Ludwig-Maximilians-Universität München
Faculty for Mathematics, Computer Science and Statistics

Master Thesis

Wearable-based Severity Detection in the Context of Parkinson's Disease Using Deep Learning Techniques

---

Author:

Jann Goschenhofer

Supervisors:

Prof. Dr. Bernd Bischl

Janek Thomas, M.Sc.

4th January 2019

Abstract

One major challenge in the medication of patients with Parkinson's disease is that the severity of the disease, reflected in the patients' motor state, can not be measured using accessible biomarkers. In this thesis we therefore develop and examine statistical models to detect the motor state of those patients based on sensor data from a wearable device on a minute level. We find that deep learning models consistently outperform classical machine learning models applied on hand-crafted features in this time series classification task. Furthermore, our results suggest that treating this problem as a regression instead of an ordered regression or a classification task works best. For consistent model evaluation and training, we adapt the Leave-one-subject-out validation scheme to the training of deep learning models. We also employ a class-weighting scheme to successfully handle the problem of high class imbalances in this domain. In addition, we propose a customized performance measure that reflects the requirements of the involved medical staff on the model. To mitigate the problem of limited availability of high quality training data, we successfully employ a transfer learning technique which helps to improve the final model performance substantially. Since trust in the model's predictions plays a crucial role in the adoption of those techniques, we additionally extend our model to quantify prediction uncertainty. Our results suggest that deep learning techniques offer a high potential to automatically detect motor states of patients with Parkinson's disease with great success.

Contents

1 Introduction

Parkinson's disease (PD) is one of the most common diseases of the elderly and, after Alzheimer's, the second most common neurodegenerative disease in general (Sama et al. 2012). Two million Europeans are affected (Ahlrichs and Lawo 2013) and Pringsheim et al. (2014) estimate 1% of the population aged 60 or older in industrial nations to suffer from PD. PD in its core is a protein disorder involving alpha-synuclein that is misfolded and leads to so called Lewy-inclusion bodies (Braak et al. 2003). These specific proteinaceous agglomerations lead to cell-death primarily involving dopaminergic neurons in the substantia nigra in the brain stem (Jellinger 2009). The resulting lack of Dopamine creates a specific motor disorder, that is parkinsonism (Berardelli et al. 2001). Lack of Dopamine is associated with the inability of the basal ganglia to create sufficient movement amplitude and speed, especially for overlearned motor activities. Key symptoms of parkinsonism are bradykinesia and hypokinesia. Additionally, tremor is observed in 30% to 80% of the PD patients (Hoehn and Yahr 1998).

The motor capabilities of the patients are distinguishable into different states, termed motor states. These can vary substantially over the course of a day within hours, as the main therapeutic relief, a medication named L-DOPA, has a short plasma half-life of only 80 minutes. The most prominent symptom is the tremor, a small but heavy shaking of the hand which comes along with partial paralysis of the patient's body (Jankovic 2008) and is also referred to as bradykinesia. In contrast to bradykinesia, an overpresence of dopaminergic medication can make affected patients execute involuntary choreatic movement patterns which may remind the untrained observer of a bizarre dance. This hyperkinetic motor state is termed dyskinesia (Tsipouras et al. 2011). In a very basic approximation, a patient with Parkinson's disease (PxPD) can be in three motor states: 1) the bradykinetic state, also referred to as the OFF state, 2) a seemingly normal state, also referred to as the ON state, and 3) the dyskinetic state.

Theoretically, if the true motor state of PxPD was known at all times, the medication dose could be optimized in such a way that the patient has an improved chance to spend the entirety of his waking day in the ON state. An example for such a closed-loop approach can be found in Diabetes therapy. Patients suffering from Diabetes can continuously measure their blood sugar level and apply the individual, correct medication dose of Insulin in order to balance the disease. This is due to the fact that the blood sugar level serves as a biomarker for Diabetes which can be measured with low effort by the patient herself. In contrast, there is not yet a validated method that allows for effortless, autonomous and continuous monitoring of the PD motor state. Such monitoring could theoretically be provided by a medical expert who evaluates the patient in a continuous fashion, which is economically and practically not feasible. However, an inexpensive, autonomous, precise method to assess the PD motor state might allow for major improvements in customized, individual medication of the PxPD ultimately in order to prolong healthy motor state phases resulting in a tremendous increase in the patients' quality of life.

Against this backdrop, we approach to fill this gap and try to evaluate the motor state using a convenient wearable device located at the patients' wrist. Therefore, we collected data from PxPD in a free-living setting and let medical staff evaluate the patients to obtain high quality labels of the data on a one minute level. Moreover, we propose methods to model the relation between this raw sensor data and the disease severity labels of the PxPD. What is more, we introduce a variety of deep learning models which we compare with standard machine learning techniques as well as a simple baseline model. We discuss the framing of this problem as a regression, ordered regression or classification task. Finally, we propose a transfer learning approach and we also show an extension of these models to quantify prediction uncertainty.

Advancements in both, wearable devices equipped with motion sensors and statistical modelling tools accelerated the scientific community in researching solutions for motor state detection of PxPD since early 2000s. Wearable devices with embedded motion sensors allow the measurement of patients' movement behavior and the use of algorithms to map this input to different motor states. Ghika et al. (1993) did pioneering work in this field by proposing a first computer-based system for tremor measurement. Ahlrichs and Lawo (2013) offer a comprehensive overview on the use of machine learning and wearable devices in a variety of PD related problems. Lonini et al. (2018) report that the use of one wrist-located sensor is sufficient for this task and that it is more beneficial to collect few data from many patients than many data from few individuals. Parts of the literature in this domain focuses on the manual extraction of features from the sensor data that are then fed into machine learning algorithms such as Support Vector Machines (SVM) (Hssayeni et al. 2018; Sama et al. 2012) or vanilla Neural Networks (Hammerla et al. 2015; Keijsers, Horstink, and Gielen 2006; 2003; Tsipouras et al. 2011). Hssayeni, Burack, and Ghoraani (2016) compare the performance of SVMs, k-means classification and the self-organizing tree map algorithm based on extracted features with deep learning models that take raw sensor data as an input. The deep learning based models perform best in their setup. In addition, Eskofier et al. (2016) compare traditional machine learning models such as SVM, AdaBoost, PART and k-nearest neighbor classification with the use deep Convolutional Neural Networks (CNNs). The deep learning models show the most promising results in their case. Um et al. (2018) also apply CNNs on this task, using the same data set that was used in this thesis but do not compare their model with alternative algorithms. In table 9, we present a structured overview of this related literature. In the present setting, where many data is collected from few individuals, a leave-one-subject-out (LOSO) validation is necessary to yield unbiased performance estimates of the models (Saeb et al. 2017). Thus, it is surprising that only a minority of the reviewed literature (Eskofier et al. 2016; Hssayeni, Burack, and Ghoraani 2016; Hssayeni et al. 2018; Tsipouras et al. 2011; Um et al. 2018) deploy a valid LOSO validation scheme. It is also noteworthy, that only Keijsers, Horstink, and Gielen (2003) propose modelling approaches with a continuous response, while the rest of the literature tackles this problem as a classification task to distinguish between the different motor

states. This is remarkable, as medical doctors need information about the patients on a more granular level in order to optimize their medication. Amongst the deep learning approaches, it is surprising that none of the related investigations offer a valid method to detect the optimal amount of training epochs for the model. Um et al. (2018) track the performance on the validation set whilst training. Then, they take the epoch with the lowest training error as the optimal amount of training time, which bares the problem of overfitting. Eskofier et al. (2016), Keijsers, Horstink, and Gielen (2003), Hammerla, Halloran, and Ploetz (2016) and Hammerla et al. (2015) do not even mention the training time or set the values arbitrarily. Within this work, we try to close the main literature gaps in this domain: we discuss the optimal framing of this problem, propose a valid LOSO validation strategy for iterative algorithms such as CNNs, and compare deep learning models with classical machine learning approaches. In addition, we investigate different model architectures for time series classification (TSC) that apply to the present task. In the following, we give an overview on relevant literature in this domain.

TSC, the classification of data that contains a temporal component, was announced as one of the most challenging data mining problems in the last century by Yang and Wu (2006). Its applications occur in a broad range of fields from economics (Caiado, Crato, and Peña 2006), the analysis of electronic health records (Rajkomar et al. 2018), to human activity recognition (Ronao and Cho 2016). TSC can be distinguished in two different subgroups: 1) traditional machine learning methods and 2) deep learning based methods. In addition, it is noteworthy that we use the term TSC for both, the modeling of a categorical as well as a metric response.

In a meta survey, Bagnall et al. (2017), who also maintain the popular UCR Time Series Dataset (Dau et al. 2018), give a broad overview on classical machine learning TSC techniques. The most prominent representatives of this model class are Dynamic Time Warping (DTW) (Rakthanmanon et al. 2013), Shapelet Transform (Hills et al. 2014) and Collection of Transformation Ensembles (COTE) by Bagnall et al. (2015). DTW is a location- and scale-invariant distance measure for sequence data and often used in combination with a Nearest Neighbor classifier as a standard instance-based benchmark classifier in TSC (Fulcher and Jones 2014). Shapelet Transform is a distance-based feature extraction algorithm that leverages sub-sequences of time series as features for classification tasks. COTE creates an ensemble of 37 different classifiers over different features extracted automatically from the time series. Despite the strong performance of these algorithms, they share a huge computational burden which makes them unpractical in a setting with multivariate time series of great length (Fawaz et al. 2018a), as is the case in this thesis. Furthermore, there are also feature-based traditional machine-learning approaches in TSC where features such as variance, mean and skewness are extracted from (multivariate) time series (Casale, Pujol, and Radeva 2011; Christ, Kempa-Liehr, and Feindt 2016; Nanopoulos, Alcock, and Manolopoulos 2001; Wang, Wirth, and Wang 2007) and then passed to any classifier. More advanced features such as self-similarity and trend are also employed (Wang, Smith, and Hyndman 2006). Fulcher and Jones (2014) offer a comprehensive

review of feature-based TSC methods. For the sake of comparability, a machine learning model trained on a set of handcrafted features is used as a baseline model within this thesis.

While distance-based methods are resource-consuming during model inference, feature-based methods require a domain expert to manually engineer meaningful features. This leads to labour intensive, problem tailored models that are hard to adapt to different tasks. In contrast to that, deep learning methods promise to mitigate these problems by their inherent feature extraction property and they therefore become more and more adopted in the TSC community. Fawaz et al. (2018a) provide a comprehensive overview on deep learning techniques for TSC comparing nine deep learning architectures on the UCR time series dataset (Dau et al. 2018) and on a multivariate data set by Baydogan (2017). Amongst them is the Fully Convolutional Net (FCN) by Wang, Yan, and Oates (2017), a one-dimensional CNN which contains Global Average Pooling (GAP) instead of regular max-pooling. Furthermore, they reference a variant of the FCN variant that employs skip connections inspired by the ResNet for image classification (He et al. 2015). Also, the encoder model by Serrà, Pascual, and Karatzoglou (2018), another extension of the FCN which replaces the GAP layer with an Attention layer, is part of the study. Karim et al. (2018b) and Karim et al. (2018a) propose the combination of a FCN with a Long-Short-Term-Memory (LSTM) which shows discussion-worthy results due to unorthodox application of the LSTM layer. Furthermore, Wang et al. (2018) introduce a stacked architecture based on the use of subsequent wavelet-decompositions. This follows the rationale that the different parts of the net can focus on signal representations with varying resolution. Yang et al. (2015) propose a deep architecture for multivariate time series that fuses the different channels at a later stage. Yet, they do not compare their work with alternative deep learning architectures. Bai, Kolter, and Koltun (2018) introduce the temporal convolutional net which consists of stacked CNN layers that use different levels of dilations to increase the receptive field of the net allowing it to take into account more global information. Their work is a simplification of the WaveNet by Oord et al. (2016) which was initially used for sequence-to-sequence prediction. The most promising of these models are employed in this thesis and will thus be further explained in Section 4.

The remainder of this thesis is organized as follows: we introduce the data that was used in this study in the upcoming section 2. In section 3, we discuss the different optional frames for the present problem and the models that were experimented with are introduced in section 4. The results of the different modelling approaches are discussed in section 5 and in section 6 we show an extensional approach for the quantification of prediction uncertainty within this task. Finally, we summarize the thesis in section 7 and give an outlook on future research in this domain.

2 Data

As introduced in chapter 1, we gathered data from various PxPD to model the relation between raw movement sensor data and the severity of Parkinson's disease. This way, we collected three Parkinson's related data sets that were used throughout this thesis. We describe the measurement devices, the preprocessing steps, the labeling process and the three data sets in the following.

2.1 Measurement Device

Inertial measurement units (IMUs) contain different sensors that allow for the continuous recording of movement patterns and are standard components of wearable devices such as smart watches, smartphones or fitness trackers. The two main sensors of such IMUs are an accelerometer and a gyroscope which respectively track the acceleration and the rotation of the body in three spatial dimensions. Within this study, two wrist-worn wearable IMU devices were employed: the Mircosoft band 2 (MSB2) (Microsoft 2018), a regular fitness tracker, and the Shimmer device (Shimmer 2018) which is explicitly designed for use in medical studies. While the Shimmer device stores data locally on an SD-card, the Microsoft band 2 sends recorded data packages to a bluetooth-connected receiving device, making it more prone to data issues due to problems with the wireless connection. Both devices measure acceleration and rotation on the X, Y and Z axis with a standard frequency of 62.5Hz. Whilst some literature suggests the ankle, the leg or the torso as location for the sensors (e.g.: Hssayeni et al. (2018); Keijsers, Horstink, and Gielen (2006); Tsipouras et al. (2011)), the wrist was chosen as the only sensor location in this thesis as it is the most comfortable location for a wearable to be used in the patients' daily life. This is supported by the work of Lonini et al. (2018), who show that the usage of a single wrist-worn sensor is sufficient for the detection of Parkinson-related symptoms.

2.2 Preprocessing

Due to problems with the bluetooth connection, the measurement points of the MSB2 were partially not equi-distant as some data packages were sent to the receiver device with time delay. As a solution, we resampled the data via linear spline interpolation. We chose a resampled, final frequency of 20Hz in alignment to the work of Hammerla et al. (2015) who suggest that Parkinson's related patterns do not exceed this frequency. A standard procedure in human activity recognition is the segmentation of continuous sensor data streams into smaller subsamples, also termed windows. As the data in this study was annotated by a medical doctor on a minute-level, we also segmented the data in windows of one minute length. With a frequency of 20Hz, this results in 1200 measuring points per variable per window. The availability of lots of training data is crucial for the use of deep learning algorithms. Thus, we segmented the windows with an overlap of 80% between neighbouring windows for training data set augmentation. This augmentation led to an increase in training data by almost the factor of 5 and this

technique is widely used in sensor data based prediction problems (Eskofier et al. 2016; Zeng et al. 2014) and (Hammerla, Halloran, and Ploetz 2016). It is important to note that data set augmentation was done for the training data only. In Human Activity Recognition (HAR), one tries to detect which activity a subject is performing. In contrast to that, we want to detect how a subject is performing activities as we are interested in the patients' underlying motor states. In the optimal case, the developed models should be able to detect the motor states irrespective of the patients' specific activities. Both sensors record the movement of the patients in all three spatial dimensions X, Y and Z. The direction of the movement might be important for HAR related tasks. In contrast, this is not the case for detecting the way in which those tasks are executed where this information might even be confusing for the models. Thus, we decided to use the L_2-norms of the accelerometer and gyroscope as model input only. Furthermore, we normalized the data to a $[0, 1]$ range using the quantile transformation as implemented in the scikit learn package (Pedregosa et al. 2011). We chose this normalization method as it is less sensitive to outliers, compared to standard procedures such as the min-max normalization, and maps all values to a range of $[0, 1]$, which is a standard procedure in working with deep neural networks. In addition to that, our first empirical experiments showed the tendency that quantile normalized data allows more stable model training compared to alternative methods such as robust scaling, min-max scaling or z-normalization.

2.3 Labels for Disease Severity

The disease severity is measured on a scale that is a combined version of the UPDRS (Goetz et al. 2008) and the mAIMS scale (Lane et al. 1985). The UPDRS scale is based on a diagnostic questionnaire for physicians to rate the severity of the bradykinesia of PxPD on a scale from $\{0, ..., 4\}$ with 0 representing the healthy and 4 the heavily bradykinetic state. The mAIMS scale is similar to the UPDRS but in contrast used for the diagnosis of dyskinetic patients on a scale from $\{0, ..., 4\}$. It as well increases from the healthy state towards heavy dyskinesia. In this thesis, both scales were combined and the UPDRS scale was flipped to cover the whole disease spectrum. The resulting label scale takes values in $\{-4, ..., 4\}$ where $y = -4$ means a patient is in a heavily bradykinetic state, $y = 0$ is assigned to a patient in a healthy state and $y = 4$ resembles a heavy dyskinetic motor state. The sensor data was labeled by a medical doctor who shadowed the PxPD during one day in a free living setting. Thus, the rater monitored each patient while the patient lived through their regular day doing regular activities and the rater diagnosed and recorded the patients' motor state in each minute. Meanwhile, the patients wore the IMU on their right wrist, constantly measuring their movement behavior.

2.4 Data Sets

Throughout this thesis, the following three data sets are used:

1. data set I: the main dataset containing movement patterns and labels for the minute-wise disease-severity for 28 patients.

2. data set II: the exogeneous test data set which contains movement patterns and severity labels for 10 patients.

3. data set III: the data set used for the transfer learning approach. It contains movement data from 50 patients with Parkinson's disease and 50 healthy controls.

In the following, we describe the key properties of the three data sets,

2.4.1 Data Set I

Data was collected from 28 PxPD in April 2016 using a MSB2 IMU. In total, 9356 samples with a window length of one minute and two variables, the norm of the accelerometer and gyroscope respectively, were extracted from the data of those 28 patients. Figure 1 shows examples of the input data for three different classes. The median value of collected windows per patients is 368, the minimum value is 47 and at most 700 minutes were collected from one patient. The label distribution of this data set I suffers from high label class imbalance as shown in figure 2. Similar to Um et al. (2018), the patients within this data set I were on average 67 (std. dev.: 10) years old and suffered from the disease since 11 years (std. dev.: 5) at the time of measurement. It is important to note, that there exist two versions of this data set: the data set I that was augmented using the strategy from chapter 2.2 that is only used for training and the original, non-augmented data set that was used for testing only. We will refer to the latter as LOSO data set I in the following. The augmented training data sets contains 45944 windows.

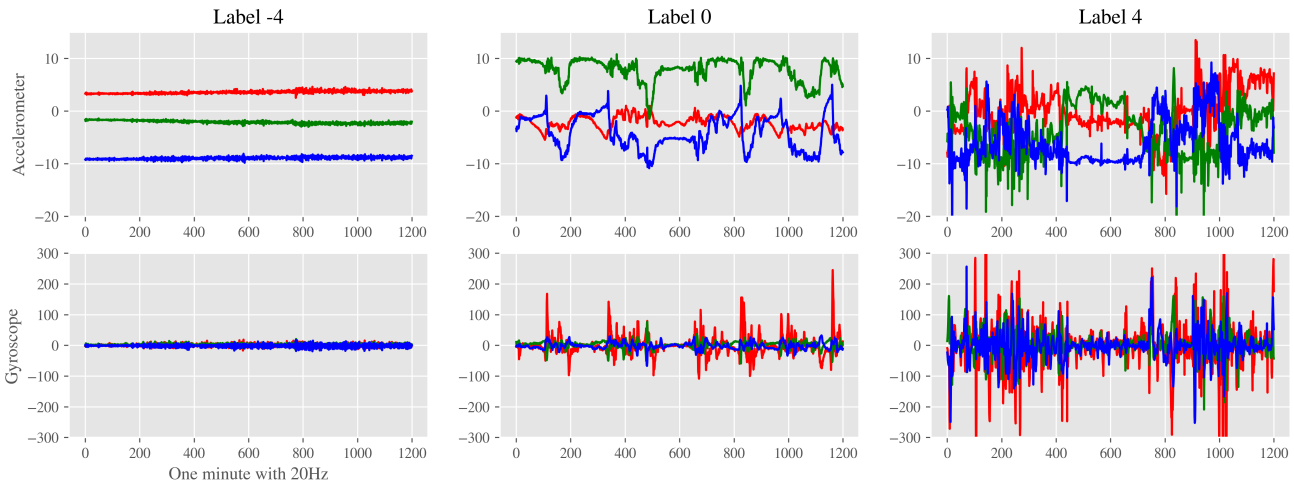


Figure 1: Input sensor data from data set I of examples for the three extreme classes -4 (bradykinesia), 0 (ON) and 4 (dyskinesia). Accelerometer data streams are shown in the first and gyroscope measurements are shown in the second row. Each window consists of 1200 measurement points, depicted on the x-axis of each of the six plots. Acceleration is measured in the unit g where $1g = 9.81m/s^2$ and rotation is measured as angular velocity in degrees per second $°/sec$.

2.4.2 Data Set II

The second data set was collected from 10 patients in spring 2018 following a protocol similar to that of data set I. One main difference, next to the measurement with a Shimmer instead

of a MSB2 device, is the smaller duration of the monitoring. The median value of collected windows per patients is 24, the minimum value is 8 and at most 85 minutes were collected from one patient of data set II. Furthermore, the label distributions for both data sets differ quite heavily, which limits the comparability of the model performance between the two data sets I and II. This is illustrated further in figure 2. Still, we report results for the models trained on data set I on data set II as this offers the external validation of the models.

2.4.3 Data Set III

Next to the supervised prediction approach, we evaluate the use of transfer learning methods on the problem at hand. Therefore, we use this data set III which also consists of sensor-data measured for 40 patients with an MSB2 device. Among those patients, the half is healthy and the other half suffers from Parkinson's disease and the patients wore the measurement device also whilst performing regular activities. The difference with respect to data sets I and II lies in the labels which are binary, stating if a window belongs to a healthy or an unhealthy patient. The labeling effort for this data set is obviously a lot smaller than for the data sets covering the granular disease severity. In total, this data sets contains 70175 windows, of which 52.20 % belong to healthy patients. This data set was solely used for the pretraining of the models as described in more detail in chapter 4.3.

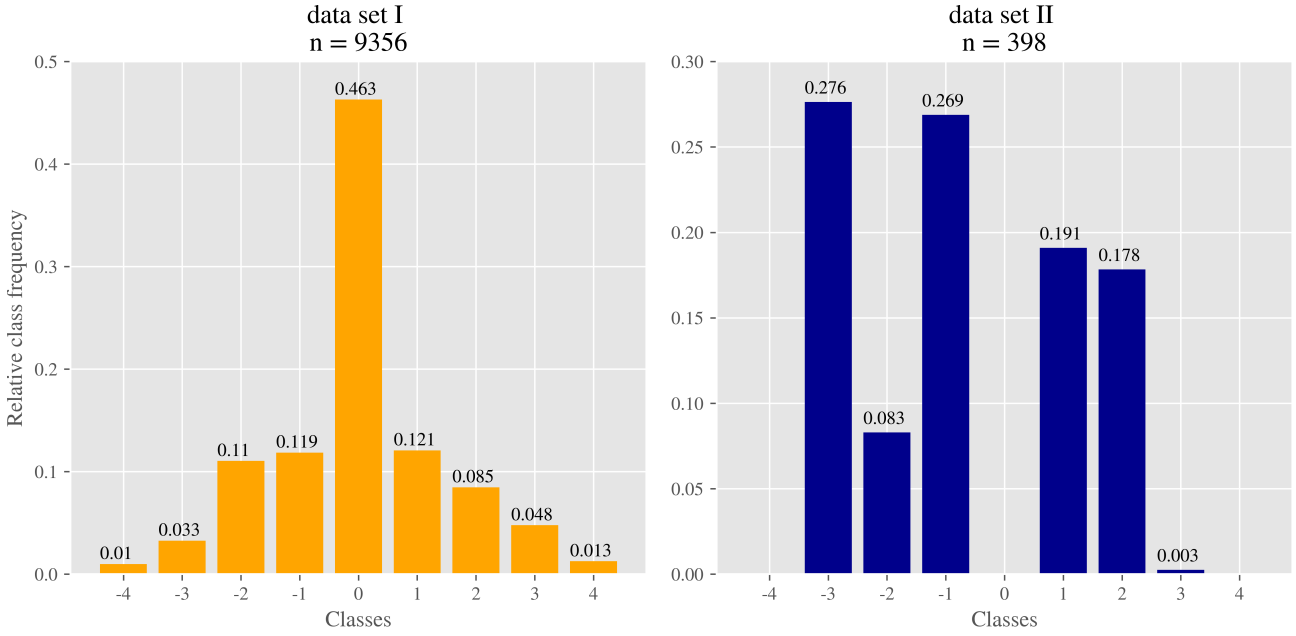


Figure 2: Label distribution of both disease severity data sets I and II. In data set I, the label distribution is highly centered around the middle class 0, the healthy state for PxPD. In contrast, the data collected within data set II show not a single class 0 sample.

3 Problem Frame

In this chapter, we discuss two peculiarities of the present problem: 1) the perception of the task as a classification, an ordered regression or a metric regression problem and 2) the high class imbalance in the data.

3.1 Framing of the Task

The goal of this thesis is the mapping of raw movement sensor-data to the severity of the disease, which is measured on a discretized scale with values $y \in \mathcal{J} = \{-4, ..., 4\}$ where $\mathcal{J}$ is referred to as target space in the following. This gives rise to the discussion on whether this problem should be taken as a classification, an ordered regression or a regression task. Hence, one purpose of this thesis is to compare the appropriateness of these different frames for this problem.

The majority of previous research in this domain (see table 9) treats the problem as binary classification due to the difficulty of the actual problem. This simplifies it significantly by aiming to just detect whether the PxPD symptoms occur, regardless of their severitiy. The granular label scheme utilized in this thesis follows an inherent, ordinal structure. For instance, a patient with $y = -4$ suffers from heavier bradykinesia than one with $y = -3$. In contrast, simple classification treats all class labels as if they were independent, thus fully neglecting the structural information in the data. The straight-forward way of including this ordinal information is the interpretation of the labels as a metric scale which can then be used in a regression setting. However, this implies the strong assumption of a linear relationship between the levels of the labels which does not always hold. For example, the change in the motor state from $y = -4$ to $y = -3$, formulated as $\delta_{-4,-3}$, could have a totally different meaning than $\delta_{-2,-1}$ though they would be equi-distant on a metric scale. Another technique for such problem settings is ordered regression, also referred to as ordinal classification. Ordered regression takes the ordinal structure of the data into account but does not make the strong linearity assumption. Thus, this model class is methodologically located at the intersection of classification and metric regression. Niu et al. (2016) give a comprehensive overview on statistical concepts in this domain such as the proportional odds model or generalized additive models with ordinal response.

Domingos (2012) offers a useful methodological framework to break down any machine learning model in three sub-modules: 1) the formal representation of the model, i.e. a neural net, 2) the evaluation function which is optimized to train the model and 3) the method to optimize the model. Within this framework, modules can be changed independently. This allows us to train the same model representation with the same optimizer in a classification, an ordered regression and a regression setting by interchanging the second module and adapting the labels y accordingly. Here it is noteworthy that the formal model representation, more specific the final layer of the CNN, has to be adjusted slightly for the different problem frames. For

classification, the final layer outputs a vector of softmax probabilities $p_j, j \in \mathcal{J}$, for each class. In the ordered regression, the final layer is replaced by a more complex, multi-task-head which is explained in more detail in chapter 3.1.2.1 and for regression, the CNN simply outputs one metric regression score without the use of any activation function. This concept allows the comparison of those three setups to answer the question of which frame suits the problem the best. The loss functions and the associated empirical risk for the three possible frames as well as the corresponding label transformations are explained in the following.

We express the above mentioned evaluation function as the empirical risk $\mathcal{R}_{\text{emp}}$ which is minimized throughout model training. Deep learning models are generally trained in a batch-wise manner, where the optimizer makes subsequent gradient descent steps based on the empirical risk of the batches. The batch-wise empirical risk is defined as the sum of the losses of the individual samples $L(y_i, X_i), i \in \{1, ..., N_b\}$, in one batch with a batch size of N_b. Thus, this batch-wise empirical risk can be formulated as:

$$\mathcal{R}_{emp} = \sum_{i=1}^{N_b} L(y_i, X_i) \tag{1}$$

and the definition of the loss $L(y_i, X_i)$ depends on the framing of the problem. In the following, we describe the losses and label transformations for each of the three possible problem frames.

3.1.1 Classification

As we deal with a multi-class problem in this thesis, we use the multi-class cross entropy loss L_i^{mXE} for classification which can be formulated as:

$$L^{\text{mXE}}(y_i, X_i) = -\sum_{j \in \mathcal{J}} \mathbb{1}[y_i = j] \cdot log(p_j(X_i, \theta)) \tag{2}$$

where $p_j(X_i, \theta) \in [0, 1]$ is the probability output for class $j \in \mathcal{J}$ from the final output layer of the model which is activated with a softmax function to map the model scores to probabilities. They depend on the sample input X_i and the model parameters θ.

3.1.2 Ordered Regression

We experimented with two versions of ordered regression models in the thesis: 1) a Multioutput CNN architecture and 2) the introduction of a distance penalization on the loss function. In the following, both approaches are explained in detail.

3.1.2.1 Multi-output CNN

There exist a broad variety of techniques for ordered regression: model-based approaches such as Herbrich, Graepel, and Obermayer (1999) and also approaches where the ordinal regression problem is transformed into multiple binary classification problems as proposed in Frank and Hall (2001), Li and Lin (2006), Chen et al. (2017) and Niu et al. (2016). As we want to use neural networks for ordered regression, we choose to implement an architecture where a single CNN architecture is trained jointly on a variety of binary ranking-based sub-tasks (2016). This approach shows strong performance compared to a regular regression approach on an age estimation task on portraits of Asian people, which motivates its usage in this thesis.

The above mentioned transformation allows the model to exploit the ordinal structure in the data. Imagine an ordered classification problem with $K = 3$ ordinal classes such that $y \in \{low, medium, high\}$ as illustrated in table 1 where a fictive person who earns a medium salary implicitly earns a low salary but does not receive an high salary. In this approach, labels are transformed into $K - 1$ rank-based labels. This is based on the idea that a sample with $y = medium$ has a rank of 1 which implies that this sample is also $y = low$. This label-transformation follows the formula in equation 3:

$$y_{i,k} \begin{cases} = 1, \text{ if } (y_i > r_k) \\ = 0, \text{ otherwise} \end{cases} \tag{3}$$

where r_k is the rank for the k-th sub-problem for $k \in \{1, ..., K-1\}$.

Table 1: Illustration of the transformation of categorical labels into rank-based labels for the three class example. The third sample is assigned class *medium* and thus it has the binary value 1 for y^1 meaning that it is higher than *low* but smaller than *high*.

	Original	Rank-based	
i	y_i	$y_{i,1}$	$y_{i,2}$
1	$1 \hat{=}$ *low*	0	0
2	$3 \hat{=}$ *high*	1	1
3	$2 \hat{=}$ *medium*	1	0
$\vdots$			

Following this label transformation, the authors propose a Multioutput CNN architecture, where each of the $K - 1$ outputs refers to one binary ranking-based sub-task. These are optimized jointly to train one single CNN corpus. Thus, the sub-task k is optimized on a binary classification problem minimizing the binary cross entropy loss:

$$L_k^{XE}(y_i, X_i) = -[y_{i,k} \cdot log(p_k(X_i, \theta)) + (1 - y_{i,k}) log(1 - p_k(X_i, \theta))] \tag{4}$$

where $p_k(X_i, \theta) \in [0, 1]$ is the probability output for the k-th sub-task, also termed the k-th head of the CNN, given input sample X_i. Thus, in total we yield $K - 1$ such probability outputs for each input sample. In order to train the CNN jointly on those $K - 1$ sub-tasks, those individual losses are combined to one cumulative loss, termed L^{ranks}:

$$L^{\text{ranks}}(y_i, X_i) = \sum_{k=1}^{K-1} L_{i,k}^{XE}(y_i, X_i) \tag{5}$$

The back-propagation procedure for this Multioutput CNN architecture is shown in more detail in the appendix of their work. For inference, the $K-1$ outputs are summed up:

$$\hat{y}_i = \sum_{k=1}^{K-1} p(X_i, \theta) - 4 \tag{6}$$

where the scalar 4 is subtracted from the sum over all probability outputs to map the predictions back to the initial label scale $y \in \mathcal{J}$, yielding a continuous output. In the ideal case, for an unseen sample X_u with the true label $y_u = -4$ the model should output a probability score of 0 for all sub-task resulting in $\hat{y}_u = \sum_{k=1}^{K-1} 0 - 4 = -4$. The authors finally mention that a consistency constraint should be applied over the sub-tasks but leave this open to further research.

3.1.2.2 Distance-based Penalization

A more subtle and intuitive approach to bridge classification and metric regression is the introduction of a cost matrix to weight misclassifications according to the distances between predicted and true labels. This way, we force the model to learn the inherent ordinal structure of the data as it is penalized higher for predictions that are very distant to the true labels.

This is achieved by the introduction of another distance-based weighting factor in the multi-class cross-entropy loss. Thus, the resulting distance-sensitive cross entropy classification loss, referred to as $L^{\text{ordinal XE}}$, can be formulated as:

$$\begin{aligned} L^{\text{ordinal XE}}(y_i, X_i) &= -(|y_i - \hat{y}_i|)^d \sum_{j \in \mathcal{J}} \mathbb{1}[y_i = j] \cdot log(p_j(X_i, \theta)) \\ &= (|y_i - \hat{y}_i|)^d L^{\text{mXE}}(y_i, X_i) \end{aligned} \tag{7}$$

where $\hat{y}_i = \arg\max_{j \in \mathcal{J}} p_j(X_i)$ denotes the class label prediction for sample i and $(|y_i - \hat{y}_i|)^d$ the norm of degree d of the distance between the predicted and the true class. There is no label transformation needed w.r.t. the basic classification frame and this modification can also be understood as a weighting scheme for the regular, multi-class cross entropy loss $L^{\text{XE}}(y_i, X_i)$. We refer to this model as ordinal classification CNN and use the L_1-norm for the distance-based penalization by setting $d = 1$. Due to the limited scope of this thesis, we leave the experimentation with norms of differing degrees open to future research.

3.1.3 Regression

The discrete class labels are mapped to a metric scale $y \in [-4, 4]$ for regression. Furthermore, each of the deep learning models now contains one single output neuron that is activated with the identity function yielding continuous, unbound prediction scores. As loss function, the quadratic loss $L^{\mathrm{QL}}(y_i, X_i)$ is used:

$$L^{QL}(y_i, X_i) = (y_i - \hat{f}(X_i, \theta))^2 \tag{8}$$

where $\hat{f}(X_i, \theta)$ is the model prediction given input data from sample X_i and the model parameters θ.

3.2 Class Imbalances

As discussed in chapter 2 and figure 2, the patient data suffers from high label imbalance towards the center of the scale $y \in \mathcal{J}$. Thus, any machine learning model will be biased towards predicting the middle classes and hardly ever predict the more extreme classes. Therefore, we aim to develop methods that are able to deal with imbalanced data in this work. Learning from imbalanced data is a well-known topic in research and He and Garcia (2009) provide a comprehensive overview on this issue.

A straightforward way of dealing with this problem is to reweight the loss contribution of different train data samples. This way, the algorithm incurs heavier loss for errors on samples from minority classes than for those of majority classes, which incentivizes more focus on the minority classes while training. In this thesis, we use the weighting scheme proposed by King and Zeng (2001), which is also implemented in the scikit framework (Pedregosa et al. 2011). The weights for the J classes $j \in \mathcal{J}$ are calculated as follows in order to upweight the contribution of minority classes to the global performance measure:

$$\begin{aligned} c_j &= \frac{N}{N_j} \\ c_{j,\text{normalized}} &= J \cdot \frac{c_j}{\sum_{j \in \mathcal{J}} c_j} \end{aligned} \tag{9}$$

where J describes the amount of possible classes (9 in this work), N is the total amount of samples, N_j the total amount of samples for class j and thus c_j is the inverse relative frequency of class j in the data. In a second step, the weights $c_j, j \in \mathcal{J}$ are normalized such that the sum of the weights is equal to the amount of classes. This normalization follows the rationale that in the non-reweighted case, each class would be assigned an imaginary weighting factor of 1, yielding a sum of 9 in total. In the following, we will refer to the individual weight of one sample as ω_i which is the weight $c_{j,\text{normalized}}$ associated with the label y_i of this sample i such that $y_i = j$.

Deep learning models are generally trained in a batch-wise manner, where the optimizer makes

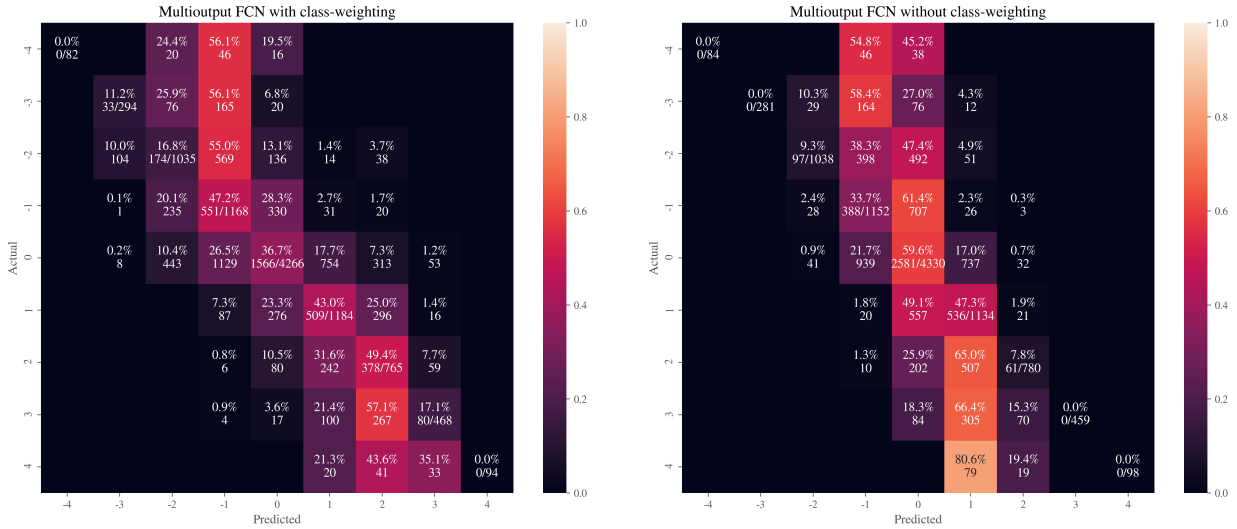
subsequent gradient descent steps based on the empirical risk of the batches, formalized as $\mathcal{R}_{\text{emp}}$. The empirical risk is defined as the sum of the losses of the individual samples in the batch. In contrast to the standard case where individual losses are summed equally weighted, we now employ the above defined weights ω_i of the individual samples $(X_i, y_i), i \in \{1, ..., N\}$ to calculate the weighted sum of losses, referred to as the weighted empirical risk $\mathcal{R}_{\text{wemp}}$. The weighted empirical risk for one batch is defined as:

$$\mathcal{R}_{\text{wemp}} = \sum_{i=1}^{N_b} \omega_i L(y_i, X_i) \tag{10}$$

for an arbitrary loss function $L_i(y_i, X_i)$ where N_b is the amount of samples in this batch and ω_i refers to the individual weights as described above. In a setting with no reweighting scheme, all weights would be set to $\omega_i = 1, i \in \{1, ..., N_b\}$ which would results in the regular empirical risk. This scheme allows for the reweighting of the empirical risks associated with any of the above defined losses. For instance, the weighted empirical risk for the multi-class cross entropy can simply be formulated as:

$$\mathcal{R}^{\text{mXE}}_{\text{wemp}} = \sum_{i=1}^{N_b} \omega_i L^{\text{mXE}}(y_i, X_i) = -\sum_{i=1}^{N_b} \omega_i \sum_{j \in \mathcal{J}} \mathbb{1}[y_i = j] \cdot log(p_j(X_i, \theta)) \tag{11}$$

This reweighting scheme was found to be crucial for the performance of any model experimented with in this thesis and in figure 3 we demonstrate its effect on the model predictions. As the class imbalance also affects the calculation of the performance measures, we include this reweighting scheme for the model training as well as for the model comparison in chapter 5.



(a) Class-weighted **(b)** Regular

Figure 3: Illustration of the effect of the class weighting scheme. Confusion matrices of two versions of the Multioutput CNN validated on the LOSO data set I are shown. The model is heavily biased towards the center of the labels in its predictions when trained with a regular empirical risk (right) and reweighting can tackle this problem effectively (left).

4 Models

Within this chapter we present the models that were used throughout this thesis. We start with the explanation of the baseline models and then discuss the different deep learning approaches that we deployed. Finally, we discuss a transfer learning approach.

4.1 Baseline Models

In order to put the performance of the deep learning based modeling approaches in perspective, we compare them with two baseline models. The most simple one, referred as majority vote, is a model that always predicts the majority class $\hat{y} = 0$. The second model is a Random Forest (Breiman 2001) which is trained on features that were manually extracted from the raw sensor data, similar to the work of Eskofier et al. (2016); Hammerla et al. (2015); Hssayeni, Burack, and Ghoraani (2016) and Sama et al. (2012) that applied this method as well in the context of Parkinson's disease. From each sample window of both accelerometer and gyroscope sensor signal norms, we extracted a total of 34 features which describe 17 different statistical measures as shown in table 11 in appendix C. This is a standard procedure in TSC and the presented features are inspired by existing software solutions (Christ et al. 2018) as well as literature (Casale, Pujol, and Radeva 2011; Christ, Kempa-Liehr, and Feindt 2016). We specifically chose a random forest over comparable feature-based machine learning models due to its low dependency on hyperparameter settings. The Random Forest was not specifically tuned as we only use it as a baseline model. Furthermore, the implementation in the scikit-learn package (Pedregosa et al. 2011) was used. The chosen model is parameterized as follows: it grows 500 trees, employs the Gini-coefficient as split-criterion, requires a minimum of 2 samples for a split, subsampling is done via bootstrapping, there is no maximum for the nodes of a leaf and no restriction in depth of the trees and each tree chooses from a maximum of $\sqrt{\#\text{ features}}$ randomly selected features at each split. In addition, a class-weighting scheme similar to that described in 3.1 is applied to account for class imbalances.

4.2 Deep Learning Approaches

In chapter 1, we gave a broad overview on recent developments in time series classification (TSC) with deep learning. Most popular approaches include Convolutional Neural Networks (CNNs) with one-dimensional layers. For an in-depth explanation of CNNs and neural networks in general we refer the reader to Goodfellow, Bengio, and Courville (2016) and for a comprehensive benchmark study on TSC, we refer to the recent work of Fawaz et al. (2018a). The most fundamental architecture in this domain is based on the Fully Convolutional Net (FCN) which was introduced as a strong baseline model by Wang, Yan, and Oates (2017) and was proven to work well on a variety of TSC problems (Fawaz et al. 2018a; Nweke et al. 2018; Um et al. 2018). In the light of these results, we decided to focus on one-dimensional CNN architectures in this thesis. Within the following chapter, we introduce the FCN in more detail and explain the

extensions that we applied to the architecture.

4.2.1 Fully Convolutional Net

The FCN, depicted in table 2 and figure 4, is a baseline deep learning architecture for the handling of one-dimensional data and was introduced as a strong baseline for TSC by Wang, Yan, and Oates (2017). The FCN basically consists of three stacked convolutional layers with decreasing kernel sizes that are followed by a batch normalization layer whose output is then activated with a ReLU function. Batch normalization decreases the variability in the batches to improve the generalization performance, resulting in faster training and model convergence (Ioffe and Szegedy 2015). This property of the batch normalization was crucial for this project due to the high computational cost of the many model runs required by the LOSO-validation scheme. A stride = 1 is used and the feature maps are padded accordingly such that they do not decrease in dimensionality. Instead of local pooling after the convolutional blocks, a global average pooling operator is used at the end of the convolution part to squash the resulting feature maps to a vector of their averages. This reduces the amount of weights in the final linear output layer drastically and helps mitigating the problem of overfitting (Lin, Chen, and Yan 2013).

Table 2: Illustration of the FCN architecture. Each of the three convolutional blocks is followed by a ReLU activation function. The final output depends on the problem frame (nine softmax outputs for the classification, eight for the ordinal regression and one, non-activated output for the regression problem).

Name	Type	Kernel	Channels	Output
Input	input		2	1200x2
CNN block 1	convolution	7x1	128	1200x128
	batch normalization		128	1200x128
CNN block 2	convolution	5x1	256	1200x256
	batch normalization		256	1200x256
CNN block 3	convolution	3x1	128	1200x128
	batch normalization		128	1200x128
GAP	global average pooling	1x1200		1x128
Output	linear output			9/ 8/ 1

4.2.2 FCN Inception

Inception modules, introduced by Szegedy et al. (2015), are mainly used in computer vision and led to substantial performance increases on image classification tasks. With the expectation of a similar positive effect in the current problem, we apply them as an extension to the basic FCN architecture.

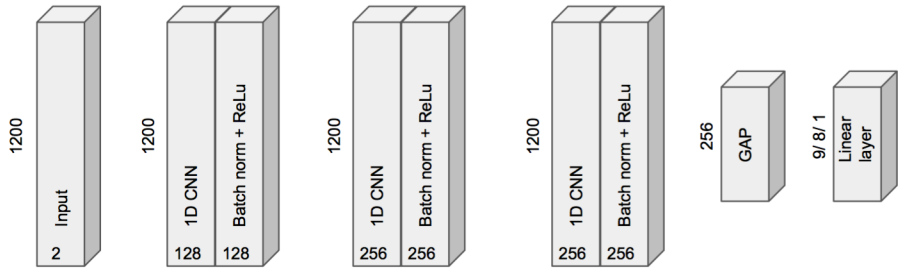


Figure 4: Visualization of the FCN architecture as outlined in table 2. One-minute sample windows with two sensor streams are given as an input to the net. As the data was resampled to 20Hz, each window has a length of 1200. The data is then subsequently fed to three convolutional blocks of depth 128, 256 and 256. The resulting feature maps and ReLU-activated and batch-normalized after each convolutional block. The final feature maps are averaged within the global average pooling layer and a linear layer connects the GAP output with the final output neurons of the net.

Inception modules, illustrated in figure 5, are motivated by the observation that the kernel size of the convolutional layers are often chosen rather arbitrarily by the deep learning practitioner. The rationale is to give the model the opportunity to choose from different kernel sizes for each convolutional block and distribute the amount of propagated information amongst the different kernels. This relieves the researcher from the burden to determine reasonable kernel sizes. Also, it provides the net with the ability to detect similar features on different scales: e.g. objects can occur in a huge and a small format in the original data. Whilst a layer with a fixed, huge kernel size will only be able to detect the objects at a bigger scale, an inception module can capture the same object in different scales due to multiple-size kernels.

In one inception module, the same input is fed through several branches of convolutional and max-pooling operations in parallel. In the last step, the outputs from the several branches are concatenated depth-wise in the channel dimension and fed to the next architectural block. One requirement for this concatenation is that the outputs of the single branches share the same dimensions. Similarly to the FCN in chapter 4.2.1, this is achieved by a same padding strategy. As shown in figure 5, branches 2-4 of the module convolve the input with kernels of different sizes. Branch 5 applies max-pooling to the input followed by a convolution with a kernel of size 1 to ensure the depth of the block matches those of the others. Before the actual convolution in branches 2-4, the input is convolved with a kernel of size 1 to reduce the depth of the feature block for computational efficiency as shown in 2015. In branch 1, only one kernel of size 1 is applied following the rationale of the residual connections as further explained in chapter 4.2.3. All convolutional layers are followed by a batch normalization layer for faster model convergence and activated with the ReLU function. Finally, the feature maps resulting from the various branches are concatenated depth-wise and fed into the next architectural block.

The final FCN Inception architecture essentially follows that of the FCN from chapter 4.2.1 with simple convolutional layers being replaced by above mentioned inception modules. The final architecture is illustrated in table 3.

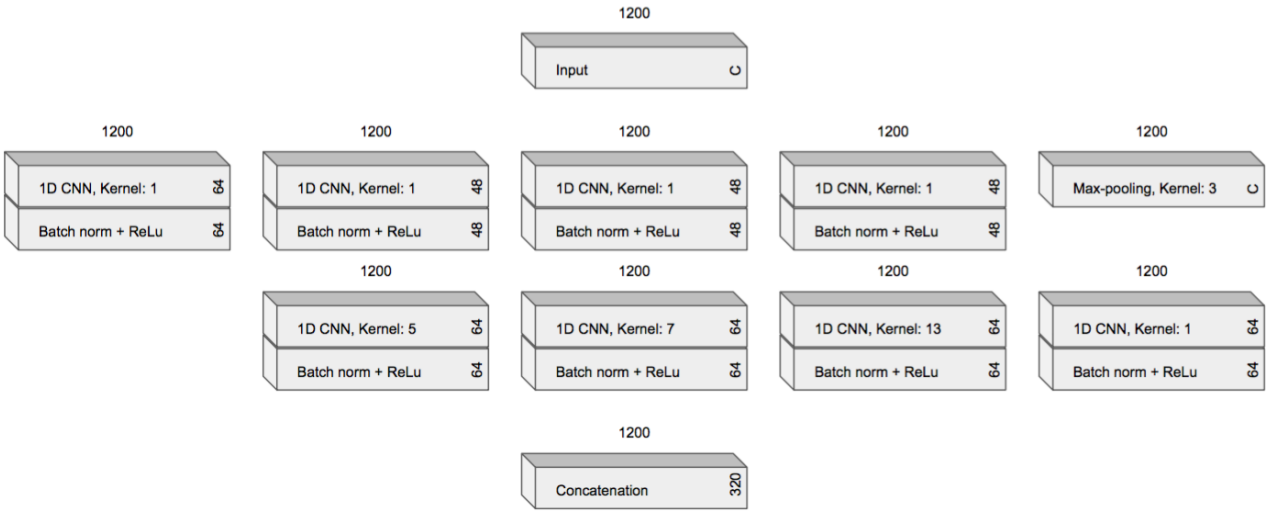


Figure 5: Illustration of one inception module. The flow from the input to the output of the module goes from top to bottom. Data with a length of 1200 and a channel-depth depending on the position of the module in the architecture is fed as an input to the module. This data is then convolved in parallel by the 5 branches of the module. The first branch serves as a projection layer and has a similar function as in the ResNet. Branches 2-4 extract features with different convolutional kernels and the 5th branch applies max-pooling to the input, similar to 2015. For the sake of faster computation, the depth of the blocks in branches 2-4 is reduced using a convolutional layer with a kernel size of 1 to reduce the depth of the block. In a last step, the outputs from the feature maps are concatenated depth-wise and fed to the next part of the architecture.

Table 3: Illustration of the FCN Inception architecture. Each convolutional layer in the architecture is batch normalized and followed by a ReLU activation. The final output depends again on the problem frame (9 softmax outputs for the classification, 8 for the ordinal regression and 1 non-activated output for the regression problem). The first module gets an input of depth 2 as two sensor streams are fed into the model.

Name	Type	Kernel(s)	Channels	Output
Input	input		2	1200x2
Inception 1	inception module 1	1, 5, 7, 13	320	1200x320
Inception 2	inception module 2	1, 5, 7, 13	320	1200x320
Inception 3	inception module 3	1, 5, 7, 13	320	1200x320
GAP	global average pooling	1200		1x320
Output	linear output			9/ 8/ 1

4.2.3 FCN ResNet

Similar to the inception modules, the introduction of residual learning by He et al. (2015) has met with great enthusiasm in the deep learning community. The main advantage of such Residual Networks (ResNet) over regular CNNs is the usage of shortcut- or skip-connections between subsequent layers. Those skip-connections allow the gradient to flow around the initial CNN layers and skip certain layers in case they do not contribute to the model performance. This helps tackling the vanishing gradient problem and thus allows for the training of much deeper network architectures.

Unlike the inception modules, this model class was already deployed for TSC by Wang, Yan,

and Oates (2017) and proven to be a strong competitor for the classic FCN in a benchmark study by Fawaz et al. (2018a). Especially on multivariate TSC problems, ResNet was shown to outperform the standard FCN according to their results. In contrast, Wang, Yan, and Oates (2017) argue that the ResNet is prone to overfitting and thus found it to perform worse than the FCN in generalization. This motivated us to also employ this model architecture for comparison with the standard FCN.

The functioning of the residual modules is illustrated in figure 6. Within the architecture, the input data can flow through two branches. First, it can propagate through the subsequently stacked convolutional layers in the central block of the architecture. Second, the data can make use of the skip-connection and bypass the feature extraction block of this current module. The central block is constructed similar to the FCN and convolved feature maps are both, activated with the ReLU and batch-normalized. As the outputs of stream 1 and 2 are required to share the same dimensions, Szegedy et al. (2016) introduced a linear projection layer to map the output of the skip connection to that of the main stream. A regular convolutional layer with a kernel size 1, an input depth C and the output depth Z is used for this linear projection.

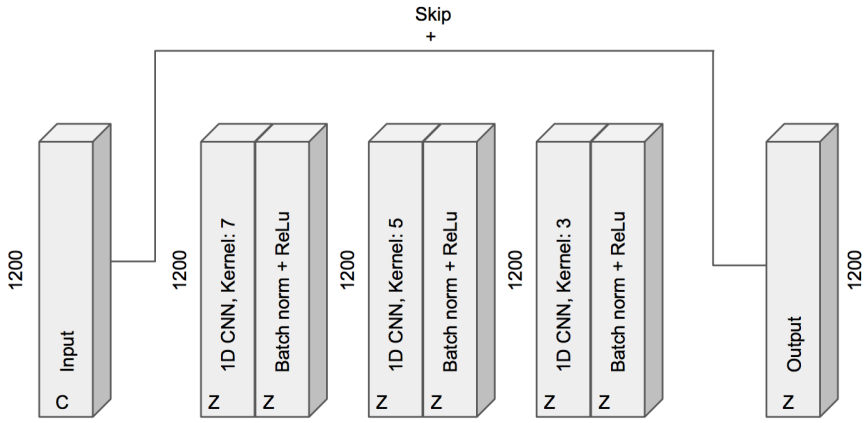


Figure 6: Illustration of one residual module. The flow from the input to the output of the module goes from left to right. Data with a length of 1200 and a channel-depth depending on the position of the module in the architecture is fed as an input to the net. The architecture then consists of two streams: 1) the skip connection at the top of the graphic and 2) the convolutional block in the middle of the architecture. The net could make use of the skip-connection to let the data flow almost untouched around the convolutional blocks of this module. Finally, the outputs from both streams are added element-wise and fed to the next architectural block. The depth of the blocks in stream 2) can be controlled by the researcher.

Theoretically, one could stack an infinite number of residual modules, as the model could make use of the residual connections to bypass unnecessary convolutional blocks. In the current setup, we decided to stack three of such modules in order to gain a general impression if this extension is reasonable for our problem. Therein, each of the modules is similar to the standard FCN in order to provide some comparability among architectures. The depth Z of the individual residual modules were chosen as proposed in Wang, Yan, and Oates (2017). The complete architecture is illustrated in table 4.

Table 4: Illustration of the FCN ResNet architecture. Three residual modules with different depth are concatenated and followed by a GAP layer. The model output is again depending on the problem framing.

Name	Type	Kernel(s)	Channels	Output
Input	input		2	1200x2
Residual 1	residual module	7, 5, 3	64	1200x64
Residual 2	residual module	7, 5, 3	128	1200x128
Residual 3	residual module	7, 5, 3	128	1200x128
GAP	global average pooling	1200		1x128
Output	linear output			9/ 8/ 1

4.2.4 Residual Connected Flow (RCF)

Wang et al. (2018) just recently proposed an alternative extension of the standard FCN module for TSC using multi-level wavelet transformations. Their basic idea is to decompose the original signal to different levels of resolution. Each of those decompositions serves as the input to one CNN. Finally, the outputs from the several CNNs are combined to one global prediction which allows the joint training of the whole model. Next to the introduction of their architecture, the authors conducted a benchmark study on the UCR time series archive (Dau et al. 2018) showing promising results compared to the standard FCN. Previous research on PxPD (Hammerla et al. 2015) employs a band-pass filter to isolate the frequency range related to their problem and extract features from this filtered signal. As specifying the frequencies is highly domain-specific, resource-intensive and to some extent subjective, we propose this more generalizable methodology that is able to learn to focus on the correct frequency ranges. Similar to (Wang et al. 2018), we combine the multi-level discrete wavelet decomposition (DWT) with the FCN. The key rationale of their work is the training of deep models on different frequency space representations of the original signal. However, whilst the authors allow the parameters of the wavelet kernel to be dynamic, we apply the Haar-kernel in a static manner for the DWT as recommended in Jothimani, Shankar, and Yadav (2016).

The discrete wavelet transform is used to map a signal into frequency space whilst preserving its temporal relation as opposed to the Fourier transform (Jensen and Cour-Harbo 2001). Therefore, it recursively decomposes the signal into high and low-frequency components, referred to as detail and approximation coefficients. Due to the time-preservation property, the original signal can be reconstructed from these decompositions in a loss-less manner. In one step, the original signal is filtered with a low-pass and a high-pass filter. For each level, this results into two representations of the input signal: one to capture the details of the signal and one to capture its general, denoised information as shown in figure 8 and referred to as l and h, respectively. Similar to Wang et al. (2018), the denoised, general representation h filter from level j is further decomposed in the next level $j + 1$. Thus, the classifier on level $j + 1$ learns from a lower-granular representation than that on level j, allowing it to focus on more global

features of the signal.

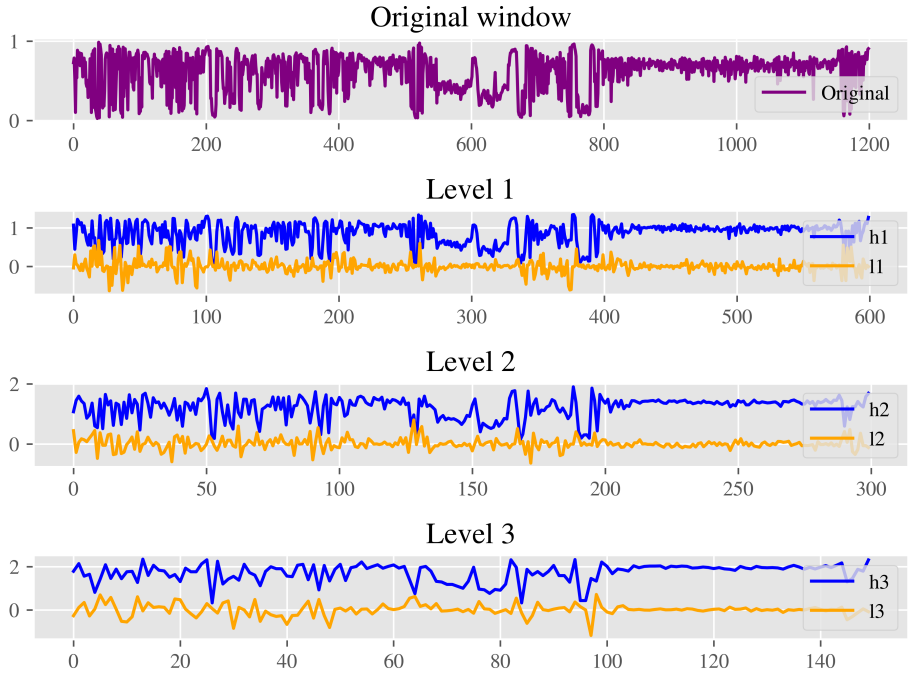


Figure 7: Multi-level wavelet decomposition of the accelerometer norm of one examplary sample window. The signal is decomposed with a high-pass filter and a low-pass filter at each level, with the output referred to as h_j and l_j for each respective step. The high-pass-filtered representation h_j is then fed into the following DWT decomposition step. Thus, the signal is getting smoother and less noisy from step to step.

The final model basically consists of three modules of the same architecture which are trained on different versions of the data. Those architectures follow that of the FCN explained in chapter 4.2.1 but can be replaced by any customized architecture. The first module is trained on more detailed data than the final module due to the subsequent discrete wavelet filtering which is explained above. Hypothetically, this allows the overall net to focus on different properties of the data. To some extent, this could be interpreted as a joint ensemble learning approach. For instance, one would expect the final module to extract more global features than the first module. Predictions are made from all three modules, depicted as $\hat{y}_j, j \in \{1, 2, 3\}$ and combined for the final prediction. In order to allow the model to weight the several predictions adaptively, we connect them using simple linear layers $f_j, j \in \{2, 3\}$, taking $\hat{y}_j, \hat{y}_{j-1}$ as an input and outputting $\hat{y}_J, J \in \{II, III\}$. In addition to using their proposed architecture in a regression setting, this learnable weighting scheme also differentiates our approach from the work of 2018. The choice of different level sizes for this architecture is left open for further research. Throughout this work, we use two sensor streams as model input which results in a channel depth of 2 for the first convolutional kernels. Consequently, a channel depth of 4 is required as each sensor stream is further decomposed into two streams and the resulting signals are concatenated depth-wise.

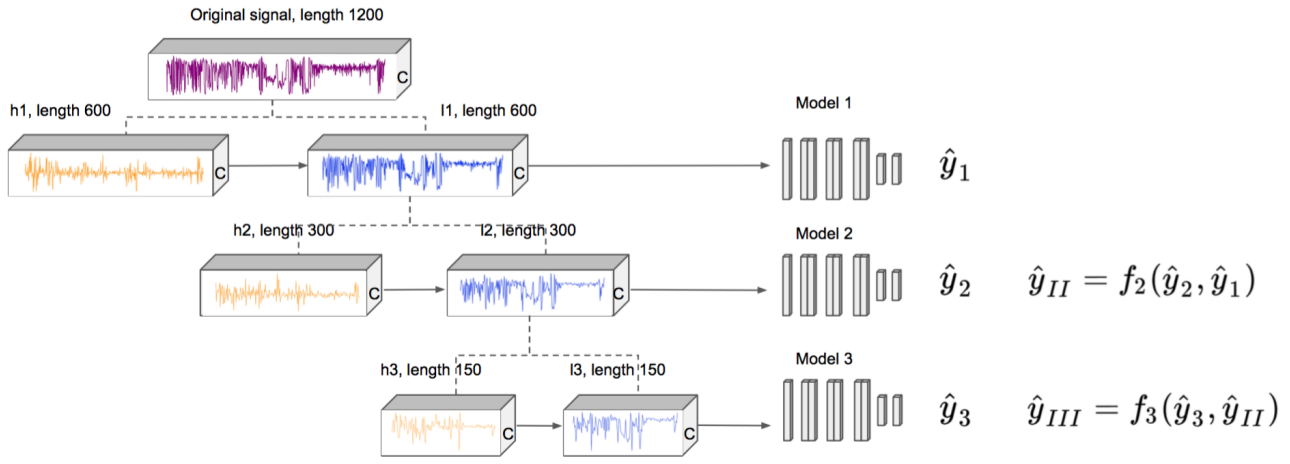


Figure 8: Architecture of the Residual Connected Flow (RCF) network which relies on multi-level wavelet-decomposed versions of the original signal. In total, three modules are trained on versions of the data in three resolutions. The outputs from the single modules are subsequently concatenated via simple, linear layers f_2, f_3 with y_{III} being the final model prediction. Note that the entire architecture is directly optimizable based on the loss defined over y_{III} which allows for joint training of the three different modules.

4.2.5 FCN Broad

Pathologically, the disease severity changes rather slowly over time. In other words, it does not make sense that the severity of the disease changes rapidly over neighbouring time windows. Thus, we hypothesized that additional input information and a broader view on the data context could have a positive impact on the model performance. We refer to this model class as FCN broad, as we allow the model to process information form a more broad context.

Therefore, we include the following extension in our models: the raw input data from the previous sample window X_{t-1} and the following sample window $X_t + 1$ were padded to the initial sample window X_t for which a prediction should be made. In the regular CNN-based approaches and as described in chapter 2, we use a channel depth of 2 for the input to the convolutional blocks as each input sample t comprises of 2 time series $X_{\text{accnorm},t}, X_{\text{gyronorm},t}$ that each contain 1200 data points. Thus, one sample tuple t can be formulated as $(y_t, (X_{\text{accnorm},t}, X_{\text{gyronorm},t})), t \in \{1, N\}$. In the FCN broad, we apply a channel depth of 6 as the raw data from the previous and the future sample are added in the depth dimension. This results in one sample tuple to be represented as $(y_t, (X_{\text{accnorm},t}, X_{\text{gyronorm},t}, X_{\text{accnorm},t-1}, X_{\text{gyronorm},t-1}, X_{\text{accnorm},t+1}, X_{\text{gyronorm},t+1})), t \in \{1, N\}$. This is illustrated in figure 9.

4.3 Transfer Learning

One of the most important requirements for the successful training of deep neural networks with strong generalization performance is the availability of enough train data. Next to heavy regularization or data set augmentation, one prominent method to fight overfitting and improve the model's generalization performance is transfer learning. In computer vision, it has been observed that early layers of CNNs tend to learn similar features, even when trained on totally different tasks (Yosinski et al. 2014). This motivated the idea to reuse CNN architectures that

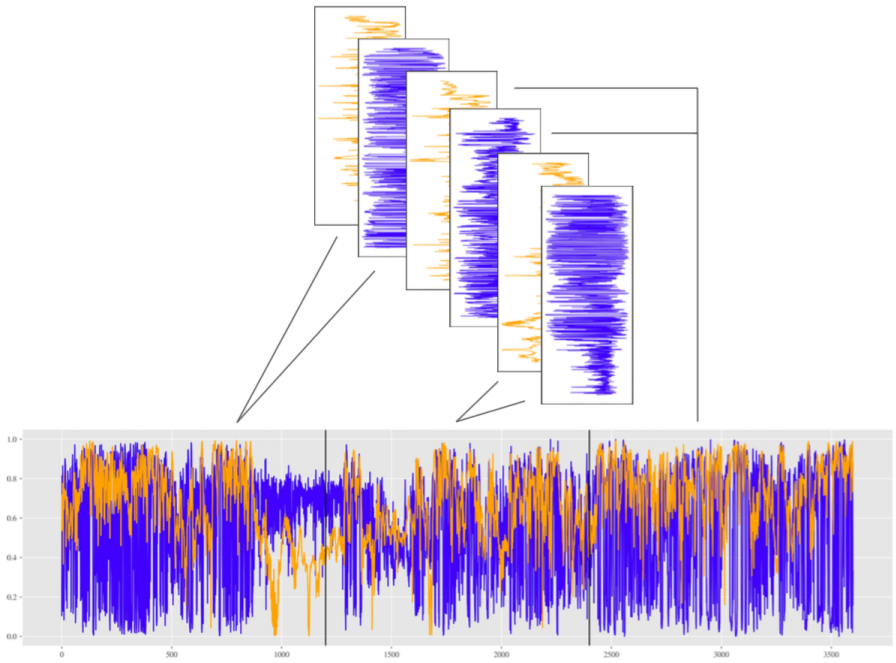


Figure 9: Visualization of the procedure for the FCN broad: the past and future window are concatenated depth-wise to the current window, whose label is used for model training. This allows the model to include more global information in its prediction and results in an overall channel depth of 6. Depicted are the normalized streams of the accelerometer norm (blue) and the gyroscope norm (orange).

were trained on source task A and transfer their inherent learned knowledge, manifested in the model weights θ_A, to the target task B. The model is then fine-tuned on the specific data for task B which leads to faster model convergence and, dependent on the similarity of tasks A and B, to an improvement in model performance. The rationale behind this is that the model trained on target task B can benefit from features that were already learned on source task A. Especially in image-based applications, transfer learning is applied with great success (Csurka 2017; Hoo-Chang et al. 2016). This is facilitated by the availability of large, multi-class image data bases such as ImageNet (Deng et al. 2009) and the open sourcing of pre-trained model architectures in common deep learning libraries. As shown in Oliver et al. (2018), transfer learning is one of the most promising semi-supervised learning strategies and recommended in the presence of suitable source data sets.

As TSC is still a niché in the deep learning community, there has not yet been much research on the adoption of transfer learning to this problem set. To fill this gap in literature, Fawaz et al. (2018b) just recently published a study on transfer learning in the time series domain. Within their study, they applied transfer learning to all pairwise combinations of the UCR data sets using the FCN architecture. Their results show that transfer learning can improve but also deteriorate the performance of the model on the target task. The latter effect can occur when source and target task are very dissimilar. In the majority of their experiments, transfer learning did not affect the model performance in any way. In addition, they introduce a distance measure for the selection of optimal source tasks for a given target task. At the time of this writing, there is only one possible data set in our project group that could be used for

transfer learning. Thus, we do not make use of this measure based selection of source tasks. Still, this could be an interesting topic for future research.

As a source task for the motor state detection, we use the data set III that is explained in more detail in chapter 2. This data set contains 72.918 one-minute sample windows of sensor data from patients that suffer from Parkinson's disease as well as from healthy controls. This data set was constructed by collecting sensor data during daily activities of the two patient groups similar to the collection protocol for our target task. Thus, we hypothesized that this source task is similar enough to the target task for transfer learning to have a positive effect. The data set is equally balanced. We used the data of those patients to train a binary classifier based on the respective target architecture that detects if a sample window is attributed to a patient that suffers from Parkinson's disease or to an healthy one.

Those source models are trained with the same hyperparameters and architectural settings as the target models. The results on this transfer learning approach are described in chapter 5.4.2.

5 Results

In this chapter we provide answers to the three following research questions:

1. Which problem frame suits the problem best?
2. Which model architecture is most suitable?
3. Can transfer learning methods improve model performance on this task?

In the following, we explain the non-trivial validation scheme, discuss the final performance measure, show the postprocessing procedure and discuss the results from the model comparison as well as the transfer learning extension.

5.1 Validation Scheme

To validate the performance of the previously described models, we want to use a method to estimate the generalization performance on new unseen data. This is the typical strategy to obtain an estimate for the model performance in practice (Hastie, Tibshirani, and Friedman (2009)). The application of this work will be a final model which predicts motor states based on movement data from one test patient without having seen any of the movement data from this particular patient before. As stated in Varoquaux et al. (2017), "cross-validation relies on independence between the train and test sets". If we would use regular cross validation (CV) strategies and randomly split the one-minute samples in train, validation and test sets, data gathered from one patient could occur in different sets. This would contradict the assumption of independence and a complex model could exploit this subject-specific information and produce a too optimistic estimate of the generalization performance. Thus, we use the LOSO validation strategy which is often applied in settings were many data are gathered from few subjects (Bao and Intille 2004; Eskofier et al. 2016; Fisher et al. 2016; Varoquaux et al. 2017). Thereby, we train a model on all subjects except one subject which is left out. The resulting models are then tested on the left out subjects, yielding an unbiased performance estimate for each subject. Saeb et al. (2017) discuss the benefits of the LOSO validation compared to regular cross validation in the context of disease monitoring with wearable devices. They show empirically as well as with a simulation study that general CV tends to overestimate the generalization performance of prediction models. In line with their argumentation, we use the LOSO strategy in our problem.

One peculiarity in the application of deep learning models is their heavy dependence on hyperparameters. The amount of iterations for which the model is trained is one of the most important ones. One epoch means one pass of all training samples through the neural net. Early stopping (Goodfellow, Bengio, and Courville 2016) is one of the standard methods to determine this optimal amount of epochs e^*. Thereby, the model's performance is continually measured on a validation data set during the training process. Model training is stopped,

when the validation performance did not improve for an amount of subsequent epochs which is referred to as patience. This requires the introduction of a tuning step to determine e^* in each of the LOSO folds, which in turn requires another split of the data set. We will refer to this as inner fold and to the actual LOSO fold as outer fold in the following. In a setting with unlimited computational resources, one would run a proper LOSO validation in the inner folds, determine the optimal e^*, train the model on the whole data except the left out subject and evaluate the trained model on that subject. With a total amount of 28 patients, this would result in the training of $28 \cdot 27 = 756$ models for the validation of one specific architecture. As this exceeds the computational resources of this thesis, we propose a cheaper solution.

We split the data in the inner fold by a 80/20 ratio depending on the timely structure of the data for each of the fold's training patients. The first 80% one minute windows per patient are used for training and the last 20% one minute windows are used for validation. Algorithm 1 illustrates the validation scheme. Those inner folds are used to determine the optimal train epochs e_j^* for each patient. Opposed to the outer folds, the model can therein exploit the individual patient effects as data from all patients occurs in the inner train as well as in the inner test data. Thus, the inner problem is more easy than the outer, proper LOSO problem. Still, we use this inner loop only for the determination of e^* and not for an estimate of the generalization performance. Therefore, we assume that the model converges in the outer and inner loop behave similarly. We argue this to be a valid assumption which is necessary due to a limited access to computational power. Comparable work in this context such as Um et al. (2018), Eskofier et al. (2016) and Hssayeni, Burack, and Ghoraani (2016) even skip this step and set the parameter e^* arbitrarily.

Algorithm 1 LOSO validation scheme

Input: Data set I split in LOSO data sets $D_j, \forall j \in \{1, ..., 28\}$ which are again split in inner train and inner valid data sets $D_{\text{j, inner train}}, D_{\text{j, inner valid}}$, where $D_{\text{j, inner train}}$ contains the first 80% and $D_{\text{j, inner valid}}$ contains last 20% of the data samples of patient j
Data set II for testing
Maximum amount of train budgets e_{max}
Model architecture $f(X, \theta)$

Output: $J = 28$ LOSO models
Final model $f^*(D, \theta)$ trained on all LOSO data for e^{**} which can be deployed
Performance measures for each LOSO fold
Performance measures for the performance of $f^*(X, \theta)$ on data set II

for *LOSO patient j in patient list from data set I* **do**
Train inner model $f_{j,inner}(D_{\text{j, inner train}})$ for max e_{max} epochs and get the optimal $e*_j$ for this fold via early stopping with validation data ($D_{\text{j, inner valid}}$)
Train outer model $f_{j,outer}(D_{-j})$ for e_j^* and get predictions and performance measures for this LOSO patient

end

Calculate e^{**} as the median of all $e_j^*, \forall j \in \{1, ..., 28\}$
Train the final, deployable model $f^*(D, \theta)$ on data set I for e^{**} epochs
Test the model with data set II and get predictions and performance measures

5.2 Performance Measure

In this chapter, we describe the performance measure and the class weighting scheme.

5.2.1 Expert-based Performance Measure

As this work results in a model that predicts a severity index, the end-user, which is the medical doctor, will base his diagnosis on the model's prediction. Thus, it is crucial to design an adequate performance measure in strong conjunction with the practicioners, which reflects the practical requirements on the model. We introduce this customized performance measure, referd to as Custom-loss in the following. Due to the limited scope of this thesis, the Custom-loss is used as an outer loss for model evaluation only and not deployed as a directly optimized inner loss. In appendix F, we show how this Custom-loss could be optmized directly as an inner loss.

5.2.1.1 Motivation

From discussions with the involved medical doctors, we found that misclassifying a sample in a class that is two levels distant to the true label is a lot worse than missing the true label by one level. This suggests a non-linear loss function. Another important finding is that misclassifying in the wrong direction of the scale (e.g. $\hat{y}_i = -1, y_i = 1$) has a higher negative impact than misclassifying in the correct direction (e.g. $\hat{y}_i = 3, y_i = 1$) though the distance in both scenarios has the same magnitude. The rationale behind this is that an exaggerated diagnosis which follows the true pathological scenario does not harm the patient as strong as an opposing diagnosis. Plain speaking: considering a dyscinetic patient too dyscinetic is less problematic than diagnosing bradykinesia, the opposing disease pattern. This is due to the fact that in the latter case a medication would be prescribed that would lead to an effect that is totally diametral to the desired one. This finding suggests an asymmetric loss function. Furthermore, we found that the cost of misclassifying a patient in the wrong pathological direction increases with the severity of the disease: diagnostic errors weigh heavier on patients with strong symptoms compared to patients that are only mildly affected by the disease. Thus, $L(\hat{y}_i = -3, y_i = -4) \overset{!}{>} L(\hat{y}_i = -2, y_i = -3)$. In summary, three main requirements on the loss function were identified:

1. Non-linearity
2. Asymmetry
3. Dependence of the strength of the asymmetry on the true label values

5.2.1.2 Asymmetric Losses

Asymmetric loss functions are also applied in economic forecasting as an under- or overestimation of economic variables has differing consequences. In this context, Elliott, Timmermann, and Komunjer (2005) introduced the following, parametrized loss function:

$$L^{\text{asym}}(\alpha, p, y_i, \hat{y}_i) = [\alpha + (1 - 2\alpha) \cdot \mathbb{1}(y_i - \hat{y}_i < 0)] \cdot (|y_i - \hat{y}_i|)^p \tag{12}$$

where $\alpha \in [0, 1]$ controls the asymmetry of the loss function, such that

$$\alpha \begin{cases} \in]0.5, 1[, & \text{penalization of underestimation} \\ = 0.5, & \text{symmetric loss} \\ \in [0, 0.5[, & \text{penalization of overestimation} \end{cases} \tag{13}$$

and p controls the norm of the loss, with $p = 1$ yielding an absolute and $p = 2$ yielding a quadratic loss measure. The loss resulting from $p = 2$ is also referred to as Quad-Quad-loss and $p = 1$ is known as Lin-Lin-loss (Christoffersen and Diebold 1996). Following the first requirement, we set $p = 2$ in the following. Elliott, Timmermann, and Komunjer (2005) and Ahn and Tsuchiya (2017) employ this loss to overcome underestimation in econometric forecasting and Toth (2015) applies it in a deep learning setting on an engineering problem. The third requirement is the inclusion of an asymmetry, that is dependent on the values of the true label and increases towards the extreme values of the label scale $y \in \{-4, ..., 4\}$. To facilitate the inclusion of this requirement, we propose a new asymmetric loss function that shares the main properties of the Quad-Quad-loss, but differs in some details. We refer to it as Custom-loss in the following. The proposed loss function has the form:

$$L^{\text{custom}}(\alpha_{\text{custom}}, y_i, \hat{y}_i) = [\alpha_{\text{custom}} + sign(\hat{y}_i - y_i)]^2 \cdot (\hat{y}_i - y_i)^2 \tag{14}$$

where $\alpha_{custom} \in [-1, 1]$ controls the asymmetry such that:

$$\alpha_{\text{custom}} \begin{cases} \in [0, 1[, & \text{penalization of overestimation} \\ = 0, & \text{symmetric loss} \\ \in]-1, 0], & \text{penalization of underestimation} \end{cases} \tag{15}$$

5.2.1.3 Comparison of Custom- and Quad-Quad-loss

One similarity of both losses is that they multiply the quadratic loss with a factor that depends on the variable α, which governs the strength of the asymmetry, and on the fact that the algorithm overestimates ($\hat{y} > y$) or underestimates ($\hat{y} < y$) the true label. For better comparison, both penalization factors are shown in equation 16, where $\alpha_{\text{custom}} \in [-1, 1]$ and $\alpha_{\text{quad-quad}} \in [0, 1]$. Overestimation ($\hat{y} > y$) is heavily penalized for $\alpha_{\text{custom}} = -1$ and $\alpha_{\text{quad-quad}} = 1$ whilst underestimation ($\hat{y} < y$) is heavily penalized for $\alpha_{\text{custom}} = 1$ and $\alpha_{\text{quad-quad}} = 0$. From both equations, we see that the Custom-loss maps back to the quadratic loss for $\alpha_{\text{custom}} = 0$ and the Quad-Quad-loss resembles the quadratic loss up to a scaling factor of 0.5 for $\alpha_{\text{quad-quad}} = 0.5$.

$$\begin{aligned} &\text{Penalization factor Custom: } 1 + \alpha_{custom}^2 + 2\alpha_{custom} sign(\hat{y} - y) = [\alpha_{custom} + sign(\hat{y} - y)]^2 \\ &\text{Penalization factor Quad-Quad: } \alpha_{quad-quad} + (1 - 2\alpha_{quad-quad}) \cdot \mathbb{1}(y - \hat{y} < 0) \end{aligned} \tag{16}$$

In the numerical example in table 5 we show that the Custom-loss collapses exactly to the quadratic loss in the symmetric setting whilst the Quad-Quad-loss is proportional to it up to a constant. One can see that the Custom-loss penalizes over- and underestimation more heavily than its Quad-Quad-counterpart and both losses share the same properties of being non-linear and asymmetric.

Table 5: Illustrative example of the similar behavior of the Quad-Quad-loss and the Custom-loss in different scenarios. Presented are edge-case scenarios with the most heavy penalizations and a symmetric loss.

Penalization	Prediction	Loss factor Custom	Loss factor Quad-Quad
		$\alpha_{custom} = 1$	$\alpha_{quad-quad} = 0$
Overestimation	$\hat{y} > y$	$[1 + sign(\hat{y} - y)]^2 = 4$	$[0 + \mathbb{1}(\hat{y} > y)] = 1$
	$\hat{y} < y$	$[1 + sign(\hat{y} - y)]^2 = 0$	$[0 + \mathbb{1}(\hat{y} > y)] = 0$
		$\alpha_{custom} = -1$	$\alpha_{quad-quad} = 1$
Underestimation	$\hat{y} > y$	$[-1 + sign(\hat{y} - y)]^2 = 0$	$[1 - \mathbb{1}(\hat{y} > y)] = 0$
	$\hat{y} < y$	$[-1 + sign(\hat{y} - y)]^2 = 4$	$[1 - \mathbb{1}(\hat{y} > y)] = 1$
		$\alpha_{custom} = 0$	$\alpha_{quad-quad} = 0.5$
Symmetric	$\hat{y} > y$	$[0 + sign(\hat{y} - y)]^2 = 1$	$[0.5 + 0] = 0.5$
	$\hat{y} < y$	$[0 + sign(\hat{y} - y)]^2 = 1$	$[0.5 + 0] = 0.5$

The final step is the conditioning of the asymmetry on the true label values, as motivated in the third requirement. The factor that controls the strength of the asymmetry is α_{custom}. Thus, we need to connect y with α_{custom}. One straightforward way towards this goal is the introduction of α^*_{custom}:

$$\begin{gathered} \alpha_{\text{custom}} = \frac{y}{4} \cdot \alpha^*_{\text{custom}} \\ \text{where } y \in \{-4, ..., 4\},\ \alpha^*_{\text{custom}} \in [0, 1] \text{ and thus } \alpha_{\text{custom}} \in [-1, 1] \end{gathered} \tag{17}$$

where the constant denominator 4 is introduced to ensure that $\alpha_{custom} \in [-1, 1]$ and α^*_{custom} is restricted to $[0, 1]$ as the sign that governs the direction of the asymmetric penalization is controlled by the true labels. Thus, the model will be heavily penalized for the overestimation of negative labels and for the underestimation of positive labels. We now see that the asymmetry of the function is reciprocally connected to the magnitude of the label y in both, the negative as well as the positive direction: e.g. the loss for $y = 1$ is more symmetric than the loss for $y = 4$. Furthermore, the loss collapses to a regular quadratic loss for $y = 0$, the median value of the label scale $y \in \{-4, ..., 4\}$. The behavior of the Custom-loss is further illustrated in figure 10.

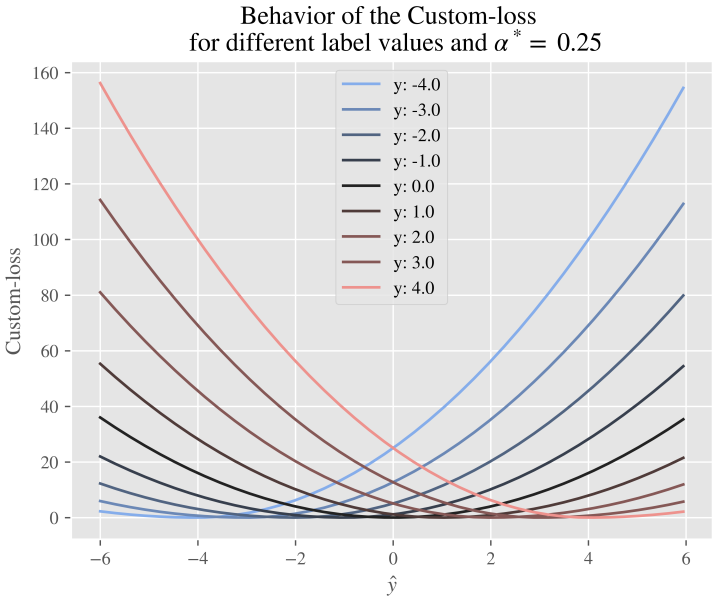


Figure 10: Illustration of the behavior of the Custom-loss on the y-axis for different y and the corresponding predictions $\hat{y}$ on the x-axis. The loss collapses to a symmetric quadratic loss for true labels of $y = 0$ and is becoming increasingly asymmetric towards the extremes of the label-scale. This illustrates the direct dependency of the asymmetry of the loss on the true label values.

In order to get a common understanding of the properties of the desired loss function, we evaluated different cost matrices in close discussion with the medical doctors involved in the project. These cost matrices also served as a basis of discussion to determine the optimal value α^*_{custom} for the present problem. Based on their domain expertise, the involved medical doctors decided to choose $\alpha^*_{custom} = 0.25$. Figure 11 shows the costs associated with different predictions for all possible label values.

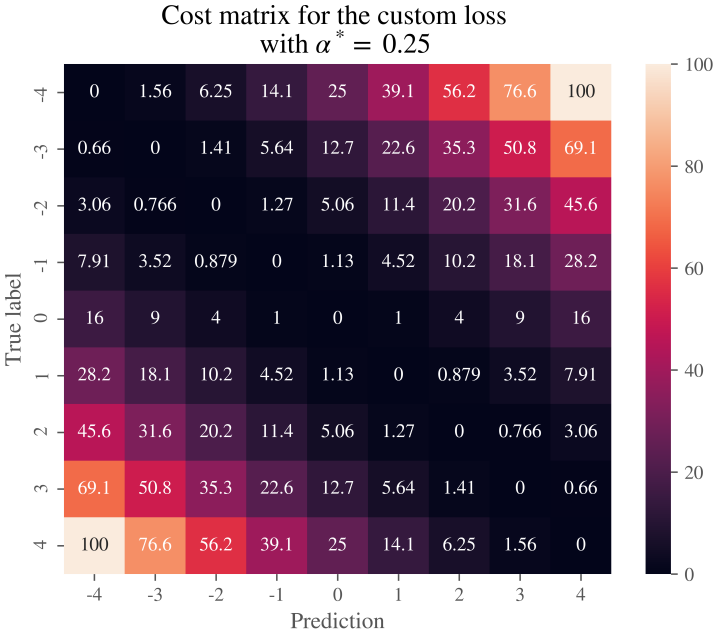


Figure 11: Cost matrix associated with $\alpha^*_{custom} = 0.25$. It is observable that the degree of asymmetry increases with more extreme labels. E.g. $L^{\text{custom}}(\hat{y}_i = -4, y_i = -3, \alpha^*_{custom} = 0.25) = 0.66$ whilst $L^{\text{custom}}(\hat{y}_i = 2, y_i = -1, \alpha^*_{custom} = 0.25) = 0.879$. Furthermore, one can observe the symmetry of the loss in the center of the scale.

5.2.2 Class Weighting

As described in chapter 2, the problem at hand inhibits a highly imbalanced label distribution. This might lead to biases in the interpretation of the performance measures such as the accuracy paradox (Thomas and Balakrishnan 2008) that can lead to overestimation of a model's performance. In chapter 3.2 we discuss and justify our reweighting scheme for the handling of class imbalances in detail and explain its inclusion in the performance measure calculation in this chapter. We include this reweighting scheme as well in the calculation of the performance measures such that the weighted metric $metric_w$, based on an arbitrary loss $L(y_i, f(X_i, \theta))$, can be formulated as:

$$metric_{\mathrm{w}} = \frac{1}{N} \sum_{i=1}^{N} \omega_i L(y_i, f(X_i, \theta)) \tag{18}$$

where the loss contribution of each sample is getting multiplied with the ω_i factor, which controls for class imbalances. The regular, unweighted version is formulated as:

$$metric = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(X_i, \theta)) \tag{19}$$

5.3 Postprocessing

Labels for the motor state of the PxPD were collected on a highly granular minute-level, resulting in models that predict the motor state for unseen sensor data on a minute-level as well. From a clinical perspective, it is unreasonable that the motor state of a patient changes rapidly from one minute to another. The transitions are more smooth in practice. Still, the models treat the individual one-minute-windows of sensor data independently. This leads to the effect that neighbouring windows can be predicted to have very dissimilar motor states due to an abrupt change in the movement pattern but not due to a change in the symptom. To balance this effect, we smooth the predicted motor state day-curves similar to Hammerla et al. (2015). It is reasonable to smooth with future predictions as the deployed model will not be used for real-time monitoring. While smoothing, we want to assign a higher weight to the current window than to windows that lie farther in the past or in the future. Therefore, we draw weights from a symmetric Gaussian that is controlled by two parameters: M, the amount of the neighbouring points, and the standard deviation *std* of the Gaussian, following the scipy implementation by Jones, Oliphant, Peterson, et al. (2001). These weights are then normalized to a sum of 1 and used as a filter function to convolve the raw prediction curves. The effect of smoothing is further illustrated in figure 12.

As the parameters of the Gaussian are hyperparameters, we tune them specifically for each of the LOSO patients. Thus, for the predictions of one patient we use the Nelder-Mead optimizer by Nelder and Mead (1965) to detect the optimal (M_i^*, std_i^*) for this patients which minimizes the weighted Custom-loss over all other LOSO patients. We use the Nelder-Mead optimizer

as it allows for multivariate, gradient-free optimization. One could experiment with related optimization methods such as Bayesian Optimization in future research. For the test patients from data set II, we optimize M and std over the predictions of all patients from the LOSO data set I.

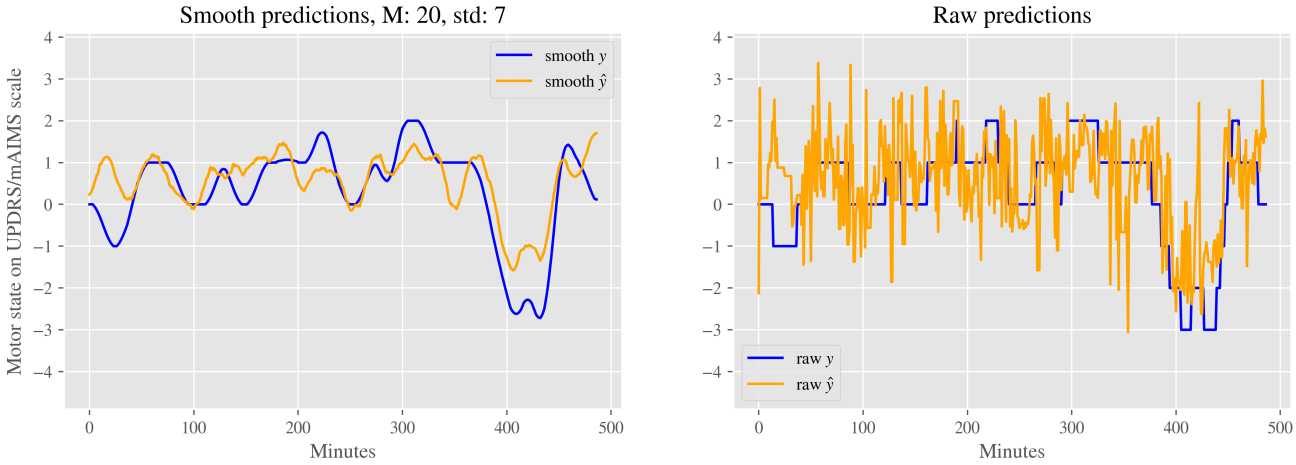


Figure 12: Illustration of the effect of smoothing on model predictions for an exemplary patient from data set I with arbitrary but reasonable M and std values. Predictions are shown in orange, true label values in blue color.

5.4 Results

In this section, we report and discuss the performance of the different models as well as the transfer learning approach. As performance measure for the model comparison, we use the weighted Custom-loss as explained in chapter 5.2. A summary with additional standard evaluation metrics is presented in table 10 in appendix B. The performance is calculated on the model predictions that were smoothed in the postprocessing step following the method from chapter 5.3.

Due to the cost-intensive LOSO validation, which results in the training of $28 \cdot 2 + 1 = 57$ single models per run, a complete run of one architecture on an NVIDIA Tesla P100 GPU takes 60-70 hours, depending on the convergence rate of the single models. Due to limited access to computational resources, we were not able to tune the hyperparameters of the various model architectures explicitly. The hyperparameter setting that we finally chose for all model runs is influenced by findings from early experiments as well as recommendations in related literature. This hyperparameter setting is summarized in table 6. The only hyperparameter that we explicitly tune is the amount of training time using early stopping as described in chapter 5.1. The specific tuning of hyperparameters can have major effects on the model performance in machine learning. Thus, it is highly recommended to invest further time and budget in the fine-tuning of the proposed models.

Table 6: Hyperparameters that were used for the training of all deep learning models. Adam (2014) was chosen as an optimizer. The learning rate in conjunction with the batch size was set due to observations in early experiments. The remaining parameters of the optimizer were set to the default settings. As described in chapter 5.1, the task in the inner folds is more easy than that in the outer folds, requiring less train budget to converge. Therefore, we chose a rather small patience = 6 for early stopping.

Hyperparameter		Value
Batch size		256
Patience		6
Adam optimizer	Learning rate	5e-5
	Weight decay	1e-6
	Beta 1	0.9
	Beta 2	0.999

5.4.1 Model Comparison

In table 7, we show the model performance of the different architectures in the different problem frames evaluated on the non-augmented version of data set I as well as the data set II, referred to as LOSO and Test respectively. We include the baseline model, referred to as Majority Vote, the Random Forest trained on the extracted features from chapter 4.1 and the various deep learning approaches. Note that the lower the weighted Custom-loss of a model, the better its performance.

Table 7: Results for the different models in different problem frames, measured with the class-weighted version of the Custom-loss. The models are validated on data set I in a LOSO setting as well as on data set II, referred to as LOSO and Test respectively. Smaller values in the performance measures suggest better model performance. The FCN architecture in the regression setting performs best on the LOSO data.

Frame	Model	weighted Custom-loss	
		LOSO	Test
Baseline	Majority vote	2.900	0.530
Classification	FCN	0.800	0.329
	Random Forest	1.542	0.348
Ordinal	FCN	0.752	0.160
	Multioutput FCN	0.922	0.150
Regression	FCN	**0.635**	0.141
	FCN Inception	0.726	**0.121**
	FCN ResNet	0.841	0.238
	RCF	0.776	0.188
	FCN Broad	0.673	0.127
	Random Forest	1.310	0.255

The FCN architecture was deployed in all three problem frames. From table 7, we yield the following order in terms of the FCN model performance validated on the LOSO data: classification (weighted Custom-loss = 0.800) < ordered regression (weighted Custom-loss = 0.752) < regression (weighted Custom-loss = 0.635). This order also holds for the model's generalization performance on the Test data set II: classification (weighted Custom-loss = 0.329) < ordered regression (weighted Custom-loss = 0.160) < regression (weighted Custom-loss = 0.141). Next to the FCN, which we applied in all three problem frames, we run the Random Forest model only in the classification and the regression setting. This yields a similar performance ordering: classification (weighted Custom-loss LOSO = 1.542, Test = 0.348) < regression (weighted Custom-loss LOSO = 1.310, Test = 0.255). Thus, we conclude that the regression frame suits this task the best and therefore run the different deep learning models in this frame only.

The comparison of the deep learning models with the classic machine learning model, that is the Random Forest, offers another interesting finding. Both model classes were applied in the classification and in the regression frame. In both frames, we observe that the deep learning models outperform the classic machine learning models on both validation data sets. Even the worst performing deep learning architecture, the FCN ResNet (weighted Custom-loss LOSO = 0.841, Test = 0.238), performs better than the Random Forest (weighted Custom-loss LOSO = 1.310, Test = 0.255) in the regression frame and the FCN (weighted Custom-loss LOSO = 0.800, Test = 0.329) outperforms the Random Forest (weighted Custom-loss LOSO = 1.542, Test = 0.348) in the classification frame as well. This finding justifies our focus on deep learning approaches for the problem at hand.

In their research, Niu et al. (2016) claim the Multioutput CNN architecture to outperform regular regression models in an ordinal regression task. Based on our work, we can not support this claim as the Multioutput FCN shows weaker performance than each of the deep learning architectures which were deployed in the regression frame. With this statement, one has to bare in mind, that the authors used this Multioutput CNN architecture for age estimation on Asian portraits, a task that differs from ours.

Looking at the results from the regression frame which were validated on the LOSO data, we obtain the following performance ordering: Random Forest (weighted Custom-loss LOSO = 1.310) < FCN ResNet (weighted Custom-loss LOSO = 0.841) < RCF (weighted Custom-loss LOSO = 0.776) < FCN Inception (weighted Custom-loss LOSO = 0.726) < FCN Broad (weighted Custom-loss LOSO = 0.673) < FCN (weighted Custom-loss LOSO = 0.635). This suggests that the most simple model architecture, the FCN, performs best on this task. Surprisingly, the more advanced architectures, such as the FCN Inception, the FCN ResNet or the RCF, which are partly inspired by the great success of their 2D counterparts in Computer Vision, do not manage to outperform this rather simple model architecture. At this point, one could argue that the problem at hand is not complex enough to require such sophisticated

models. The increased capability of these more advanced models comes at the price of an heavy increase in model weights. For instance, the FCN consists of 283145 weights, whilst the FCN Inception contains 514809, the FCN ResNet 512385 and the RCF contains 811807 model parameters. Though those models employ strategies such as the residual connections to prevent overfitting, they could still be prone to this problem. Furthermore, such intricate models call for more data availability in order to achieve successful training.

While the regression FCN model performs best on the LOSO data set I, the FCN Inception outperforms it on the Test data set II. When interpreting this result, one should keep in mind that the Test data set II has major differences with the LOSO data set I on which the model was trained: it contains a lot less samples (398 vs. 9356 windows), the data was collected with a different device (Shimmer vs. MSB2) and the class distribution differs heavily from that of the LOSO data set I (see figure 2). Thus, we base our decision in the model comparison mainly on the results from the LOSO validation and conclude that the FCN in the regression frame performs best amongst the frames and model classes that we applied to the problem.

5.4.2 Transfer Learning

For transfer learning, we pretrained a FCN model on the source task of distinguishing movement windows from PxPD and healthy patients. Therefore, we trained the FCN model on the balanced data set III, where 80% of the data are randomly selected as training data and the remaining 20% are used for validation. As this source model is only used for the initialization of the weights for the target task, we trained the model using the hyperparameters from table 6 on this binary classification task with a total budget of 60 epochs, storing the model weights after each epoch. We found the model to converge after 30 epochs and the weights of the best performing epoch with a test accuracy of 89.516% were picked for the initialization of the weights for the target task.

This pretraining was only done for the FCN architectures. Thus, we report the performance of those four pretrained models along with their non pretrained versions in table 8. Within all four experiments, pretraining on the source task increased the model performance in the LOSO setting across the different problem frames. Further, this strategy helped to even improve the performance of the FCN in the regression setting, which was found to be the best performing model in the previous chapter 5.4.1, from a weighted Custom-loss of 0.635 to 0.600 on the LOSO data set I. Thus, the pretrained FCN model in the regression setting is the final, best performing model amongst all models that we applied across the different frames. In figure 13, we show the predicted and the actual day curves for four selected LOSO patients. A visual comparison of the predictions from the pretrained and the regular regression FCN is shown in figure 19 in Appendix E. The prediction curves for all patients are depicted in appendix D. In figure 15, we show the associated confusion matrix for the validation of the model on the LOSO

data set I and on the Test data set II respectively.

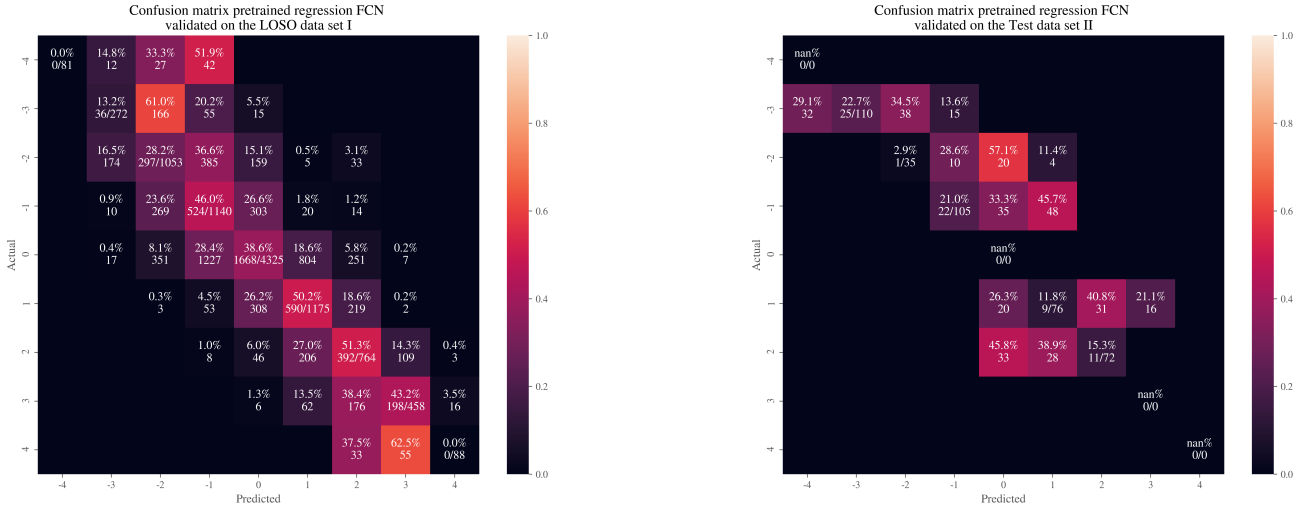
Table 8: Performance of the transfer learning approaches compared to their non-pretrained counterparts. Models were evaluated in the same way as in the previous chapter 5.4.1. Pretraining increases the performance of all models on the LOSO data. Furthermore, it seems to be disadvantageous for the Test data, which was collected with a different IMU device.

Frame	Model	weighted Custom-loss	
		LOSO	Test
Classification	FCN	0.800	0.329
	FCN Transfer	0.771	0.163
Ordinal	FCN	0.752	0.160
	FCN Transfer	0.616	0.249
	Multioutput FCN	0.922	0.150
	Multioutput FCN Transfer	0.657	0.220
Regression	FCN	0.635	0.141
	FCN Transfer	0.600	0.174

Transfer learning has the biggest effect on the performance of the Multioutput FCN in the ordered regression frame. This might be due to the fact that this model architecture basically consists of eight jointly trained binary classifiers. As the source task is also a binary classification problem, we hypothesize that this similarity in source and target task allows the Multioutput FCN to exploit the pre-extracted features the best. Validating this hypothesis further would be an exciting topic for future research.

Despite the performance increase within the LOSO validation, this transfer learning strategy lead to worse model performance on the Test data set II. Both, the data set III, that was used for the source task and the LOSO data set I were collected with an MSB2 IMU. In contrast, the Test data set II was collected with a Shimmer IMU. Hypothetically, the pretraining helps the model to better adapt to the specific data which is beneficial for its performance on data from the same device, but harmful on data from another device. This would stress the importance of standardization amongst devices. Due to the limited scope of this thesis, we leave this topic as well open for further research.

Figure 13: Illustration of the actual day curves (blue) of four selected patients from the LOSO data and the associated motor state predictions (orange) from the pretrained regression FCN, the best performing model. As explained in chapter 5.3, the data was smoothed in a postprocessing step. The model is able to capture the motor state transitions for patients 7 and 47 and also detects the OFF phase of patient 91 correctly. In contrast, the model fails to detect the clear OFF phase of patient 27 and osciliates too heavy in the ON phases at the beginning and the end of the patient's day.



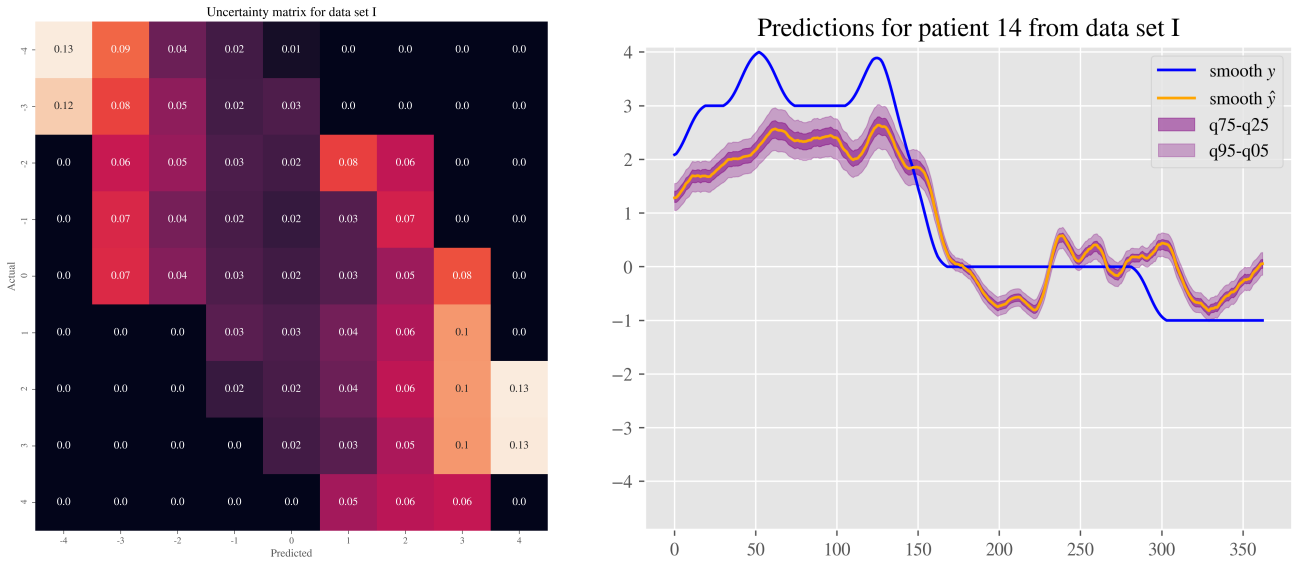
(a) LOSO data set I **(b)** Test data set II

Figure 14: Confusion matrices for predictions from the pretrained FCN in the regression frame on the LOSO data set I (left) and on the Test data set II (right). The predicted values were smoothed and then rounded for the backmapping to class labels. The confusion matrices were color-coded according to the relative values that were normalized for each actual label, the rows in the matrix. The model is hesistant to predict the extreme class labels $\{-4, 4\}$ but produces reasonable results for the other class labels. The heavy class imbalance could be a reason for this behavior.

6 Uncertainty Modeling

In deployment, the predictions from the proposed model will be used to control the medication of real humans suffering from PD. This requires a high level of trust in the model's predictions from both, the patients as well as the doctors. One approach to foster this trust is the extension of the model to output its certainty about the predictions next to the point estimates itself. Amongst the different approaches to uncertainty estimation in deep learning, we propose a first straight forward model extension based on the work of Gal and Ghahramani (2016) and compare it with deep Quantile regression. The following provides the basis for extensions of the main part of our work from sections 3, 4 and 5.

A decent part of the growing literature on uncertainty estimation in deep learning is inspired by Bayesian Statistics, as Bayesian models are designed to model posterior probability distributions instead of point estimates. For instance, Blundell et al. (2015) assume each of the weights of the neural networks to follow a Gaussian and propose a unique optimization method for the training of such models. While this delivers reasonable results, Gal and Ghahramani (2016) point to the "prohibitive computational cost" of such approaches and introduce a more straight-forward and economic strategy using the well-known Dropout regularization technique (Srivastava et al. 2014). Besides model training with Dropout, they propose to use it as well in model inference in order to receive empirical distributions of the predictions on test samples. Basically, T stochastic forward-passes are made for one test sample where the Dropout ensures that within each forward pass, different model weights are randomly set to 0. Due to this stochastic sampling, the method is also referred to as Monte-Carlo Dropout (MC Dropout). The empirical mean and variance are calculated from this predictive distribution and used as the point and uncertainty estimates, respectively. Within their work, Gal and Ghahramani (2016) justify this approach by proposing the dropout network to be the approximation of a Gaussian process. In an extensive proof, they show that minimizing the "dropout objective ... minimises the Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process". This strategy has the major benefit that, instead of other Bayesian approaches, it does not require architectural changes other than including Dropout in the model architecture. Thus, we employ this technique in our work. We insert a linear Dropout layer with a Dropout probability of 0.5 between the GAP and the final layer in the FCN architecture and sample $T = 100$ predictions for each unseen window. The obtained point estimates along with the prediction intervals for a selected LOSO patient are illustrated in figure 15b. Plots for all LOSO and Test patients are depicted in appendix G. Kendall and Gal (2017) discuss the term uncertainty itself and distinguish it into two different types: epistemic uncertainty, which measures what the model does not know and can be reduced by collecting more data, and aleatoric uncertainty, a measure for the uncertainty that is inherent in the problem. The MC Dropout uncertainty estimation quantifies epistemic uncertainty (Gal and Ghahramani 2016), highlighting predictions on input data that is very dissimilar to the ones



(a) Uncertainty matrix **(b)** Point estimates with uncertainty bands

Figure 15: Results from the uncertainty modelling approach via MC Dropout on the LOSO data set I. The left graphic shows the uncertainty matrix, where the average prediction variance for each combination of rounded predicted and actual labels is shown, similar to a confusion matrix. An example plot including predicted point estimates along with the uncertainty bands is depicted on the right. Predictions are postprocessed and ucnertainty bands are depicted as the area between the 95^{th} and the 5^{th} quantile as well as the 75^{th} and the 25^{th} quantile. Both graphics show that the uncertainty for predicted values towards the extremes of the label scale is higher than in the center. This supports the argument, that MC Dropout quantifies the epistemic model risk as there is only little data support for these extreme classes.

the model was trained on. Thus, we expect the estimated uncertainty to be especially high for input data for heavily bradkyinetic and dyskinetic samples due to the high class imbalance in the train data set I. This is supported by the resulting uncertainty matrix in figure 15a, where extreme predictions show a higher average variance than predictions in the center of the scale.

An alternative way to obtain more information from statistical models than pure point estimates is the calculation of prediction intervals. According to Meinshausen (2006), who leveraged Quantile regression to calculate prediction intervals from a Random Forest model, "Quantile regression offers [...] a principled way of judging the reliability of predictions". Quantile regression is based on the idea to predict different conditional quantiles of the data instead of the conditional mean (Koenker and Hallock 2001). As opposed to the quadratic loss, it therefore employs the pinball loss, also referred to as tilted loss, to estimate different quantiles. The resulting prediction intervals offer a clear interpretation: for instance, the predicted value of a new data sample lies with 90% probability within its associated prediction interval which is calculated from the predicted 95^{th} and 5^{th} quantile values. Rodrigues and Pereira (2018) transfer this idea successfully to deep learning and propose a multi-head CNN model that predicts a conditional mean via MSE minimization as well as a variety of quantiles for the calculation of prediction intervals. In addition, they show that this joint multi-task training has a positive regularizing effect increasing the model's generalization performance in their application.

Both approaches have advantages and disadvantages. One main benefit of MC Dropout is that

it is applicable to almost any deep learning architecture with minor to no required change in the architecture. Also, it can be applied to both, classification and regression. On the negative side, the width of the sampled predictive distribution is depending on hyperparameters, such as the dropout ratio or the activation functions, which are difficult to calibrate. Next to that, there has been recent critique by Osband (2016), arguing that MC Dropout uncertainty does not decrease with an increase of data. Thus, at the current state of research, uncertainty estimated based on MC Dropout should be interpreted with caution. Quantile regression in contrast is shown to be applicable in deep learning, but only in regression settings and including architectural adjustments. Beneficial about Quantile regression is that the interpretation of the resulting uncertainty bands has backing in statistical theory. Furthermore Rodrigues and Pereira (2018) suggest the joint training of different quantile estimates to have a regularizing effect increasing the model's generalization performance.

7 Conclusion

Within this thesis we employed a variety of different statistical learning approaches for motor state detection of patients with Parkinson's disease based on sensor data collected with a wearable device. Whilst the vast majority of related literature handles the problem as a classification problem, the high resolution of the data at hand allows us to assess the performance of several suitable approaches. We could show that treating this task as a regression problem works better than framing it as a classification or an ordered regression task. Other than most of the related research, we invested in an appropriate Leave-one-subject-out (LOSO) strategy for enhanced model validation. In its wake, we propose a LOSO validation scheme that is explicitly well suited for iterative learning algorithms such as neural networks. Furthermore, we compared the performance of different deep learning architectures with a classical machine learning model, namely a Random Forest based on manually engineered statistical features. We can draw two main findings from those examinations. First, deep learning approaches outperformed the classic machine learning approach in the present task. Secondly, the most straightforward one among the neural network architectures offered the most promising results. Although this is initially surprising, it gives reason to suspect that the task at hand is simply not complex enough such that it would require more sophisticated architectures. Such intricate models call for more data availability in order to achieve successful training. In order to assess and compare the model performance, we propose the use of a customized performance measure. This measure is a scale-variant extension of the asymmetric Quad-Quad-loss and was developed in close conjunction with the involved medical staff. Since high quality labeled data is scarce and costly in the medical domain, we also showed that transfer learning approaches can be successfully employed in the present time series classification task. As an additional extension of the final model, we provided an outlook on uncertainty modeling. Therefore we briefly introduced the Monte-Carlo Dropout method and compared it with recent advances in deep Quantile regression.

Besides the convincing results of our modelling approaches, there exist also some issues in this thesis which could be tackled in future work. The intense computational cost of the LOSO validation hindered us from properly tuning the hyperparameters of the proposed models. Thus, we suggest at least to tune the parameter setting for the best performing model with more computational budget. For the determination of the optimal amount of training time, we made the strong assumption that models trained with a standard train test split show the same convergence behavior as those trained in a valid LOSO setting. This is one weakness of this work and there is a need for the development of additional advanced cross validation strategies for such problem settings where many data are collected from few individuals. Despite the deployment of a class weighting scheme, the model performed especially bad in predicting the extreme values of the label scale. However, collecting more training data from such extreme classes could mitigate this problem. The data used in this thesis was labeled by one single

rater, which makes the labels prone to a possible subjectivity bias. One method to overcome this problem would be the collection of the patients' movement patterns on videos, which could be labeled by a group of experts in a second step. What is more, the transfer learning approach has lead to a substantial increase in model performance. Thus, pretraining the model on additional wearable-based tasks would be an interesting extension of the current work. In the final section of the thesis, we introduced methods for the estimation of prediction uncertainty. Those could be further leveraged for the application of semi-supervised learning methods in future research to overcome the limited availability of high quality labeled data.

Appendices

A Structured Literature Overview

Table 9: Overview on results from the literature on Motor State detection for PxPD. This table supports the main points from the literature review in chapter 1. The literature is structured according to different criteria such as the (modeling) method, the validation scheme and the collection of the data in a free-living or a lab setting. In the modeling method, the abbreviation MLP refers to a Multi-layer Perceptron neural network and the abbreviation FE to manual feature extraction. Further, SVM refers to a Support Vector Machine and LSTM to Long-short-term-memory neural networks. In the label section, the names of the class labels are depicted. From this column, one can infer that only two authors used continous labels and thus regression models for their task. Generally, a comparison of the different proposed approaches is difficult due to high variation in the various used data sets and methods.

Author	Method	Validation	Subjects	Sensors	Position	Setting	Labels	Labeling	Window	Results
Hssayeni et al. (2018)	Feature extraction (FE), SVM	LOSO	19	6	wrist, ankle	lab	ON/ OFF	rater	1 min	Accuracy: 90.5%
Um et al. (2018)	CNN	LOSO	30	1	wrist	free	OFF/ ON/ DYS	rater	1 min	Accuracy: 63.1%
Sama et al. (2012)	FE, SVM	10 holdout patients	20	1	belt	lab	ON/ OFF	rater	variable	Accuracy: 94%
Hssayeni, Burack, and Ghoraani (2016)	LSTM	LOSO	12	1	ankle	free	ON/ OFF	rater	5 sec	Accuracy: 73.9 %
	FE, SVM	LOSO	12	1	ankle	free	ON/ OFF	rater	5 sec	Accuracy: 65.7 %
Eskofier et al. (2016)	CNN	LOSO	10	2	wrist	lab	ON/ OFF	rater	5 sec	Accuracy: 90.9%
Hammerla, Halloran, and Ploetz (2016)	FE, MLP	Leave-one-day-out	34	2	wrist	free	OFF/ ON/ DYS/ Sleep	diary	1 min	F1: 55%
Hammerla et al. (2015)	FE, MLP	7-fold CV	34	2	wrist	lab	OFF/ ON/ DYS/ Sleep	rater	1 min	F1: 76%
Keijsers, Horstink, and Gielen (2006)	FE, MLP	Train set	23	6	trunk, wrist, leg	lab	ON/ OFF	rater	1 min	F1: 97%
Tsipouras et al. (2011)	FE, MLP	LOSO	29	6	wrist, leg, chest, waist	lab	DYS Y/N	rater	2 sec	Accuracy: 84.3%
Keijsers, Horstink, and Gielen (2003)	FE, MLP	80/20 Split	13	6	trunk, wrist, leg	free	AIMS, continous	2 raters	1min	Accuracy: 77%
Lonini et al. (2018)	FE, RF	LOSO	20	1	wrist	lab	OFF Y/N	rater	5 sec	AUC: 73%
	FE, RF	LOSO	20	1	wrist	lab	Tremor Y/N	rater	5 sec	AUC: 79%
Our approach	CNN	LOSO	28	1	wrist	free	continuous UPDRS + mAIMS	rater	1 min	rel. Accuracy (9-class): 86.95%, MAE: 0.65

B Model Results with Additional Measures

Table 10: Results for the different models measured with a variety of standard performance measures. As explained in chapter 5.2.2, we report the class-weighted next to the standard metric for MAE, MSE and the Custom-loss. The models are validated on LOSO data set I as well as on Test data set II, referred to as LOSO and Test respectively. The relaxed accuracy is an accuracy measure, within which predictions that differ by ±1 from the true class label are also counted as correctly classified. This measure is motivated by the high label noise in the data due to the fact that the data was labeled by one subjective rater only. The two accuracy measures should be interpreted with caution due to the high class imbalances.

Model		Measure															
Frame	Model	Custom		Custom weighted		Relaxed accuracy		Accuracy		F1		MAE		MAE weighted		MSE		MSE weighted	
		LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test	LOSO	Test
Baseline	Majority vote	2.837	5.381	2.900	0.529	0.702	0.460	0.463	0.0	0.293	0.0	0.960	1.819	0.661	0.168	2.101	4.015	1.982	0.392
Classification	FCN	1.483	3.297	0.800	0.329	0.809	0.510	0.340	0.198	0.366	0.254	0.890	1.388	0.312	0.112	1.337	2.747	0.592	0.271
	FCN Transfer	1.534	2.113	0.771	0.164	0.813	0.658	0.361	0.178	0.375	0.231	0.897	1.169	0.318	0.087	1.363	1.760	0.571	0.136
	Random Forest	1.696	5.247	1.542	0.348	0.802	0.470	0.459	0.103	0.394	0.153	0.802	1.716	0.465	0.121	1.295	3.897	1.071	0.263
Ordinal	FCN	1.696	1.999	0.752	0.160	0.767	0.794	0.302	0.171	0.321	0.218	0.985	1.107	0.311	0.085	1.558	1.700	0.562	0.135
	FCN Transfer	1.501	2.642	0.616	0.249	0.802	0.633	0.326	0.224	0.350	0.298	0.921	1.255	0.295	0.100	1.380	2.277	0.470	0.209
	Multioutput FCN	1.433	2.043	0.922	0.150	0.820	0.721	0.352	0.229	0.361	0.288	0.873	1.089	0.344	0.080	1.235	1.636	0.653	0.120
	Multioutput FCN Transfer	1.355	2.517	0.657	0.220	0.829	0.621	0.360	0.246	0.367	0.314	0.857	1.208	0.301	0.095	1.194	2.105	0.481	0.182
Regression	FCN	1.234	1.675	0.635	0.141	0.843	0.802	0.338	0.259	0.346	0.330	0.836	1.010	0.293	0.079	1.091	1.386	0.461	0.080
	FCN Transfer	1.093	2.094	0.596	0.174	0.870	0.658	0.388	0.171	0.407	0.237	0.772	1.167	0.273	0.089	0.968	1.793	0.432	0.147
	FCN Incept	1.328	1.651	0.726	0.121	0.841	0.839	0.370	0.239	0.380	0.299	0.842	0.953	0.304	0.068	1.155	1.348	0.525	0.099
	FCN Residual	1.529	2.613	0.841	0.238	0.809	0.605	0.309	0.211	0.334	0.270	0.924	1.267	0.336	0.101	1.382	2.204	0.626	0.204
	FCN RCF	1.407	2.538	0.776	0.188	0.837	0.626	0.345	0.241	0.354	0.311	0.863	1.245	0.316	0.091	1.227	2.118	0.560	0.157
	FCN Broad	1.298	1.780	0.673	0.127	0.835	0.764	0.339	0.229	0.347	0.292	0.852	1.071	0.294	0.077	1.140	1.505	0.491	0.108
	Random Forest	1.361	3.641	1.310	0.255	0.848	0.492	0.436	0.173	0.411	0.191	0.760	1.444	0.423	0.104	1.047	2.741	0.902	0.195

C Manually Extracted Features

Table 11: Features extracted from the raw accelerometer and gyroscope norm data. These features are used to train the random forest baseline model. The features are extracted for each signal per one-minute window sample.

Feature	Description
average peak distance	average distance between peaks
average peak frequency	amount of peaks / window length
rms	root of the mean of the squared sample points
standard deviation	standard deviation of the sample points
range	range of max and min sample points
mean	mean of the sample points
min	minimum sample point
max	maximum sample point
entropy	entropy of the probability distribution of the histogram of the sample points
variance	variance of the sample points
kurtosis	kurtosis of the sample distribution
skewness	skewness of the sample distribution
quantile 25	25% quantile value of the sample distribution
quantile 50	50% quantile value of the sample distribution
quantile 75	75% quantile value of the sample distribution
energy	sum of the squared sample points
amount of peaks	amount of detected peaks

D Prediction Curves for All Patients

In the following figures 17, 18 and 16 we show the predicted day curves form the best performing model, the pretrained regression FCN, for all patients from the Test data set II and the LOSO data set I.

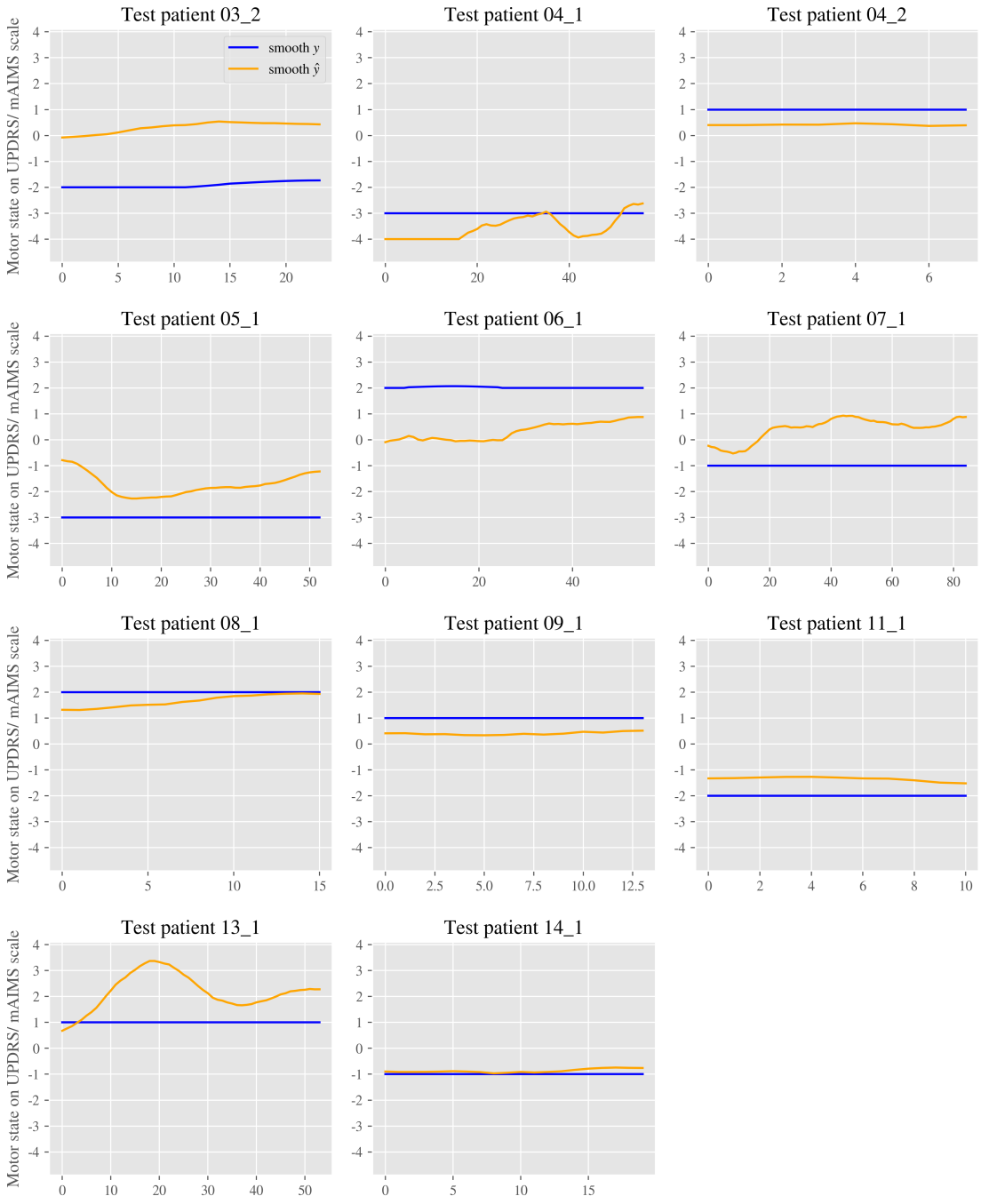


Figure 16: Illustration of the actual (blue) and the predicted day curves (orange) for the 10 Test patients from data set II. Predictions were made from the pretrained regression FCN model which was trained on all patients from the LOSO data set I.

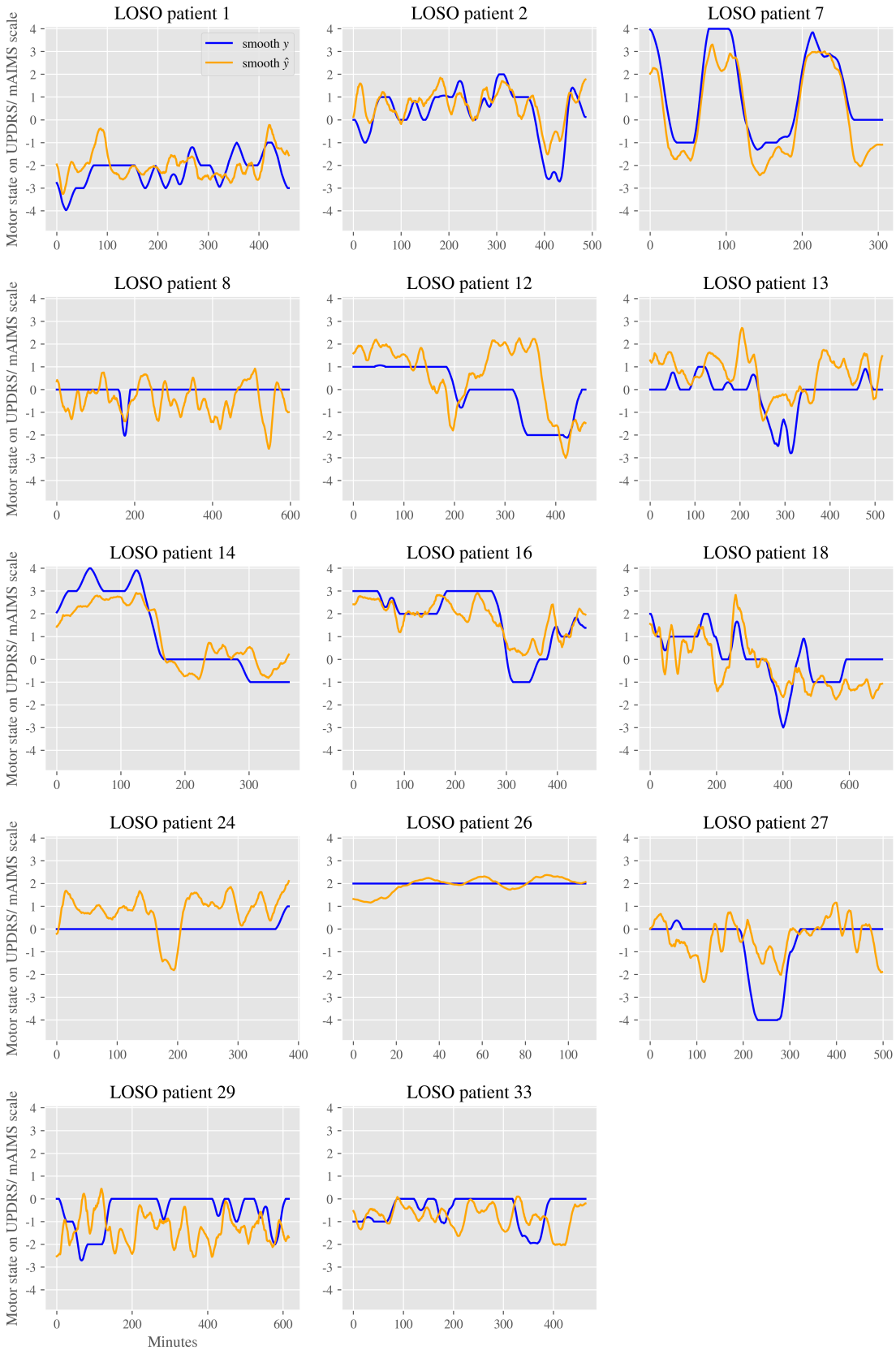


Figure 17: Illustration of the actual (blue) and the predicted day curves (orange) of the 14 first LOSO patients.

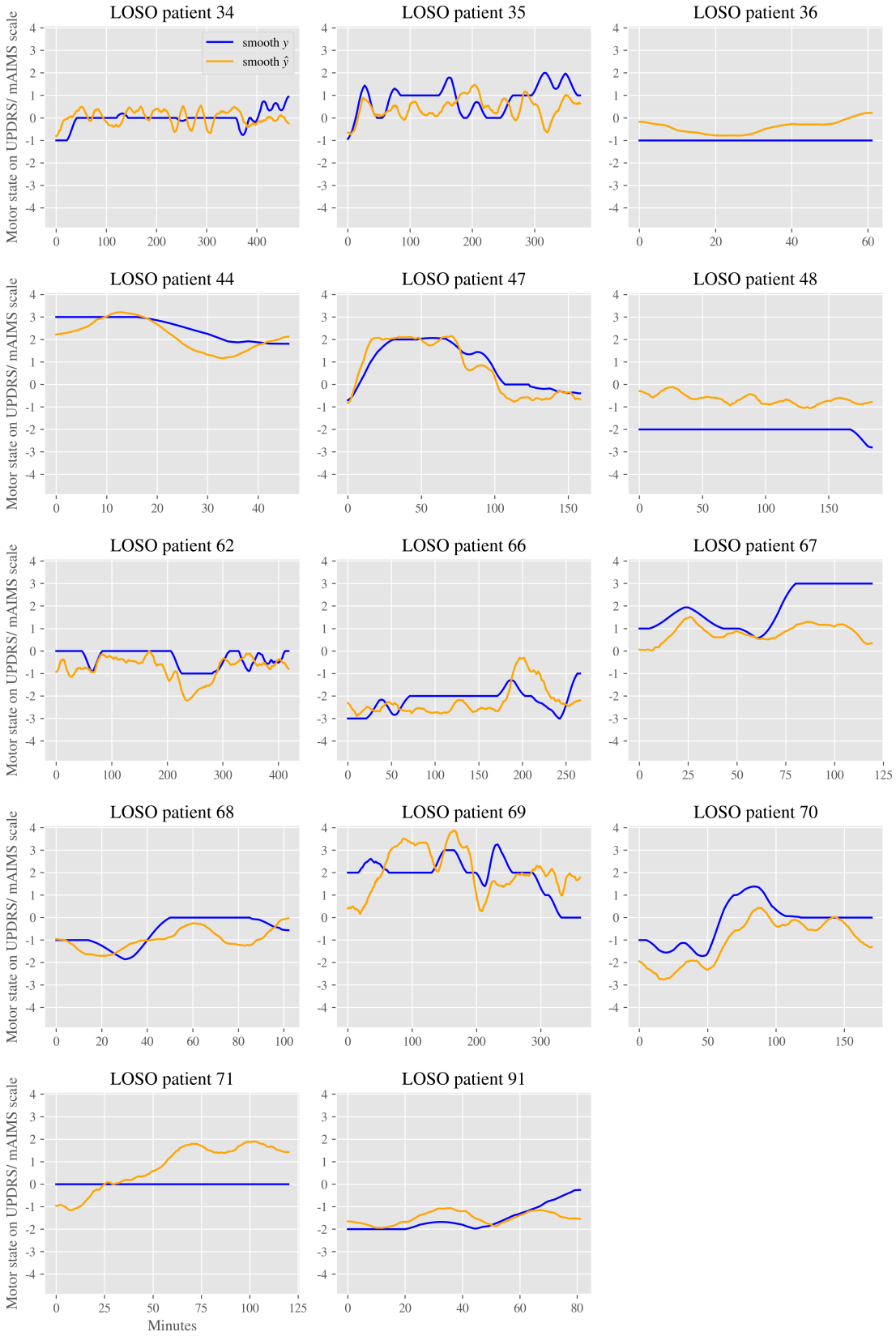


Figure 18: Illustration of the actual (blue) and the predicted day curves (orange) of the 14 last LOSO patients.

E Comparison of Transfer and Regular Model Training

A comparison of the predictions from the pretrained and the regular regression FCN for selected patients from the LOSO data set I is shown in figure 19. Therein, one can observe that the pretrained model approximates the true day curves of the patients even more accurate than the regular, non-pretrained version.

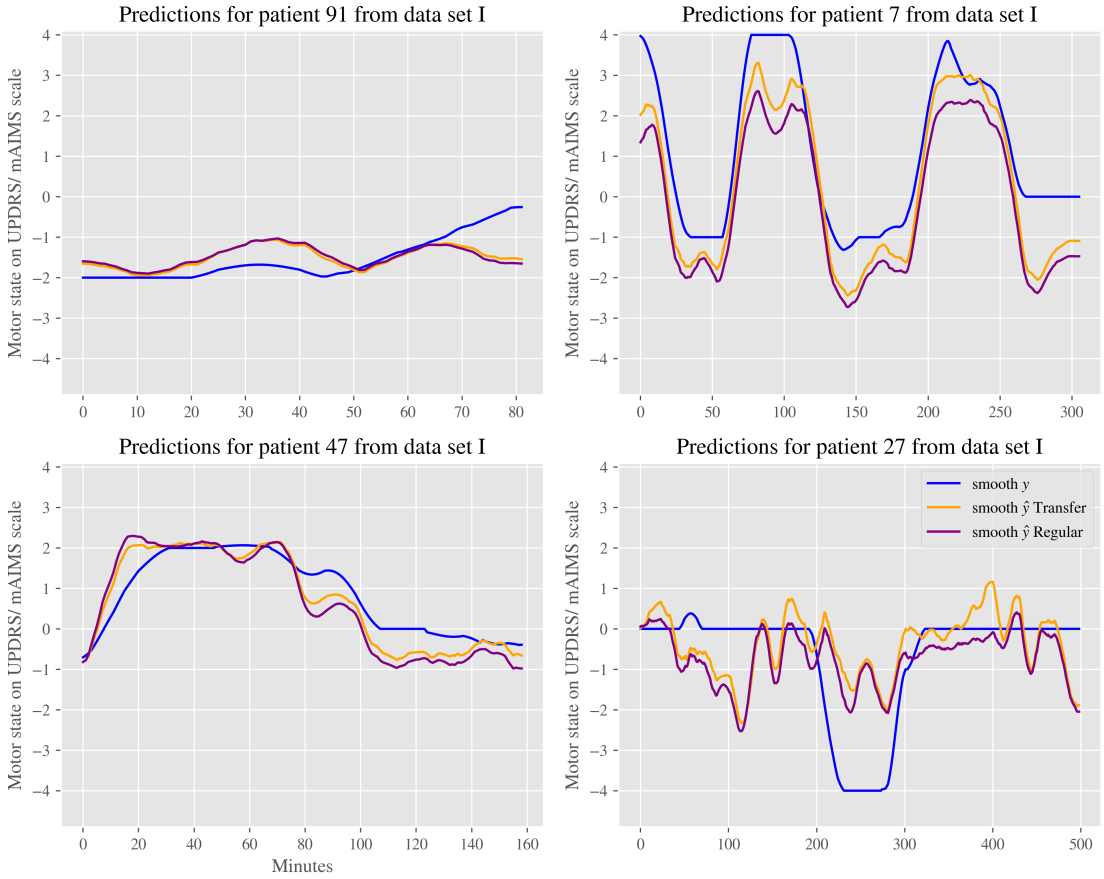


Figure 19: Comparison of the predictions from the pretrained and the regular regression FCN. Ground truth day curves are depicted in blue, predictions from the transfer learning model in orange and those from the regular regression FCN in purple. It is observable, that the predictions from the pretrained model approximate the ground truth motor states better than the regular model. This is most clear on patients 7 and 47.

F Direct Optimization of the Custom-loss

In the current setup and due to the limited scope of this thesis, the Custom-loss described in chpater 5.2.1 is used as an outer loss for model evaluation only and not deployed as a directly optimized inner loss. Still, this would be a possible extension, as the gradient of the Custom-loss can be formulated analytically. Thus, it can be deployed for gradient-based back-propagation in deep learning models and implemented within common frameworks such as tensorflow (Martın Abadi et al. 2015) and pytorch (Paszke et al. 2017). The gradient of the Custom-Loss with respect to the parameters θ of the model $f(X_i, \theta)$ can be formulated as follows:

$$
\begin{aligned}
\frac{\partial L^{\text{custom}}(y_i, f(X_i,\theta), \alpha_{\text{custom}})}{\partial \theta} &= \frac{\partial [\alpha_{\text{custom}} + sign(f(X_i,\theta) - y)]^2 (f(X_i,\theta) - y_i)^2}{\partial \theta} \\
&= (f(X_i,\theta) - y_i)^2 \cdot \frac{\partial [\alpha_{\text{custom}} + sign(f(X_i,\theta) - y_i)]^2}{\partial \theta} \\
&\quad + [\alpha_{\text{custom}} + sign(f(X_i,\theta) - y_i)]^2 \cdot 2(f(X_i,\theta) - y_i) \frac{\partial f(X_i,\theta)}{\partial \theta} \\
&= (f(X_i,\theta) - y_i)^2 \cdot 2[a_{\text{custom}} + sign(f(X_i,\theta) - y_i)] \\
&\quad \cdot \frac{\partial sign(f(X_i,\theta) - y_i)}{\partial \theta} \\
&\quad + [\alpha_{\text{custom}} + sign(f(X_i,\theta) - y_i)]^2 \cdot 2(f(X_i,\theta) - y_i) \frac{\partial f(X_i,\theta)}{\partial \theta} \\
&\overset{\frac{\partial sign(x)}{\partial x} = 2\delta(x)}{=} [\alpha_{\text{custom}} + sign(f(X_i,\theta) - y_i)]^2 \\
&\quad \cdot 2(f(X_i,\theta) - y_i) \frac{\partial f(X_i,\theta)}{\partial \theta}
\end{aligned}
\tag{20}
$$

where the derivative $\frac{\partial sign(x)}{\partial x} = 2\delta(x)$ is the Dirac delta function. The sign function $sign(x)$ is differentiable for $x \in]-\infty, 0[$ and $x \in]0, \infty[$, as $x = 0$ marks the discontinuity point of the function. As $sign(x)$ yields constant values for $x \in]-\infty, 0[$ and $x \in]0, \infty[$, its gradient in these intervals is well defined and equal to 0. If one would approximate the $sign(x)$ with a tangens hyperbolicus $tanh(x)$ to ensure differentiability in all x, the gradient would converge to infinity for x values infinitesimal close to 0. This approximation of the $sign(x)$ function would be a possible extension of the Custom-loss, but we leave this topic open to further research. In practical implementations such as tensorflow (Martın Abadi et al. 2015) and pytorch (Paszke et al. 2017), our deep learning framework of choice, $\frac{\partial sign(x)}{\partial x} \overset{!}{=} 0, \forall x \in]-\infty, \infty[$ for the whole interval, including the discontinuity point. This simplifcation is also used in the final step of equation (20). Furthermore, $\frac{\partial f(X_i,\theta)}{\partial \theta}$ represents the gradient of the neural network $f(X_i, \theta)$ w.r.t. the model weights θ. This gradient can be calculated automatically using pytorch's Autograd functionality.

G MC Dropout Uncertainty Plots

In the following figures 21 and 22 and 20 we show the predicted day curves including the uncertainty bands from the MC Dropout FCN in the regression frame for all patients from the LOSO data set I and the Test data set II respectively.

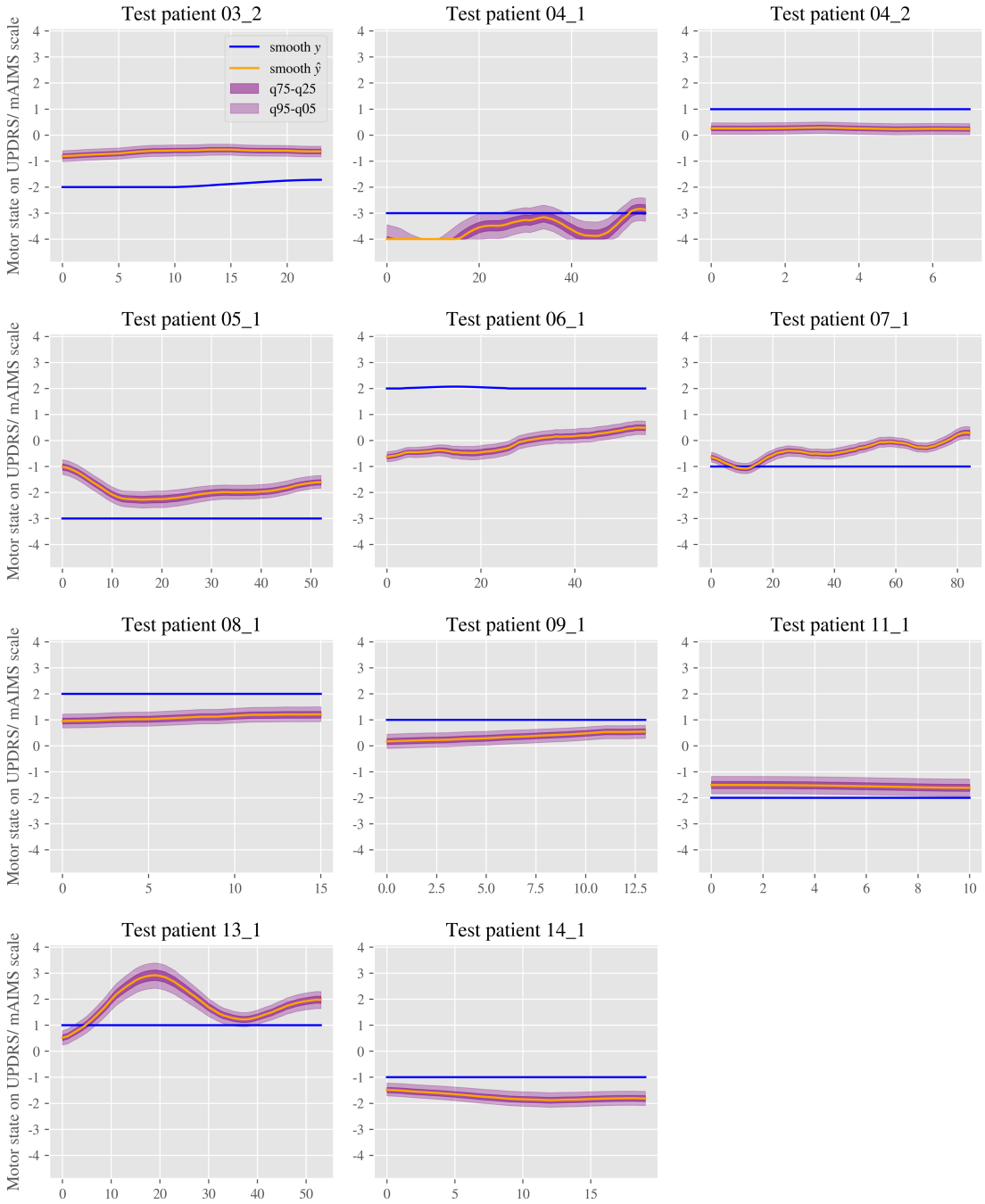


Figure 20: Illustration of the actual (blue) and the predicted day curves (orange) for the 10 Test patients from data set II. The uncertainty bands are depicted as the area between the 95^{th} and the 5^{th} quantile as well as the 75^{th} and the 25^{th} quantile.

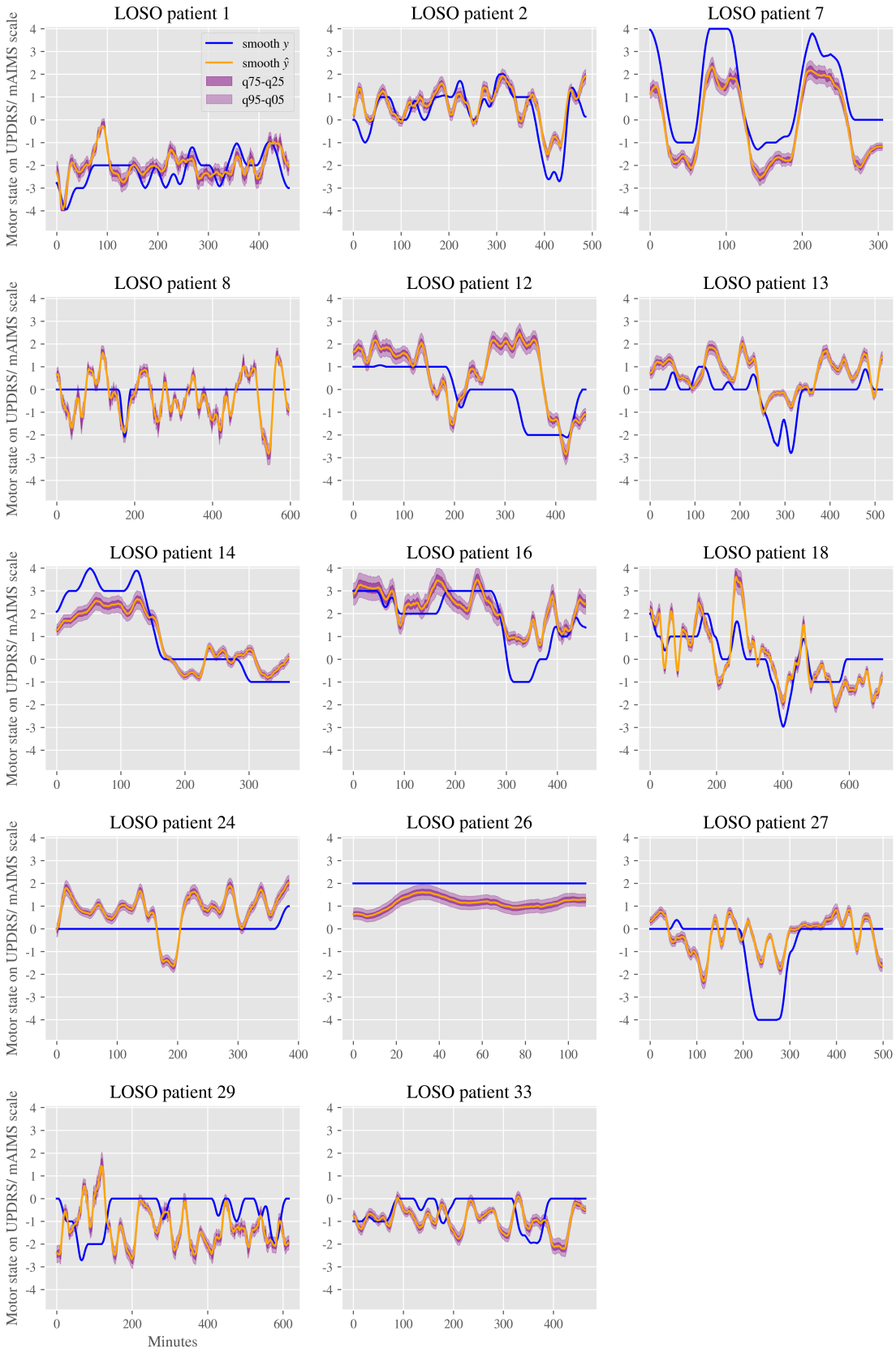


Figure 21: Illustration of the actual (blue) and the predicted day curves (orange) of the 14 first LOSO patients from the MC Dropout FCN. The uncertainty bands are depicted as the area between the 95^{th} and the 5^{th} quantile as well as the 75^{th} and the 25^{th} quantile.

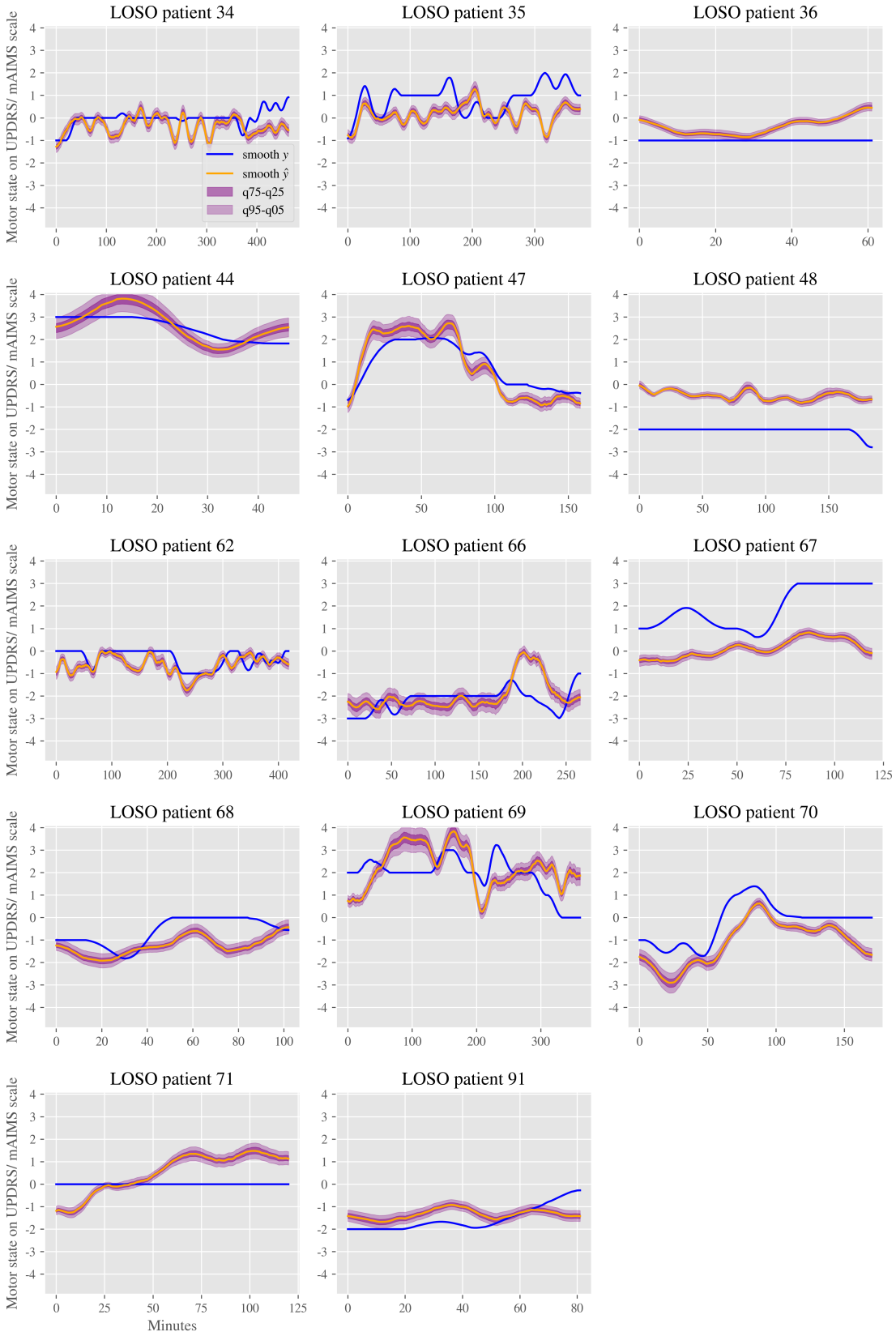


Figure 22: Illustration of the actual (blue) and the predicted day curves (orange) of the 14 last LOSO patients from the MC Dropout FCN. The uncertainty bands are depicted as the area between the 95^{th} and the 5^{th} quantile as well as the 75^{th} and the 25^{th} quantile.

List of Figures

List of Tables

References

Ahlrichs, C., and M. Lawo. 2013. "Parkinson's Disease Motor Symptoms in Machine Learning: A Review". *Health Informatics* 2.

Ahn, Y. B., and Y. Tsuchiya. 2017. "Asymmetric Loss and the Rationality of Inflation Forecasts: Evidence from South Korea". *Pacific Economic Review*.

Bagnall, A., et al. 2017. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". *Data Mining and Knowledge Discovery* 31 (3): 606–660.

Bagnall, A., et al. 2015. "Time-series classification with COTE: the collective of transformation-based ensembles". *IEEE Transactions on Knowledge and Data Engineering* 27 (9): 2522–2535.

Bai, S., J. Z. Kolter, and V. Koltun. 2018. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". *CoRR* abs/1803.01271.

Bao, L., and S. S. Intille. 2004. "Activity recognition from user-annotated acceleration data". *International Conference on Pervasive Computing*: 1–17.

Baydogan, M. G. 2017. *Multivariate Time Series Classification Datasets*.

Berardelli, A., et al. 2001. "Pathophysiology of bradykinesia in Parkinson's disease". *Brain* 124 (11): 2131–2146.

Blundell, C., et al. 2015. "Weight Uncertainty in Neural Network". Ed. by F. Bach and D. Blei. *Proceedings of the 32nd International Conference on Machine Learning* (Lille, France), Proceedings of Machine Learning Research, 37 (): 1613–1622.

Braak, H., et al. 2003. "Staging of brain pathology related to sporadic Parkinson's disease". *Neurobiology of aging* 24 (2): 197–211.

Breiman, L. 2001. "Random forests". *Machine learning* 45 (1): 5–32.

Caiado, J., N. Crato, and D. Peña. 2006. "A periodogram-based metric for time series classification". *Computational Statistics & Data Analysis* 50:2668–2684.

Casale, P., O. Pujol, and P. Radeva. 2011. "Human activity recognition from accelerometer data using a wearable device". *Iberian Conference on Pattern Recognition and Image Analysis*: 289–296.

Chen, S., et al. 2017. "Using ranking-cnn for age estimation". *The IEEE Conference on Computer Vision and Pattern Recognition*.

Christ, M., A. W. Kempa-Liehr, and M. Feindt. 2016. "Distributed and parallel time series feature extraction for industrial big data applications". *CoRR* abs/1610.07717. arXiv: 1610.07717.

Christ, M., et al. 2018. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh–A Python package)". *Neurocomputing*.

Christoffersen, P. F., and F. X. Diebold. 1996. "Further results on forecasting and model selection under asymmetric loss". *Journal of applied econometrics* 11 (5): 561–571.

Csurka, G. 2017. "Domain adaptation for visual applications: A comprehensive survey". *arXiv preprint arXiv:1702.05374*.

Dau, H. A., et al. 2018. *The UCR Time Series Classification Archive*. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Deng, J., et al. 2009. "ImageNet: A Large-Scale Hierarchical Image Database".

Domingos, P. 2012. "A few useful things to know about machine learning". *Communications of the ACM* 55 (10): 78–87.

Elliott, G., A. Timmermann, and I. Komunjer. 2005. "Estimation and testing of forecast rationality under flexible loss". *The Review of Economic Studies* 72 (4): 1107–1125.

Eskofier, B. M., et al. 2016. "Recent machine learning advancements in sensor-based mobility analysis: deep learning for Parkinson's disease assessment". *Engineering in Medicine and Biology Society*: 655–658.

Fawaz, H. I., et al. 2018a. "Deep learning for time series classification: a review". *arXiv preprint arXiv:1809.04356*.

— . 2018b. "Transfer learning for time series classification". *arXiv preprint arXiv:1811.01533*.

Fisher, J. M., et al. 2016. "Unsupervised home monitoring of Parkinson's disease motor symptoms using body-worn accelerometers". *Parkinsonism & related disorders* 33:44–50.

Frank, E., and M. Hall. 2001. "A simple approach to ordinal classification". *European Conference on Machine Learning*: 145–156.

Fulcher, B. D., and N. S. Jones. 2014. "Highly comparative feature-based time-series classification". *IEEE Transactions on Knowledge and Data Engineering* 26 (12): 3026–3037.

Gal, Y., and Z. Ghahramani. 2016. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". *International Conference on Machine Learning*: 1050–1059.

Ghika, J., et al. 1993. "Portable system for quantifying motor abnormalities in Parkinson's disease". *IEEE Transactions on Biomedical Engineering* 40 (3): 276–283.

Goetz, C. G., et al. 2008. "Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): scale presentation and clinimetric testing results". *Movement disorders: official journal of the Movement Disorder Society* 23 (15): 2129–2170.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. "Deep Learning". Book in preparation for MIT Press.

Hammerla, N. Y., S. Halloran, and T. Ploetz. 2016. "Deep, convolutional, and recurrent models for human activity recognition using wearables". *arXiv preprint arXiv:1604.08880*.

Hammerla, N. Y., et al. 2015. “PD Disease State Assessment in Naturalistic Environments Using Deep Learning.”: 1742–1748.

Hastie, T., R. Tibshirani, and J. H. Friedman. 2009. *The elements of statistical learning*. Ed. by R. Tibshirani and J. H. Friedman. Springer series in statistics. New York: Springer.

He, H., and E. A. Garcia. 2009. “Learning from imbalanced data”. *IEEE Transactions on Knowledge & Data Engineering*, no. 9: 1263–1284.

He, K., et al. 2015. “Deep Residual Learning for Image Recognition”. *CoRR* abs/1512.03385. arXiv: 1512.03385.

Herbrich, R., T. Graepel, and K. Obermayer. 1999. “Support vector learning for ordinal regression”. *International Conference on Artificial Neural Networks*.

Hills, J., et al. 2014. “Classification of time series by shapelet transformation”. *Data Mining and Knowledge Discovery* 28 (4): 851–881.

Hoehn, M. M., and M. D. Yahr. 1998. “Parkinsonism: onset, progression, and mortality”. *Neurology* 50 (2): 318–318.

Hoo-Chang, S., et al. 2016. “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning”. *IEEE transactions on medical imaging* 35 (5): 1285.

Hssayeni, M. D., M. A. Burack, and B. Ghoraani. 2016. “Automatic assessment of medication states of patients with Parkinson’s disease using wearable sensors”: 6082–6085.

Hssayeni, M. D., et al. 2018. “Wearable-based Mediation State Detection in Individuals with Parkinson’s Disease”. *arXiv preprint arXiv:1809.06973*.

Ioffe, S., and C. Szegedy. 2015. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. *CoRR* abs/1502.03167. arXiv: 1502.03167.

Jankovic, J. 2008. “Parkinson’s disease: clinical features and diagnosis”. *Journal of neurology, neurosurgery & psychiatry* 79 (4): 368–376.

Jellinger, K. A. 2009. “Recent advances in our understanding of neurodegeneration”. *Journal of neural transmission* 116 (9): 1111–1162.

Jensen, A., and A. la Cour-Harbo. 2001. “Ripples in mathematics: the discrete wavelet transform”. *Springer Science & Business Media*.

Jones, E., T. Oliphant, P. Peterson, et al. 2001. *SciPy: Open source scientific tools for Python*.

Jothimani, D., R. Shankar, and S. S. Yadav. 2016. “Discrete wavelet transform-based prediction of stock index: a study on National Stock Exchange Fifty index”. *arXiv preprint arXiv:1605.07278*.

Karim, F., et al. 2018a. “LSTM fully convolutional networks for time series classification”. *IEEE Access* 6:1662–1669.

Karim, F., et al. 2018b. "Multivariate LSTM-FCNs for Time Series Classification". *arXiv preprint arXiv:1801.04503*.

Keijsers, N. L., M. W. Horstink, and S. C. Gielen. 2006. "Ambulatory motor assessment in Parkinson's disease". *Movement disorders: official journal of the Movement Disorder Society* 21:34–44.

— . 2003. "Automatic assessment of levodopa-induced dyskinesias in daily life by neural networks". *Movement disorders: official journal of the Movement Disorder Society* 18:70–80.

Kendall, A., and Y. Gal. 2017. "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*: 5574–5584.

King, G., and L. Zeng. 2001. "Logistic regression in rare events data". *Political analysis* 9:137–163.

Kingma, D. P., and J. Ba. 2014. "Adam: A Method for Stochastic Optimization". *CoRR* abs/1412.6980. arXiv: `1412.6980`.

Koenker, R. W., and K. F. Hallock. 2001. "Quantile Regression". *Journal of Economic Perspectives* 15 (4): 143–156.

Lane, R. D., et al. 1985. "Assessment of tardive dyskinesia using the Abnormal Involuntary Movement Scale." *Journal of Nervous and Mental Disease*.

Li, L., and H.-T. Lin. 2006. "Ordinal Regression by Extended Binary Classification". *Proceedings of the 19th International Conference on Neural Information Processing Systems* (Canada), NIPS'06: 865–872.

Lin, M., Q. Chen, and S. Yan. 2013. "Network in network". *arXiv preprint arXiv:1312.4400*.

Lonini, L., et al. 2018. "Wearable sensors for Parkinson's disease: which data are worth collecting for training symptom detection models". *NPJ DIGITAL MEDICINE* 1.

Martın Abadi et al. 2015. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.

Meinshausen, N. 2006. "Quantile regression forests". *Journal of Machine Learning Research* 7:983–999.

Nanopoulos, A., R. Alcock, and Y. Manolopoulos. 2001. "Feature-based classification of time-series data". *International Journal of Computer Research* 10 (3): 49–61.

Nelder, J. A., and R. Mead. 1965. "A simplex method for function minimization". *The computer journal* 7 (4): 308–313.

Niu, Z., et al. 2016. "Ordinal regression with multiple output cnn for age estimation". *Proceedings of the IEEE conference on computer vision and pattern recognition*: 4920–4928.

Nweke, H. F., et al. 2018. "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges". *Expert Systems with Applications*.

Oliver, A., et al. 2018. “Realistic Evaluation of Deep Semi-Supervised Learning Algorithms”. *CoRR* abs/1804.09170. arXiv: 1804.09170.

Oord, A. van den, et al. 2016. “WaveNet: A Generative Model for Raw Audio”. *CoRR* abs/1609.03499. arXiv: 1609.03499.

Osband, I. 2016. “Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout”.

Paszke, A., et al. 2017. “Automatic differentiation in PyTorch”.

Pedregosa, F., et al. 2011. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* 12:2825–2830.

Pringsheim, T., et al. 2014. “The prevalence of Parkinson’s disease: A systematic review and meta-analysis”. *Movement disorders* 29 (13): 1583–1590.

Rajkomar, A., et al. 2018. “Scalable and accurate deep learning with electronic health records”. *npj Digital Medicine* 1 (1): 18.

Rakthanmanon, T., et al. 2013. “Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping”. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7 (3): 10.

Rodrigues, F., and F. C. Pereira. 2018. “Beyond expectation: Deep joint mean and quantile regression for spatio-temporal problems”. *arXiv preprint arXiv:1808.08798*.

Ronao, C. A., and S.-B. Cho. 2016. “Human activity recognition with smartphone sensors using deep learning neural networks”. *Expert Systems with Applications* 59:235–244.

Saeb, S., et al. 2017. “The need to approximate the use-case in clinical machine learning”. *Gigascience* 6 (5): 1–9.

Sama, A., et al. 2012. “Dyskinesia and motor state detection in Parkinson’s disease patients with a single movement sensor”. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*: 1194–1197.

Serrà, J., S. Pascual, and A. Karatzoglou. 2018. “Towards a universal neural network encoder for time series”. *Frontiers in Artificial Intelligence and Applications* 308:120–129.

Srivastava, N., et al. 2014. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. *Journal of Machine Learning Research* 15:1929–1958.

Szegedy, C., et al. 2015. “Going deeper with convolutions”. *Conference on Computer Vision and Pattern Recognition*: 1–9.

Szegedy, C., et al. 2016. “Rethinking the Inception Architecture for Computer Vision”. *IEEE Conference on Computer Vision and Pattern Recognition*: 2818–2826.

Thomas, C., and N. Balakrishnan. 2008. “Improvement in minority attack detection with skewness in network traffic”. *Data Mining, Intrusion Detection, Information Assurance and Data Networks Security*.

Toth, E. 2015. “Asymmetric error functions for reducing the underestimation of local scour around bridge piers: application to neural networks models”. *Journal of Hydraulic Engineering* 141 (7): 04015011.

Tsipouras, M. G., et al. 2011. “On automated assessment of Levodopa-induced dyskinesia in Parkinson’s disease”. *Engineering in Medicine and Biology Society*: 2679–2682.

Um, T. T., et al. 2018. “Parkinson’s Disease Assessment from a Wrist-Worn Wearable Sensor in Free-Living Conditions: Deep Ensemble Learning and Visualization”. *arXiv preprint arXiv:1808.02870*.

Varoquaux, G., et al. 2017. “Assessing and tuning brain decoders: cross-validation, caveats, and guidelines”. *NeuroImage* 145:166–179.

Wang, J., et al. 2018. “Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis”. *Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining*: 2437–2446.

Wang, X., K. Smith, and R. Hyndman. 2006. “Characteristic-based clustering for time series data”. *Data mining and knowledge Discovery* 13 (3): 335–364.

Wang, X., A. Wirth, and L. Wang. 2007. “Structure-based statistical features and multivariate time series clustering”, *International Conference on Data Mining*: 351–360.

Wang, Z., W. Yan, and T. Oates. 2017. “Time series classification from scratch with deep neural networks: A strong baseline”. *International Joint Conference on Neural Networks*: 1578–1585.

Yang, J., et al. 2015. “Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.”, *International Joint Conferences on Artificial Intelligence* 15:3995–4001.

Yang, Q., and X. Wu. 2006. “10 challenging problems in data mining research”. *International Journal of Information Technology & Decision Making* 5 (04): 597–604.

Yosinski, J., et al. 2014. “How transferable are features in deep neural networks?” *Advances in neural information processing systems*: 3320–3328.

Zeng, M., et al. 2014. “Convolutional neural networks for human activity recognition using mobile sensors”. *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*: 197–205.

Statutory Declaration

I hereby confirm that I composed the present thesis with the topic

Wearable-based Severity Detection in the Context of Parkinson's Disease Using Deep Learning Techniques

independently and that I have used no other sources other than those cited in the text. The text passages which are taken from other works in wording or meaning I have identified in each individual case by stating the source. This applies also to all graphics, drawings, maps and images included in the thesis. Neither this, nor a similar work, has been published or presented to an examination committee.

Date, Signature