IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2021, accepted September 23, 2021, date of publication October 4, 2021, date of current version October 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117543

Impact of Data Preparation in Freezing of Gait Detection Using Feature-Less Recurrent Neural Network

ALI HADDADI ESFAHANI[1], ZOYA DYKA[1], STEFFEN ORTMANN[2], AND PETER LANGENDÖRFER[1,3]

[1]IHP–Leibniz-Institut für Innovative Mikroelektronik, 15236 Frankfurt (Oder), Germany
[2]Carl-Thiem-Klinikum Cottbus, 03048 Cottbus, Germany
[3]BTU Cottbus-Senftenberg, 03046 Cottbus, Germany

Corresponding author: Ali Haddadi Esfahani (haddadi@ihp-microelectronics.com)

This work was supported in part by the Federal Ministry of Education and Research of Germany under Grant 03ZZ5301E.

ABSTRACT Many studies showed the feasibility of detecting Freezing of Gait (FOG) of Parkinson's patients by using several numbers of inertial sensors worn on the body and in back-end computing power. This work uses machine learning approaches for analyzing the data of one single body-worn inertial sensor system to classify and detect FOG. Long-Short-Term-Memory (LSTM) is employed as the FOG detection algorithm and the Daphnet (FOG and normal gait) dataset provides the data for model training and testing in this paper. The model considers raw data from three channels of the acceleration sensor mounted on the patient's shank and ignores all other data from other sensors. The model is patient dependent and uses sensitivity and specificity metrics to evaluate the model's performance. In this paper, we propose a novel padding method that is applied to the windows of FOG and non-FOG with zero overlaps on the training set and adapts the padding to the individual regions. This method produces windows of only one type of data and label. The proposed padding method reduces the padding amount by two orders of magnitude compared to bigger batch sizes in the sequence splitting method offered by MATLAB 2019a. The padding amount is independent of the batch size. Raw data is fed to the model in the testing mode without any pre-processing or data transformation. The standard rolling window generates fixed-size windows for the test set without overlap and the higher amount of FOG or Normal walking data which defines the label of the individual window. The model for one-second long windows applied in this work outperformed the literature results with a sensitivity of 92.57% and a specificity of 95.62% compared to 82% and 94% reported by Masiala *et al.*

INDEX TERMS Freezing of gait, long short term memory, machine learning, recurrent neural network, time-series classification.

I. INTRODUCTION

Parkinson's disease (PD) is a long-term effect on selective neurological conditions mostly diagnosed for elderly people [1]. The disease degenerates the neurons and reduces the dopamine substance which leads to alteration in the motor and non-motor symptoms like Parkinson's tremors [2]. It mainly affects motor symptoms such as gait disturbance, gait festinating, Freezing of Gait (FOG), slowness of movements, gastrointestinal disturbance, postural instability, and falls [3]. Furthermore, aging and Parkinson's disease have a substantial impact on gait nonlinear dynamics. Age, in particular, influences randomness, whereas Parkinson's disease changes regularity and stability [4]. We are aware of the fact that also the progress of the Parkinson's disease impacts parameters indicating FOG. This means that the LSTM network needs to be trained with individual Patients' data from time to time, but this is out of scope of this paper. The effects of the Parkinson's disease clearly make the execution of Activities of Daily Living (ADL) difficult and have a high influence on the Quality of Life (QOL) of the patient. Concretely, the patient's autonomy decreases over time. There are

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

also non-motor impacts of Parkinson such as depression and hallucination, sleep disorder, and personality change [5].

Although recent drug therapies can reduce the impact of PD, FOG events can remain and happen even if the patient takes his/her medicine [6], [7]. The dopamine level is mainly effective for a short time, e.g., 45 minutes, and strongly depends on the activity. Thus holding a sufficient constant level of dopamine over time is impossible with the drug intake only. However, cueing methods like visual, somatosensory, or rhythmic sounds can considerably reduce FOG episodes' duration and frequency [8], [9].

Wearables are an option for automatic assessment, analysis, and decision-making on PD signs [10]. Wearable sensors are non-invasive monitoring means that can be worn on the patient's body. Wearable sensors are becoming increasingly popular for automatic FOG detection. Such sensors and corresponding computation modules are widely adopted and employed in the biomedical research field, e.g., in terms of analyzing gait impairment of a simple ambulatory setting without using lab experts [10], [11]. A large number of researchers analyzed gait impairment quantitatively by extracting parameters such as velocity, instantaneous orientation, and position of the lower limb, step length, foot clearance, gait cycle, the proportion of heel strike, foot flat, heel off, and swing using Inertial Measurement Units (IMU) sensors. However, none of these parameters were used in analyzing FOG detection [11], [12].

Wearable devices are affordable, small and a battery can run the device for several days. They allow to mount the sensor at the positions at which they provide the most significant data for detecting FOG. In addition, their size ensures that they cannot be detected by fellow men, which is important for the Parkinson's patients. Wearable devices can be designed remarkably user friendly i.e. all the user need to do is to attach them to their body and switch them on when using the devices. Only recharging has to be done when the device is not worn. Even though recent smart phones provide powerful processing units and large memory and due to this, support complex mathematical operations, they cannot replace body-worn sensors, as the data recorded by a sensor integrated in the smartphone worn in the pocket of a trouser will contain by far less information about changes in the movement of the patient than data recorded by sensor directly attached to the limbs of the patient, e.g., to the shank. So, there will be some sensors attached to the body of the patient. Sending the raw data to the smartphone is quite challenging and will delay the assessment that may prohibit in time cueing for which max 250 ms are available.

Human activity analysis can incorporate one or multi wearable sensor nodes analyzing the detailed body movement through the data from each sensor node and provide immediate feedback. Wearable devices with a single sensor node are usually employed for activity classification but a multi-sensor setup can also provide more information from each part of the body that sensor is worn [13]. Although more information is available by multi-sensors setup, the complexity of sensor networks in data transmission, time consistency among the sensors, and assigning the main processing unit to analyze all acquired data introduce much more complexity than single-sensor setup. Moreover, body-worn sensor networks lead to patients' mobility complexities which could impose some constraints to their daily activities. From earlier experiments, it is known that a single sensor worn at the shank of the patient provides sufficient data to detect FOG. Parkinson's patients express their wish to get an unobtrusive solution and are suffering from limited mobility by complex body-worn sensor networks which will not satisfy the patients' needs. This is the reason that motivated us to focus on single sensor nodes to satisfy their needs.

Machine Learning (ML) algorithms can employ the time-series data acquired from a wearable sensor. These algorithms have a great capability to specifically learn and detect FOG events with a dataset containing annotated data from a clinical expert who recognizes and labels the data correctly. The lack of big datasets containing various types of activity under real-life conditions and the limited amount of labeled data distributed among the subjects are the main challenges researchers face in this field [14]. Nevertheless, the recent robust ML algorithms are leveraging small datasets for FOG detection with high-performance results.

This work aims to automatically detect FOG events using supervised ML algorithms and train the model with data prepared using padding, i.e., we added zero values to the data as the pre-processing step aiming at keeping the number of zeros added as low as possible. The unprocessed data representing real-life scenarios are used to test the individual model for each patient. The model shall then interpret raw data stemming from inertial sensors' channels and classify them as FOG or non-FOG. The Daphnet dataset is a publicly available dataset from which data for training and testing benchmarks in this work are used. The windowing strategy is applied to take a fixed number of sampled data representing windows containing a subset of data that does not have an overlap with neighbor windows.

The conventional windowing strategy may lead to two labels of the data in a single window where the type of data changes in the dataset, i.e. when normal walking changes to FOG or vice versa. However, in the training set, windows can only represent one type of data, Normal or FOG. The padding methods proposed and employed in MATLAB solve the issue for such problematic windows and generate windows - from the Daphnet dataset - which do not contain two labels of data. The resulting windows have only one label of data.

The first contribution of this paper is a new padding method which is introducing a constant padding amount at any mini-batch size and reducing the undesired padding amount by two orders of magnitude compared to the MATLAB method. The proposed method prepares the data in such a way that all windows contain unique labeled data and are filled with padding without removing any parts of data.

The key aspect of this architecture is that the model uses feature-less data, i.e. the raw data generated by IMU sensor

which is directly fed to the model without applying any mathematical methodology for extracting any features. The computation cost is zero in feature-less data for our architecture, while other research works extract numerous features to prepare the data before feeding in their ML models [15]–[17]. As we are aiming at using our model in a wireless body-worn sensor network, reducing the effort for processing the data is essential. Here feature-less processing helps to avoid the necessity of pre-processing the data for extracting the essential features.

Long Short Term Memory (LSTM) network is an extension of Recurrent Neural Network (RNN) capable to learn long-term dependencies over time-series data and is used as a classification model to detect FOG events. The second contribution of this paper is a new approach when training the model, i.e., training the model with half of a patient's data as well as other patients of the dataset. The individual model trained for a specific patient validates the model performance by the other half of the data from the same patient that was not used for training. Our experimental results show that our LSTM network using raw acceleration data could achieve an average sensitivity of 92.57%, specificity of 95.62%, and Area Under the Curve (AUC) score of 97.62% outperforming the state-of-the-art approaches using the Daphnet dataset significantly.

This paper is organized as follows. Section II discusses related work. Section III provides a brief description of the Daphnet dataset and the proposed padding method on training data as well as of the data representation in test set. Section IV describes LSTM in FOG detection and section V shows the details of the neural network implementation. Section VI presents the evaluation results of our approach. Section VII concludes the paper and highlights our future directions.

II. DATASET, BACKGROUND, AND RELATED WORK

A. DATASET

Daphnet is a publicly available dataset of movement data records from 10 Parkinson patients, seven males and three females, diagnosed with PD having significant variation in their motor skills [18]. Eight patients out of 10 elicited FOG events during the test, while the remaining two did not show any freezing, and their gait appeared like normal walking. The Daphnet dataset is based on a protocol composed of three main activity sessions, and each one consists of walking tasks designed to show the different activities of daily walking. The data are labeled as FOG and Normal walking data. Three sensor nodes are placed at different locations at the body, i.e., shank, thigh, and on the lower back of patients. Each sensor node provides three channels of raw data from acceleration, gyroscope, and magnetometer sensors corresponding to x, y, and z coordinates acquiring data at 64 *Hz* frequency [19]. In this work, the Daphnet dataset was used to compare our own results with the state-of-the-art work of other researchers using the same dataset.

B. TERMINOLOGIES AND EVALUATION METRICS

In this paper, we use the following terms that we will shortly define here to avoid misunderstandings. A **region** means a continuous data recording holding the same label. A region ends when the label of data is changed showing the beginning of a new region. For example, a patient walks normally, region 1, and then faces a FOG event, region 2. The FOG event finishes after a few seconds and finally the patient walks normally again, region 3, see Fig. 1. A region continues until the label of data changes thus, regions can have any length. A **window** is a part of a dataset containing a specific amount of data. This window has only one label associated to the whole data inside. So a region may consist of several windows. But towards the end of a region there might not be sufficient data to fill a complete new window. In such a case **padding** is used to increase the length of the data so it reaches to the border of the window length. Padding is adding zero values to the data.

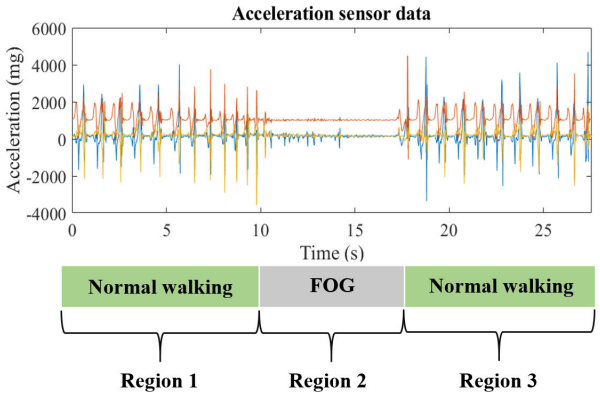

FIGURE 1. Each region is associated with continuous data recorded having the same label. When the label of data changes, a region ends and a new one corresponding to the changed label begins.

A **training set** is a part of the dataset used in the training phase, while the rest of the dataset is considered a **test set**. The **training and test sets** do not have any shared data. **Mini-Batch** is a subset of the dataset fed in the ML model in the forward process, and the model calculates the loss function in the training step according to the whole data in the mini-batch. After each mini-batch, the backward propagation computes the gradient of the loss function with respect to each learnable parameter and updates the parameters. One **epoch** means that all mini-batches in a training set, i.e., the complete training set, have gone through the neural network, and each sample in the dataset was incorporated in updating the model parameters. The **training phase** refers to having several epochs and iterations over the training set several times. The updated parameters in the model are tested in the testing phase using only the test set.

The performance evaluation in our approach is based on the assessment of windows. The label assigned to each window is compared to the ground truth of the annotation. The windows correctly labeled as FOG are counted as True Positive (TP),

whereas the wrongly classified FOG windows are counted as False Negative (FN). Correctly classified Normal walking windows are True Negatives (TN), whereas the incorrectly classified Normal walking windows are False Positive (FP). Sensitivity represents the ratio of the correctly classified FOG windows to the number of all FOG windows in the test set. Likewise, the specificity represents the ratio for Normal walking windows, see (1) and (2).

$$Sensitivity = \frac{TP}{TP + FN} \tag{1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2}$$

C. RELATED WORK

Moore *et al.* transformed the inertial data of leg movement to the frequency domain [20]. They found that FOG events have some characteristics in the frequency domain which do not appear in normal walking. The frequency components between 3 and 8 Hz exist only in FOG episodes. They also proposed the Freeze Index (FI) that detects FOG events individually. The FI is calculated as the power spectra in the ''freeze'' band (3-8 Hz) that is divided by the power in the ''locomotor'' band (0,5-3 Hz). The FOG event was detected by a FI threshold which varied patient by patient. Moore *et al.* could obtain a sensitivity of 78% and specificity of 80% in FOG event detection with a fixed FI index for all patients. The FI index was assigned globally and was the same for all subjects in the Daphnet dataset. In another approach, the FI index was calibrated individually for each patient, and the sensitivity and specificity increased to 89% and 90%, respectively [20]. The FI feature defined and introduced by Moore *et al.* is widely used in later experiments and papers [21], [22].

Assam and Seidl used the features from wavelet transformation [23]. They used different window lengths. The algorithm proposed by Assam *et al.* could not get a sensitivity better than 75%. The window length was 8 seconds bringing a considerable delay between data acquisition and decision making [23].

Mazilu *et al.* introduced the vast majority of features from the frequency domain field [24]. Mazilu *et al.* used features from the frequency domain field [24] and applied them to individual patients getting a model for each patient. For the R10-fold cross-validation approach, the model sensitivity was as high as 99% for 4-second windows applying the AdaBoost algorithm. However, as also noted by the authors [24], there is a significant probability that this high sensitivity was achieved due to the fact that purly random selection of data having a correlation between training and test sets and the model tests the data that is already seen in training. The authors evaluated the model performance also by applying to Leave One Subject Out (LOSO) i.e., they kept one patient isolated from the training stage, using that patient only to validate model performance. This approach gained a sensitivity of 66.25% for 4-second windows.

Ravi *et al.* used a Convolutional Neural Network (CNN) with one convolutional layer followed by a fully connected layer [25]. The raw inertial signal is transformed into the spectrogram as a function of time and frequency. The gyroscope and accelerometer signals are exploited in that research. The convolutional layer uses a set of filters on spectrograms of the input data. Then the weighted convolved signals are added to each other and this process is applied at each time t repeatedly. This layer extracts the features from the spectrogram. Finally, it is followed by a classification layer which consists of a fully connected layer and a Softmax layer at the end. The validation is R10Fold with 4-second windows without overlap. The architecture achieved a mean sensitivity of 66.3% and 97.7% for specificity. The R10Fold method showed a big variation in model performance for patients. The specificity recognizes the Non-FOG pattern which obtained a quite high rate.

Alsheikh *et al.* used a Deep Neural Network (DNN) [17]. Like Ravi *et al.*, they applied pre-processing to the data and transformed it into a spectrogram. A deep learning generative layer takes the input data and computes the intrinsic features. The proposed model applies training in two main steps. The first step is a pre-training procedure that generates features on each layer by unlabeled data using Deep Belief Networks (DBN). In the second step, the pre-trained weights are fine-tuned by supervised learning from labeled data. The results are 91.5% for both specificity and sensitivity. The model is validated by R10Fold cross-validation with a window length of 4 seconds without any overlap.

San-Sequndo *et al.* used raw inertial data from the Daphnet dataset transformed to feature sets proposed in other papers, e.g., Mazilu *et al.* [16]. The authors used a CNN, random forest and hidden Markov models for training and testing. They assessed the methods with LOSO and R10Fold cross-validation. The windows are 4 seconds long with a 75% overlap. The sensitivity and specificity are around 75% and 95%, respectively, when 7 consecutive windows are employed. One of the drawbacks in using several adjacent windows for a CNN model is a high time delay in classification because of large input data. Furthermore, it also needs more memory to process the input data.

San-Sequndo *et al.* took advantage of several feature sets adopted from different research fields such as Mazilu *et al.*'s features set, speech quality features using harmonicity and predictability in time and frequency domains and spectral flex, human activity recognition, and Mel frequency cepstral coefficients features in the FOG detection algorithms [16]. However, the decision trees algorithm and LOSO evaluation results were not better than 55% in sensitivity. Therefore, the model failed to recognize about half of the FOG events which is a poor result. Similarly, the specificity is 95% for non-FOG but it is an expected result because around 90% of the dataset is non-FOG. San-Sequndo *et al.* revealed that each PD subject has a personal gait style and there is a drastic difference between the model performances on each patient. San-Segundo *et al.* also used R10Fold cross-validation which

was remarkably patient dependent and did not have generalization over unseen subjects [16]. San-Segundo *et al.* used the feature set from Mazilu *et al.* and applied the random forest algorithm with R10Fold cross-validation. The corresponding achievements with zero overlap are 91% and 91.5% for sensitivity and specificity, respectively.

Masiala *et al.* used LSTM block cell and Daphnet dataset [15]. The authors applied time and frequency domain features with the windows of 4 seconds and overlap of 0.5 second into LSTM block cells. In spite of the high overlap in consecutive windows, the training and test set were isolated from each other. Their subject-dependent model considered all three sensors' data, accelerometer, gyroscope, and magnetometer, available in the dataset and classified with 87% sensitivity and 96% specificity. When only the shank sensor was considered, both sensitivity and specificity dropped to 82% and 94%, respectively. The sensitivity is higher in the LOSO approach, 84%, but specificity is lower, 86%. The sensitivity and specificity achieved by Masiala *et al.* in the subject-dependent approach are considered as the baseline and benchmark for this paper. The papers in the related work did not provide any information about dataset preparation.

To the best of our knowledge, there is no study on the effect of padding when using machine learning models in freezing of gait detection field. The literatures in the related works did not mention specifically how they prepare the regions and the windows in training and test sets. Mostly, they only address the general windowing strategy with given window length and overlap. Instead, the papers in freezing of gait field focus on the sensitivity and specificity achieved from their algorithms and extracted features from the dataset. But we are convinced that data preparation i.e. ensuring that windows contain only one type of data during the training phase greatly helps to improve the training. So we analyzed the literature for approaches to split windows of training data without losing data. We were focusing on data sets with a similar structure as the Daphnet data set used by us. Such data sets can be found for functional protein prediction and written texts. In both cases, the data represent sequences, i.e., long vectors like the sequences of acceleration data from IMU sensors in the Daphnet data set.

The sequence of data in freezing of gait, language and Amino acids show variable lengths in their nature and cause challenges in feeding machine learning algorithms. The analogy between these fields motivates the applications of the Natural Language Processing (NLP) techniques to FOG detection by changing the length of the sequence with minimum impact on the neural network. When applying machine learning to NLP and protein functional prediction the issue of variable sequence length is solved by adding pad values to the sequences to ensure sequences with the same length [26]–[31]. Zero padding is frequently used in the NLP and protein prediction and the padding operation concatenates a vector of zero values to the measured sequences [30]–[34]. Padding with zero values is also adopted in popular machine learning frameworks such as Tensorflow, Pytorch, and the Matlab machine learning toolbox [35]–[37].

TABLE 1. List of literature using daphnet dataset for FOG detection.

	Sensitivity (%)	Specificity (%)	Validation	Window length (s)	Overlap (%)
Moore *et al.* [20]	78	80	General	6	-
	89	90	Individual	6	-
Assam *et al.* [23]	75	75	Individual	8	-
Mazilu *et al.* [24]	62.05	95.15	General	1	-
	66.25	95.38	General	4	-
	98.35	99.72	Individual	1	90
	99.69	99.96	Individual	4	90
Ravi *et al.* [25]	66.3	97.7	Individual	4	0
Alsheikh *et al.* [17]	91.5	91.5	Individual	4	50
San-Segundo *et al.* [16]	95	75	General	28	75
	55	95	General	4	75
	91	91.5	Individual	4	0
Masiala *et al.* [15]	84	86	General	4	12.5
	82	94	Individual	4	12.5

This paper proposes a novel method to use padding to prepare training data for the LSTM model. The trained model was fed with raw data with a length of 1 second with zero overlap and without any additional pre-processing or feature extraction task and detects FOG windows successfully in the subject-dependent approach. Table 1 shows the results of the methods mentioned above.

III. DATASET PREPARATION

The distribution, size, and quantity of labels are different throughout any standard dataset. In the Daphnet dataset, FOG episodes do not have the same length and distribution for all patients. The total amount of FOG data is only 9.7% of the dataset, which is much lower than Normal walking data, 90.3%. Therefore the Daphnet dataset shows a massive bias between the two labels. The Daphnet dataset originally contains three labels, 0, 1, and 2, corresponding to each data recording. Label "0" relates the data before and after the experiment, such as installing the sensors on patient's body. Data with the label "0" was not considered in this work because they are not part of the experiments. Data corresponding with label "1" was part of the experiment and covered activities such as standing, walking and turning. The data with label "1" was known as Normal walking and label "2" represents the data showing freezing and was labeled as FOG in this work.

In time-series classification problems, in which the data mainly exist as one long vector containing one or more channels, it is not possible to use the whole data for training in one step and a window strategy should be applied to split the long vector into subsets [14]. This technique segments the time-series data, and generates equal- and fixed-size windows as subsets [38].

A. TRAINING SET

Each window in training should only contain a unique type of labeled data, so each window in the training set represents only FOG or Normal walking. However, in most cases, regions of FOG or Normal data do not fit entirely into consecutive windows of a given length. So windows at the end or beginning of a FOG or Normal walking region can contain data from the respective neighbor region. Although windows with data of both labels can be truncated, data representing freezing of gait faces a shortage and removing such windows decreases the amount of data which is already scarce. In contrast, this paper focuses on keeping all data from the original dataset instead of removing any data. Therefore, a solution is required to prepare data in the training set to fulfill the following two conditions: windows contain only one type data and no data is discarded. Meeting these conditions are only tricky for windows in which changes from normal walking to FOG or vice versa happen, i.e., for the most interesting windows.

Thus a methodology is required to prepare the data in such a way that each window is labeled unique. It is worthy to note that the data preparation should have the least impact on the neural network.

To solve the issue of regions of different lengths, padding with zero values is applied to the Daphnet dataset where appropriate. Thus the amount of data in each region is increased, so that afterwards, all windows are completely filled with data of one label only and overlapping with the neighbor regions is avoided.

As padding data is additional data and not part of original dataset. It is important that the padding data have the least possible impact on the calculated loss from loss function and that the model mainly focuses on actual acceleration data from the dataset. After padding, the data in the training set are fed directly to the model without any further transformation. The loss function in the LSTM model calculates the error between the predicted label and the actual window label without considering the origin of the input values. In multilayer LSTM networks, the weighted sum of zero inputs from padding data becomes zero in the corresponding synapses while the output of the activation function is calculated from the bias value and the hidden state of the previous time step of the neuron. Afterward, the layers generate new hidden and cell states from corresponding padding data. Thus it is essential to keep the zero paddings at the lowest possible amount to ensure the calculated loss values mainly stem from the actual dataset and not from padding data.

1) SEQUENCE SPLITTING

Sequence splitting in MATLAB pads and splits regions to data intervals of equal lengths which allows feeding the model with a fixed window size for the whole training phase, whereas each mini-batch has a different length. Sequence splitting fits the windows in a mini-batch by padding the regions in which data is missing, see Fig. 2.

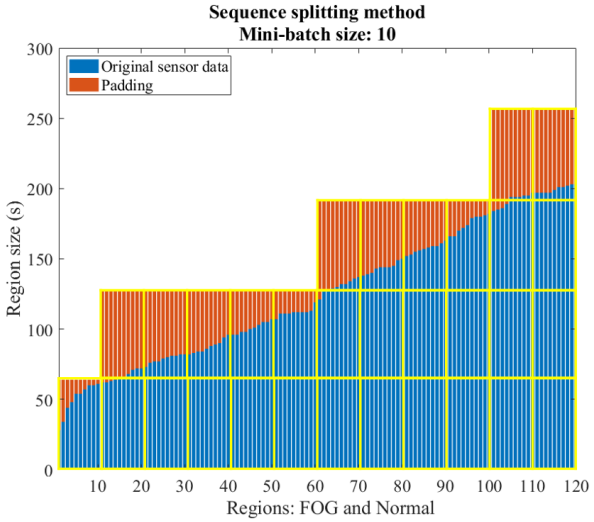

FIGURE 2. Sequence splitting method with a windows length of 64 and mini-batch size of 10. FOG and Normal walking regions, blue color, are sorted ascending by region length in this figure. Padding data, orange color, are added at the end of each region.

The mini-batch size defines a number of regions using the sequence splitting method, for example, ten regions in a mini-batch in Fig. 2. This principle has a drawback for mini-batches having a high variation in the regions' lengths. To reduce the variation of regions' length in a mini-batch, the regions are sorted in ascending length, as shown in Fig. 2, and therefore, regions inside a mini-batch show more similar lengths. Some regions get much more padding to reach the mini-batch size. For example, the regions between 100 to 103 in Fig. 2. need only a bit of padding to fill the windows. However, the window length should be the same for all windows in the mini-batch, so the regions 100 to 103 have to add and fill the fourth window with padding values. This method generates a high amount of padding to the regions.

The length of regions in the Daphnet dataset increases exponentially and bigger mini-batch sizes showing a greater difference between the shortest and longest regions in a mini-batch leading to a higher padding amount. For example, the sequence splitting method adds much less padding for regions 335 to the 385 for a mini-batch of size 25, Fig. 3-a, compared to a mini-batch size of 50, Fig. 3-b.

Although this method can prepare the dataset for LSTM training, the enormous amount of padding in larger batch sizes and the varying number of splits in each mini-batch are problematic to the model training.

2) PROPOSED PADDING

Fig. 4 shows the rolling window over the dataset, the point where FOG and Normal regions meet each other cause challenges. The rolling window at these places might contain data from both FOG and Normal regions. However, this contrasts with the requirement of having only one type of data in a window for the training set. We propose to shift the regions and to fill the rolling windows with a

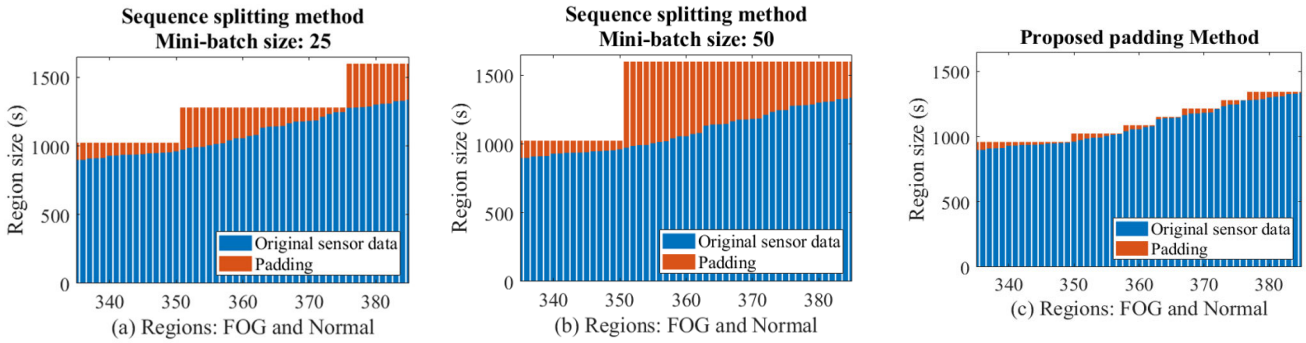

FIGURE 3. Padding produced by the Sequence splitting method provided in MATLAB and the here proposed padding method. The mini-batch sizes are 25 and 50 for (a) and (b), respectively. The mini-batch size is a parameter in sequence splitting method and the bigger batch size generates a higher padding amount. The padding amount in the proposed method is independent of the mini-batch (c). The proposed padding method fills the free space in the last window of a region. When compared to sequence splitting method, the proposed padding algorithm added far less padding data (c).

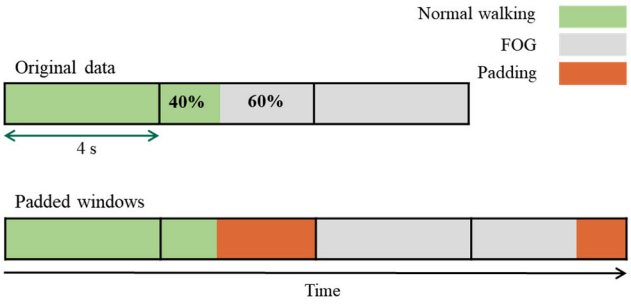

FIGURE 4. Normal walking and FOG regions are split in windows of 4 seconds. The rolling window covers parts of FOG and Normal data at the boundary of these regions. The padding adds some data to the end of regions to have data from one label only.

TABLE 2. The comparison between MATALB sequence splitting and the proposed padding method.

	Sequence splitting (MATLAB)	Proposed padding method
Regions fit the windows	Yes	Yes
Padding amount regardless of mini-batch size	No	Yes
Optional padding location	No	Yes

padding value in order to have a single type of data, FOG or Normal in each window, see Fig. 4. So each window gets the same label for its inner data. To the best of our knowledge, so far, no paper reported on the preparation of individual regions before training. So the method here cannot be compared to others and may be considered innovative, and at least it provides missing information on data preparation.

The proposed padding method processes each region individually and calculates the required padding to fill the rolling window. Unlike the sequence splitting method, regions have no impact on the amount of the padding of other regions, see Fig. 3-c.

The MATLAB method is not supporting flexible padding positions and simply adds padding data at the beginning or the end of all regions. The proposed padding method can put the padding data at the beginning or at the end of FOG or Normal regions. For example, the developer can pad at the beginning of a Normal region and at the end of a FOG region. The mini-batch size is not a coefficient of number of regions, but it is based on the number of windows, and the same amount of windows are processed at each epoch. The padding amount is constant and invariant of the batch size, see Table 2.

B. TEST SET

In a real-life situation, it is not apparent when FOG starts or is going to end. The testing environment should represent real-life scenarios in which the model assesses the data in the same form as retrieved from the sensor node. So, padding or any other transformation is not applied to the test set. In other words, raw data is fed to the model directly.

A window strategy is applied to the whole test set generating fixed-size windows and provides a unique label to a window, and unlike the training set, it does not consider each region separately. A window gets the FOG or Normal label if the whole data inside the window is FOG or Normal, respectively. However, where the FOG and Normal region meet each other and both types of data exist in the windows, see Fig. 5 label assignment is based on the threshold for the proportion of data in the window. For example, the threshold of 50% for FOG data means that the window gets FOG label if a window has over 50% of the FOG data otherwise, the window receives a Normal label.

IV. LSTM FOR FOG DETECTION

RNN models use sequential information from time-series or temporal data. RNN provides robust prediction performance as well as a remarkable ability to learn long-term dependencies between temporal data. Therefore the model delivers

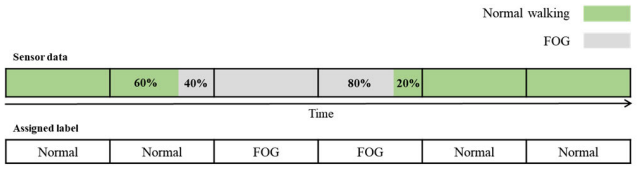

FIGURE 5. The labels of the windows are assigned based on the percentage of FOG data in the test set. In this figure, 50% of FOG data is used to assign the FOG label to the window.

information that does not only depend on currently received data but also on those with previous time stamps. Recently, LSTM cells as a robust variant of RNN models achieved good results in many domains with times series data such as machine translation and speech recognition [39]–[41]. An LSTM network contains recurrent blocks that act as memory and adapt their inputs based on the gate's function [42], see Fig. 6.

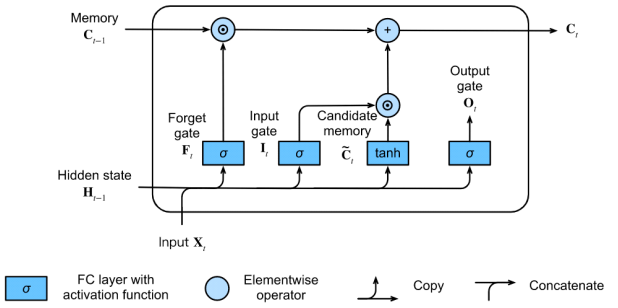

FIGURE 6. Internal components of an LSTM block [43]. LSTM components apply nonlinear transformation by the sigmoid and hyperbolic tangent activation functions across the components as well as dot-product and summation of the matrix as mathematical operations. The LSTM block gets intuition of the hidden state and the cell state, h_{t-1} and c_{t-1}, respectively, from the previous time step and generates two outputs, h_t and c_t corresponding to the states for the current input data x_t.

FIGURE 7. Used architecture with multi-layer LSTM.

The input of the LSTM model is the data from the accelerometer in the time-series format and the model memorizes data from previous time steps. The LSTM capabilities such as memory and time-series feeding style make it well suited for detecting FOG. The FOG detection model in this work is many-to-one which uses sequential data from the accelerometer sensor as temporal values belonging to a window with a unique class, whereby only one output is delivered at the last element of the window. In other words, the model predicts only one label per window. Fig. 7 illustrates the architecture of a multi-layer LSTM network that processes raw accelerometer data for classification purposes. The stacked LSTM network learns richer data representations as additional layers produce higher abstraction levels of input data over time [15]. The class of a window is predicted by using a fully connected layer, followed by a Softmax layer and finally, a classification output layer.

The training procedure yields the model to overfit during the training phase if no regularization is used. Overfitting leads the model to achieve a low error rate within the training set, whereas it leads to a high error at the test set that was isolated from the training set. The regularization techniques preserve training accuracy in the training set while the model obtains low generalization error over unseen data [44]. Several regularization techniques such as the L2-norm regularization, early stopping, data augmentation, and dropout are employed.

V. MODEL IMPLEMENTATION

This work is based on the stateless definition of LSTM, i.e., the model passes states to the next time step in the same window and it does not transmit any states to the further windows. In other words, stateless models get the intuition of dependency inside the window and they do not have any information from other windows. As dependency between the windows does not exist, the order of windows was shuffled at each epoch. Our model consists of three layers of LSTM blocks stacked on each other. The input data are fed to the LSTM model with 110 hidden units in the first layer which is followed by two additional layers with 90 and 70 hidden units. The whole model consists of 167600 learnable parameters. The type of the model is many-to-one, so the input data is a three-channel raw data from the acceleration sensor mounted on the shank and the output is a label defining the class of the input data. The LSTM layers are followed by a fully connected layer, and then the Softmax layer assigns the likelihoods of the outputs to one of the two classes. Finally, the window is classified as *FOG* or *Normal* class.

The input weights and recurrent weights are initialized with the Glorot initializer and orthogonal matrix initializer, respectively. The hyperbolic tangent function is applied to the cell candidate component of the LSTM block. The sigmoid activation function computes the nonlinear transformation for the input, forget, and output gates. The Adam optimizer updates the model weights after each mini-batch of training data. The initial learning rate was assigned 0.005 and the software updates the learning rate every 3 epochs. The decay rate factor in the gradient moving average is 0.8 for the Adam solver. The squared gradient moving averages are defined as 0.99. During the experiments, these parameters led to the best model performance.

There are 23629 windows after padding in the training set. Each mini-batch considers 100 windows and in total, 236 mini-batches were generated for each epoch. MATLAB covered 23600 windows in one epoch, and the 29 remaining windows were ignored because they do not fill a mini-batch entirely.

The L2 regularization method was applied to the training algorithm. The weight decay factor is specified as 0.0008. Early stopping terminates the training process to avoid

overfitting when the validation set is 15% worse than the training set. This strategy stops the training process when the condition becomes true for five consecutive epochs. The maximum number of epochs is 60. The augmentation method increases the FOG regions virtually to reduce the imbalance between the FOG and Normal walking data. FOG regions were copied four times after applying the proposed padding method and increase the FOG data from 9,7% of the original dataset into 33% after the augmentation method.

All the development tasks from data cleansing, data representations, training, and testing processes were done in MATLAB 2019a [45]. The proposed padding method was also developed, tested and employed in the MATLAB software. The Statistics and Machine Learning Toolbox of MATLAB 2019a was used for the training and testing algorithm. The window length is 1 second containing 64 samples for each channel, in the training and in the test set. Consecutive windows have zero overlap. The proposed padding algorithm was applied to the training data. The test set did not face any padding or any other pre-processing and the data in test set was used in the raw format. The label of windows in the test set data are based on the proportion of the input data. The test set windows with 100% of FOG or Normal data get the label corresponding to the internal data. The algorithm assigns a FOG or a Normal label when the windows have at least 90% of the corresponding data, windows with a lower proportion than 90% are not considered.

The model is trained subject-dependent, i.e., each patient has an individual model. The proportion of data taken from each patient in the training and test sets are not the same as in the related literature. Some researchers took 90% of the patient's data for training and the rest of data of the same patient, i.e. 10%, for testing [21], [25], [17], [16]. Asam *et al.* considered 75% to 25% in training and test sets, respectively [23]. Masiala *et al.* trained the model with lower data of the patient, 70%, and tested their model with the remaining 30% of the same patient's data [15]. All these papers selected the training and test sets from the dataset randomly. In this work, 50% of both FOG and Normal windows are kept out of the training phase and used for testing. The training set uses the other 50% of the data of the same patient and it also considers all other patients except for patients 4 and 10 who did not experience any FOG. The subsets of data for each patient in training and test sets are selected randomly.

Afterward, the padding method - proposed padding method - is applied to the data considered for the training set. Although the equal proportion of data in training and test sets is more challenging with respect to learning, it gives more confidence for sensitivity and specificity because the size of the test set for each patient is the largest among the studies used for comparison in this paper. Eliminating patients 4 and 10 reduces the imbalance between FOG and Normal walking labels. These two patients represent 25% of the dataset with no FOG data and only contain Normal walking data. Patients 4 and 10 were only used for determining the normal walking detection rate.

TABLE 3. Padding amount in MATLAB and the method introduced in this paper for the whole dataset using window length of 64.

	Mini-batch size	Padding amount	Total number of data in the dataset	Proportion of padding to dataset (%)
Sequence splitting (MATLAB)	10	206113	1140835	18
	20	389153		34
	50	1086753		95
	100	3006753		263
Our padding method	Any	16097		1

TABLE 4. The trained model size is in kilobyte (KB).

Learned parameters	Internal calculated parameters	Total parameters	Model size (KB)
167600	3193	170793	684

VI. RESULT AND DISCUSSION

A. PADDING

The padding amount significantly depends on the batch size in the sequence splitting method. This method adds an extensive amount of padding data to the dataset, even for a relatively small batch size. For example, it generates the same amount of padding data as the actual data from the dataset when the batch size is 50, see Table 3. However, the padding method proposed in this paper only adds 1% of the dataset as padding data at any batch size, i.e., the padding amount is invariant of the batch size. For example, if the batch size is 100, Table 3, the padding amount is two orders of magnitude smaller. So, our padding method adds much lower padding and the model calculates the loss function mainly from accelerometer data.

B. MODEL SIZE

The learned parameters of the LSTM model, as well as the internal calculation values from each layer and time step, allocate a certain amount of storage. The data type of the model parameters in MATLAB 2019a is a single-precision variable that stores both the raw and weight data as 4-byte (32-bit) floating-point values. Table 4 presents the learned parameters, the number of calculated values inside the LSTM block, and the required memory size at the application stage.

C. MODEL PERFORMANCE

The paper shows two different methodologies of padding, the sequence splitting method and the proposed padding method, which were applied to the training set resulting in two different training sets. Each of the two training sets are used in training for separate models and the trained models are evaluated using the same test set which is not manipulated

TABLE 5. Performance results using the sequence splitting method in MATLAB to prepare the data for training.

Mini-batch size	Sensitivity (%)	Specificity (%)	AUC (%)
10, 20, 50, 100	0	100	56

TABLE 6. LSTM model performance for FOG detection. The window length is 1 second without overlap.

Patient number	Sensitivity (%)	Specificity (%)	AUC (%)
1	92.5	95.9	98
2	93.6	94.3	98.4
3	92.4	93.7	97.1
4	-	100	-
5	94.1	93.3	97.3
6	88.9	97.7	98
7	90.1	94.7	97.4
8	94.9	93	97
9	94.1	93.6	97.8
10	-	100	-
Mean	**92.57**	**95.62**	**97.62**

by padding or any other data pre-processing method. The data in the test set shows the recordings from sensor data in the dataset and they are only split to fixed-size windows in order to fit the specification of our multi-stack LSTM model.

The evaluation metrics such as sensitivity, specificity and AUC compare the model performance in the two different padding methods.

1) SEQUENCE SPLITTING METHOD

The sequence splitting method in MATLAB was used to prepare the training set. Different models with different mini-batch sizes, 10, 20, 50, and 100 were trained. Table 5 reveals the average performance results of all the trained models through all patients using the sequence splitting method. The models could not even detect one single FOG event correctly, whereas all the data in the test set were classified as Normal walking achieving 100% specificity. AUC gained a low metric value of 55%.

2) PROPOSED PADDING METHOD

The results in Table 6 show the achieved sensitivity, specificity and AUC for each patient in the Daphnet data set as well as the mean sensitivity of 92.57% and specificity of 95.62% and the mean AUC of 97.62%. Our model score gained 100% specificity in patients 4 and 10 who have only Walking data and did not have any false classification.

Since we used Daphnet as a benchmark dataset for FOG detection, we can compare the performance of our model with the results reported in the literature. As the related literatures used different machine learning models and features pools and they did not apply padding methodologies, the comparison of this work to others is done using the model performance parameters i.e., sensitivity, specificity, and window length.

In most of the works, the window length is 4 seconds. Mazilu *et al*. used 1-second windows and achieved 98% and 99% for sensitivity and specificity, respectively [24]. However, Mazilu *et al*. mentioned that the high values might be because of a bias in their experimental setup, which is high from a purely random 10-fold cross-validation evaluation procedure [24]. Tremendous overlapping between windows, around 90%, leads to data leakage from the test set to the training set. Without any overlap, the sensitivity and specificity drop to about 91% and 91.5%, respectively [16]. Bachlin *et al*. employed windows of 4 seconds, which are four times longer than those used in our approach and achieved lower sensitivity, 88.6%, and 92.4% for specificity [19].

Masiala *et al.* also used LSTM and the same dataset [15]. Although Masiala *et al*. did not mention their neural network architecture and training configuration, they tested their model in a subject-dependent way as we did. The sensitivity, 92.57% that we achieved with our approach is more than 10% better than theirs, 82%, which is a significant improvement. Our model's specificity, 95.62%, represent a 1% increase compared to their specificity of 94%. This is even more impressive as we used 1-second windows, whereas they used 4-second windows.

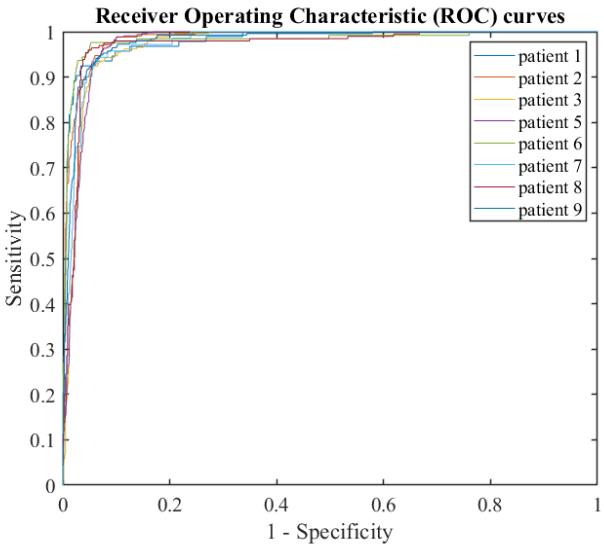

FIGURE 8. ROC curves for each patient. Patients 4 and 10 are excluded as they only have normal walking data. Windows size is 1 second without overlap.

Receiver Operating Characteristic (ROC) curves depicted in Fig. 8 show that our model achieves very similar results for the data and the average AUC is 97.62. Our average AUC is 1% higher than the one achieved by the model of Masiala *et al.* The AUC achieved by San-Segundo *et al.* is similar to ours but they used 4-second windows with 75% overlap [15], whereas we used 1-second windows without any overlap.

VII. CONCLUSION AND OUTLOOK

In this paper, we proposed a new method to apply padding that calculates the amount of padding for each region individually, regardless of the mini-batch size, leading to a constant padding amount for all batch sizes. The total amount of padding values is around 1% of the entire dataset. The so prepared data windows were later used to train and test our innovative ML approach for FOG detection, which was our main goal. We used the Daphnet dataset, which was acquired by wearable inertial sensors. Our LSTM model uses the three channels of the acceleration sensor mounted on the shank of the patients. The LSTM model received a fixed size length of data. The windows in the training set contain only one type of data, FOG or Normal and padding is used to ensure equal window size. Our patient-dependent approach assigns the training set using 50% of data from the target patient in addition to all other patients except for patients 4 and 10. The other 50% of the data of the target patient was used for testing. The test set is split into windows as well, but the regions were not padded in order to represent the actual signal in real life. Each window in the test set was labeled based on the proportion of the contained data. Data from patients 4 and 10 were only used when they were target patients.

Our approach using the proposed padding method outperformed all other approaches that were discussed in this paper using 1-second windows with respect to sensitivity, specificity, and AUC, achieving 92.57%, 95.62%, and 97.62%, respectively. Our LSTM model architecture used the feature-less data with zero pre-processing computation cost. In contrast to these convincing results, models trained with subsets produced by the sequence splitting method of MATLAB were not successful in FOG detection.

We need to admit that we cannot clearly state which of the applied means, i.e., the new type of padding or individualized training leads to the improved performance. We will do a thorough analysis of the impact of the different approaches in our future work.

In addition, the model assessment requires more extensive datasets to cover more patients and more complex gait activities. In order to provide these, we are currently cooperating with clinicians to gather data using inertial sensors from Parkinson's patients. In addition, we will apply alternative deep learning architectures such as gated recurrent units and LSTM combined with feature engineering. Once the results are retrieved, we want to compare them with our findings, reported here.

REFERENCES

[1] S. Sveinbjornsdottir, "The clinical symptoms of Parkinson's disease," *J. Neurochem.*, vol. 139, pp. 318–324, Oct. 2016, doi: 10.1111/jnc.13691.
[2] L. C. Triarhou, "Dopamine and Parkinson's disease," in *Madame Curie Bioscience Database*. Austin TX, USA: Landes Bioscience, 2013.
[3] Parkinson's Foundation. (2019). *Movement Symptoms*. Accessed: Nov. 10, 2020. [Online]. Available: https://www.parkinson.org/Understanding-Parkinsons/Movement-Symptoms
[4] S. Lahmiri, "Gait nonlinear patterns related to Parkinson's disease and age," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 7, pp. 2545–2551, Jul. 2019, doi: 10.1109/TIM.2018.2866316.
[5] Parkinson's Foundation. (2019). *Non-Movement Symptoms*. Accessed: Jun. 1, 2019. [Online]. Available: https://www.parkinson.org/Understanding-Parkinsons/Symptoms/Non-Movement-Symptoms/Hallucinations-Delusions
[6] N. Giladi, M. P. McDermott, S. Fahn, S. Przedborski, J. Jankovic, M. Stern, and C. Tanner, "Freezing of gait in PD: Prospective assessment in the DATATOP cohort," *Neurology*, vol. 56, no. 12, pp. 1712–1721, Jun. 2001, doi: 10.1212/WNL.56.12.1712.
[7] J. D. Schaafsma, Y. Balash, T. Gurevich, A. L. Bartels, J. M. Hausdorff, and N. Giladi, "Characterization of freezing of gait subtypes and the response of each to levodopa in Parkinson's disease," *Eur. J. Neurol.*, vol. 10, no. 4, pp. 391–398, Jul. 2003, doi: 10.1046/j.1468-1331.2003.00611.x.
[8] P. Arias and J. Cudeiro, "Effect of rhythmic auditory stimulation on gait in Parkinsonian patients with and without freezing of gait," *PLoS ONE*, vol. 5, no. 3, p. e9675, Mar. 2010, doi: 10.1371/journal.pone.0009675.
[9] W. R. Young, L. Shreve, E. J. Quinn, C. Craig, and H. Bronte-Stewart, "Auditory cueing in Parkinson's patients with freezing of gait. What matters most: Action-relevance or cue-continuity?" *Neuropsychologia*, vol. 87, pp. 54–62, Jul. 2016, doi: 10.1016/j.neuropsychologia.2016.04.034.
[10] W. Maetzler, J. Domingos, K. Srulijes, J. J. Ferreira, and B. R. Bloem, "Quantitative wearable sensors for objective assessment of Parkinson's disease," *Movement Disorders*, vol. 28, no. 12, pp. 1628–1637, Oct. 2013, doi: 10.1002/mds.25628.
[11] S. Qiu, Z. Wang, H. Zhao, L. Liu, and Y. Jiang, "Using body-worn sensors for preliminary rehabilitation assessment in stroke victims with gait impairment," *IEEE Access*, vol. 6, pp. 31249–31258, 2018, doi: 10.1109/ACCESS.2018.2816816.
[12] S. Qiu, Z. Wang, H. Zhao, and H. Hu, "Using distributed wearable sensors to measure and evaluate human lower limb motions," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 4, pp. 939–950, Apr. 2016, doi: 10.1109/TIM.2015.2504078.
[13] M. Seiffert, F. Holstein, R. Schlosser, and J. Schiller, "Next generation cooperative wearables: Generalized activity assessment computed fully distributed within a wireless body area network," *IEEE Access*, vol. 5, pp. 16793–16807, 2017, doi: 10.1109/ACCESS.2017.2749005.
[14] J. Camps, A. Samà, M. Martín, D. Rodríguez-Martín, C. Pérez-López, S. Alcaine, B. Mestre, A. Prats, M. C. Crespo, J. Cabestany, and À. Bayés, "Deep learning for detecting freezing of gait episodes in Parkinson's disease based on accelerometers," in *Proc. Int. Work-Conf. Artif. Neural Netw.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 10306, 2017, pp. 344–355, doi: 10.1007/978-3-319-59147-6_30.
[15] S. Masiala, W. Huijbers, and M. Atzmueller, "Feature-set-engineering for detecting freezing of gait in Parkinson's disease using deep recurrent neural networks," Sep. 2019, *arXiv:1909.03428*. [Online]. Available: http://arxiv.org/abs/1909.03428
[16] R. San-Segundo, H. Navarro-Hellín, R. Torres-Sánchez, J. Hodgins, and F. de la Torre, "Increasing robustness in the detection of freezing of gait in Parkinson's disease," *Electronics*, vol. 8, no. 2, pp. 1–15, 2019, doi: 10.3390/electronics8020119.
[17] M. A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H. P. Tan, "Deep activity recognition models with triaxial accelerometers," in *Proc. Workshops 13th AAAI Conf. Artif. Intell.*, Nov. 2015.
[18] D. Roggen, M. Plotnik, and M. Plotnik. (2013). *Daphnet Freezing of Gait Data Set, UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait
[19] M. Bächlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Troster, "Wearable assistant for Parkinson's disease patients with the freezing of gait symptom," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 436–446, Mar. 2010, doi: 10.1109/TITB.2009.2036165.
[20] S. T. Moore, H. G. MacDougall, and W. G. Ondo, "Ambulatory monitoring of freezing of gait in Parkinson's disease," *J. Neurosci. Methods*, vol. 167, no. 2, pp. 340–348, Jan. 2008, doi: 10.1016/j.jneumeth.2007.08.023.
[21] S. Mazilu, A. Calatroni, E. Gazit, D. Roggen, J. M. Hausdorff, and G. Tröster, "Feature learning for detection and prediction of freezing of gait in Parkinson's disease," in *Proc. Int. Workshop Mach. Learn. Data Mining Pattern Recognit.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 7988, 2013, pp. 144–158, doi: 10.1007/978-3-642-39712-7_11.

[22] Y. Xia, Q. Gao, and Q. Ye, "Classification of gait rhythm signals between patients with neuro-degenerative diseases and normal subjects: Experiments with statistical features and different classification models," *Biomed. Signal Process. Control*, vol. 18, pp. 254–262, Apr. 2015, doi: 10.1016/j.bspc.2015.02.002.

[23] R. Assam and T. Seidl, "Prediction of freezing of gait from Parkinson's disease movement time series using conditional random fields," in *Proc. 3rd ACM SIGSPATIAL Int. Workshop Use GIS Public Health (HealthGIS)*, 2014, pp. 11–20, doi: 10.1145/2676629.2676630.

[24] S. Mazilu, M. Hardegger, Z. Zhu, D. Roggen, G. Troester, M. Plotnik, and J. Hausdorff, "Online detection of freezing of gait with smartphones and machine learning techniques," in *Proc. 6th Int. Conf. Pervasive Comput. Technol. Healthcare*, 2012, pp. 123–130, doi: 10.4108/icst.pervasivehealth.2012.248680.

[25] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," in *Proc. IEEE 13th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2016, pp. 71–76, doi: 10.1109/BSN.2016.7516235.

[26] A. L.-D. Rio, M. Martin, A. Perera-Lluna, and R. Saidi, "Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction," *Sci. Rep.*, vol. 10, no. 1, pp. 1–14, Dec. 2020, doi: 10.1038/s41598-020-71450-8.

[27] A. T. Müller, J. A. Hiss, and G. Schneider, "Recurrent neural network model for constructive peptide design," *J. Chem. Inf. Model.*, vol. 58, no. 2, pp. 472–479, Feb. 2018, doi: 10.1021/acs.jcim.7b00414.

[28] A. L.-D. Rio, A. Nonell-Canals, D. Vidal, and A. Perera-Lluna, "Evaluation of cross-validation strategies in sequence-based binding prediction using deep learning," *J. Chem. Inf. Model.*, vol. 59, no. 4, pp. 1645–1657, Apr. 2019, doi: 10.1021/acs.jcim.8b00663.

[29] W. Li, L. Zhu, Y. Shi, K. Guo, and E. Cambria, "User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models," *Appl. Soft Comput.*, vol. 94, Sep. 2020, Art. no. 106435, doi: 10.1016/j.asoc.2020.106435.

[30] G. Piao and J. G. Breslin, "Financial aspect and sentiment predictions with deep neural networks," in *Proc. Companion Web Conf. Web Conf. (WWW)*, 2018, pp. 1973–1977, doi: 10.1145/3184558.3191829.

[31] M. Nabil, A. Atyia, and M. Aly, "CUFE at SemEval-2016 task 4: A gated recurrent model for sentiment classification," in *Proc. 10th Int. Workshop Semantic Eval. (SemEval)*, 2016, pp. 52–57, doi: 10.18653/v1/S16-1005.

[32] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Process. Lett.*, vol. 21, no. 9, pp. 1120–1124, Sep. 2014, doi: 10.1109/LSP.2014.2325781.

[33] M. Cliche, "BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs," Apr.2017, *arXiv:1704.06125*. [Online]. Available: http://arxiv.org/abs/1704.06125

[34] H. Feng and R. Lin, "Sentiment classification of food reviews," Sep. 2016, *arXiv:1609.01933*. [Online]. Available: http://arxiv.org/abs/1609.01933

[35] (2021). *TensorFlow*. Accessed: Mar. 29, 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences

[36] (2019). *PyTorch*. Accessed: Mar. 28, 2021. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.utils.rnn.pad_sequence.html

[37] Mathworks. *Long Short-Term Memory Networks*. Accessed: Feb. 1, 2021. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

[38] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Jan. 2015, pp. 3995–4001.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," Sep. 2014, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473

[40] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Sep. 2014, pp. 3104–3112.

[41] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012, doi: 10.1109/MSP.2012.2205597.

[42] C. Nicholson. (2019). *A Beginner's Guide to LSTMs and Recurrent Neural Networks*. Accessed: Jul. 10, 2021. [Online]. Available: https://wiki.pathmind.com/lstm

[43] A. Zhang and Z. C. Lipton. (2020). *Long Short-Term Memory (LSTM)*. Accessed: Nov. 27, 2020. [Online]. Available: https://d2l.ai/chapter_recurrent-modern/lstm.html

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[45] *MATLAB Deep Learning Toolbox*. Accessed: Oct. 15, 2020. [Online]. Available: https://www.mathworks.com/products/deep-learning.html

ALI HADDADI ESFAHANI received the B.E. degree in electrical engineering from Azad University, in 2013, and the M.S. degrees in micro and nano systems and automotive software engineering from Technische Universität Chemnitz, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the IHP–Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany. His current research interests include time-series classification, machine learning, and model compression for embedded applications.

ZOYA DYKA received the Diploma degree in radiophysics and electronics from Taras Shevchenko National University of Kyiv, Ukraine, in 1996, and the Ph.D. degree from the Technical University of Cottbus-Senftenberg, Germany, in 2012. Since 2000, she has been with the IHP–Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany. Since 2013, she has been leading a young researchers group in the field of tamper resistant crypto ICs. Since 2018, she has also been leading a research group in the field of resilient CPSoS. She has authored more than 40 peer-reviewed technical articles and filed five patents in the security area of which four are granted. Her research interests include design of efficient hardware accelerators for cryptographic operations, SCA countermeasures, and anti-tampering means and resilience.

STEFFEN ORTMANN received the Diploma degree in computer science and the Ph.D. degree in engineering. Since 2019, he has been the Head of the Thiem-Research GmbH, a 100% research-subsidiary of a large community hospital, and the Carl-Thiem-Klinikum Cottbus, Germany, where he is also coordinating the science activities. Before that, he gathered more than 15 years of experience in leading national and European mHealth and eHealth research projects in a non-university research institute. He has published more than 60 refereed technical articles about wearables and medical sensors, mHealth applications, and the implementation of concepts and applications in telemedicine. His research interests include medical wearables, infrastructure for tele-medical innovations, eHealth, and the digital hospital.

PETER LANGENDÖRFER received the Diploma and Ph.D. degrees in computer science. Since 2000, he has been with the IHP–Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany. In the IHP–Leibniz-Institut für Innovative Mikroelektronik, he is leading the Wireless Systems Department. From 2012 to 2020, he was leading the chair for security in pervasive systems at the Technical University of Cottbus-Senftenberg. Since 2020, he owns the chair wireless systems at the Technical University of Cottbus-Senftenberg. He has published more than 150 refereed technical articles, filed 17 patents of which ten have been granted already. He worked as a Guest Editor for many renowned journals, such as *Wireless Communications and Mobile Computing* (Wiley) and *ACM Transactions on Internet Technology*. He is highly interested in security for resource constraint devices, low power protocols, and efficient implementations of AI means and resilience.