.38%

Table 9: Validation-accuracies for various sampling intervals k for MLP and FCN architectures. It can be observed that performance does not suffer much even if the samples are reduced from 101 to just 11 data-points.

original data. The resulting parameter search revealed that there is no difference in performance of an MLP with a single hidden layer when only 11 data-points are used instead of the total 101. The best accuracy observed on the reduced dataset even outperformed the original one slightly, scoring 72.60% accuracy. As a result of the conducted experiments, the following conclusions can be drawn:

- Almost no information is contained within the temporal dependencies of a signal (i.e. they can be fully represented by 11 points).
- Interaction within signals also seem to be of limited importance (i.e. the behaviour in the regions removed from the dataset is not relevant).
- Not all re-samplings $\tilde{X}_k$ seem to yield an equally accurate representation, depending on the selection of points. For example $k = 6$ and $k = 7$ feature the lowest accuracies across all evaluated architectures, suggesting that some parts of the waveform are more important than others.

The last part of this section investigates if the theory by Slijepcevic et al. (2020) about the increased robustness of the classifier when training with all trials instead of the mean waveform can be verified for neural network architectures. Table 10 lists the obtained accuracies when using each individual trial during the training process. The results for the validation- and test-set are provided for the mean waveform (i.e. the model is trained with all trials, but just the mean waveform is predicted) or all trials in total (i.e. the classifier tries to predict *each* trial. Aggregating those individual labels into one common class per gait analysis session will be discussed in the next section.

First of all, it needs to be remarked that the training process takes considerably more time and computing power if each individual trial is used, especially for the more complex architectures. As a general observation, there seems to be a lot of variability within the trials of a single session, as the performance on the individual trials is almost always lower than on the mean waveform. This signifies that there exist at least some trials within a session, that get classified differently than the average. Furthermore, while the performance on the validation-set generally decreased (compared to the models being trained on the mean waveform), this is not necessarily the case for

the test-set, with some architectures achieving higher accuracies than previously (see Table 6). In combination with the smaller gap between the accuracies of the mean waveform and all trials on the test-set, it does indeed seem like a more robust classifier was created.

Architecture	Training-Accuracy	Validation-Accuracy		Test-Accuracy	
	AllTrials	*Mean*	*AllTrials*	*Mean*	*AllTrials*
MLP30	71.40%	65.07%	62.49%	56.98%	56.62%
MLP90	75.19%	67.81%	62.57%	57.41%	54.81%
MLP2	82.84%	63.70%	58.76%	52.33%	53.44%
MLP-D	69.27%	69.18%	62.17%	**61.77%**	**58.75%**
1DCNN-strided	80.56%	64.38%	61.22%	58.43%	55.01%
1DCNN-dilated	92.70%	63.70%	58.84%	52.18%	51.68%
2DCNN-dilated	98.00%	60.96%	58.45%	53.34%	54.11%
2DCNN-1DKernels	81.47%	67.12%	62.65%	56.34%	53.22%
LSTM	84.76%	65.07%	61.46%	53.92%	52.22%
FCN (tuned)	100%	65.07%	62.25%	57.56%	52.35%
FCN (original)	98.99%	61.64%	58.84%	54.07%	48.19%
ResNet	100%	66.44%	59.87%	53.78%	52.03%
InceptionTime	100%	63.70%	60.11%	47.97%	49.22%
IMG (GADF)	97.96%	63.70%	56.62%	53.34%	50.98%
IMG (GASF)	88.99%	59.59%	55.11%	50.15%	49.70%
IMG (MTF)	97.53%	55.48%	52.97%	49.71%	46.89%
IMG (RCP)	97.33%	61.64%	57.73%	51.89%	51.59%

Table 10: Depicting the performance achieved when training with all individual trials instead of taking the mean. Accuracies for the validation-set are given for both, the aggregated data (i.e. *Mean*) as well as all individual trials.

Most notably, the **MLP-D**, seems to benefit from more data being available (see the comparison on the whole *TRAIN* set in Table 7), achieving the best accuracy on the test-set so far with **61.77%**. Another notable fact is the low training-accuracy achieved by this network, suggesting that it is indeed somehow able to extract more generalizable information than the other architectures. In general, the performance of the MLPs when relying on individual trials is impressive, with the perceived small advantage of the convolutional networks reported for the mean waveform, seeming to disappear entirely, with good performances being observed across all architectures.

7.5. Influence of post-processing

According to the results obtained by Slijepcevic et al. (2020), aggregation of all individual trials for a single session by using a majority voting-based approach, should be able to take advantage of the more robust classifier built in the previous section to produce a more accurate result. The

comparison between the original score (highest accuracy using the mean waveform, independent of which data has been used to train the model - denoted as *baseline*) and models trained with either aggregated or individual trials followed by *late fusion* via majority voting) is presented in Table 11.

For the validation-set, accuracy did generally not increase when majority voting was applied, while better performance can be observed for the test-set on some architectures. Interestingly, in some cases the classifier obtained from training with the mean waveform seemed to be more robust, yielding the best result on the test-set, especially for the **2DCNN-1DKernels** model. The **MLP-1D** managed to increase its accuracy by +0.2% with applied majority voting, achieving the highest performance observed on the test-set during all experiments. With the exception of **ResNet**, none of the convolutional networks could improve its results when compared to the mean waveform, suggesting a lower generalizability of the learned patterns.

Architecture	Validation-Accuracy			Test-Accuracy		
	Baseline	*Mean*	*AllTrials*	*Baseline*	*Mean*	*AllTrials*
MLP30	**65.07%**	61.64%	64.38%	56.98%	57.12%	**57.99%**
MLP90	**67.81%**	65.07%	65.07%	**57.41%**	54.51%	57.27%
MLP2	**70.55%**	64.38%	61.64%	**57.56%**	57.41%	55.52%
MLP-D	**71.92%**	69.86%	67.81%	61.77%	56.40%	**61.92%**
1DCNN-strided	**73.97%**	67.81%	63.01%	**58.43%**	55.96%	56.98%
1DCNN-dilated	**73.97%**	70.55%	58.90%	**57.85%**	57.27%	52.33%
2DCNN-dilated	**73.97%**	67.81%	63.01%	**60.3%**	58.58%	57.27%
2DCNN-1DKernels	**73.97%**	67.12%	65.75%	56.54%	**57.12%**	54.51%
LSTM	**69.86%**	64.38%	62.33%	**53.92%**	51.16%	52.76%
FCN (tuned)	**71.92%**	65.75%	65.75%	**57.56%**	54.94%	54.36%
FCN (original)	**64.38%**	56.85%	58.90%	**55.67%**	52.62%	48.98%
ResNet	**70.55%**	60.27%	63.01%	53.78%	49.85%	**56.25%**
InceptionTime	**69.86%**	63.01%	65.75%	**54.51%**	54.36%	49.85%
IMG (GADF)	**69.18%**	62.33%	62.33%	**55.81%**	54.94%	**55.81%**
IMG (GASF)	**63.70%**	61.64%	58.22%	**52.18%**	52.47%	51.89%
IMG (MTF)	56.85%	52.05%	**57.53%**	49.71%	**51.60%**	51.02%
IMG (RCP)	**64.38%**	63.01%	57.53%	51.89%	**55.09%**	54.80%

Table 11: Demonstrating the effect of *late fusion* via majority voting applied to different architectures. The column denoted by *Mean* shows the accuracy obtained when using majority voting in combination with a model that has been trained on mean waveforms, while each individual trial has been used for training in the *AllTrials* column. *Baseline* on the other hand reports the best result without majority voting, obtained by using the mean waveform.

Overall, there are three important trends to observe:

- When compared to the results in Table 7, majority voting *never* decreased performance for models being trained on all individual trials, in most cases actually increasing it by

1-5%.

- For model trained on the mean waveform, majority voting did not help in most cases, probably because the resulting models are more sensitive to perturbations (i.e. less robust)
- The high accuracy on the validation-set for the *baseline* suggests that most architectures extracted patterns from this set that are not generalizable, thus *overfitting* the validation data.

Considering that all networks (with the exception of the unmodified **FCN**) have been optimized for the mean waveform, the combination of using majority voting in combination with a classifier trained on all trials seems to be a promising approach. The accuracy boost achieved by *late fusion* seems to apply even if the performance of the architecture is already high, indicating that it would increase even further with a better classifier. Optimizing the networks to utilize all input data therefore seems to be a promising approach for future research. However, because results indicate that it is more difficult to obtain general features from all individual trials, it is unlikely that overall performance would increase by more than 5-10%.

7.6. Class separability

The goal of this section is to provide more insights into the problem of classifying gait disorders according to body locations itself, by exploring the distinctiveness of each class. For a "good" classification, the overlap between classes should be small, as this indicates that they can be separated reasonably well. Only the results obtained for the **1DCNN-strided** architecture are presented here, since this architecture achieved the best results for the class fusion introduced in Section 6.3 (i.e. it is the most accurate model for detecting whether a gait disorder originates from the lower or upper leg). The confusion matrix for this network is depicted in Figure 23.

The matrix illustrates that some of the classes are more likely to be confused with each other because their characteristics seem to be very similar (i.e. they are hard to distinguish). This is especially true for the *ankle* class, where 32% of the instances are misclassified as *calcaneus*. However, when combining such hard to differentiate classes, a considerably higher accuracy could be reached and the theoretical performance of the fused *UL* at *LL* classes is highlighted in the figure. If the classification task is simplified to only three categories, an accuracy of **87.67%** could be obtained. A parameter search was conducted using this new classification problem (i.e. *Healthy - UL - LL* in an attempt to train a model that is able to improve on the performance of the classifier using the original five. Unfortunately, both tasks seem to be equally difficult, because none of the models examined during the extensive search was able to surpass the value of 87.67% accuracy. In accordance to the results of the previous sections, this seems to support the theory that the network actually *overfitted* on the validation-data and has already found an

optimal solution.

1DCNN-strided		Real				
		Healthy (n = 29)	*Hip* (n = 32)	*Knee* (n = 28)	*Ankle* (n = 25)	*Calcaneus* (n = 32)
Prediction	*Healthy*	100.0%	3.1%	10.7%	4.0%	9.4%
	Hip	0.0%	71.9%	17.9%	16.0%	3.1%
	Knee	0.0%	21.9%	60.7%	4.0%	0.0%
	Ankle	0.0%	0.0%	10.7%	44.0%	0.0%
	Calcaneus	0.0%	3.1%	0.0%	32.0%	87.5%

Upper Leg: 86.76%

Lower Leg: 82.46%

Figure 23: Confusion matrix for the **1DCNN-strided** architecture. All values are in percentages of the real occurrences (i.e. each columns sums up to 100%). For example the value in the *Hip-Hip* cell signifies that 71.9% of all patients with hip injuries were classified as such. The highlighted areas simulate the effect of fusing commonly confused classes together, resulting in a total accuracy of 87.67% for the classifier (+13% from the original result).

The research on class separability is concluded by investigating the influence of combining data from both legs, with the the results being summarized into the confusion matrix of just the affected leg displayed in Figure 24. In case of the **1DCNN-strided** architecture, the accuracies for all classes, with the exception of *knee* are decreased, indicating that the model learns a narrow representations for four classes and a single broader one, that is used to fit everything that does not match any other category (in this case classifying such data as a *knee* injury). Furthermore, it can be deduced that information from the unaffected leg is especially helpful for recognizing *healthy*-patients, while the effects on other classes remain mixed (dominated by misclassifications as *knee*). Calculating the average accuracy per class (i.e. the mean of the diagonal elements) reveals that it is reduced by almost 10% (from 72.82% down to 63.28%) if the signals for the unaffected leg are omitted from the classification. Ass such, the conclusion drawn by (Slijepcevic et al., 2020, 2018b), claiming that inclusion of the unaffected leg results in a better overall accuracy can be verified for neural networks.

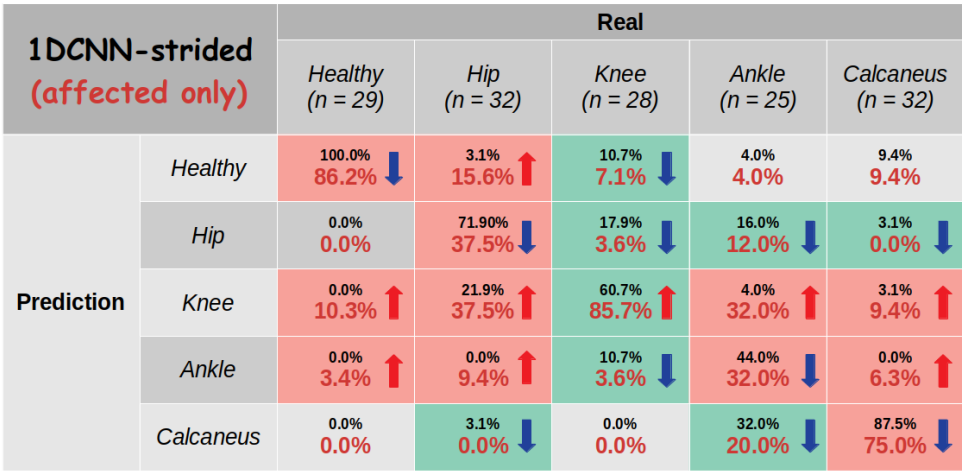

Figure 24: Comparison of confusion matrices for the **1DCNN-strided** network for GRF-measurements using either all data or only the affected leg. All values are in percentages of the real occurrences (i.e. each columns sums up to 100%). The small black numbers give the percentages obtained with the unaffected side included, while the red numbers illustrate the results for the affected side only. Increases/Decreases are highlighted by arrows, while the background colour indicates a negative (red) or positive (green) effect.

7.7. Data Selection

The implications of the selection process applied to the data obtained from the *GaitRec* archive are discussed in this section. Due to the considerable increase in data (especially when readmissions are considered) and thus computing time required, experiments in this section were restricted to the two architectures achieving the highest accuracy on the test set, namely the **MLP1** and **1DCNN-dilated** models. Furthermore, in contrast to the previous sections, the *TRAIN* set had to be used during the training process because the *TRAIN-BALANCED* set does not include samples with the examined characteristics. Results are only available for the mean waveform, but it is expected that the general behaviour would be similar, even if other input forms are used.

When looking at persons with injuries in both legs, there are a total of 34 (*hip*: 10, *knee*: 6, *ankle*: 1, and *calcaneus*: 17) available in the test-set. The inclusion/exclusion of such patients during training and/or testing had the following consequences on the examined architectures:

- **MLP1**: This architecture manages to classify 14 out of these 34 correctly, when included in training and testing. Excluding such data from the whole process increased total accuracy by +0.1%, with the most notable changes being an increase in the accuracy of the

calcaneus class (+13%) and, surprisingly, a decrease for the healthy patients (-11%).

- **1DCNN-dilated**: 50% of the patients with injuries in both legs are classified correctly (17 out of 34), while excluding them from both training and testing boosted overall accuracy by +2.5%, increasing the values for both *knee* (+10%) and *healthy* (+5) with only a slight decrease in results for the *calcaneus* class (-4%).

Excluding those patients from just the training-data while maintaining them in the test-set resulted in a worse performance for both architectures. Otherwise, due to the low sample size, there is little indication to which method should be preferred in general. However with only a 50% chance of the impairment location being classified correctly, there seems to be little benefit in maintaining such data, as it will not see practical use. Since the GRF-measurement classification task appears to be difficult enough without this additional constraint, the recommended approach would be to discard such data from both the training- and test-set, focussing on the development of a classier for injuries in a single leg.

Including data from *readmissions* provides the advantage of increasing the amount of available data for training by a factor of four, while, on the other hand, skewing the class balance even further (since such data is not available for healthy persons). This is reflected in the experiments, deteriorating performance for the *healthy* group considerably, with accuracy decreasing by 11-20%. Results obtained for the other classes remained mixed and seemed to be dependent on the architecture. For the **MLP1** a considerable increase in accuracy was observed for the *hip* class, paired with an equally strong decrease for the *knee* class, whereas the exact opposite applied when the **1DCNN-dilated** network was employed. For now, there is no definite answer to the question on how to best utilize such data, as its usefulness is severely restricted by the lack of measurements for non-injured persons.

In contrast to the somewhat ambiguous results above, experiments conducted with patients wearing orthopaedic/shoes insoles give definite evidence on how to proceed with such data. Both models show a decreased accuracy for all classes (except one that seems to act as a general category for everything that can not be classified anywhere else) resulting in a low performance in total. This demonstrates that the impact of the distortion caused by orthopaedic aids should not be underestimated and it is generally recommended to remove such samples from the dataset. This can be safely done without limiting the clinical usefulness of the classifier, since GRF-measurements can be performed equally well when taking off one's shoes and walking barefoot or in socks, a practice that should not be too hard to adopt in a clinical setting.

7.8. Summary

The obtained results demonstrate that classification of gait disorders according to the body location of the associated impairment is a difficult task to solve. High inter- and intra-class variability prevent the classifier from accurately separating the data, signifying that either the method is not good enough to extract the appropriate features or the information is simply not contained in the results. Especially the fact that no higher accuracy than 73.97% can be achieved on the validation-set is concerning. An analysis of the corresponding waveforms (see Figures 21 & 19) seems to indicate that the actual learning is limited around 60% and deviations from that number are just random fits of the data, that "happen" to be a good choice for the validation-set. This theory is supported by the accuracies obtained for the *TEST* set, which rarely surpass that value.

Furthermore, the information needed to achieve those 60% seems to be fairly obvious and is contained within only a few data-points, as down-sampling did not decrease accuracy by much and even a simple MLP with just 30 neurons is able to extract it. A major problem seems to be the distinction between the *ankle* and *calcaneus* classes, with classifiers usually just performing well on only one of them. Closer inspection revealed that the main decision point between those two classes is an earlier maxima in the F_ap component, but unfortunately only about 75% of the *calcaneus* instances actually posses that characteristic. At the same time approximately 25% *ankle* signals contain that feature as well. Therefore, higher accuracy for one of these classes usually results in lower performance for the other. *Hip* and *knee* injuries are easier to distinguish but rarely achieve a score larger than 70% (*hip*) or 60% (*knee*) respectively. While several of the proposed techniques (majority voting, training with all trials) seemed to be able to mitigate those issues, the improvements remain rather small, never able to catch up to the performance observed on the validation-set.

Image-transformations, normalization methods and neural network architectures did not seem to influence the result much, largely achieving identical values independent of the type. However, the conducted experiment indicate that having more data available during training is able to improve accuracy and maybe a larger dataset could be used to alleviate many of the current issues. Neural networks are known to benefit from a huge amount of data (Roh et al., 2019) and if generalizable patterns exist more sampled would definitely be advantageous.

8. Conclusion

The main contribution of this research is the evaluation of different neural network based approaches for classifying GRF-measurements according to the location of gait impairment. Results suggest that, despite the high accuracy achieved (compared to previous research), the

amount of information contained within the data is actually limited. None of the employed architectures has been able to break the threshold of 74% accuracy on the validation-set, even though extensive searches across the parameter space have been performed. The fact that the maximal accuracy for several different models is exactly the same might signify that there is no better solution obtainable with the given dataset. However, there are still areas for future research that might end up yielding a better performance. Making use of all individual trials during the training process in combination with a majority voting based approach for the final classification of each sample could lead to a boost in accuracy. Especially if the parameters of those models are fine-tuned on non-aggregated input, such an approach could perform better than the ones used in this research. Experiments indicate that majority voting does not lose its advantages even when applied to networks that already obtain high accuracy, reducing the task to finding a model that performs well on all individual trials. Furthermore, due to the observed differences between architectures on the accuracies obtained for single classes, investigating ensembles of neural networks (similar to the approach used by Fawaz et al. (2019b) but consisting of different architectures) seems to be a promising direction for future research.

Nonetheless, the results obtained from the experiments raise serious doubts that the aforementioned threshold can be surpassed by a large margin. The GRF-signals might simply not contain enough information for an accurate classification, as the information contained within temporal dependencies and the interaction between signals seems to be largely redundant. None of the methods specifically developed for categorizing time-series yielded a significantly better accuracy than a simple MLP consisting of a single hidden layer. Perceived (although small) differences on the validation-set did not transfer to the independent test-set, because it overestimated the actual accuracy by approximately 10% (and in most cases even more). Therefore, fine-tuning of the architectures generally ended up in finding a good fit for the validation-set, with a somewhat limited ability to generalize well on the test-set.

Especially the *ankle* and *knee* classes are hard to identify resulting in a low performance for both of them. This seems to be a general problem for the *ankle* class, suggesting that it might be due to the characteristics of the class itself (i.e. the waveforms are too similar to other classes), while the complications for the *knee* class seem to be of a different nature. The drop in accuracy on the *TEST* data might signify that the variability of the waveforms within this group is too high to be accurately captured by the validation-set, making this class difficult to categorize. While the latter could be combated with more available data, finding a good representation for the *ankle* class seems to be the real bottleneck and the key to improving accuracy.

Apart from such concerns, research revealed that it is possible to extract the same amount of information (i.e. reaching the same level of accuracy) even if the signals are downsampled by a factor of 0.1, resulting in only 11 points per measure. This implies that all relationships and correlations between signals other than at those exact time-steps are irrelevant for the classification,

probably because of the aforementioned high variability. This applies to the classes in general as well as to the individual persons. The performance decrease observed when using majority voting in combination with the mean waveform can only be explained by either outliers or individual trials that are distinctly different from each other. It seems to be entirely possible for the averaged waveform of all trials (from the same session), to be classified into one category, while the majority of the trials itself would fall into another when analysed individually. This raises some concerns about the robustness of the classifier, calling for further research on that topic.

Due to the reasons listed above and the fact that the *TEST* set defined by Horsak et al. (2020) has not been used in any publications yet, a comparison to previous research (Slijepcevic et al., 2017, 2018a,b, 2020) is difficult. For example, Slijepcevic et al. (2020) used the same dataset (i.e. *TRAIN-BALANCED*) for training a SVM, forming an independent test-set by randomly selecting 35% of the samples. Since this test-set is nearly balanced with high probability, the overall accuracies can not be compared and performance for individual classes is not reported in their research. However, their best accuracy of 62% is almost identical to the highest score observed in this research (61.92%) indicating that both approaches work equally well (and maybe separate the data in a similar fashion). In order to verify this hypothesis, an evaluation of the SVM on the dataset from Horsak et al. (2020) would be needed and might reveal some additional insights into the task.

In any case, the purpose of successfully establishing a first benchmark on the dataset provided by Horsak et al. (2020) was achieved, hopefully facilitating comparisons for future research against a common baseline. Unfortunately it seems like small subsamples from the *TRAIN-BALANCED* set might not be a good enough indication for the actual performance of a classifier, making this common reference a much needed opportunity to assess further research.

References

Abdulhay, E., Arunkumar, N., Narasimhan, K., Vellaiappan, E., and Venkatraman, V. (2018). Gait and tremor investigation using machine learning techniques for the diagnosis of Parkinson disease. *Future Generation Computer Systems*, 83:366--373.

Alaqtash, M., Sarkodie-Gyan, T., Yu, H., Fuentes, O., Brower, R., and Abdelgawad, A. (2011). Automatic classification of pathological gait patterns using ground reaction forces and machine learning algorithms. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 453--457, Boston, MA. IEEE.

Alharthi, A. S. and Ozanyan, K. B. (2019). Deep Learning for Ground Reaction Force Data Analysis: Application to Wide-Area Floor Sensing. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 1401--1406, Vancouver, BC, Canada. IEEE.

Bagnall, A., Bostrom, A., Large, J., and Lines, J. (2016). The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. *arXiv:1602.01711 [cs]*. arXiv: 1602.01711.

Bagnall, A., Lines, J., Hills, J., and Bostrom, A. (2015). Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522--2535.

Baker, R. (2013). *Measuring walking: a handbook of clinical gait analysis*. Mac Keith Press, London. OCLC: 829386814.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157--166.

Bhanja, S. and Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting.

Chau, T. (2001a). A review of analytical techniques for gait data. Part 1: fuzzy, statistical and fractal methods. *Gait & Posture*, 13(1):49--66.

Chau, T. (2001b). A review of analytical techniques for gait data. Part 2: neural network and wavelet methods. *Gait & Posture*, 13(2):102--120.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8(1):6085.

Christian, J., Kröll, J., Strutzenberger, G., Alexander, N., Ofner, M., and Schwameder, H. (2016). Computer aided analysis of gait patterns in patients with acute anterior cruciate ligament injury. *Clinical Biomechanics*, 33:55 -- 60.

Costilla-Reyes, O., Scully, P., and Ozanyan, K. B. (2018). Deep Neural Networks for Learning Spatio-Temporal Features From Tomography Sensors. *IEEE Transactions on Industrial Electronics*, 65(1):645--653.

Cui, Z., Chen, W., and Chen, Y. (2016). Multi-Scale Convolutional Neural Networks for Time Series Classification. *arXiv:1603.06995 [cs]*. arXiv: 1603.06995.

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2018). *The UCR Time Series Classification Archive*.

DiPietro, R. and Hager, G. D. (2020). Chapter 21 - Deep learning: RNNs and LSTM. In Zhou, S. K., Rueckert, D., and Fichtinger, G., editors, *Handbook of Medical Image Computing and Computer Assisted Intervention*, pages 503 -- 519. Academic Press.

Eckmann, J.-P., Kamphorst, S. O., and Ruelle, D. (1987). Recurrence Plots of Dynamical Systems. *Europhysics Letters (EPL)*, 4(9):973--977.

Elias, P., Sedmidubsky, J., and Zezula, P. (2015). Motion Images: An Effective Representation of Motion Capture Data for Similarity Search. In Amato, G., Connor, R., Falchi, F., and Gennaro, C., editors, *Similarity Search and Applications*, volume 9371, pages 250--255. Springer International Publishing, Cham. Series Title: Lecture Notes in Computer Science.

Ertuğrul, Ö. F., Kaya, Y., Tekin, R., and Almalı, M. N. (2016). Detection of Parkinson's disease by Shifted One Dimensional Local Binary Patterns from gait. *Expert Systems with Applications*, 56:156--163.

Eskofier, B. M., Federolf, P., Kugler, P. F., and Nigg, B. M. (2013). Marker-based classification of young–elderly gait pattern differences via direct PCA feature extraction and SVMs. *Computer Methods in Biomechanics and Biomedical Engineering*, 16(4):435--442.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019a). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917--963.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. (2019b). InceptionTime: Finding AlexNet for Time Series Classification. *arXiv:1909.04939 [cs, stat]*. arXiv: 1909.04939.

Ferrarin, M., Bovi, G., Rabuffetti, M., Mazzoleni, P., Montesano, A., Pagliano, E., Marchi, A., Magro, A., Marchesi, C., Pareyson, D., and Moroni, I. (2012). Gait pattern classification in children with Charcot–Marie–Tooth disease type 1A. *Gait & Posture*, 35(1):131--137.

Figueiredo, J., Santos, C. P., and Moreno, J. C. (2018). Automatic recognition of gait patterns in human motor disorders using machine learning: A review. *Medical Engineering & Physics*, 53:1--12.

Gamboa, J. C. B. (2017). Deep Learning for Time-Series Analysis. *arXiv:1701.01887 [cs]*. arXiv: 1701.01887.

Garcia, G. R., Michau, G., Ducoffe, M., Gupta, J. S., and Fink, O. (2020). Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms. *arXiv:2005.07031 [cs, eess, stat]*. arXiv: 2005.07031.

Goh, K., Lim, K., Gopalai, A., and Chong, Y. (2014). Multilayer perceptron neural network classification for human vertical ground reaction forces. In *2014 IEEE Conference on Biomedical Engineering and Sciences (IECBES)*, pages 536--540, Kuala Lumpur, Malaysia. IEEE.

Guennec, A. L., Malinowski, S., and Tavenard, R. (2016). Data Augmentation for Time Series Classification using Convolutional Neural Networks.

Hatami, N., Gavet, Y., and Debayle, J. (2017). Classification of Time-Series Images Using Deep Convolutional Neural Networks. *arXiv:1710.00886 [cs]*. arXiv: 1710.00886.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527--1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735--1780.

Horsak, B., Slijepcevic, D., Raberger, A.-M., Schwab, C., Worisch, M., and Zeppelzauer, M. (2020). GaitRec, a large-scale ground reaction force dataset of healthy and impaired gait. *Scientific Data*, 7(1):143.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-Excitation Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132--7141, Salt Lake City, UT. IEEE.

Hüsken, M. and Stagge, P. (2003). Recurrent neural networks for time series classification. *Neurocomputing*, 50:223--235.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. arXiv: 1502.03167.

Kantz, H. and Schreiber, T. (2003). Advanced embedding methods. In *Nonlinear Time Series Analysis*, pages 143--173. Cambridge University Press, Cambridge, 2 edition.

Karim, F., Majumdar, S., and Darabi, H. (2019). Insights into LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, 7:67718--67725. arXiv: 1902.10756.

Karim, F., Majumdar, S., Darabi, H., and Chen, S. (2018a). LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, 6:1662--1669.

Karim, F., Majumdar, S., Darabi, H., and Harford, S. (2018b). Multivariate lstm-fcns for time series classification.

Karimi-Bidhendi, S., Munshi, F., and Munshi, A. (2018). Scalable Classification of Univariate and Multivariate Time Series. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1598--1605, Seattle, WA, USA. IEEE.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

Kirtley, C. (2006). *Clinical gait analysis: theory and practice*. Elsevier, Churchill Livingstone, Edinburgh. OCLC: 845846752.

Köhle, M. and Merkl, D. (1996). Identification of gait patterns with self-organizing maps based on ground reaction force.

Köhle, M. and Merkl, D. (2000). Analyzing human gait patterns for malfunction detection. In *Proceedings of the 2000 ACM symposium on Applied computing - SAC '00*, pages 41--45, Como, Italy. ACM Press.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in artificial intelligence*, 5(4):221--232.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84--90.

Lakany, H. (2008). Extracting a diagnostic gait signature. *Pattern Recognition*, 41(5):1627--1637.

Last, M., Kandel, A., and Bunke, H., editors (2004). *Data mining in time series databases*. Number v.57 in Series in machine perception and artificial intelligence. World Scientific, New Jersey ; London. OCLC: ocm56760428.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521:436--44.

Levinger, P., Lai, D. T., Begg, R. K., Webster, K. E., and Feller, J. A. (2009). The application of support vector machines for detecting recovery from knee replacement surgery using spatio-temporal gait parameters. *Gait & Posture*, 29(1):91--96.

Lines, J. and Bagnall, A. (2014). Ensembles of Elastic Distance Measures for Time Series Classification. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 524--532. Society for Industrial and Applied Mathematics.

Lines, J., Taylor, S., and Bagnall, A. (2018). Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):1--35.

Liu, C.-L., Hsaio, W.-H., and Tu, Y.-C. (2019). Time Series Classification With Multivariate Convolutional Neural Network. *IEEE Transactions on Industrial Electronics*, 66(6):4788--4797.

Lozano-Ortiz, C. A., Muniz, A. M. S., and Nadal, J. (2010). Human gait classification after lower limb fracture using Artificial Neural Networks and principal component analysis. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 1413--1416, Buenos Aires. IEEE.

Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., and Webb, G. I. (2019). Proximity Forest: An effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607--635. arXiv: 1808.10594.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11--24.

Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification. *arXiv:1706.08838 [cs]*. arXiv: 1706.08838.

Marsland, S. (2009). *Machine Learning: An Algorithmic Perspective*. Chapman&Hall/CRC, 1st edition.

Mezghani, N., Husse, S., Boivin, K., Turcot, K., Aissaoui, R., Hagemeister, N., and de Guise, J. (2008). Automatic Classification of Asymptomatic and Osteoarthritis Knee Gait Patterns Using Kinematic Data Features and the Nearest Neighbor Classifier. *IEEE Transactions on Biomedical Engineering*, 55(3):1230--1232.

Muniz, A. and Nadal, J. (2009). Application of principal component analysis in vertical ground reaction force to discriminate normal and abnormal gait. *Gait & Posture*, 29(1):31--35.

Nayak, S., Misra, B., and Behera, D. H. (2014). Impact of data normalization on stock index forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*, 6:357--369.

Ngiam, J., Chen, Z., Chia, D., Koh, P. W., Le, Q. V., and Ng, A. Y. (2010). Tiled convolutional neural networks. pages 1279--1287.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks. *arXiv:1211.5063 [cs]*. arXiv: 1211.5063.

Perry, J. (1992). *Gait analysis: normal and pathological function*. SLACK, Thorofare, NJ. OCLC: ocm27816876.

Roh, Y., Heo, G., and Whang, S. E. (2019). A Survey on Data Collection for Machine Learning: a Big Data -- AI Integration Perspective. *arXiv:1811.03402 [cs, stat]*. arXiv: 1811.03402.

Slijepcevic, D., Horsak, B., Schwab, C., Gorgas, A.-M., Schüller, M., Baca, A., Breiteneder, C., and Zeppelzauer, M. (2017). Ground reaction force measurements for gait classification tasks: Effects of different PCA-based representations. *Gait & Posture*, 57:4--5.

Slijepcevic, D., Zeppelzauer, M., Gorgas, A.-M., Schwab, C., Schuller, M., Baca, A., Breiteneder, C., and Horsak, B. (2018a). Automatic Classification of Functional Gait Disorders. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1653--1661.

Slijepcevic, D., Zeppelzauer, M., Schwab, C., Raberger, A.-M., Breiteneder, C., and Horsak, B. (2020). Input representations and classification strategies for automated human gait analysis. *Gait & Posture*, 76:198--203.

Slijepcevic, D., Zeppelzauer, M., Schwab, C., Raberger, A.-M., Dumphart, B., Baca, A., Breiteneder, C., and Horsak, B. (2018b). P 011—Towards an optimal combination of input signals and derived representations for gait classification based on ground reaction force measurements. *Gait & Posture*, 65:249--250.

Smirnov, D. and Nguifo, E. M. (2018). Time Series Classification with Recurrent Neural Networks.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv:1206.2944 [cs, stat]*. arXiv: 1206.2944.

Soares, D. P., de Castro, M. P., Mendes, E. A., and Machado, L. (2016). Principal component analysis in ground reaction forces and center of pressure gait waveforms of people with transfemoral amputation. *Prosthetics and Orthotics International*, 40(6):729--738.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929--1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*. arXiv: 1409.4842.

Valdés-Sosa, P., Sanchez, J. M., Vega-Hernández, M., Melie-Garcia, L., Castellanos, A., and Canales-Rodríguez, E. (2006). Handbook of Time Series Analysis: Recent Theoretical Developments and Applications. In *Handbook of Time Series Analysis: Recent Theoretical Developments and Applications*, pages 461 -- 491.

Verstraete, D., Ferrada, A., Droguett, E. L., Meruane, V., and Modarres, M. (2017). Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings. *Shock and Vibration*, 2017:1--17.

Vieira, A., Sobral, H., Ferreira, J. P., Ferreira, P., Cruz, S., Crisostomo, M., and Coimbra, A. P. (2015). Software for human gait analysis and classification. In *2015 IEEE 4th Portuguese Meeting on Bioengineering (ENBENG)*, pages 1--1, Porto, Portugal. IEEE.

Wallot, S. and Mønster, D. (2018). Calculation of Average Mutual Information (AMI) and False-Nearest Neighbors (FNN) for the Estimation of Embedding Parameters of Multidimensional Time Series in Matlab. *Frontiers in Psychology*, 9:1679.

Wang, Z. and Oates, T. (2015a). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks.

Wang, Z. and Oates, T. (2015b). Imaging Time-Series to Improve Classification and Imputation. *CoRR*, abs/1506.00327. _eprint: 1506.00327.

Wang, Z., Yan, W., and Oates, T. (2016). Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *arXiv:1611.06455 [cs, stat]*. arXiv: 1611.06455.

Williams, G., Lai, D., Schache, A., and Morris, M. E. (2015). Classification of Gait Disorders Following Traumatic Brain Injury:. *Journal of Head Trauma Rehabilitation*, 30(2):E13--E23.

Wu, J. and Wang, J. (2008). PCA-Based SVM for Automatic Recognition of Gait Patterns. *Journal of Applied Biomechanics*, 24(1):83--87.

Wu, J., Wang, J., and Liu, L. (2007). Feature extraction via KPCA for classification of gait patterns. *Human Movement Science*, 26(3):393--411.

Wu, Y., Chen, P., Luo, X., Wu, M., Liao, L., Yang, S., and Rangayyan, R. M. (2017). Measuring signal fluctuations in gait rhythm time series of patients with Parkinson's disease using entropy parameters. *Biomedical Signal Processing and Control*, 31:265--271.

Yazdanbakhsh, O. and Dick, S. (2019). Multivariate Time Series Classification using Dilated Convolutional Neural Network. *arXiv:1905.01697 [cs, stat]*. arXiv: 1905.01697.

Zeng, W., Liu, F., Wang, Q., Wang, Y., Ma, L., and Zhang, Y. (2016). Parkinson's disease classification using gait analysis via deterministic learning. *Neuroscience Letters*, 633:268--278.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162--169.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. (2016). Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science*, 10(1):96--112.

Appendices

A. Thesis summary

With the recent advancements and success of machine learning techniques for automated classification of data, several attempts have been made to employ such methods in the field of gait analysis. Various studies have been conducted, often making use of both kinetic and kinematic data obtained from measurements using a 3-dimensional gait analysis system. However, due to the relatively high cost and time requirements of such a device, it is not commonly used in clinical practice. Instead, experienced clinicians often rely on a simple quantification of Ground Reaction Force (GRF) measurements in combination with a visual inspection in order to assess the clinical status of the analysed motor function. An automated approach for classifying GRF-measurements would not only be able to support clinicians in their diagnosis, but could additionally be used for planning and coordinating the treatment by personalizing the gait training according to the recognized movement disorder.

The most exhaustive research (focussing solely on GRF measurements obtained by using force plates) conducted in the field so far, tries to map gait impairments to the location (the joint) of the corresponding injury using the five main classes: *Healthy - Hip - Knee - Ankle - Calcaneus*, providing a non-invasive option for assessing locomotive functions. However, the problem remains a difficult one, with the highest classification accuracies reported (62%, obtained by employing a Support Vector Machine) being lower than desirable in a clinical setting. Because temporal dependencies and relationships between the individual components of GRF-measurements have been neglected in previous attempts, this thesis tries to expand on those approaches by employing different versions of neural networks for the classification task. The main advantage of such advanced architectures is their ability to operate on multivariate input, enabling them to take advantage of correlations between the different signals. Additionally, those methods have been specifically developed (e.g. LSTM - Long Short-Term Memory) or adapted (e.g. Convolutional Neural Networks - CNNs) for time-series data, promising to extract a maximum of information from the available measurements. The networks evaluated in this research have either been previously used for GRF data (but with a different classification goal) or have shown good performance across various time-series data, making it likely that they will achieve high accuracy on the investigated task.

Experiments have been conducted using the *GaitRec*-data, using the provided balanced dataset for training the classifiers and the independent test set for evaluating their performance. Parameters for the individual architectures have been estimated by splitting the data used for training into a train- (80%) and validation-set (20%), using the model achieving the highest accuracy on

the validation-set for further testing. The following four main components of neural networks have been verified, independently and in combination, to determine the best architecture for the GRF classification task: *(1)* Multi-Layer Perceptron (MLP), *(2)*1-dimensional convolution (1D-CNN), *(3)* 2-dimensional convolution (2D-CNN) and *(4)* LSTM. Additionally, more complex models proposed for time-series classification such as *FCN* (Fully Connected Network), *ResNet* (Residual Network) and *InceptionTime*, and time-series to image transformations have been investigated as well.

However, experiments revealed that the advanced feature extraction ability of such complex techniques is not able to detect beneficial patterns within the data, showing minimal to no increase in accuracy when compared to much simpler methods. Generally, it seems like the amount of information contained within the GRF-measurements is highly redundant, with only a few data-points actually contributing to the performance of the classifier. None of the more sophisticated networks is able able to outperform a rather simple single-layered MLP (achieving the highest observed accuracy of 61.91%) on the independent test-set, suggesting that there is no advantage to analysing temporal and inter-signal dependencies of the data. On the contrary, the conducted research indicates that the amount of data-points can be reduced by a factor of ten without a significant drop in accuracy, achieving almost identical performance on the validation-set. Furthermore, even though the convolutional approaches are able to score a higher accuracy on the validation-set (presumably due to *overfitting*), no model is able to surpass the accuracy threshold of 74%, although several different architectures achieve 73.97%. Such behaviour could signify that the optimal solution for the validation-set has been found and no further increase might be possible using neural networks.

Despite those slightly surprising results concerning general classification accuracy, evaluation of different pre- and post-processing methods has been able to reveal some insights into the problem itself, indicating promising field for future research. While the performance seems to be independent of the input format and neural network type, experiments demonstrate that additional normalization of the input is advisable, resulting in higher scores on validation- and test-set. Additionally, some positive effects when training with all individual trials instead of the mean waveform have been observed, possibly leading to a more robust classifier when employed. Especially in combination with *late fusion* of individual trials via majority-voting, higher accuracies have been achieved on various architectures. While the aggregation of trials after the actual classification is clearly advantageous, more research is necessary in order to determine if a better classification result can be achieved when both methods are combined.

B. Zusammenfassung der Masterarbeit

Aufgrund der raschen Entwicklung und dem fortschreitenden Erfolg von künstlicher Intelligenz (besonders im Bereich des sogenannten *Deep Learnings*), ist es nicht weiter verwunderlich dass zahlreiche Versuche unternommen wurden, diese Methoden auch für die Ganganalyse zu nutzen. Viele Forschungsprojekte zu diesem Thema verwenden dabei sowohl kinetische als auch kinematische Daten, die mittels eines 3D Ganganalysesystems erhoben werden. Aufgrund der hohen Anschaffungskosten und der relativ zeitintensiven Messungen, finden solche Systeme jedoch nur selten Anwendung im klinischen Alltag. Viel verbreiteter ist die Analyse der Bodenreaktionskräfte (erhoben mittels Kraftmessplatten) gekoppelt mit einer visuellen Inspektion durch erfahrenes Personal. Eine derartige Analyse könnte erheblich von einer automatischen Klassifizierung solcher Daten profitieren, die dazu in der Lage wäre quantitative und nachvollziehbare Information sowohl für die Diagnose als auch für die Planung und Kontrolle des Rehabilitationsprozesses beizusteuern.

Die bisher größte Studie, welche sich ausschließlich mit der Klassifikation von Bodenreaktionskräften beschäftigt, versucht eine Verbindung zwischen der jeweiligen Beeinträchtigung und dem von der Verletzung betroffenen Gelenk herzustellen. Dabei wird zwischen den folgenden fünf Klassen zu unterschieden: *Gesund - Hüfte - Knie - Knöchel – Calcaneus*. Eine solche Klassifizierung erweist sich jedoch als schwierig und die Genauigkeit der Methode (62% unter Benutzung einer *Support Vector Machine*) ist noch erheblich unter der erstrebenswerten Anforderung für eine medizinische Anwendung. Da zeitliche Abhängigkeiten und Beziehungen zwischen den einzelnen Komponenten einer Bodenreaktionskraftmessung in den bisherigen Studien nicht berücksichtigt wurden, wird im Rahmen dieser Masterarbeit versucht diese zusätzliche Information mithilfe neuronaler Netze zu analysieren um eine höhere Genauigkeit zu erzielen. Dabei kommt eine Anzahl speziell entwickelter (z.Bsp. LSTM - Long Short-Term Memory) oder adaptierter (z.Bsp. Convolutional Neural Networks – CNNs) Methoden für die Zeitreihenanalyse zum Einsatz. Die angewendeten Modelle wurden entweder bereits erfolgreich für Bodenreaktionskraftdaten getestet oder zeigten gute Ergebnisse bei Experimenten mit unterschiedlichen Arten von Zeitreihen und repräsentieren den gegenwärtigen Stand der Forschung in diesem Bereich.

Zur Evaluation dieser Netzwerkarchitekturen wurde die *GaitRec*-Daten herangezogen, welche sowohl einen balanzierten Trainingsdatensatz als auch unabhängige Testdaten zur Verfügung stellen. Um die optimalen Parameter der individuellen Architekturen zu erheben, wurde der Trainingsdatensatz in ein Trainings- (80%) und Validierungs-set (20%) geteilt, und Kandidaten für weitere Tests wurden anhand der erzielten Genauigkeit am Validierungs-set ausgewählt. Die folgenden vier Komponenten neuronaler Netzwerke wurden (unabhängig und in Kombination) getestet um das beste Modell für Bodenreaktionskraftdaten zu finden: *(1)* Multi-Layer Per-

ceptron (MLP), *(2)*1-dimensional convolution (1D-CNN), *(3)* 2-dimensional convolution (2D-CNN) and *(4)* LSTM. Zusätzlich wurden komplexere Modelle, die große Erfolge in der multivariaten Zeitreihenanalyse erzielen konnten implementiert (*FCN* - Fully Connected Network, *ResNet* - Residual Network und *InceptionTime*) und auch Umwandlungen von Zeitreihen zu Bildern (um von der hohen Genauigkeit aktueller Bilderkennungsanwendungen zu profitieren) untersucht.

Die durchgeführten Experimente zeigen jedoch dass auch solche komplexen Methoden nicht dazu in der Lage sind, weitere Muster in den Daten zu identifizieren und die erzielte Genauigkeit kann nur minimal bis gar nicht gesteigert werden. Generell erscheint der Informationsgehalt der Daten größtenteils redundant zu sein, da nur wenige Datenpunkte dazu in der Lage sind die Qualität der Klassifizierung zu beeinflussen. Das beste Ergebnis am unabhängigen Testset wurde von einem relativ simplen Netzwerk (61.91%, MLP mit nur einem Layer) erzielt und konnte von keiner der anderen Methoden übertroffen werden. Dies legt nahe, dass weder in den zeitliche Abhängigkeiten noch in den Korrelationen zwischen den Komponenten relevante Information enthalten ist, die für eine bessere Klassifikation genutzt werden könnte. Im Gegenteil, erste Tests mit reduzierten Datensätzen zeigen dass die Anzahl der Datenpunkte auf mindestens ein Zehntel reduziert werden kann, ohne signifikant an Genauigkeit zu verlieren. Das relativ gute Abschneiden der CNNs auf dem Validierungs-set, bei dem mehrere verschiedene Architekturen eine Genauigkeit von 73.97% erreichen, suggeriert ebenfalls einen limitierten Informationsgehalt. Da keines der Modell dazu in der Lage ist die Schwelle von 74% zu überschreiten, könnte dies bedeuten dass keine (global) bessere Lösung existiert und somit auch kein besseres Ergebnis erzielt werden kann.

Trotz dieser etwas enttäuschenden Resultate in Bezug auf die Klassifikationsgenauigkeit, zeigen die Experimente dass entsprechende Schritte bei der Vor- und Nachbearbeitung der Daten dazu in der Lage sind das Resultat zu beeinflussen. Während sowohl das Eingabe-Format als auch der Netzwerktyp nur eine untergeordnete Rolle spielen, zeigen die Versuche deutlich bessere Ergebnisse wenn die Daten erneut normalisiert wurden, und zwar unabhängig von der dabei gewählten Methode. Zusätzlich konnten positive Effekte beobachtet werden wenn mit sämtlichen Daten (die während eines einzelnen Ganganalyse-Termins erhoben wurden) gearbeitet wurde anstatt den Mittelwert der individuellen Messungen zu verwenden. Insbesondere in Kombination mit einer späten Aggregation (d.h. Klassifikation jedes einzelnen Trials) via Mehrheitsentscheidung ist es möglich die Genauigkeit zusätzlich zu steigern. Während der Einfluss der Mehrheitsentscheidung durchgehend positiv ausfällt, ist bezüglich der verschiedenen Kombinationen der Fusionierung individueller Trials weitere Forschung notwendig um deren Effekte und Auswirkungen zu verstehen.