Automatic detection of myoclonus bursts: A follow-up study

Research

University of Groningen

July 31, 2024

Author:
Joy Kwant

Primary supervisor:
Prof. Dr. Michael Biehl

Secondary supervisor:
Elina van den Brandhof

External supervisor:
Dr. Jelle Dalenberg

university of groningen
faculty of science and engineering

Abstract

Myoclonus refers to brief, shock-like involuntary muscle contractions that can vary in intensity and frequency. Electromyography (EMG) signals measure muscle activity and can record myoclonus bursts in patients with this movement disorder. Currently, clinicians manually detect myoclonus bursts, a process that requires expert analysis and is time-consuming due to the significant amount of EMG data. Machine learning offers potential solution to this problem.

This research is a follow-up to the study "Automatic detection of myoclonus bursts" [1], which made the first step in myoclonus burst detection using machine learning. The focus of this study is to potentially improve burst detection by using hybrid models that combine Convolutional Neural Networks with Long Short-Term Memory (CNN+LSTM) and Gated Recurrent Units (CNN+GRU).

Both models demonstrated the ability to identify bursts that corresponded to the annotated bursts in the EMG data of myoclonus patients Additionally, the models predicted bursts in the EMG data of healthy controls, which only contains non-bursts data. Although the models have shown improvements and potential in detecting myoclonus bursts, further refinements are needed to increase their accuracy and reliability.

Contents

List of Figures

List of Tables

1 Introduction

Myoclonus refers to "brief, shock-like, involuntary" muscle contractions that can vary in intensity and frequency [2]. These muscle contractions can happen to everyone and are considered normal if they occur occasionally, such as in the case of hiccups [3]. However, if these contractions occur frequently or disrupt daily activities, they could indicate an underlying and potentially serious condition, such as epilepsy [1]. Myoclonus can be caused by a variety of factors, including medications, neurological disorder, and genetic influences [1, 4], and it can take in different forms affecting various parts of the body [3, 1].

For classification of myoclonus, physicians commonly distinguish myoclonus by its physiological origin, which refers to where the impulse for the contraction originates [3]. The primary subtypes include cortical, subcortical, segmental, spinal, peripheral and brainstem [3, 5]. This subtype classification is crucial in diagnosing and choosing the most effective treatment strategy [2], because the treatment of the subtypes differs from each other. [4, 6].

Electromyography (EMG) signals measure the muscle activity in voltage (millivolts) over time (milliseconds) and consist of two types; surface and intermuscular EMG [3, 7]. Surface EMG places electrodes on the skin covering the muscle, while intermuscular EMG inserts needle electrodes or fine wires directly into the muscle [8]. Researchers typically use surface EMG for myoclonus burst measurement.

For burst classification, the clinician first identifies the myoclonus bursts in the data and then classifies the bursts in one of the subtypes [1]. When dealing with significant amounts of EMG data from patients, manually detecting myoclonus bursts is a highly time-consuming task, requiring expert and accurate analysis, which can lead to delays in diagnosis and treatment. Fortunately, machine learning techniques offer a potential solution to speed up diagnostics [1].

Machine learning is the study of developing algorithms and models that enable machines to learn from input data, allowing them to make decisions and predictions [9]. Machine learning is able to process large datasets, which reduces the computational cost while being able to maintain high accuracy and efficiency in tasks as pattern recognition, classifications, and feature extractions in a diverse range of fields. Several studies have implemented successful machine learning techniques in problems with similar EMG data sets [10, 11, 12, 13]. This inspired the previous study "Automatic detection of myoclonus bursts in EMG data" [1] to use machine learning for the detection of bursts in EMG data of myoclonus.

This research is a follow-up to the study "Automatic detection of myoclonus bursts in EMG data" [1] and in collaboration with the Next Move in Movement Disorders (NEMO) study of Expertise Centre Movement Disorders Groningen, University Medical Centre Groningen (UMCG) [14]. The previous study made the first step in the automatic detection of myoclonic bursts.

The study suggested that the most suited model for the bursts detection in EMG data is a recurrent neural network (RNN) with long short-term memory (LSTM), because of its ability to handle sequential data, integrate contextual information and use supervised learning with the labelled data ('non burst' and 'burst'). The proposed model resulted in a detection accuracy of 76% for testing and 50% for the entire dataset.

The goal in this follow-up project is to improve myoclonus burst detection by proposing a new model architecture. I will consider the research following question:
Can the detection algorithm be improved with respect to implementation, performance and robustness?

In this paper, I will discuss the previous study and investigate related studies to determine which model to examine for potential improvement in section 2. In section 3, I will discuss the methodology. The results will be presented in section 4 and analysed in the section 5. Finally, the paper will conclude with the findings and suggestions for future work in sections 6 and 7, respectively.

2 State of the art

'Automatic detection of myoclonus bursts in EMG data' [1] made the first step in detecting myoclonus bursts in EMG data using a deep learning solution. This section will begin by summarizing their process and findings.

2.1 Previous study

The research used EMG data from patients of UMCG with the movement disorder myoclonus, obtained while performing specific tasks: resting (1), postural (17), and dynamic (22, 23), which will be explained in section 3.1. In this data set, the myoclonus bursts were labelled manually.

The author suggested that EMG data from Task 17 would be most effective for creating an optimal detection model, because patients experienced the most bursts and had the highest number of labelled bursts.

Due to much noise in the data, a denoising Butterworth filter with a bandwidth of 10,450 and an order of 6 was applied for preprocessing. For the detection strategy, they decided to use the window technique to classify 256-windows as 'burst' if bursts were present, or 'non burst' otherwise. They used this dataset for both burst and non-bursts samples and applied data augmentation to create additional bursts samples. The bursts and non-burst samples were distributed among the training, validation and test set, ensuring that each set contained samples from each patient.

Analysing the EMG data, they detected that models capable of processing this sequential data and incorporate the contextual information would be best suited for the detection.

A recurrent neural network (RNN) model meets these requirements but can struggle with long term dependencies. The Long term-short memory (LSTM) extension addresses this problem. Therefore, the research proposed a RNN model with LSTM. By experimenting with different model architectures and hyperparameter settings, they retrieved the optimal combination that achieved the highest accuracy. This looked as follows:

The model was trained with 250 epochs and batches of size 64. The input was an array of size 256x1. The model architecture was as follows: A LSTM layer with 300 hidden units, a dropout layer with 0.25 drop rate, a dense layer with 128 units and ReLU and a dense output layer with 2 units and softmax activation. The Adam optimizer was used with a rate of 1-e5. See figure 1.

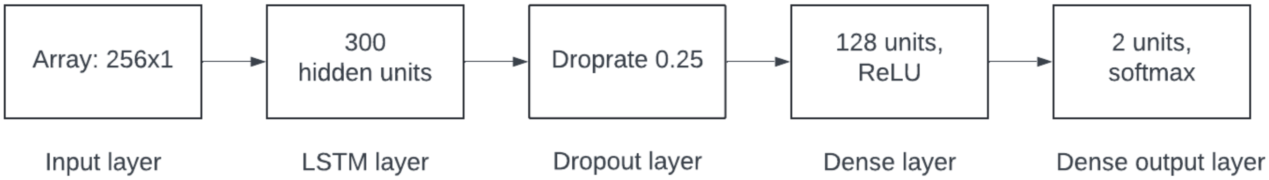


Figure 1: The architecture of the best model of [1].

The research resulted in an accuracy of 76% with the testing set and 55% with the entire dataset, which included all four tasks (also explained in section 3.1). For future work, the author suggested adding more complexity to the model, data cleaning and modifying the training, validation, and test data set composition to enhance the performance on new, unseen data (model generalization).

2.2 Related work

As mentioned in section 1, several studies successfully applied machine learning to similar data sets. To further this research, this section will also review other studies that applied machine learning, particularly RNNs, on similar data. By examining these approaches, I aim to discover potential improvements and innovations for this follow-up research.

[15] evaluated LSTM, Gated Recurrent Unit (GRU), and Bidirectional RNNs for classifying EMG signals related to five movements of the right upper limb. They pre-processed their EMG signals with a Butterworth filter and applied a window technique with 250 milliseconds (ms) windows and 190 ms overlap. The model architecture consisted of two RNN layers and one fully connected (FC)/dense layer. With the Grey Wolf Optimization, they tuned their hyperparameters to obtain the optimal model, achieving accuracies between approximately 95% and 99% for all RNNs. This study demonstrated the impact of RNNs on the EMG signal classification in terms of effectiveness and speed.

Much research has been done on the EMG data of gesture recognition. [16] and [17] used LSTM for hand gesture classification, while [12] for human movements such as walking.

[12] developed an optimal detector tool for human movements using an architecture of two LSTM layers, one FC (dense) layer with softmax activation function.

[17] applied various filters for denoising and signal smoothing, including a finite impulse response (FIR) filter, a low-pass filter, and an additional moving average Filter (SMA). They also standardize the data. They compared a one-dimensional convolutional neural network (1D-CNN) model, comprised of three 1D-CNN layers and one max pooling layer, with a LSTM model, comprised of two LSTM layers and two dense layers. The LSTM model achieved an accuracy of approximately 95%, while the 1D-CNN model achieved 85%.

The study of [16] applied mean subtraction, the absolute value, SMA and normalization to filter and smooth the signals. They evaluated six LSTM models, where the sixth model, comprised of one dense layer, one LSTM layer followed by another dense layer using softmax activation function, obtained the highest accuracy of 87% ± 7%.

The studies [18, 19, 20] proposed a CNN with LSTM model for the recognition of hand gestures, and [21] for muscle classifications.

[18] obtained an average recognition accuracy of approximately 99% and 97% for two datasets. They applied standardization as preprocessing step, scaling the data to have a mean of zero and a standard deviation of one. The architecture included one convolution layer, one max pooling layer, one LSTM layer, and one dense layer with a softmax transfer function, using a hyperbolic tangent activation function.

The CNN-LSTM model of [19] achieved a mean recognition accuracy of 90.55% ± 9.45% using the window technique with a windows size of 300 datapoints and Short-Time Fourier Transform for preprocessing. The model consisted of a convolutional layer with ReLU, one flatten layer, one LSTM layer and one FC layer with softmax.

Study [20] obtained recognition accuracies between 87.0% and 99.7% depending on the different gesture recognition dataset. Their model included two convolution layers, two locally connected layers, three FC layers, one LSTM layer, followed by another FC layer with softmax.

For muscle classification, [21] succeeded to accurately classify various muscle groups in patients. Their architecture consisted of two (1D) convolution layers, one FC layer, one LSTM layer, another FC layer, and ended with a FC layer with softmax.

[22] used a CNN with GRU model for hand gesture recognition, achieving a recognition accuracy of approximately 76%. Their model used ReLU activation and included two convolution layers, one dropout layer, followed by two more convolution layers, one dropout and one max pooling layer, one GRU layer, followed by three blocks of one dense layer and one dropout layer.

The research of [13] compared CNN with LSTM and GRU for the recognition of lower limb activity. They pre-processed the data using Butterworth and wavelet denoising filters, followed by normalization. Additionally, they applied a windowing technique with a 256-window sample and a 25% overlap. The model architecture consisted of two convolution layers, one max pooling layer, two LSTM or GRU layers, and two FC layers. The CNN+LSTM model achieved an accuracy of approximately 97%, while the CNN+GRU model achieved an accuracy of approximately 98%.

Observing these studies, the accuracy of detection using machine learning models such as LSTM, CNN+LSTM and CNN+GRU is remarkably high. However, the architectures obtained in these studies vary, and no single architecture emerges as the best model. Based on the findings of the previous study and the related works using CNN and RNN, I will examine the hybrid models of CNN with LSTM and CNN with GRU for the detection of myoclonus and draw inspiration from these architectures to improve the detection model of the previous study.

3 Methodology

In this section, I will discuss the methodology of this research. First the data for the research will be discussed, followed by the explanation of the preprocessing techniques applied on this data. Next, I will explain the different neural network architectures for the proposed framework, along with the metrics used to evaluate the model's performance. This will be followed by an overview of the implementation of the code. And finally, an overview of the settings that will be tested.

3.1 Data

This study used the EMG data from the Next Move in Movement Disorders (NEMO) study of Expertise Centre Movement Disorders Groningen, University Medical Centre Groningen (UMCG). They collected data from approximately two hundred selected UMCG patients, diagnosed with a movement disorder or as healthy control [14]. The EMG data of patients with the movement disorder (cortical) myoclonus, polyminimyoclonus and myoclonus dystonia (gen negative/SCGE) were selected for myoclonus burst detection.

For non-bursts data, I used the healthy control group and a small set of non-burst annotations from a few patients with the movement disorder. This approach differs from the previous study, which did not incorporate the healthy control group.

The NEMO team obtained EMG data by attaching sixteen sensors to the patient's arms, hands, and face during procedures of rest, posture and action. For this research, I only used the right and left sensors of the biceps (BiR and BiL), the extensor (ExR and ExL) and flexor (FlR and FlL), which are denoted as sensors 3 up to 8, respectively. The patients with the movement disorder had sensor data only from the side of the body where the myoclonus primarily occurs. Therefore, for the 'burst' samples, data from sensors 3, 5 and 7 from the right side, or 4, 6 and 8 from the left side are used, depending on the patient's condition. For the healthy control group, data from all six sensors are used.

The procedure of 'rest' is represented by Task 1, where the patients rested their hands on their legs with palms facing upwards. The procedure of 'posture' is represented by Task 17, where the patients kept their hands straight in front of the body with vertical palms. Lastly, the procedures of 'action' are represented by Task 22 and 23, where the patients preformed a "finger-to-nose maneuver" (22 for the left arm and 23 for the right arm) [2, 1].

Each of these tasks lasted for around thirty seconds, providing an average of 60.000 data points at a measurment frequency of 2000 Hz. Afterwards, the NEMO team manually labelled the onset and offset of the bursts in the EMG data, creating a dataset primarily used for analyzing burst duration differences across multiple myoclonus subtypes (publication under revision). From a few patients, they also annotated segments of EMG data where with certainty no bursts activity occurred, which was used for the non-burst data. The EMG data from the healthy control group consists of non-bursts only.

This research continued to focus on bursts detection in EMG data from Task 17, as it contains the most identified bursts and where the patients experienced the most bursts [1].

3.2 Data preprocessing

This research used the same preprocessing techniques applied in the previous study, as they improved the effectiveness of the detection model's training. I will briefly go through these techniques.

3.2.1 Filtering and rectification

The previous study has shown that bandpass filtering the EMG data had a positive effect on model performance. Therefore, I used the same Butterworth bandpass filter for the EMG data as the previous and other studies have shown to be effective [1, 13, 23]. This filters helps removing the low- and high-frequency noise, preserving clear muscle activity in the EMG signals [24]. The filter applied a bandwidth of 10,450 with order six, decreasing the impact of frequencies outside the range of 10 up to 450 Hz.

After applying the filter, the absolute value of the signal is computed. This rectification converts the signal to all positive values, thereby improving the measured cortical signals in the EMG data [25]. This process helps in better identifying the amplitude and strength of muscle contractions of the muscle activity.

3.2.2 Windowing

The manually labelled data set, with their own assigned start and end times of myoclonus burst, creates a potential human bias when training a model with these labelled bursts. With the windowing technique I will not constrain the model to detect these exact start and end points, but rather classifying sample windows into one of two classes: 'bursts' or 'non-bursts'. The labelled bursts in the dataset have a duration of approximately 50-70 milliseconds, which corresponds to 100-140 datapoints. Therefore, a window size of 256 datapoints, the first power of two that guarantees capturing the entire burst, was chosen for the data samples and was used for this research.

3.2.3 Augmentation

The total number of bursts across all participants than can be sampled with a window size of 256 datapoints is relatively small. Therefore, I augmented the data to increase the number of burst samples for the training, validation and testing datasets. I sampled ten different positions of each burst within a 256 data point window, creating ten data samples per burst. Through this augmentation strategy, the model learns to detect burst without bias towards their specific location with the time window.

3.2.4 Scaling

One preprocessing technique that was tested in this research is the application of normalization, standardization or neither to the dataset to improve the model accuracy. Both techniques aim to make the data consistent for machine learning.

Normalization scales the data to a specific range, bringing all features to a consistent scale, without disturbing the difference in their range of values. [26]. For this research, the Min-Max normalization was applied with a range of [0,1]. The equation is as follows:

$$\boldsymbol{x'} = \frac{\boldsymbol{x} - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Standardization scales the data to have a mean of zero and a standard deviation of one [26], achieving a consistent scale across all features, improving their comparability.[26]. This is also known as the Z-score standardization. The equation is as follows:

$$\boldsymbol{x'} = \frac{\boldsymbol{x} - \mu}{\sigma} \tag{2}$$

,where μ is the mean of x and σ is the standard deviation of x.

3.3 Proposed Framework

3.3.1 Convolutional neural network (CNN)

Convolutional neural networks (CNNs) are well-known neural networks that are specialized in analysing local patterns in data, such as signals (1D), images (2D) and videos (3D) [27].

By extracting relevant features and patterns using convolutional operations, CNNs can be used for classification, recognition, segmentation or detection. This research explored the detection of the EMG sequence data; therefore, the most suitable CNN model is the one-dimensional CNN.

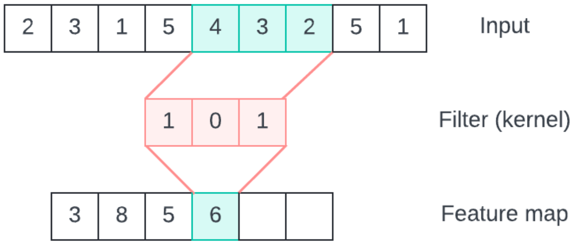


Figure 2: Convolution.

The basic architecture of a neural network consists of the input, hidden and output layers. In a convolutional neural network, the hidden layers primarily consist of three types of layers: the convolution, the pooling and the fully connected (dense) layer [28].

The convolution layers in the 1D CNN extracts valuable features by sliding a filter (kernel) of size k over the 1D input sequences and computing dot products with the signal values, generating a feature map capturing patterns [18], see Figure 2. In a convolution layer, you specify the number of filters. Each filter applies the convolution operation to the input, generating a feature map. Consequently, the number of filters determines the number of feature maps produced. In this research, the number of filters is one of the parameters that was tested with various values. To add non-linearity to the outcome, the CNN applies an activation function to the feature map after the convolution layer. In this research, the rectified linear activation unit (ReLU) function was used.

$$ReLU(x) = max(0, x) \tag{3}$$

, where x is an input value.

Next, the result passes through pooling layers, which reduce the spatial dimensionality of the feature maps and summarize the most important features [28, 29]. This research applied max pooling that selects the maximum value within the pooling window, as shown in Figure 3, highlighting the most important features related to the muscle activity extracted by the convolutional layer.

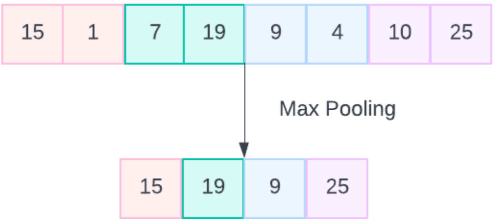


Figure 3: Max pooling.

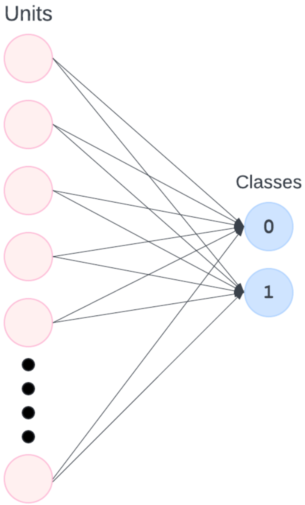


Figure 4: Fully connected layer.

Finally, the fully connected (dense) layer converts the feature maps into a 1D array with a size equal to the number of classes, see Figure 4. Each entry corresponds to the predicted score or probability of a specific class. In this research, the fully connected layer was applied after the recurrent neural networks' variants.

Even though CNNs typically contain these three layers, additional layers can be added to improve model performance. In this research, I expored the use of the dropout layer, which was tested in both CNNs and RNNs . The dropout layer drops a fraction of the input data during training, to add regularization and therefore prevent overfitting [30].

In this research, our model must classify the windows samples into the two categories: 'non-burst' (0) and 'burst' (1). This categorization falls under binary classification. To achieve this, the sigmoid activation function was used in the final layer (the dense layer) of the neural network. The sigmoid function outputs probabilities in the range of 0 to 1, allowing us to interpret the result as the probability that an input sample belongs to the class 'non-burst' (0) or 'burst' (1). The sigmoid activation function is as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

, where x is an input value.

3.3.2 Recurrent Neural Network (RNN)

What the recurrent neural network (RNN) differs from the convolutional neural network (CNN) is its ability to analyse sequential time-series data, such as text, video or in our research, EMG data (time vs amplitude) [31]. The RNN processes the input sequentially in the hidden layers with recurrent connections. Due to these recurrent connections, the RNN can use the previous outputs as extra input for analyzing the next data points, leading to an inherent memory that contains information about all previous sequences.

However, when as RNNs becomes deeper, training can become slower and less effective due to the vanishing gradient problem.This problem occurs during backpropagation when the gradients decrease to very small values, which causes the weights in the earlier layers of the network to stop updating and hinders the network's ability to learn from long-term dependence [32]. This problem leads to deficient performance of capturing the long-term dependencies in RNNs.

Long Short-Term Memory (LSTM)

To solve this problem, they introduced an extension of the RNN, namely the Long Short-Term Memory (LSTM) [33]. The LSTM layer consists of multiple specialized memory LSTM cells, see Figure 5, with gating mechanisms that regulate information flow and therefore improve the long-term dependencies. Within the LSTM cell, there are three gates: the forget gate, the input gate, and the output gate each with its own bias and weight vectors.

The forget gate, f_t, determines the proportion of information that should be removed ("forgotten") from the cell state C_t [1, 18]. The current data point X_t and the previous hidden state h_{t-1} are passed through a sigmoid function with their own weights W_{fx} and W_{fh} and bias b_f. The output, f_t, ranges from 0 to 1, determining to use all information (1) or none (0).

$$f_t = \sigma(W_{fx}X_t + W_{fh}h_{t-1} + b_f) \tag{5}$$

The input gate, i_t, controls the addition of new information into the LSTM memory unit C_{t-1}. Here, X_t and h_{t-1} are passed through a sigmoid function with weights W_{ix} and W_{ih} and bias b_i to determine how new current information to forget, similar to the forget gate.

$$i_t = \sigma(W_{ix}X_t + W_{ih}h_{t-1} + b_i) \tag{6}$$

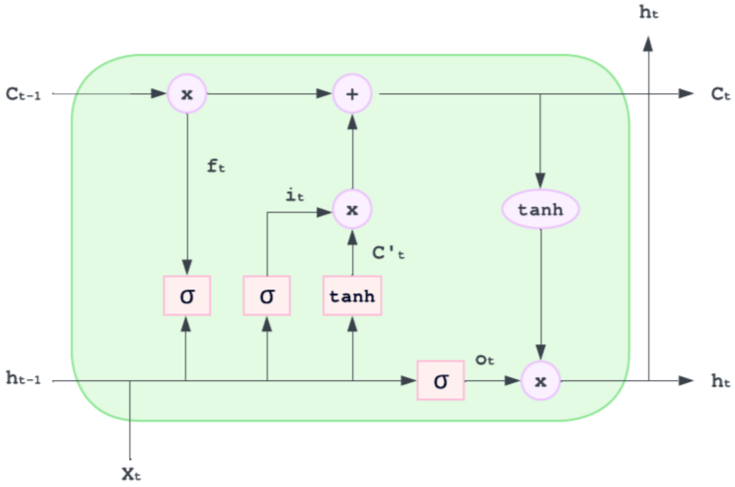


Figure 5: The LSTM cell.

Then the new information C'_t (cell activation), is processed through a hyperbolic tangent function with X_t and h_{t-1}, using weight W_C and bias b_C. The outputs of the I_t and C'_t are multiplied to incorporate the new information into C_t, see equation 9.

$$C'_t = tanh(W_C X_t + W_C h_{t-1} + b_C) \tag{7}$$

The output gate, o_t, determines the proportion of information from the current cell state C_t for current hidden state h_t. Here, X_t and h_{t-1} are passed through a sigmoid function with weights W_{ox} and W_{oh} and bias b_o.

$$o_t = \sigma(W_{ox} X_t + W_{oh} h_{t-1} + b_o) \tag{8}$$

The current cell state, C_t, is obtained by multiplying the previous cell state C_{t-1} by the forget gate output and adding the result to the product of the input gate output and C'_t.

$$C_t = f_t \odot C_{t-1} + i_t \odot C'_t. \tag{9}$$

This C_t value is then passed through a hyperbolic tangent function and multiplied by the output gate value to produce the new hidden state h_t.

$$h_t = O_t \odot tanh(C_t) \tag{10}$$

Gated Recurrent Unit (GRU)

Another solution to the vanishing gradient problem is the gated recurrent unit (GRU), a modification of the LSTM [34, 35]. Similar to LSTM, GRU processes sequential data through gates, but is uses only the hidden state to handle the memory, removing the separate memory cell (C_t in LSTM). Additionally, the GRU cell consists of two different gates, namely the update and reset gate, see Figure 6.

The update gate, u_t, decides the amount of past information to be preserved for the future. The current data point X_t and the previous hidden state h_{t-1} are passed through a sigmoid function with their own weights W_{ux} and W_{uh} and bias b_u.

$$u_t = \sigma(W_{ux} X_t + W_{uh} h_{t-1} + b_U) \tag{11}$$

The reset gate, r_t, controls how much past information to forget. Here, X_t and h_{t-1} are passed through a sigmoid function with weights W_{rx} and W_{rh} and bias b_r.

$$r_t = \sigma(W_{rx} X_t + W_{rh} h_{t-1} + b_r) \tag{12}$$

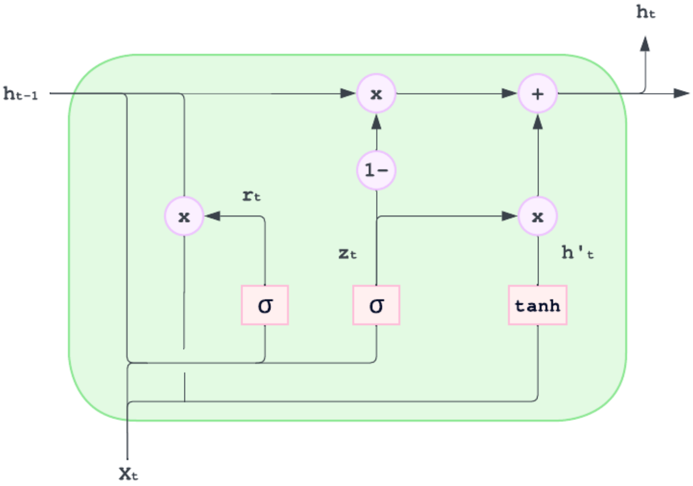


Figure 6: The GRU cell.

The internal activation h'_t, captures relevant past information. h'_t is computed by the output of r_t multiplied by h_{t-1}, combined with X_t, and passed through a hyperbolic tangent function with the weight $W_{h'x}$ and $W_{h'h}$ and bias $b_{h'}$.

$$h'_t = tanh(W_{h'x}X_t + W_{h'h}(r_t \odot h_{t-1}) + b_{h'}) \tag{13}$$

The current hidden state, h_t, is derived by multiplying u_t with h'_t and adding it to the results of u_t multiplied by h_{t-1}.

$$h_t = u_t \odot h'_t + (1 - u_t) \odot h_{t-1} \tag{14}$$

Having one fewer gate, GRU uses fewer parameters, increasing simplicity, efficiency, and speed. However, GRU might to not process complex sequential patterns or very long-term dependencies as good as LSTM. Both extensions have their advantages and disadvantages; therefore, in this study I examined both RNN extensions [36].

In this study, I applied hybrid convolutional neural network models, which extract relevant features from EMG data, with variants of the recurrent neural network, the LSTM and GRU, handling the sequential time series data with their inherent memory. These hybrid models were explored for the detection of bursts in the EMG data and examined for an improvement of [1].

3.4 Model evaluation

To develop the optimal model, I use the three datasets: training, validation, and testing set, with a 60-20-20% distribution respectively.

The training set is used to train the model, while the validation set, which contains new unseen data, is used to finetune the hyperparameters. Based on the accuracy improvements from the validation set, the optimal model is saved. The test set, also containing new unseen data, is used to evaluate the final model's performance, ensuring its generalizability to new data.

To evaluate the performance of the various models tested in this research, I used the following metrics: Accuracy, Loss, Precision and Recall.

Accuracy measures the percentage of correct predicted labels by calculating the ratio of correctly predicted labels to the total number of true labels:

$$Accuracy = \frac{Correct\ predicted\ labels}{Total\ number\ of\ true\ labels} \tag{15}$$

Loss evaluates how well the model predicts the true label. A lower loss indicates better performance in predicting the true labels, while a higher loss implies less accurate predictions. For the loss calculation, binary cross entropy (BCE) is used, which measures the differences between predicted and true binary labels. The equation is as follows [37]:

$$BCE = -\frac{1}{M}\sum_{m=1}^{M}[y_m \times log(h_\theta(x_m)) + (1 - y_m) \times log(1 - h_\theta(x_m))] \tag{16}$$

, where M is the number of samples, y_m is the true label of the m-th sample, x_m is the predicted label input of the m-th sample and h_θ the model with neural network weights θ.

The metrics Precision and Recall use True Positives (TP), True Negatives (TN), False positives (FP), and False Negatives (FN) to evaluate the prediction performance of a model. True Positives (TP) are the number of positive samples correctly identified as positive. True Negatives (TN) are the number of negative samples correctly identified as negative. False Positives (FP) are the number of samples that are incorrectly identified as positive, and False Negatives (FN) are the number of samples that are incorrectly identified as negative [15].

Precisions measures how accurate the positive predictions are. [38]. It computes the percentage of positive predictions that are correct. The equation is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

Recall measures the model's ability to correctly identify positive samples. It computes the percentage of positive samples that are correctly predicted as positive. The equation is as follows:

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

After obtaining the optimal CNN+LSTM and CNN+GRU models, I evaluated their performance on the training, validation, and testing sets using confusion matrices.

The confusion matrix provides an overview of predicted and true labels with the TP, TN, FP and FN, as shown in Figure 7. In this research, a positive label (1) represents the detection of a burst, while a negative label (0) represents the absence of a burst.

Additionally, the models' predictive performance are evaluated by applying them to EMG data from the test set, which includes both myoclonus patients and healthy controls, performing not only Task 17 but also the other discussed tasks. The evaluation included several decision thresholds, which determines the cutoff for classifying a sample as positive ('burst') or negative ("non-burst").

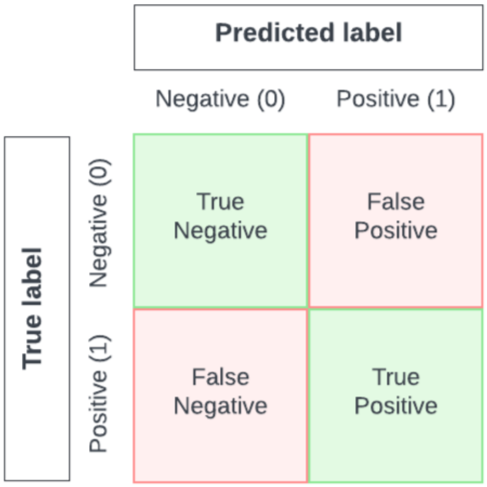


Figure 7: The confusion matrix.

3.5 Implementation

The programming language used in this research was Python 3.10.10. The most important Python package for this research was Keras 3.3.3. This research relied on the Keras framework for creating, training, and testing the neural networks. Keras also provided useful callbacks and offered the advantage of tensors, which are optimized multidimensional arrays designed for GPU processing.

Other helpful packages in this research were sklearn for the normalization or standardization of the data confusion matrix, NumPy for arrays structures and matplotlib for the visualization of the loss and accuracy during the training and testing process.

The models were trained on Linux Ubuntu 22.04 system with an NVIDIA RTX3090 GPU and CUDA version 11.2 [39].

3.5.1 Samples

First, I filtered out the patients labelled with the movement disorder or as healthy controls from the NEMO data. Patients with a movement disorder were labelled with the types (cortical) myoclonus and myoclonus dystonia (gen negative/SCGE). Patients with polyminimyoclonus were included and classified under the myoclonus group.

Within each group (myoclonus, myoclonus dystonia and healthy control), I divided the patients into 60%-20%-20% distributions for the 'burst' or 'non-burst' training, validation, and testing set, respectively.

More specifically, there were twenty patients with myoclonus, twenty six patients with myoclonus dystonia, and fourty healthy controls. Twelve patients with myoclonus and sixteen patients with myoclonus dystonia were placed in the 'burst' training set, while twenty four healthy controls were placed in the 'non-burst' training set.

The creation of samples differed for the burst and non-burst data. For each training, validation, and test set, I followed the same process:

Burst sets
For each patient in the respective list (train/validation/test), I looped through the list of tasks, retrieved the list of timestamps and then looped through each sensor. For each sensor, if the EMG data existed, I applied the Butterworth filter and rectified the data. Then, I retrieved the corresponding list of bursts, align the bursts annotations with the time stamps and ordered the bursts chronologically.

Next, I looped through the list of bursts. For each burst, I retrieved the start and end points and extended the burst window to a size of 256 data points. I created samples by extracting the corresponding EMG data for each window and applied data augmentation to generate ten burst samples per burst, with each sample placing the burst at a different position within the window. The samples from the patients were placed in one list and returned to the corresponding train/-validation/testing set.
Some movement disorder patients also had non-burst annotations. For these, I retrieved the non-burst data and checked if it did not overlap with burst-labelled data. If it did, I remove any overlapping bursts and non-bursts from both lists. For each non-bursts, I sampled the non-burst data with a 75% overlap if the non-burst is longer than 256 datapoints.

For each training, validation, and testing list, I also maintained a corresponding label list. Initially, this list was created with the same size of the corresponding burst list and filled with one's.

Healthy control group
For each patient in the respective list (train/validation/test), I retrieved the list of timestamps and the EMG data from all six sensors. For each patient, I selected the EMG data from one sensor and denoised it with the Butterworth filter. The sensor number changed after processing a patient. The non-burst samples were created by retrieving the EMG data within a window of 256

datapoints window with a 25% overlap. All non-burst samples from the patients were stored in one list and returned to the corresponding data set.

After retrieving the non-burst samples, I merged the two non-burst sample lists together for training, test, and validation sets. Then, I checked if the amount of burst and non-burst samples was equal, such that I had a 50% division in each set. If the non-burst sample list is lesser, I created additional samples using the same process but selected data from a different sensor than before and cropped the list of non-burst samples to the same length as the burst samples.

After the check, I added the non-burst samples to the corresponding burst sample list and appended the number of zero's, which is equivalent to the number of non-burst samples (and burst samples), to the label list of the corresponding set.

Next, I applied normalization or standardization to the dataset. The scalers, obtained by the training data, were saved using `joblib`. Depending on the model, the suitable scaler should be applied to the new, unseen data before entering the model.

Finally, the sample lists are reshaped into a 3D shape array, ready for the training and testing the model.

3.5.2 Models

The models are implemented using `Keras`. I looped over the list of batch sizes, epochs, and filters values. For example, with [100] epochs, [16,32] batch sizes and [16, 32] filters, the model was trained and tested with the combinations (100,16,16), (100,16,32), (100,32,16) and (100,32,32).

In a dictionary named `parameters`, I stored the following parameters: `model_number`, `layer`, `hidden_units`, `max_pool`, `droplayer`, `droprate`, `adam_rate`, `dense_units` and `kernel_size`.

To observe how well a model performed, I did not rely on the results of a single training run. Instead I took the average two runs. Therefore, in the implementation, the model could be trained for a certain number of `REPEATS`. For each `repeat`, the obtained losses and accuracies of training, validation, and test along with the parameters were saved in a `csv` file. After all the repeats, a separate `csv` saved the mean and standard deviations of all test accuracies along with the parameter settings. For each repeat, the loss values of the training and validation set during the training of X epochs were stored in separate Dataframe and were plotted after all repeats.

3.5.3 Confusion matrices and prediction plots

The confusion matrices for the training, validation and test set were generated with the `confusion_matrix` function of `sklearn` and were plotted with the package `sns`.

For the predictions plots on the EMG data, the timestamps and the EMG data of the specific test patient with a specific task and sensor were first retrieved. Both the raw unfiltered EMG data and the filtered, scaled EMG were saved in two separate plots, which were the same for both models.

A loop then slid over the EMG data with a step size of 1, saving each 256-window sample into a separate list. Another list recorded the start times of these sample, which were the timestamps of the middle data point of the 256-windows.

The EMG samples were then scaled with the scaler associated with the trained model. The `predict` function of `Keras` returned a list of probabilities for each sample, representing the likelihood of being a “burst”. Based on the selected decision threshold, each sample was classified as either ‘burst’ or ‘non-burst’. The start and end times of detected bursts were listed in two separate lists. Finally, these times were plotted as vertical lines on the scaled and filtered EMG data plot to visualize the detected bursts.

3.6 Models and Settings

In this research, I performed a grid search to obtain the optimal model implementing the CNN+LSTM and CNN+GRU architecture.

These are the parameter settings that were tested:

1. **The number of layers.**
   - Layer 1: CNN and LSTM/GRU
   - Layer 2: CNN, CNN and LSTM/GRU
   - Layer 3: CNN, CNN and LSTM/GRU, LSTM/GRU
   - Layer 4: CNN, CNN, CNN and LSTM/GRU, LSTM/GRU
   - Layer 5: CNN, CNN, CNN and LSTM/GRU, LSTM/GRU, LSTM/GRU

After each CNN layer, the ReLU activation function is applied.

2. **The number of Drop layers**, which are placed after a CNN and LSTM/GRU block:
   - Drop layer 1: CNN block, LSTM/GRU block, Drop
   - Drop layer 2: CNN block, Drop, LSTM/GRU block, Drop

3. **The number of epochs, batch sizes, filters, hidden and dense units**:
   - Epochs = [50, 100, 300]
   - Batch sizes, filters, hidden and dense units = [8,16,32,64,128]

4. **Data preprocessing**
   - Standardization, Normalization or None

These are the fixed parameters:

- Max pooling = 2
- Kernel size = 3
- Dropout rate = 0.5
- Learning rate of the Adam optimizer = 0.001

The network tail of the model consist of two Dense layers. The first layer applied the ReLU activation function, while the second applied the sigmoid activation function.

4 Results

In this section, I present the findings obtained during this research, aimed at improving the model for the detection of bursts. I begin by showing the amount of samples and providing examples of the input samples using the different scaling techniques, each with the 256-window. Following this, I showcase the visual representations of the best obtained convolution neural network (CNN) with long short-term memory (CNN+LSTM) and CNN with gated recurrent unit (CNN+GRU) models.

Next, I present plots showing the loss and accuracy during the training of three hundred epochs for both the training and validation sets for each model, indicating the models' training and validation performance. The evaluation metrics of the test set are then presented to show the performance of both models on unseen data samples. Additionally, the confusion matrices for the training, validation, and test sets are provided to evaluate the models' prediction accuracy.

Finally, I visualize the prediction performance of the models on unseen EMG data by plotting the EMG data with annotated bursts and predicted bursts in separate plots. This visualization is given for both patients with movement disorders, for the 'burst data' analyses, and healthy controls, for the 'non-burst' data analyses.

4.1 Data samples

In this research, the 'bursts' samples for training, validation and test sets, were created based on the annotated bursts from each patient performing Task 17, recorded by sensors on a specific body side. The number of annotated burst per sensor is presented in Table 1. Because of the difference in annotated bursts, some sensors had more samples than the other sensors, as can be seen for sensor four which has the half amount of samples than than sensor six.

Additionally, during data augmentation ten new samples were created from a burst. However Table 1 shows that during data augmentation for 'burst' samples, not all generated samples had a window size of 256 data points. This deviation occurred because some bursts were labelled near the beginning or end of the EMG data, causing the random positioning to prevent the window from capturing 256 data points.

For the 'non-burst' samples, data was retrieved from one sensor per patient performing Task 17. For each patient, samples were created by sliding the 256-window with an overlap of 10% over the EMG data. As shown in Table 1, sensors 7 and 8 had fewer 'non-burst' samples than the other sensors. Table 1 further shows the distribution of samples is well-balanced across the right and left sides for each muscle sensor.

In terms of burst samples in the sets, the distribution was as follows: 8,764 samples for the training set, 2,712 for the validation set, and 3,037 for the test set. An equal number of 'non-burst' samples were created for each set, maintaining a 50% split between 'burst' and 'non-burst' samples.

Overall, the dataset consisted of 17,528 samples in the training set, 5,424 samples in the validation set and 6,074 in the test set, making a total of 29,026 samples in this study.

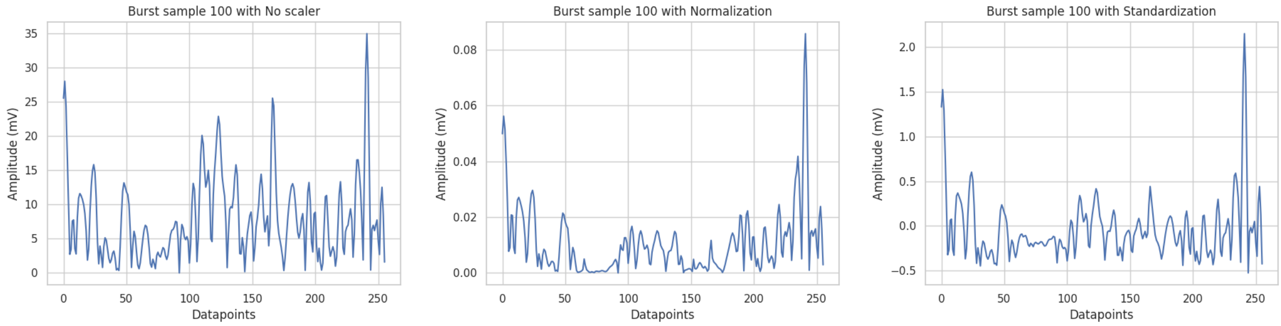
Sensor	Annotated Bursts	Augmented Bursts Samples	Non Bursts Samples	Total Samples
3: BiR	211	2110	2538	4648
4: BiL	150	1500	3222	4722
5: ExR	269	2681	2732	5413
6: ExL	311	3104	2222	5326
7: FlR	260	2596	1919	4515
8: FlL	253	2522	1880	4402

Table 1: The number of annotated bursts and the resulting 'burst' and 'non-burst' samples for each sensor. The sensors include the right and left biceps (BiR and BiL), the right and left extensors (ExR and ExL), and the right and left flexors (FlR and FlL).

4.2 Input burst and non burst samples

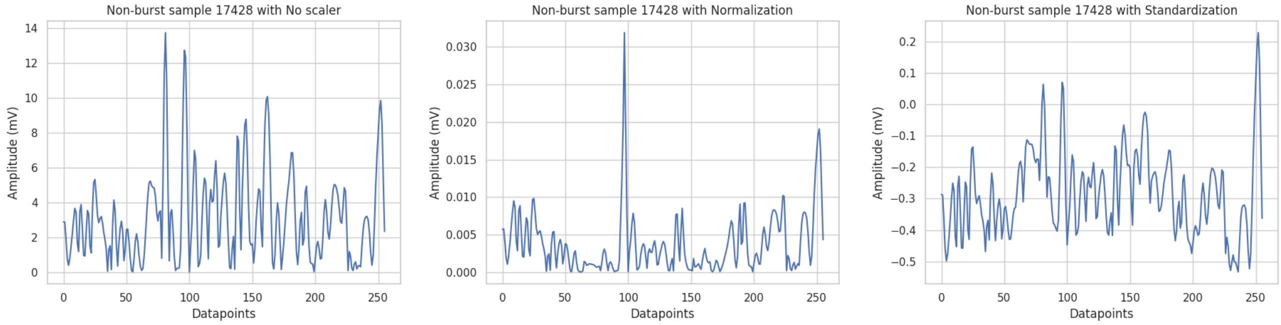
Before the training and testing of the model, various scaling preprocessing methods, including preprocessing methods, including normalization, standardization, and no scaling, were evaluated to examine there impact on the models' performances.

Figures 8 and 9, illustrate the effects of these scaling technique on a 'burst' and 'non-burst' sample, respectively. In Figures 8(b) and 8(c), both normalization and standardization highlighted the burst peak in the 'burst' sample around data point 190. In Figure 9, normalization highlighted the peak around data point 100, while standardization scales the group of peaks and highlights the peak at around 256 datapoints of the 'non-burst' sample.



(a) No scaler (b) Normalization (c) Standardization

Figure 8: Plots of the amplitude (mV) across 256 data points for sample 100 ('burst'): (a) without scaling, (b) with normalization, and (c) with standardization applied.



(a) No scaler (b) Normalization (c) Standardization

Figure 9: Plots of the amplitude (mV) across 256 data points for sample 17428 ('non-burst'): (a) without scaling, (b) with normalization, and (c) with standardization applied.

After applying these preprocessing techniques and training the models with different parameter settings and architectures, the differences in performance was minimal. However, standardization consistently yielded slightly better results in the evaluation metrics and stability for both models. Therefore, standardization was chosen as the preprocessing method for both optimal models.

4.3 The optimal models

The performance of various models was evaluated using the training, validation and testing set in terms of the evaluation metrics: loss, accuracy, precision, and recall. While many models showed optimal results in terms of the evaluation metrics, visualising their training process with the training and validation set revealed inconsistency and very spiked curves in the loss and accuracy plots.

The optimal models of CNN+LSTM and CNN+GRU, shown in Figures 10 and 11, were chosen not only for their average low loss and high accuracy, precision, and recall values of the test set, but also because of their stability during training with the training and validation set. The results on these models' performance and stability are discussed in the following sections.

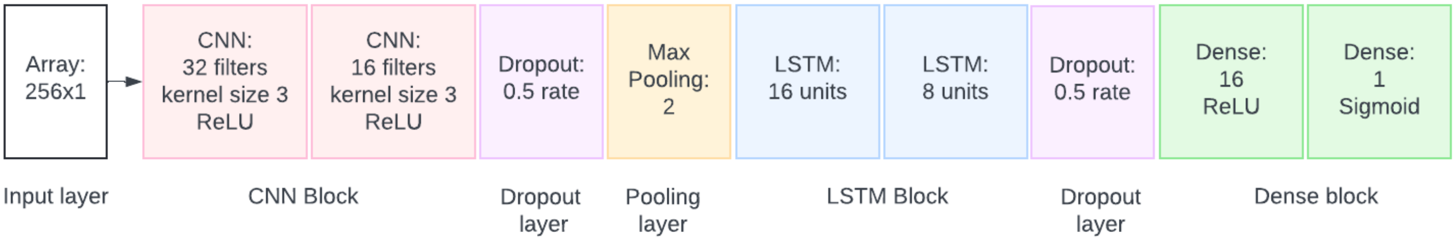


Figure 10: The architecture of the best performed CNN+LSTM model.

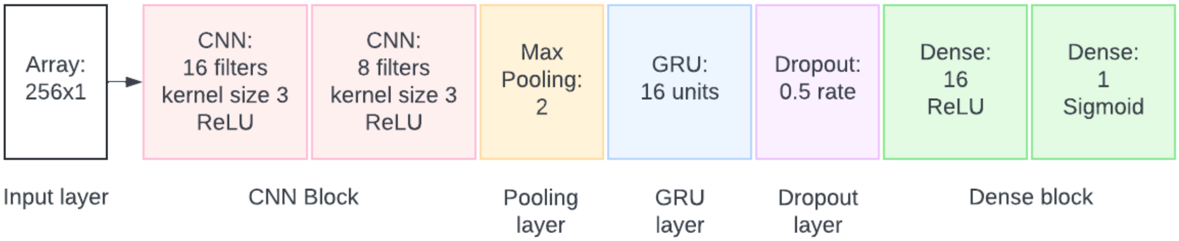


Figure 11: The architecture of the best performed CNN+GRU model.

Figures 10 and 11 visually represent the architectures of the CNN+LSTM and CNN+GRU model, respectively.

The CNN+LSTM model with batch size 32 and standardization, includes the following layers:

- **Convolutional block**: Two convolution layers, each with a kernel size of 3 and ReLU activation. The first layer includes 32 filters, while the second has 16 filters.
- **Dropout Layers**: Two drop layers, each with dropout rate 0.5. One layer is placed after the convolutional block, and the other after the LSTM block.
- **Max Pooling Layer**: One max pooling layer with a pool size of 2.
- **LSTM block**: Two LSTM layers, with the first layer containing 16 units and the second layer containing 8 units.
- **Dense block**: And at the end there are two dense layers. The first layer contains 16 units and applies the ReLU activation, while the second layer has 1 unit and applies sigmoid activation layer.

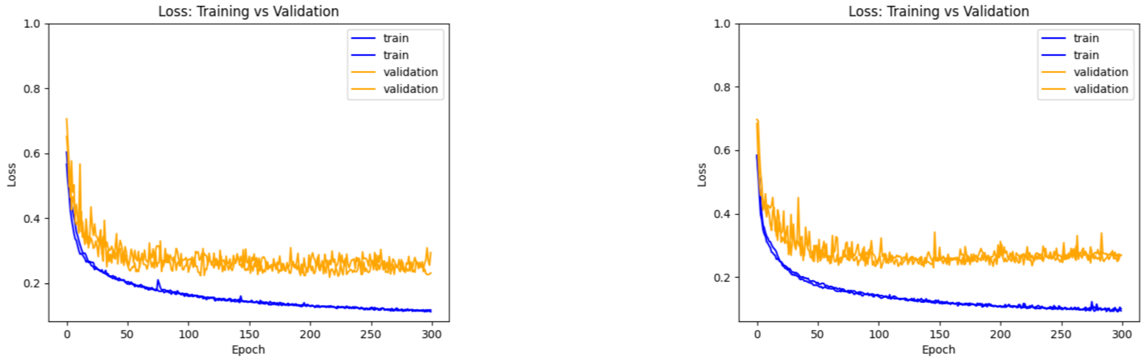
The CNN+GRU model with batch size 16 and standardization, includes the following layers:

- **Convolutional Block**: Two convolution layers, each with a kernel size of 3 and ReLU activation. The first layer includes 16 filters, while the second has 8 filters.
- **Max Pooling Layer**: One max pooling layer with a pool size of 2.
- **GRU Layer**: One GRU layer containing 16 units.
- **Dropout Layer**: One drop layer with drop rate 0.5.
- **Dense block**: And at the end there are two dense layers. The first layer contains 16 units and applies the ReLU activation, while the second layer has 1 unit and applies sigmoid activation layer.

Both these models used the Adam optimizer with learning rate 0.001, and the data were standardized before training and testing these optimal models.

4.4 Loss and accuracy of the training and validation sets

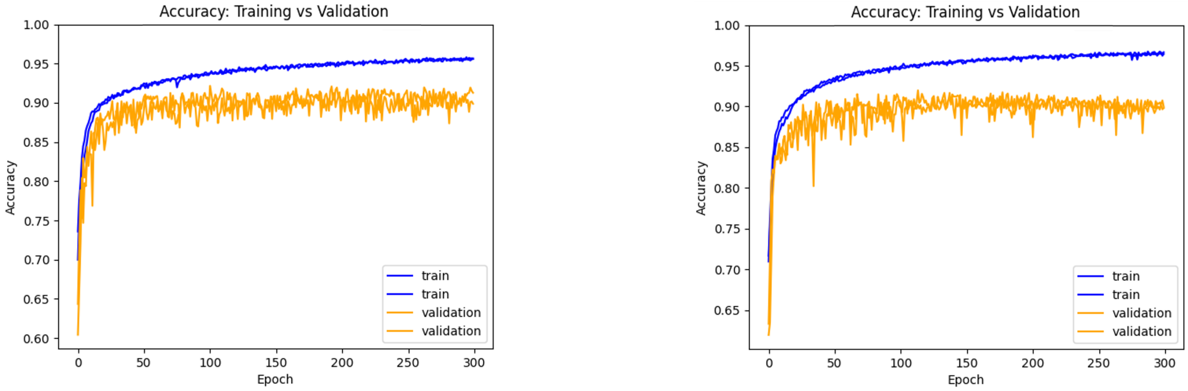
During the training of the models over 300 epochs using the training and validation set, the history of loss and accuracy values was recorded. The loss and validation plots in Figures 12 and 13 show the stability that distinguished the optimal models of CNN+LSTM and CNN+GRU from other models that displayed less consistent performance. The loss plots in Figure 12 show that the CNN+LSTM model stabilized around a loss value of 0.3, whereas the CNN+GRU model showed a slight increase towards the end, also around a loss value of 0.3. In contrast, other CNN+GRU models showed greater instability in the last epochs compared to the selected model. The accuracy plots in Figure 13 confirm the stability of both models, with each stabilizing at an accuracy value of approximately 0.90.



(a) CNN+LSTM

(b) CNN+GRU

Figure 12: Plots showing the training and validation loss values over 300 epochs of two rounds for: (a) CNN+LSTM and (b) CNN+GRU.



(a) CNN+LSTM

(b) CNN+GRU

Figure 13: Plots showing the training and validation accuracy values over 300 epochs of two rounds for: (a) CNN+LSTM and (b) CNN+GRU.

4.5 Evaluation metrics of the test set

The performance of both the CNN+LSTM and CNN+GRU models on the unseen data of the test set, evaluated with the discussed metrics, is presented in Table 2.

The table shows that both models presented similar results, with only minimal difference in their performance metrics. The high scores for loss, accuracy, precision, and recall indicate robust performance in accurately predicting the true labels and predicting the positive samples, which corresponds to the ’burst’ samples.

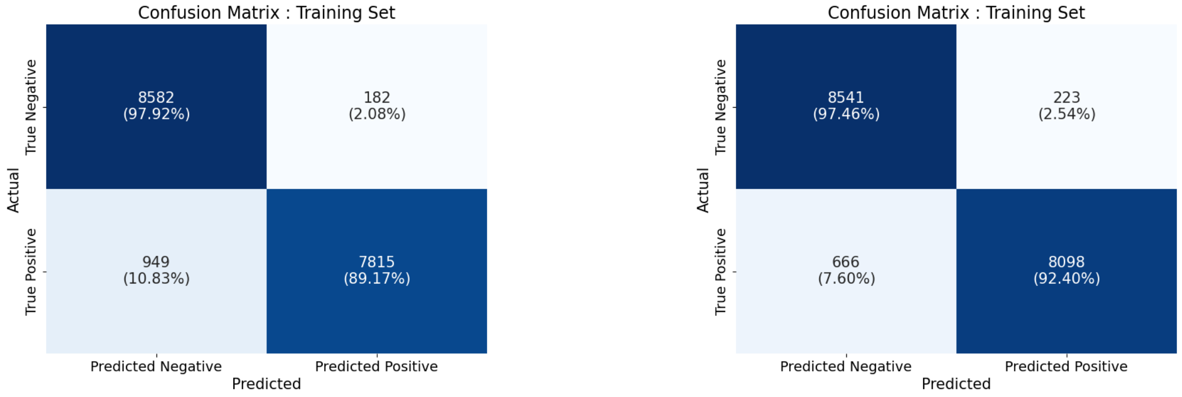
Model	Loss	Accuracy	Precision	Recall
CNN+LSTM	0.2784348130	0.8847547174	0.8708346486	0.9035232067
CNN+GRU	0.27918699384	0.8921633363	0.8555223942	0.9436944127

Table 2: Comparison of evaluation metrics (loss, accuracy, precision, and recall) between of the CNN+LSTM and CNN+GRU models on the test set.

4.6 Confusion of the three sets

For the prediction performance of both models’ on the training, validation and test set is evaluated with the confusion matrices. Figures 14, 15, and 16 show these confusion matrices for the training, validation, and test sets of the CNN+GRU and CNN+LSTM models. Each matrix displays the number of samples for True Positive (TP, bottom right), False Positive (FP, top right), False Negative (FN, bottom left), and True Negative (TN, top left) cases. Additionally, the matrices include the percentage of positive predictions (TP and FN) and negative predictions (TN and FP) separately.

The confusion matrices for the training set, shown in Figure 14, reveal that True Negatives (TN) are higher than True Positives (TP) for both models. Specifically, CNN+LSTM had 8,582 TN and 7,815 TP, while CNN+GRU had 8,541 TN and 8,098 TP. indicating better performance in predicting ‘non burst’.



(a) CNN+LSTM

(b) CNN+GRU

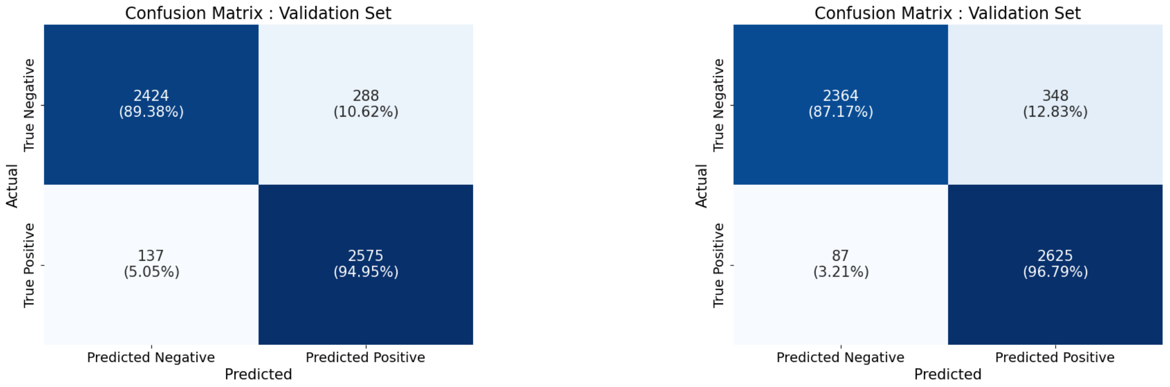
Figure 14: Confusion matrices for the train set, displaying the number of samples for True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) cases, along with the percentage of positive predictions (TP and FN) and negative predictions (TN and FP) separately: (a) CNN+LSTM and (b) CNN+GRU.

However, the validation and test set show that both models perform better in predicting ‘bursts’ samples. The matrix of the validation set, illustrated in Figure15, shows that CNN+LSTM had 2,424 TN and 2,575 TP, and CNN+GRU had 2,364 TN and 2,625 TP. The confusion matrices show that approximately 11% of ’non-burst’ samples were falsely predicted as positive by CNN+LSTM, and about 13% by CNN+GRU. Conversely, around 5% of ’burst’ samples were misclassified as negative by CNN+LSTM, and approximately 3% by CNN+GRU.

Similarly, the matrix of the test set, illustrated in Figure16, shows that CNN+LSTM had 2,630 TN and 2,744 TP, and CNN+GRU had 2,553 TN and 2,866 TP. The confusion matrices show that approximately 13% of ’non-burst’ samples were false predicted as positive by CNN+LSTM, and about 16% by CNN+GRU. Conversely, around 5% of 'burst' samples were misclassified as negative by CNN+LSTM, and approximately 3% by CNN+GRU.

Comparing the CNN+LSTM and CNN+GRU models in Figures 14 through 16, the confusion matrices show that the LSTM+CNN model performs better at predicting 'non-burst' samples. Specifically, CNN+LSTM achieved True Negative (TN) percentages of 97.92% in the training set, 89.38% in the validation set, and 86.60% in the test set. In contrast, the CNN+GRU model achieved TN rates of 97.46% in the training set, 87.17% in the validation set, and 84.06% in the test set.

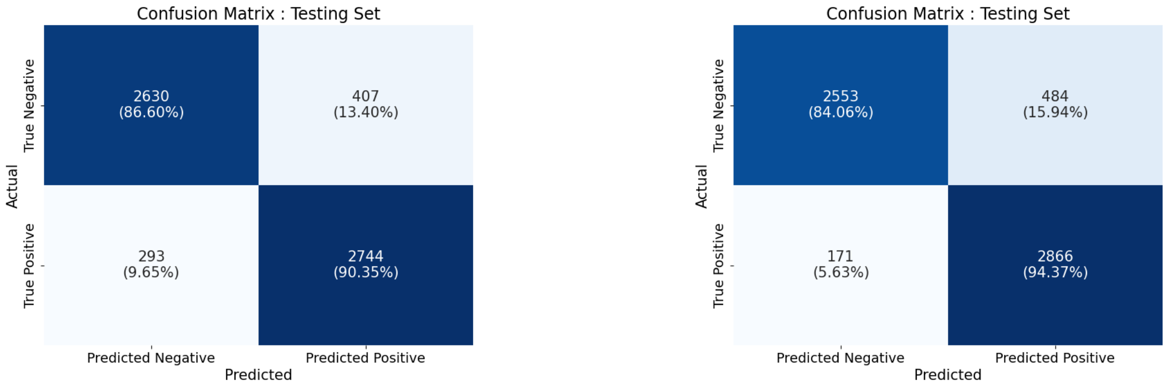
However, the CNN+GRU performs better in predicting the 'burst' samples than the CNN+LSTM models. Specifically, CNN+GRU achieved True Positives (TP) percentages of 92.40% in the training set, 96.79% in the validation set, and 94.37% in the test set. In contrast, the CNN+LSTM model achieved TN rates of 89.17% in the training set, 94.95% in the validation set, and 90.35% in the test set.



(a) CNN+LSTM

(b) CNN+GRU

Figure 15: Confusion matrices for the validation set, displaying the number of samples for True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) cases, along with the percentage of positive predictions (TP and FN) and negative predictions (TN and FP) separately: (a) CNN+LSTM and (b) CNN+GRU.



(a) CNN+LSTM

(b) CNN+GRU

Figure 16: Confusion matrices for the test set, displaying the number of samples for True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) cases, along with the percentage of positive predictions (TP and FN) and negative predictions (TN and FP) separately: (a) CNN+LSTM and (b) CNN+GRU.

4.7 Predictions on EMG test patients, all tasks

As mentioned in the section 3, the models were applied to EMG data from patients in the movement disorder group and the healthy control group, taken from the test data set, performing the Tasks 17, 1, 22 and 23. For each patient, EMG data was plotted with manually labelled bursts for myoclonus patients and no labels for healthy controls. Additionally, separate plots were created with the filtered and scaled EMG data, showing the predicted bursts at decision thresholds 0.5, 0.8 and 0.9.

4.7.1 Myoclonus patients

After analysing the predicted burst on myoclonus patients across all tasks (1, 17, 22 and 23), several distinct cases were observed. These cases are not related to specific sensors, tasks or patients as they are consistently observed in all EMG data.

The distinct cases are presented in this section by different patients performing various tasks.

Annotated and Predicted Bursts aligned in EMG data of Patient NEMO_024
In Figure 17, the manually labelled bursts in the EMG data from patient NEMO_024 are shown. These annotations were used to evaluate the accuracy of the predicted bursts, generated by the model. Figure 18 shows the pre-processed data, which was filtered with the Butterworth filter to remove noise, rectified to convert negative values to positive, and standardized for consistency. The pre-processed data was used for the prediction process.

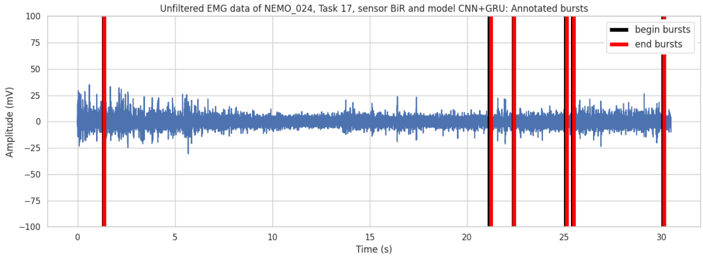


Figure 17: Manually labelled bursts in the unfiltered EMG data of patient NEMO_024 performing Task 17.

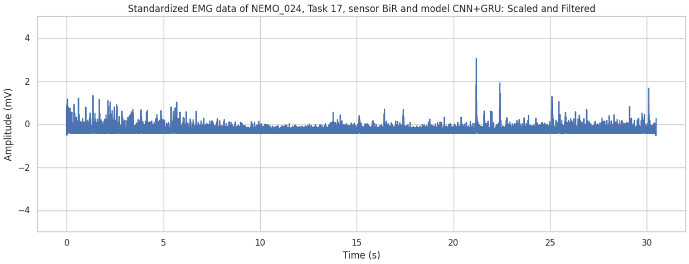


Figure 18: Filtered and scaled EMG data of patient NEMO_024 performing Task 17.

Figures 19 and 20 display the predicted bursts of both models at a decision threshold of 0.9. The plots show that while the annotated bursts of Figure 17 were present in the prediction plots, additional bursts were also predicted that were not annotated. Moreover, the plots highlight that high peaks in the filtered, rectified, and scaled data were usually marked as bursts.

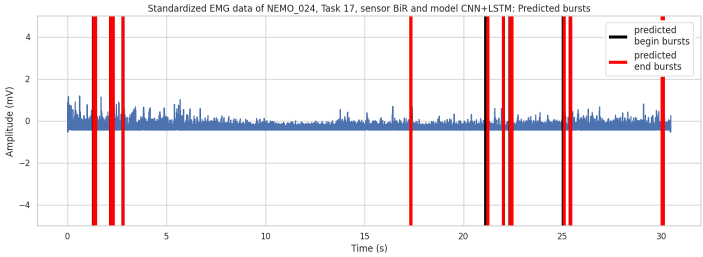


Figure 19: Burst predictions by the CNN+LSTM model on the EMG data of patient NEMO_024 performing Task 17, with decision threshold 0.9.

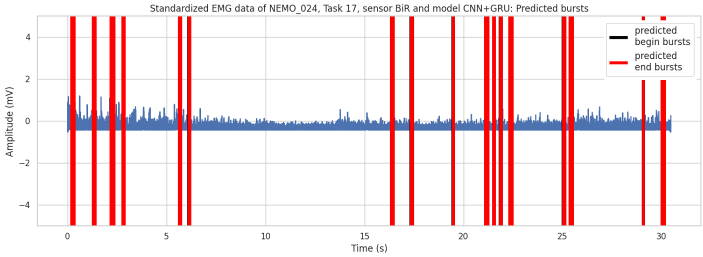


Figure 20: Burst predictions by the CNN+GRU model on the EMG data of patient NEMO_024 performing Task 17, with decision threshold 0.9.

Zoomed-in view on one burst of Patient NEMO_024, capturing bursts.
In Figure 21, I zoomed in on a specific annotated burst with a start time of approximately 25.06 seconds and an end time of approximately 25.12 seconds of patient NEMO_024 performing Task 17. To examine the predictions of both models, Figure 22 shows the results for each model with the three decision thresholds.

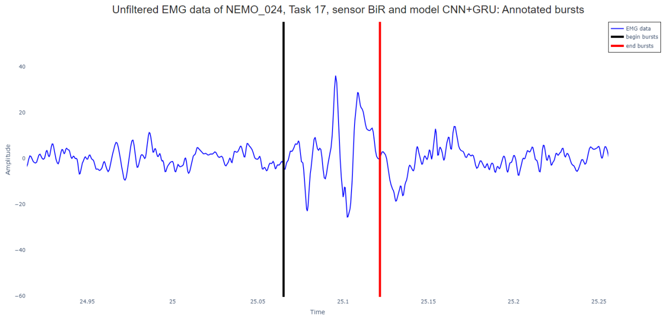
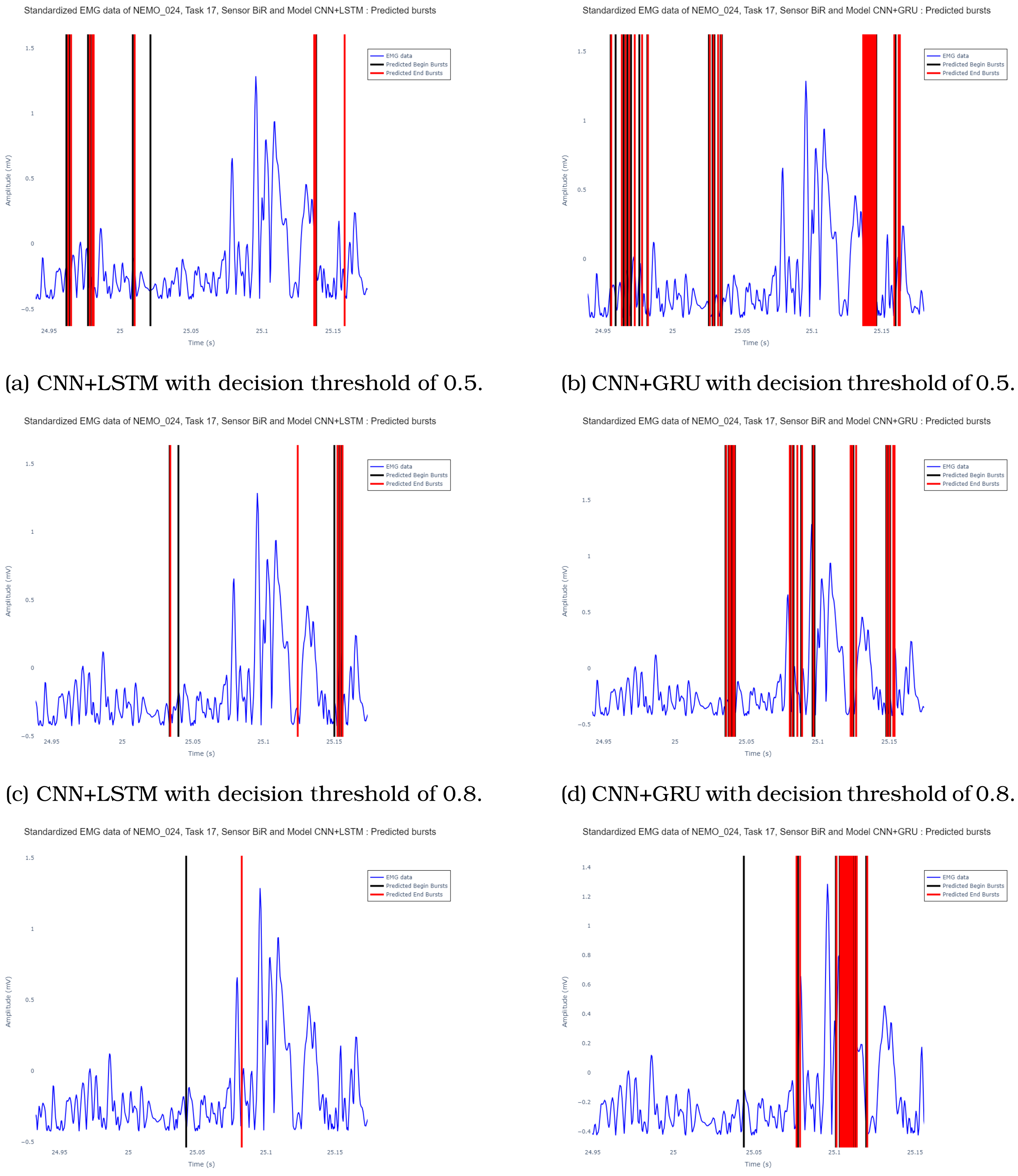


Figure 21: Zoomed-in view of a manually labeled burst in the unfiltered EMG data of patient NEMO_024 performing Task 17. The burst starts at approximately 25.06 seconds and ends at approximately 25.12 seconds.

Figures 22(a), 22(b) and 22(c) illustrate the cases where the predicted bursts fully captured this interval, while Figures 22(d), 22(e) and 22(f) show cases where the predicted bursts overlap with the annotated burst. This demonstrates that both models can predict the annotated burst, either by capturing the entire interval or by overlapping with the it, depending on the chosen decision threshold and model.

Figure 22, also shows that bursts prediction by the models at the lower thresholds usually have longer duration or consist of multiple smaller predicted bursts compared to those at a higher decision thresholds. This is evident from the many red lines in the plots at 0.5 decision thresholds, which is often observed across different patients and tasks.



(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

(c) CNN+LSTM with decision threshold of 0.8.

(d) CNN+GRU with decision threshold of 0.8.

(e) CNN+LSTM with decision threshold of 0.9.

(f) CNN+GRU with decision threshold of 0.9.

Figure 22: Zoomed-in view of predicted bursts with a decision thresholds 0.5 (a-b), 0.8 (c-d) and 0.9 (e-f), in the filtered and scaled EMG data of patient NEMO_024 during Task 17. The predictions, generated by the CNN+LSTM (a,c,e) and CNN+GRU (b,d,f) , are shown in the same time range as Figure 21 (24.95 to 25.15 seconds).

Annotated and Predicted Bursts in EMG data of Patient NEMO_026, no aligned bursts at high decision thresholds

However, there are cases where the annotated bursts were not predicted at a higher decision threshold, but only predicted at lower decision threshold. This can be seen in Figure 25 with the EMG data from patient NEMO_026.

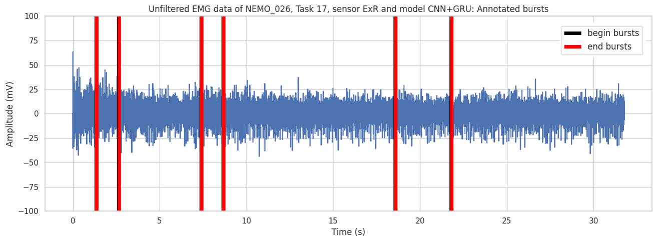


Figure 23: Manually labelled bursts in the unfiltered EMG data of patient NEMO_026 performing Task 17.

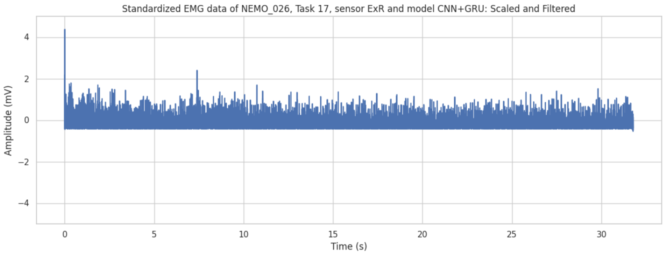
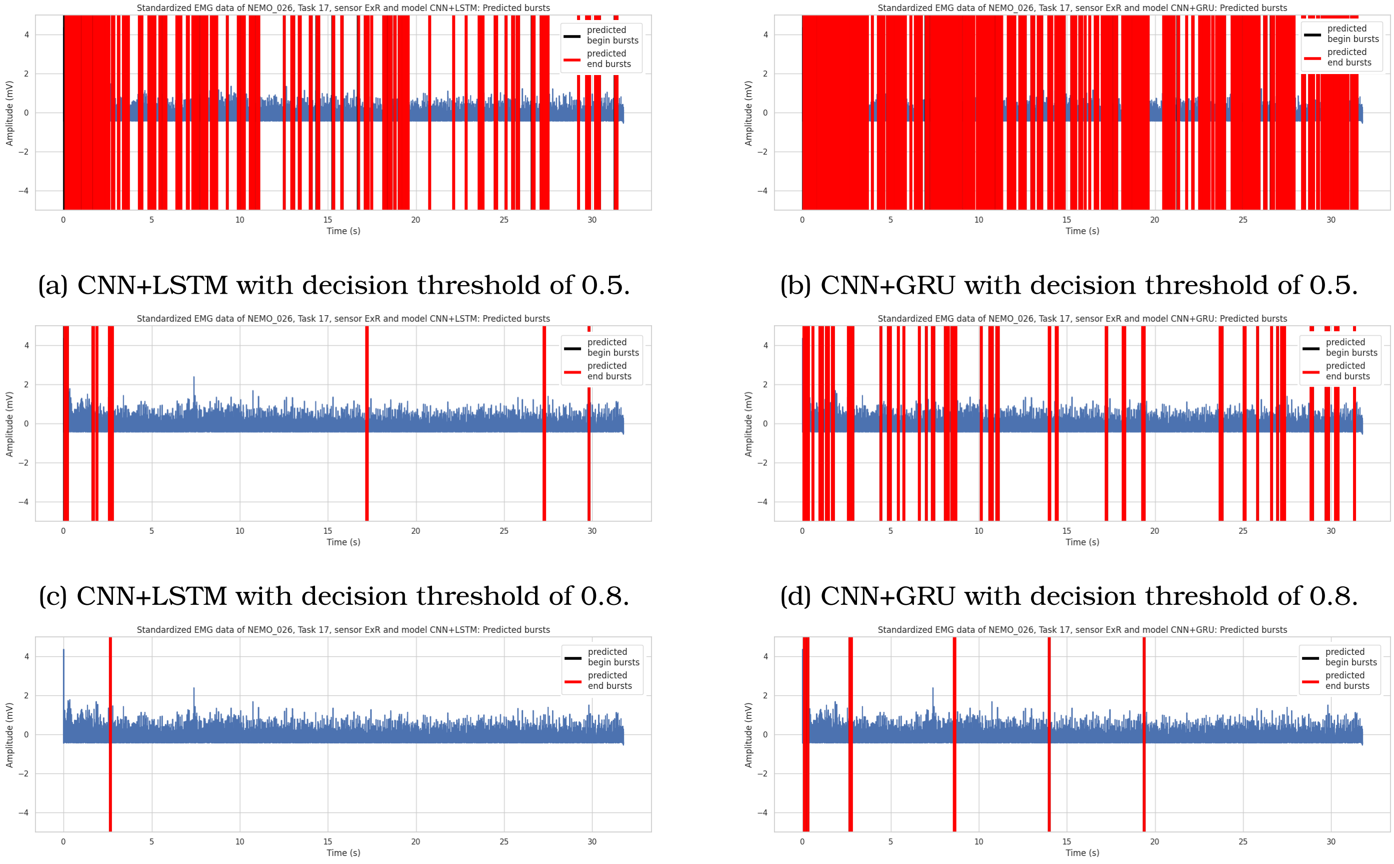


Figure 24: Filtered and scaled EMG data of patient NEMO_026 performing Task 17.



(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

(c) CNN+LSTM with decision threshold of 0.8.

(d) CNN+GRU with decision threshold of 0.8.

(e) CNN+LSTM with decision threshold of 0.9.

(f) CNN+GRU with decision threshold of 0.9.

Figure 25: Burst predictions by the CNN+LSTM (a,c,e) and CNN+GRU (b,d,f) model on the EMG data of patient NEMO_026 performing Task 17, with decision threshold 0.5 (a-b), 0.8 (c-d) and 0.9 (e-f).

Most annotated burst are not predicted by the CNN+LSTM model for threshold 0.8 and 0.9, as seen in Figure 25(c) and Figure 25(e), respectively. Similarly, some annotated bursts are also not predicted by the CNN+GRU model, but predicted more annotated bursts than CNN+LSTM, as shown in Figure 25(d) and Figure 25(d).

Zoomed-in view of bursts of Patient NEMO_026, capturing no bursts or close-by. Additionally, there are cases where bursts are not predicted at all. This can be seen in the zoomed in-view of the predicted burst by the CNN+LSTM in Figure 28(a), where no burst is marked, compared to the zoomed-in view of the manually labelled burst in Figure 26 within the same range. Conversely, the CNN+GRU model does predict a burst in this range, as shown in Figure 28(b).

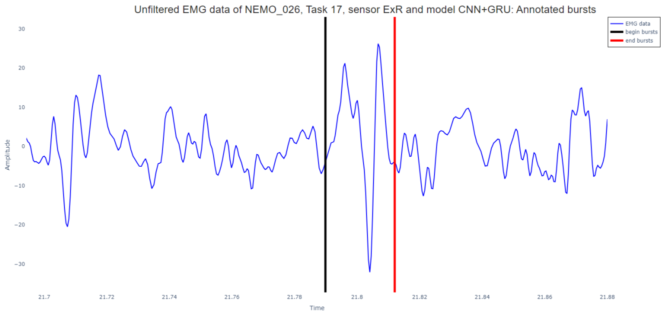


Figure 26: Zoomed-in view of a manually labeled burst in the unfiltered EMG data of patient NEMO_026 performing Task 17. The burst starts at approximately 21.79 seconds and ends at approximately 21.81 seconds.

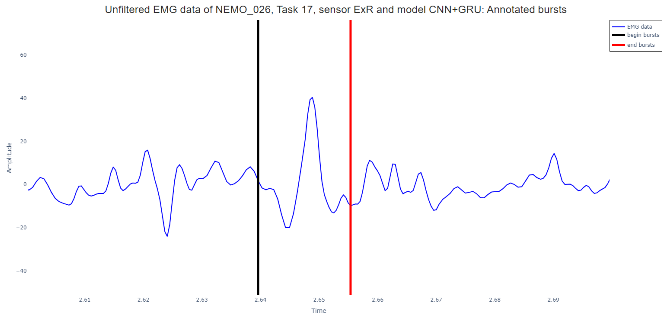
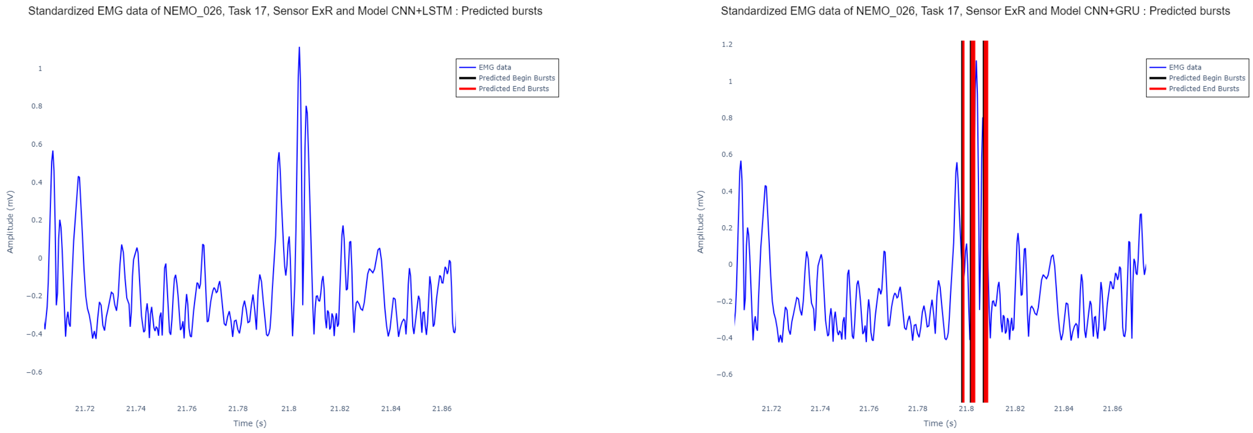


Figure 27: Zoomed-in view of a second manually labeled burst in the unfiltered EMG data of patient NEMO_026 performing Task 17. The burst starts at approximately 2.64 seconds and ends at approximately 2.655 seconds.

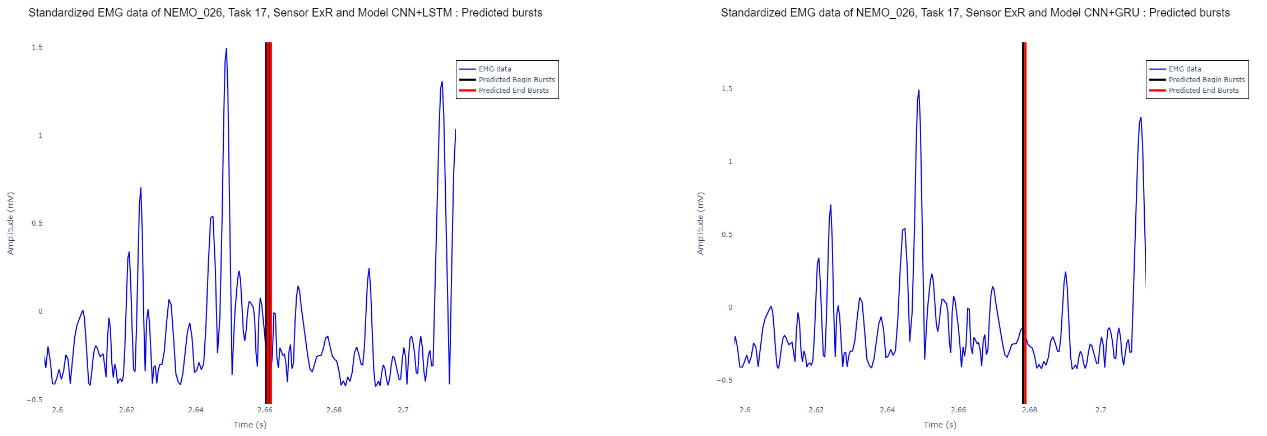


(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

Figure 28: Zoomed-in view of predicted bursts with a decision thresholds 0.5, in the filtered and scaled EMG data of patient NEMO_026 during Task 17. The predictions, generated by the CNN+LSTM (a) and CNN+GRU (b) models, are shown in the same time range as Figure 26 (21.72 to 21.86 seconds).

Moreover, some predicted bursts are close to the annotated burst, but do not capture the annotated burst, which can be seen in Figures 27, 29(a) and 29(b). These cases show that due to the difference in the EMG data, the model also has difficulty determining the bursts and therefore is not able to predict the annotated bursts or predict bursts close to the annotated burst.



(a) CNN+LSTM with decision threshold of 0.9.

(b) CNN+GRU with decision threshold of 0.9.

Figure 29: Zoomed-in view of a predicted bursts with a decision thresholds 0.9, in the filtered and scaled EMG data of patient NEMO_026 during Task 17. The predictions, generated by the CNN+LSTM (a) and CNN+GRU (b) models, are shown in the same time range as Figure 27 (2.6 to 2.7 seconds).

Annotated and Predicted Bursts in EMG data of Patient NEMO_093, peaks in the filtered, rectified and scaled data

When looking at the results of the previous patient, NEMO_026, and compare the filtered, rectified and scaled data without predictions, shown in Figure 24, with the predicted plots of patient NEMO_026, shown in Figure 25, it revealed that outstanding peaks in the scaled data did not always correspond to burst predictions.

However, in other EMG data, they can. During Task 22 with patient NEMO_093, the peaks in the scaled data were clearly visible, as shown in Figure 31, and aligned with the annotated bursts, as shown in Figure 30. In Figure 32, both models marked bursts around these peaks with a decision threshold 0.9.

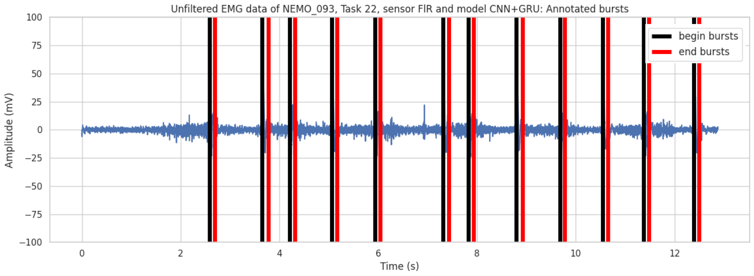


Figure 30: Manually labelled bursts in the unfiltered EMG data of patient NEMO_093 performing Task 22.

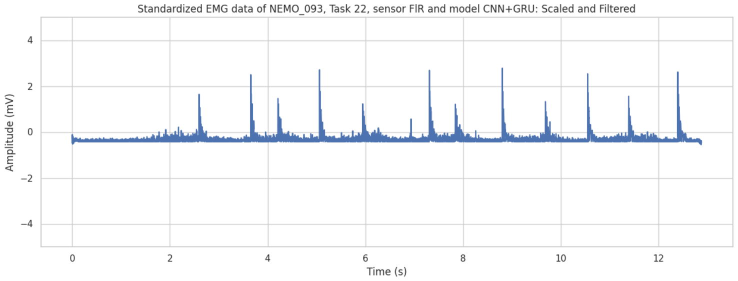
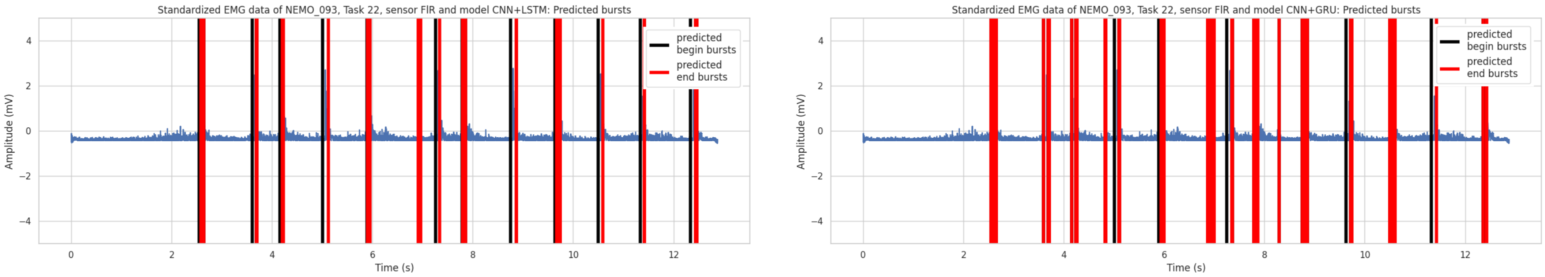


Figure 31: Filtered and scaled EMG data of patient NEMO_093 performing Task 22.



(a) CNN+LSTM with decision threshold of 0.9.

(b) CNN+GRU with decision threshold of 0.9.

Figure 32: Burst predictions by the CNN+LSTM (a) and CNN+GRU (b) models on the EMG data of patient NEMO_093 performing Task 22, with decision threshold 0.9.

Zooming in on two specific notated bursts at 5 and 6 seconds, as shown in Figure 33 and as peaks in Figure 31, revealed that the models predicted these peaked bursts with both high and lower decision thresholds, as illustrated in Figure 34, but also smaller peaks in between for decision threshold 0.8, by model CNN+GRU, and 0.5, by both models. This indicates the models' sensitivity towards peaks in the EMG data and showing high accuracy in predicting annotated bursts.

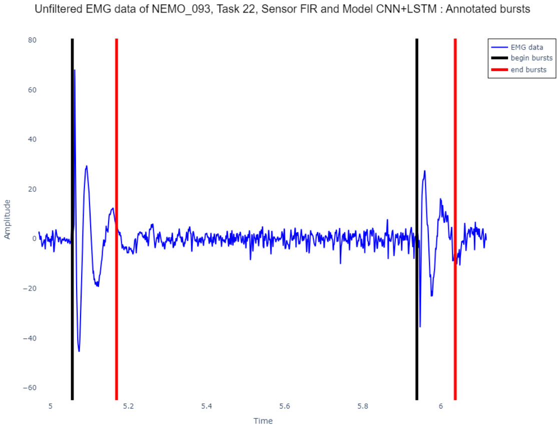
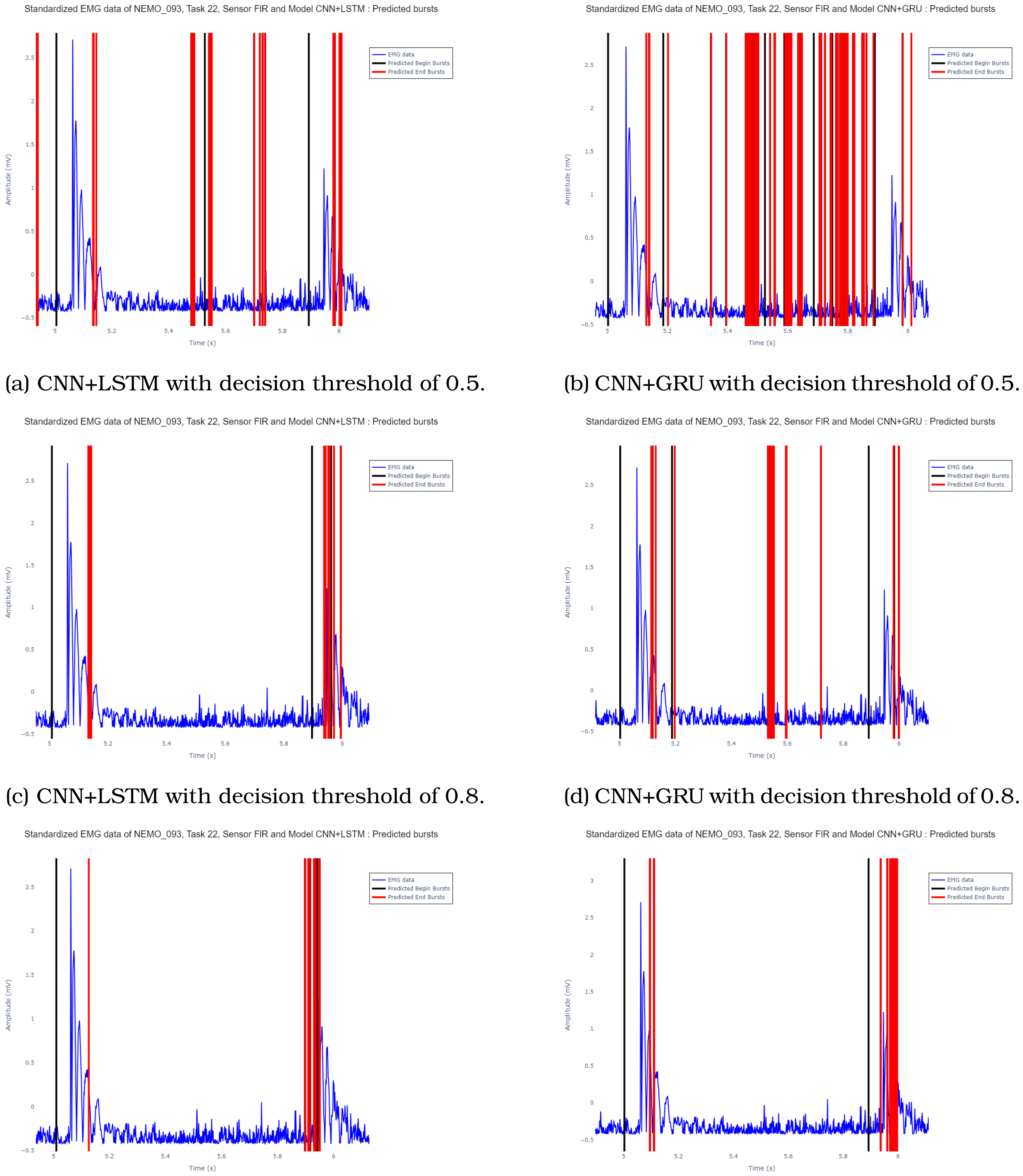


Figure 33: Zoomed-in view of manually labeled bursts in the unfiltered EMG data of patient NEMO_093 performing Task 22. The burst starts at approximately 5.05 and 5.95 seconds and ends at approximately 5.18 and 6.03 seconds.



(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

(c) CNN+LSTM with decision threshold of 0.8.

(d) CNN+GRU with decision threshold of 0.8.

(e) CNN+LSTM with decision threshold of 0.9.

(f) CNN+GRU with decision threshold of 0.9.

Figure 34: Zoomed-in view of predicted bursts with a decision thresholds 0.5 (a-b), 0.8 (c-d) and 0.9 (e-f), in the filtered and scaled EMG data of patient NEMO_093 during Task 22. The predictions, generated by the CNN+LSTM (a,c,e) and CNN+GRU (b,d,f) models, are shown in the same time range as Figure 33 (5.0 to 6.1 seconds).

4.7.2 Healthy Controls

To evaluate the models' performance in predicting 'non-bursts,' I analysed the EMG data from healthy controls, which guaranteed 'non-burst' in all data.

After analysing the predictions plots of the healthy controls across all tasks, I observed that also for these patients there are several distinct cases but are not related to sensors, tasks or patients. The scaled data sometimes showed detectable patterns, though not consistently.

Predicted Bursts in EMG data of Patient NEMO_002, with a high level of muscle activity
In many cases, when there was high level of muscle activity in the EMG data of patients performing one of the tasks, such as those of patient NEMO_002 performing Task 17 in Figure 35, and patient NEMO_003 performing Task 1 in Figure 38, the scaled EMG data also showed numerous peaks, as seen in Figure 36 of patient NEMO_002 and 39 of patient NEMO_003.

As a results, both models predicted bursts that often matched the outstanding peaks at a high decision threshold of 0.9. However, not all such peaks were marked as bursts by both models, as seen in Figures 37 and 40. Smaller peaks were also marked as bursts, indicating that the models did not show a consistent pattern of burst prediction based on peak size.

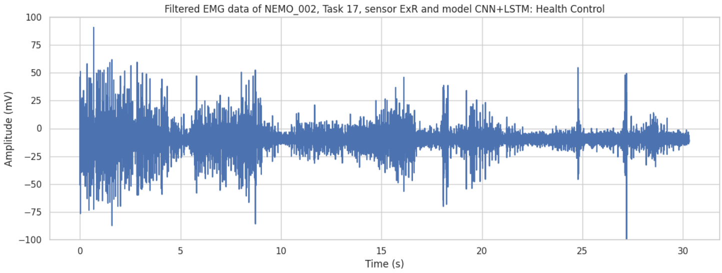


Figure 35: Unfiltered EMG data of patient NEMO_002 performing Task 17.

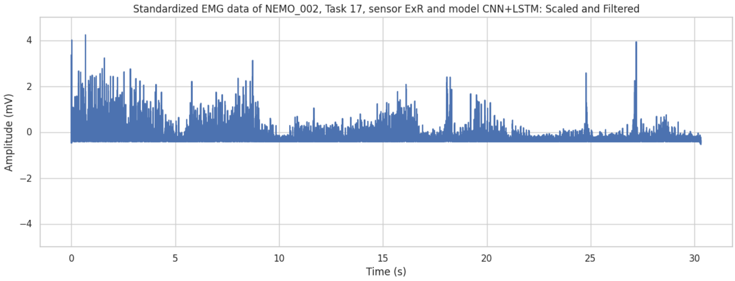
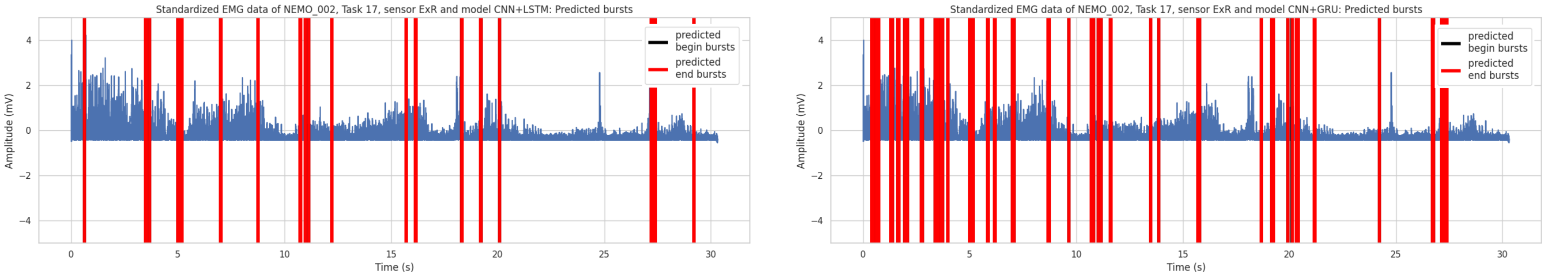


Figure 36: Filtered and scaled EMG data of patient NEMO_002 performing Task 17.



(a) CNN+LSTM with decision threshold of 0.9.

(b) CNN+GRU with decision threshold of 0.9.

Figure 37: Burst predictions by the CNN+LSTM (a) and CNN+GRU (b) models on the EMG data of patient NEMO_002 performing Task 17, with decision threshold 0.9.

Predicted Bursts in EMG data of Patient NEMO_003, high level of muscle activity

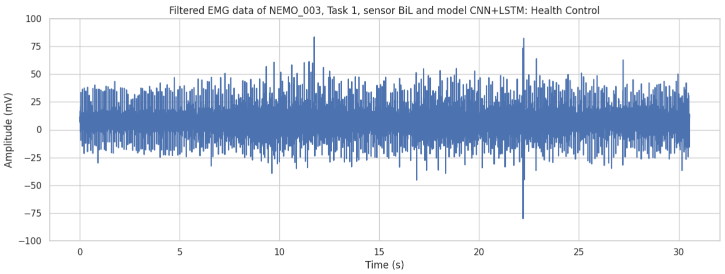


Figure 38: Unfiltered EMG data of patient NEMO_003 performing Task 1.

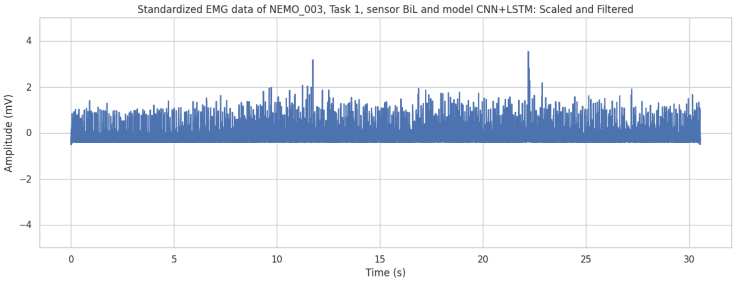
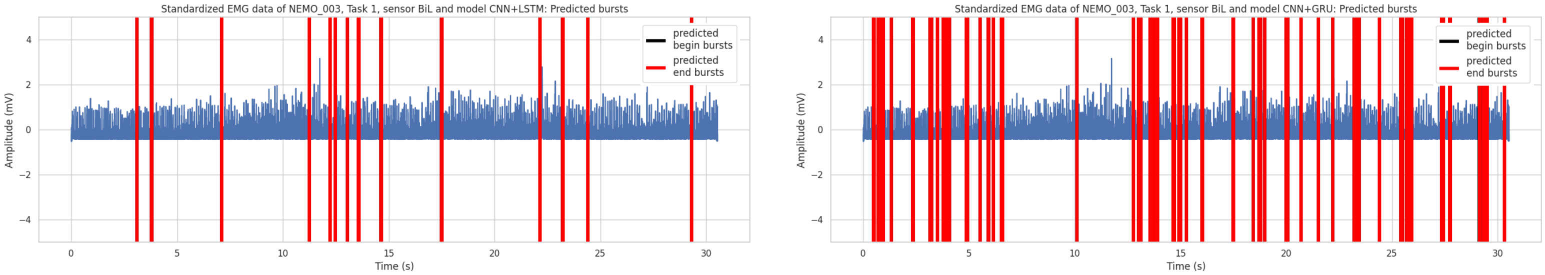


Figure 39: Filtered and scaled EMG data of patient NEMO_003 performing Task 1.



(a) CNN+LSTM with decision threshold of 0.9.

(b) CNN+GRU with decision threshold of 0.9.

Figure 40: Burst predictions by the CNN+LSTM (a) and CNN+GRU (b) models on the EMG data of patient NEMO_003 performing Task 1, with decision threshold 0.9.

Predicted Bursts in EMG data of Patient NEMO_008, distinct peak
Another case involved consistent EMG data with occasional distinct peaks, which were also visible in the scaled EMG data. The data of patient NEMO_008 performing Task 23 showed this in Figures 41 and 42. Both models predicted this as burst with at decision threshold of 0.9 or 0.8, indicating the sensitivity to peaks, as seen in Figure 43.

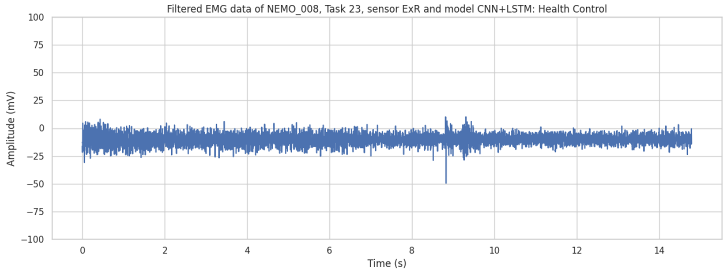


Figure 41: Unfiltered EMG data of patient NEMO_008 performing Task 23.

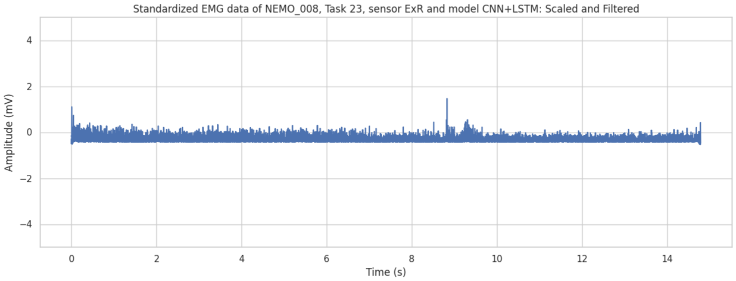
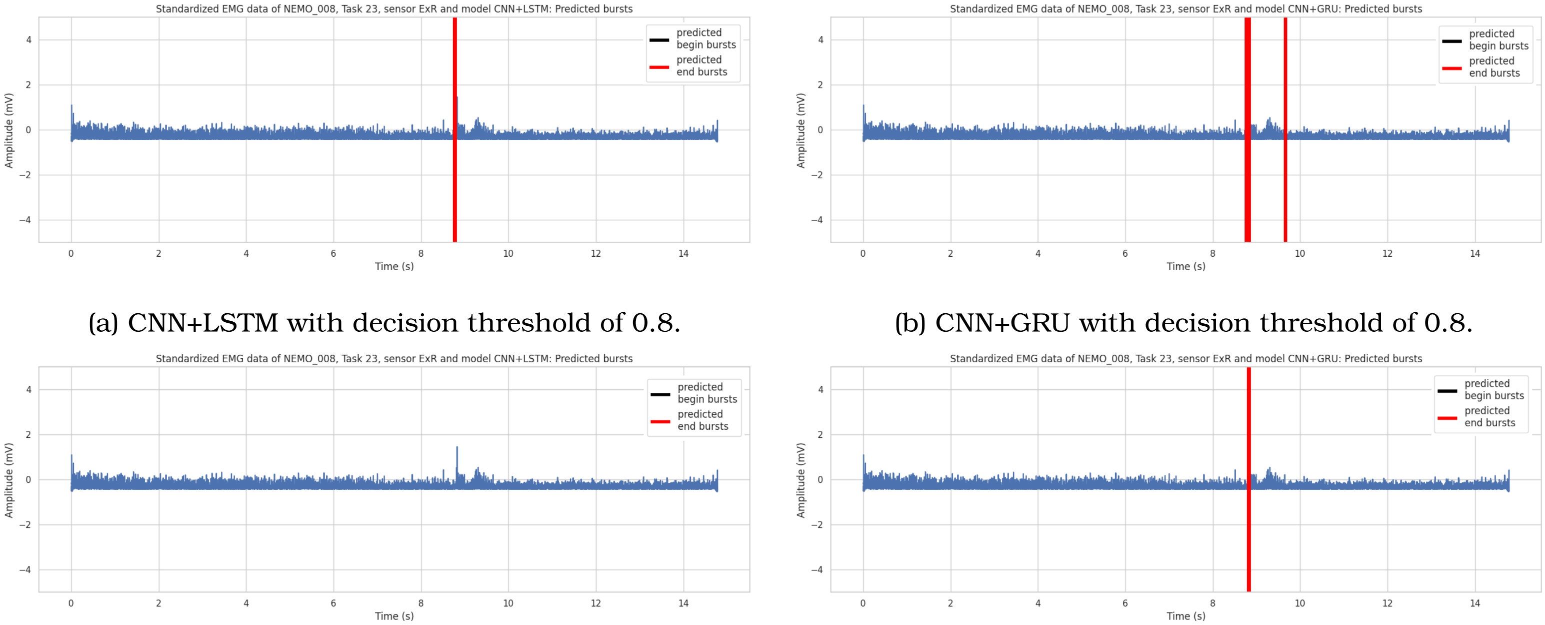


Figure 42: Filtered and scaled EMG data of patient NEMO_008 performing Task 23.

Predictions.



(a) CNN+LSTM with decision threshold of 0.8.

(b) CNN+GRU with decision threshold of 0.8.

(c) CNN+LSTM with decision threshold of 0.9.

(d) CNN+GRU with decision threshold of 0.9.

Figure 43: Burst predictions by the CNN+LSTM (a,c) and CNN+GRU (b,d) models on the EMG data of patient NEMO_008 performing Task 23, with decision threshold 0.8 (a-b) and 0.9 (c-d).

Predicted Bursts in EMG data of Patient NEMO_005, no burst at higher decision thresholds Nonetheless, there were cases where no bursts were predicted in the scaled EMG data with consistent peaks at the higher thresholds 0.9 and 0.8. This is illustrated in Figure 46, showing patient 005 performing Task 22, with unfiltered data presented in Figure 44 and filtered and scaled data in Figure 45. This demonstrates the models' ability to correctly detect non-bursts at higher decision thresholds.

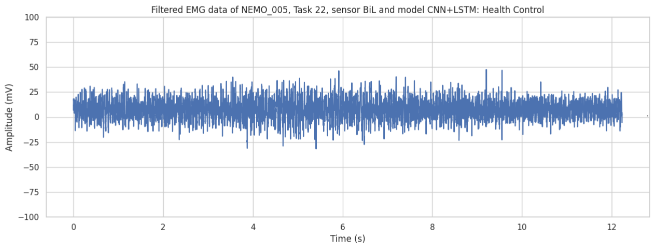


Figure 44: Unfiltered EMG data of patient NEMO_005 performing Task 22.

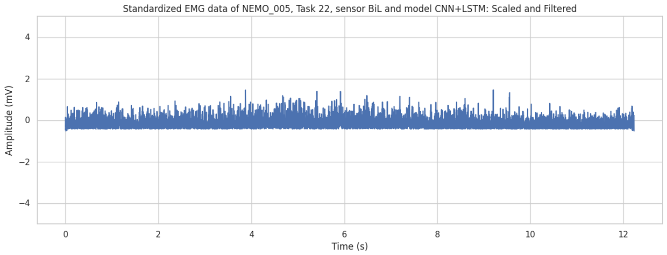
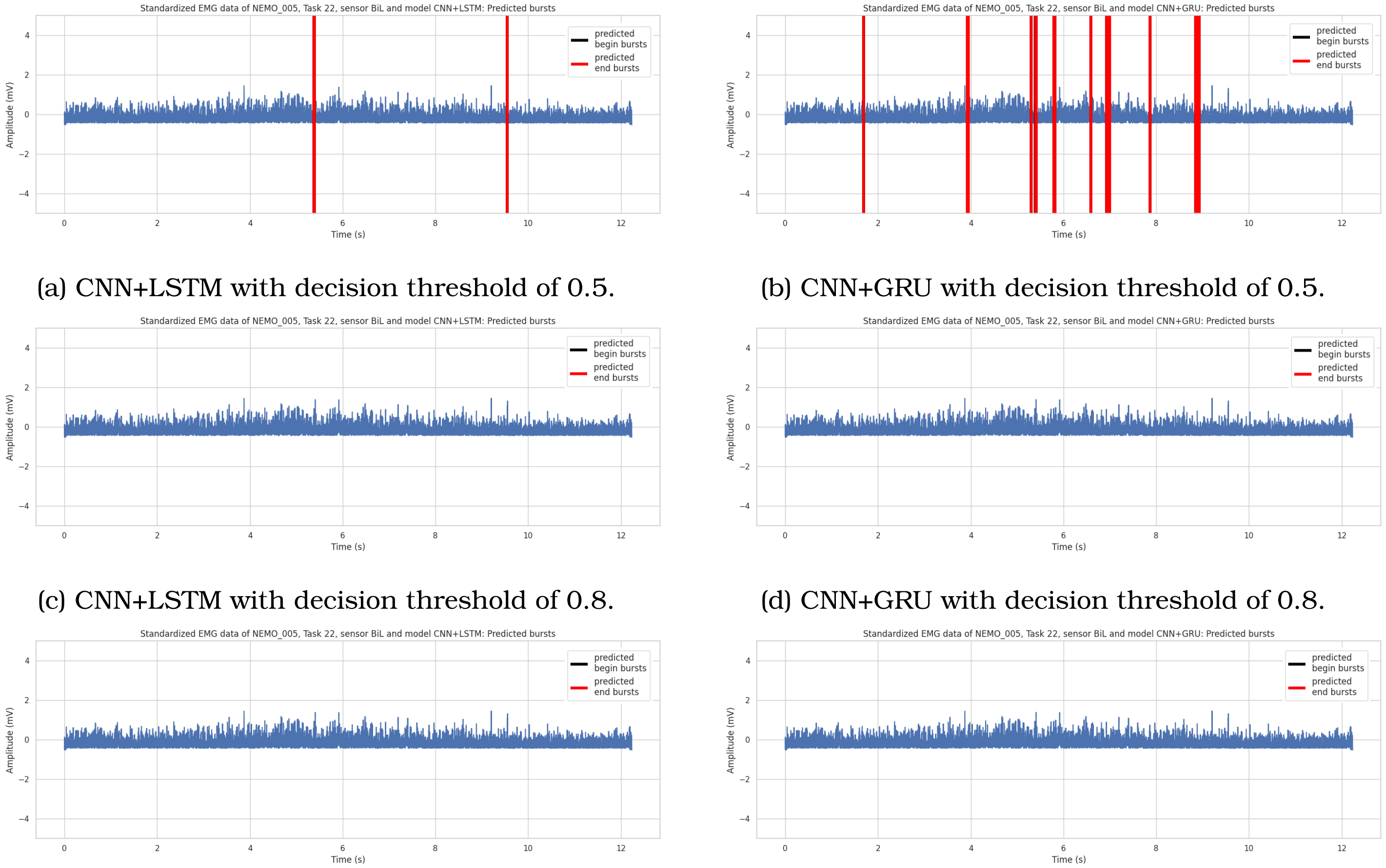


Figure 45: Filtered and scaled EMG data of patient NEMO_005 performing Task 22.



(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

(c) CNN+LSTM with decision threshold of 0.8.

(d) CNN+GRU with decision threshold of 0.8.

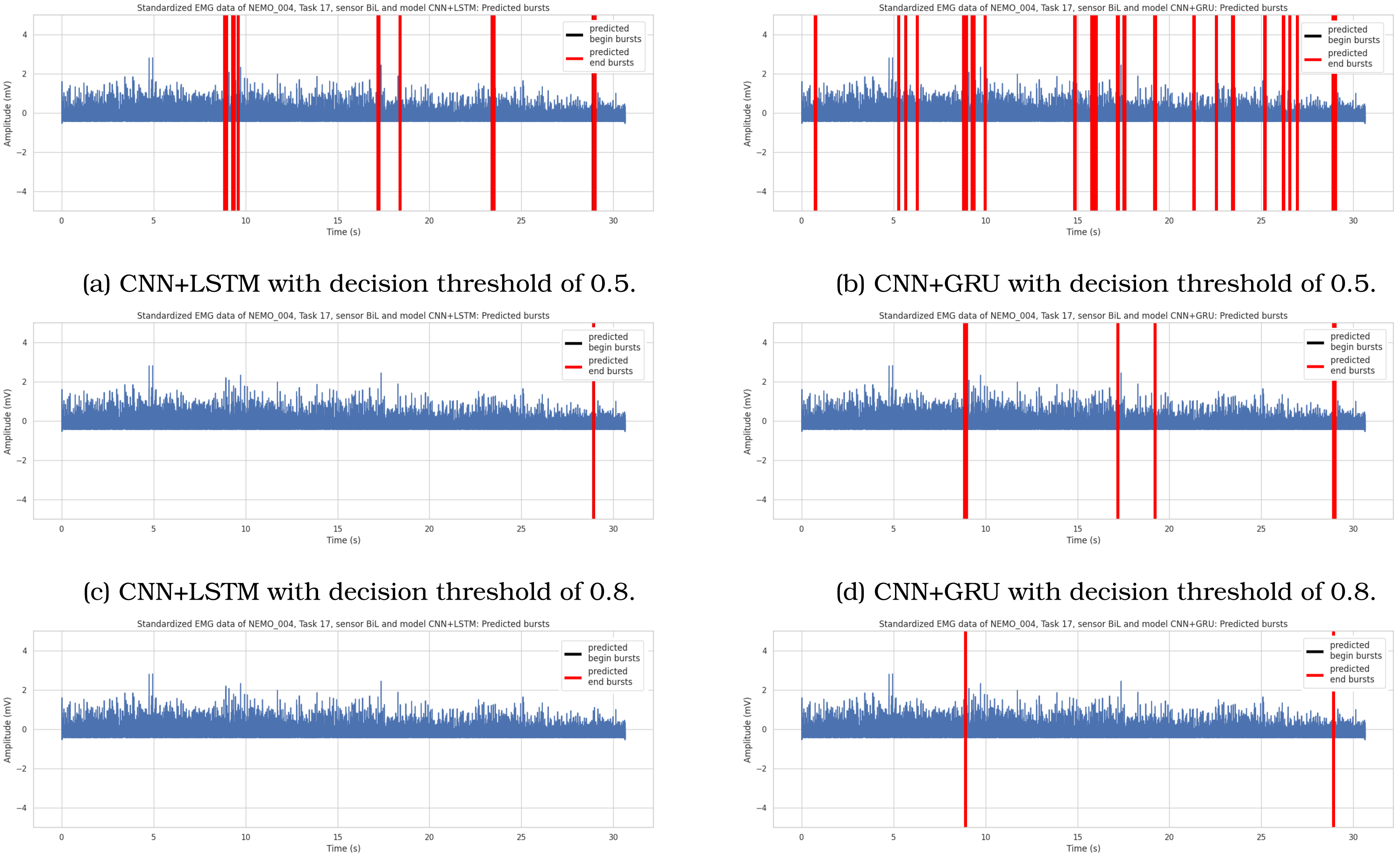
(e) CNN+LSTM with decision threshold of 0.9.

(f) CNN+GRU with decision threshold of 0.9.

Figure 46: Burst predictions by the CNN+LSTM (a,c,e) and CNN+GRU (b,d,f) models on the EMG data of patient NEMO_005 performing Task 22, with decision threshold 0.5, (a-b) 0.8 (c-d) and 0.9 (e-f).

Predicted Bursts in EMG data of Patient NEMO_004, number of bursts predictions from CNN+LSTM and CNN+GRU models

Figure 47 presents the burst predictions for patient NEMO_004 performing Task 1. At the same decision threshold, the CNN+LSTM model predicted fewer bursts compared to the CNN+GRU model. This observation is in agreement with the confusion matrices in Figures 15 and 16, which showed the prediction performance of both models on samples of the validation and test sets.



(a) CNN+LSTM with decision threshold of 0.5.

(b) CNN+GRU with decision threshold of 0.5.

(c) CNN+LSTM with decision threshold of 0.8.

(d) CNN+GRU with decision threshold of 0.8.

(e) CNN+LSTM with decision threshold of 0.9.

(f) CNN+GRU with decision threshold of 0.9.

Figure 47: Burst predictions by the CNN+LSTM (a,c,e) and CNN+GRU (b,d,f) models on the EMG data of patient NEMO_004 performing Task 17, with decision threshold 0.5, (a-b) 0.8 (c-d) and 0.9 (e-f).

5 Discussion

The goal of this study is to improve the previous study [1] in the detection of bursts in EMG data of myoclonus patients. After reviewing the findings of the previous work and exploring the use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in EMG data in other fields, I decided to use hybrid models of CNN with long short-term memory (LSTM) and CNN with gated recurrent unit (GRU) for burst detection in myoclonus. Through testing various models with different parameter settings and comparing the evaluation metrics, loss and accuracy plots, I obtained the optimal models of both architectures. These models were then applied to unseen EMG data from myoclonus patients and healthy control group.

This section discusses the main findings from the evaluation of the optimal models with the training, validation and test set. Additionally, I will discuss the strengths and weaknesses of the optimal models based on their performance with unseen EMG data from both myoclonus patients and healthy controls. Finally, I will discuss the changes made in this study compared to the previous one.

5.1 The evaluation on the training, validation and training set

The high values of the evaluation metrics in the test set in Table 2 show that the prediction performance of both models indicate robust performance in accurately predicting the true labels and positive samples, which are the 'burst' samples. Specifically, the CNN+LSTM model achieved an accuracy of approximately 88% for predicting the true labels in the test set, while the CNN+GRU model achieved an accuracy of approximately 89%..

As mentioned in Section 7, the previous study achieved an accuracy of 76% on a different test dataset. Therefore, a direct comparison of these accuracy metrics is not possible.

The confusion matrices of the unseen data from both the validation and test set show that both models performed well in predicting true labels. Specifically, the CNN+LSTM predicted approximately 87% True Negativesv(TN) and 92% True Positives (TP) over the two sets. While the CNN+GRU predicted approximately 85% TN and 95% TP over the two sets. This indicates that both models are more effective at predicting True Positives, or 'burst' samples, compared to True Negatives, or 'non-burst' samples.

Furthermore, the confusion matrices reveals that the CNN+LSTM performs slightly better at prediction the 'non-burst' samples, whereas the CNN+GRU models performs slightly better in predicting the 'burst' samples.

5.2 Predictions in EMG data of Myoclonus patients

Before discussing the results of the predictions in the EMG data of myoclonus patients, it is important to mention that the manually labelled data from NEMO should not be considered ground truth. Although bursts are labelled in data, this labelling does not guarantee that all bursts are identified or that the intervals between labelled bursts can be classified as 'non-burst'. Therefore, the data was not used for 'non-burst' samples. Consequently, when evaluating the models' predictive performance, 'burst' predictions that do not match the annotated bursts in the NEMO data should not be classified as False Positives.

Due to this absence of ground truth, this research cannot compare the prediction performances between the CNN+LSTM and CNN+GRU models on the 'burst' data. Similarly, it is not possible to compare the prediction performance across different decision thresholds or determine the optimal thresholds.

The predictions on the EMG data from the myoclonus patients showed potential in burst detection but also needs improvements. Both the CNN+LSTM and CNN+GRU models showed strengths and weaknesses in their performance.

One strength of the models is their ability to predict the annotated bursts in the EMG data. The models were able to make accurate predictions, sometimes at higher decision thresholds and other times only at lower thresholds. Another strength is their detection around peaks in the EMG data. This was seen in the results of patient NEMO_093 in Figures 30 through 34 where the models accurately predicted bursts at the prominent peaks, which were also manually labeled.

However, the both models also showed weakness. Both models sometimes detected bursts close to the annotated bursts or failed to predict a annotated bursts at a decision threshold of 0.5 or higher. Therefore, the biggest weakness of the models is the inconsistency when it comes to predict burst in EMG data. This inconsistency may be due to the models' sensitivity towards outstanding peak or prominent peak behaviour, but also to the wide variability in muscle activity in patients, making it difficult to determine bursts.

There are also debatable points regarding the models' performance. One such point is the detection of multiple bursts that are not annotated. It is possible that these bursts were undetected by manual labelling, which could indicate a strength of the models for predicting multiple bursts. However, there is also the possibility that many of these predictions are False Positives, which indicate a weakness of both models. Therefore, without the ground truth, this point is debatable.

Another debatable point is the models' sensitivity to smaller peaks in the EMG data. By the absence of the ground truth, it is unclear whether these small peaks represent small myoclonus bursts or are False Positives.

5.3 Predictions in EMG data of Healthy controls

The data from healthy controls guarantees 'non-burst' data, allowing this research to evaluate the models' performance on 'non-burst' predictions. Both the CNN+LSTM and CNN+GRU models displayed strengths and weaknesses in their predictions on EMG data from healthy controls.

One strength observed in the results of the models, is predicting no bursts with higher decision thresholds when the muscle activity is consistent. Additionally, the results show that distinct peaks in the data are predicted as burst. This indicates the models' sensitivity to distinct peaks, which is a strength for the detection of bursts in the EMG data.

However, this sensitivity can also be a weakness of the models, as it leads to False Positives when these peaks suck peaks occur in 'non-burst' data, and possibly in 'burst' data.

Additionally, the results of the 'non-burst' data also showed that when the data contains high level of muscle activity, both models predicted burst around the high peaks, including the smaller ones. This is seen with patient NEMO_002 and NEMO_003 in Figures 35 through Figure 40. The reason for these results may be because the models are trained to detect outstanding peaks or prominent peak behaviour in the EMG data. This leads to sensitivity to these peaks in the 'non-burst' data and results in 'burst' predictions at these peaks rather than 'non-burst', even at a higher decision threshold.

Another cause could be the difference in EMG data of the healthy controls, which show less spiked data but also data with prominent spikes. Therefore, the proposal to train with more 'non-burst' data to better distinguish 'burst' peaks from 'non-burst' peaks could help with these shortcomings.

The comparisons between the results of the CNN+LSTM and the CNN+GRU models show that the CNN+LSTM model predicts fewer bursts than the CNN+GRU, and thus performs better in predicting 'non-burst' data. This was also demonstrate in the confusion matrices of the three sets, see Figures 14, 15 and 16.

Lastly, a higher decision threshold showcase fewer bursts, suggesting a high decision threshold is the optimal threshold for 'non-burst' data. However, due to the absence of ground truth, this research cannot determine the optimal decision threshold for distinguishing between 'burst' and 'non-burst' data. Therefore, this research is unable to select the optimal threshold based on the 'non-burst' and 'burst' data.

Both the predictions on EMG data of the myoclonus patient and healthy controls highlight the potential of using CNNs with LSTM and GRU architectures for the detection of bursts. While the results show significant strengths, they also reveal weakness and raise debatable points that needs to be examined in the future.

5.4 Changes from the previous study.

Several important modifications are made in this research compared to the previous study. One major change was the inclusion of EMG data from healthy controls for the 'non-burst' sampling. This makes the data set more reliable but can make the models a bit more biased towards 'non-burst' data from healthy controls, potentially making it more challenging to predict 'non-burst' data of myoclonus patients. Due to the absence of ground truth this cannot be tested.

Another change is the division of patient data. This research applies the 60-20-20 split between patients for the training, validation, and test, respectively. Patients with the movement disorders are categorized into the two types and each type divided with the 60-20-20 division. Similarly, the health group is split into the 60-20-20 division. This division differs from the previous study, which divided the augmented burst samples over the three datasets.

Additionally, the datasets are expanded with EMG data from all six sensors. Although not all patients had data from each sensor, including data from all six sensors increases the total number of samples.

Finally, this research also includes the use of different scalers, or none, in the preprocessing step to evaluate their effectiveness on the model performance.

6 Conclusion

This follow up study examined hybrids models combining convolutional neural network (CNN) with long short-term dependencies (LSTM) and gated recurrent unit (GRU) for the detection of myoclonus bursts.

Both models showed the ability to predict bursts that aligned with the annotated bursts in the EMG data of myoclonus patient, as well as predicting bursts that were not annotated. Due to the absence of ground truth, it could not be determined if these additional bursts were False Positives and should be investigated in the future.

The models also predicted bursts in the EMG data of healthy controls, which only contains 'non-bursts'. The cause of these mistakes probably lies in the models' sensitivity to outstanding peaks and prominent peak behaviour.

Additionally, due to the absence of ground truth EMG data, it was not possible to definitively compare the CNN+LSTM and CNN+GRU models to determine the optimal model for burst detection. However, for non-burst detection the CNN+LSTM model performed better than the CNN+GRU model.

The research question was:
Can the detection algorithm be improved with respect to implementation, performance and robustness?

During this research, significant improvements were made in data usage. The inclusion of the healthy control group for "non-burst" samples and the distribution of patient data across training, validation and testing made the data more reliable, improving the overall quality of training and testing for burst detection models.

When evaluating the performance of both models on the training, validation and testing set, the results showed higher values for the evaluation metrics compared to the previous model, which was tested on a different dataset. Due to this and the absence of ground truth, this research can not conclude that the obtained models are improvements of the previous model.

While there have been made improvements and the models show potential in the burst detection myoclonus, there remains room for further improvement.

7 Future Work

Although this research has shown significant improvements and valuable applications of the model, there are several opportunities for further improvements and developments.

Improvements
The annotations of the non-burst data from patients with the movement disorder suitable for 256-window samples was very limited. This study primarily used non-burst data from the healthy control group. One approach to further enhance the model's accuracy is to train and test using more non-burst data from patients with the disorder.

For the preprocessing of data, the study [13] also applied a wavelet denoising filter after the Butterworth filter, which contributed to improved signal clarity. Their models achieved high accuracy (approximately 98%). Further denoising the EMG data could potentially enhance the model's ability to detect burst with even greater accuracy.

In this research, fixed parameters were used due to time constraints. For future work, a new model can be trained to determine if these fixed parameters affect model's performance, or if the chosen parameter values already provide the optimal result.

Another approach to potentially achieve even higher accuracy is to train and test a model with different window sizes. This research used a window size of 256 datapoints based on the findings of the previous study. To further this study, models can be created with different window sizes to explore their impact on performance.

Lastly, expanding the dataset to include data from the other tasks (1,22 and 23), and potentially other tasks, could further improve the training and testing the models.

Application
To extend this research, the model can be applied for burst detection in EMG data from other movement diseases.

Follow-up Study
A follow-up study could focus on the next stage of the problem: classifying the labelled burst data into the subtypes of myoclonus and its neural origin (cortical or subcortical) by training classifiers.

Ground truth
Finally, most important for future work is to acquire accurate ground truth data. This provides a more precise comparison between the actual bursts and the predicted bursts, allowing better evaluation of the models

References

[1] Thomas Heerdink. Automatic detection of myoclonus bursts in emg data. 2023.

[2] Sterre van der Veen, Amber Maliepaard, A.M. Madelein van der Stouwe, Jelle Dalenberg, Inge Tuitert, Jan Willem J. Elting, and Marina A.J. Tijssen. Validating an age-old hypothesis: Substantiating the short burst duration in cortical myoclonus. *Movement Disorders*, 2024.

[3] John N. Caviness. Myoclonus. *Continuum (Minneap. Minn.)*, 25(4):1055–1080, aug 2019.

[4] Kojovic Maja, Carla Cordivari, and Kailash Bhatia. Review: Myoclonic disorders: A practical approach for diagnosis and treatment. *Therapeutic advances in neurological disorders*, 4:47–62, 01 2011.

[5] Sterre van der Veen, John N. Caviness, Yasmine E.M. Dreissen, Christos Ganos, Abubaker Ibrahim, Johannes H.T.M. Koelman, Ambra Stefani, and Marina A.J. Tijssen. Myoclonus and other jerky movement disorders. *Clinical neurophysiology practice*, 7:285–316, oct 2022.

[6] John N. Caviness. Treatment of myoclonus. *Neurotherapeutics*, 11(1):188–200, jan 2014.

[7] Hatice Tankisi, David Burke, Liying Cui, Mamede de Carvalho, Satoshi Kuwabara, Sanjeev D. Nandedkar, Seward B. Rutkove, Erik Stålberg, Michel J.A.M. van Putten, and Anders Fuglsang-Frederiksen. Standards of instrumentation of emg. *Clinical Neurophysiology*, 131:243–258, 2019.

[8] Wikipedia. Electromyography. Last accessed 5 July 2024.

[9] Tom Mitchell. *Machine learning*. McGraw-Hill Science/Engineering/Math, 1997.

[10] Ying Wei, Jun Zhou, Yin Wang, Yinggang Liu, Qingsong Liu, Jiansheng Luo, Chao Wang, Fengbo Ren, and Li Huang. A review of algorithm & hardware design for ai-based biomedical applications. *IEEE Transactions on Biomedical Circuits and Systems*, 14:145–163, 2020.

[11] Usman Rashid, Imran Khan Niazi, Nada Signal, Dario Farina, and Denise Taylor. Optimal automatic detection of muscle activation intervals. *Journal of Electromyography and Kinesiology*, 48:103–111, 2019.

[12] Marco Ghislieri, Giacinto Luigi Cerone, Marco Knaflitz, and Valentina Agostini. Long short-term memory (LSTM) recurrent neural network for muscle activity detection. *Journal of NeuroEngineering and Rehabilitation*, 18(1):153, oct 2021.

[13] Ankit Vijayvargiya, Bharat Singh, Nidhi Kumari, and Rajesh Kumar. semg-based deep learning framework for the automatic detection of knee abnormality. *Signal, Image and Video Processing*, 17:1087–1095, 2022.

[14] A.M. Madelein van der Stouwe, Inge Tuitert, Ioannis Giotis, Joost Calon, Rahul Gannamani, Jelle R. Dalenberg, Sterre van der Veen, Marrit R. Klamer, Alex C. Telea, and Marina A.J. Tijssen. Next move in movement disorders (NEMO): developing a computer-aided classification tool for hyperkinetic movement disorders. *BMJ Open*, 11(10):e055068, oct 2021.

[15] Marcos Aviles, José Manuel Alvarez-Alvarado, Jose-Billerman Robles-Ocampo, Perla Yazmín Sevilla-Camacho, and Juvenal Rodríguez-Reséndiz. Optimizing rnns for emg signal classification: A novel strategy using grey wolf optimization. *Bioengineering*, (1), 2024.

[16] Alejandro Toro-Ossaba, Juan Jaramillo-Tigreros, Juan Tejada, Alejandro Peña, Alexandro López-González, and Rui Alexandre Castanho. Lstm recurrent neural network for hand gesture recognition using emg signals. *Applied Sciences*, 12, 09 2022.

[17] Anton Vasiliev and Alexey Melnikov. Application lstm neural networks for biological signal classification. In *2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 750–755, 11 2022.

[18] Mai H. Abdelaziz, Wael A Mohamed, and Ayman S. Selmy. Hand gesture recognition based on electromyography signals and deep learning techniques. *Journal of Advances in Information Technology*, 2024.

[19] Lorena Isabel Barona López, Francis M. Ferri, Jonathan Zea, Ángel Leonardo Valdivieso Caraguay, and Marco E. Benalcázar. Cnn-lstm and post-processing for emg-based hand gesture recognition. *Intelligent Systems with Applications*, 22:200352, 2024.

[20] Yu Hu, Yongkang Wong, Wentao Wei, Yu Du, M. Kankanhalli, and Wei dong Geng. A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition. *PLoS ONE*, 13, 2018.

[21] Esteban Velásquez, Jan Cornelis, Lubos Omelina, and Bart Jansen. Muscle classification via hybrid cnn-lstm architecture from surface emg signals. In *2023 24th International Conference on Digital Signal Processing (DSP)*, pages 1–5, 2023.

[22] Duanyuan Bai, Dong Zhang, Yongheng Zhang, Yingjie Shi, and Tingyi Wu. Gesture recognition of semg signals based on cnn-gru network. *Journal of Physics: Conference Series*, 2637, 2023.

[23] Marcos Aviles, José Manuel Alvarez-Alvarado, Jose-Billerman Robles-Ocampo, Perla Yazmín Sevilla-Camacho, and Juvenal Rodríguez-Reséndiz. Optimizing rnns for emg signal classification: A novel strategy using grey wolf optimization. *Bioengineering*, 11(1), 2024.

[24] Carlo J. De Luca, L. Donald Gilmore, Mikhail Kuznetsov, and Serge H. Roy. Filtering the surface emg signal: Movement artifact and baseline noise contamination. *Journal of biomechanics*, 43 8:1573–9, 2010.

[25] L Myers, Madeleine Lowery, Mark O'Malley, C Vaughan, Conor Heneghan, A Gibson, Yolande Harley, and R Sreenivasan. Rectification and non-linear pre-processing of emg signals for cortico-muscular analysis. *Journal of neuroscience methods*, 124:157–65, 05 2003.

[26] Elisha Blessing and Hubert Klaus. Normalization and standardization: Methods to preprocess data to have consistent scales and distributions. 12 2023.

[27] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[28] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv*, abs/1511.08458, 2015.

[29] Medium. Convolutional neural networks, explained. Last accessed 2 July 2024.

[30] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[31] IBM. What are recurrent neural networks? Last accessed 30 June 2024.

[32] Barak Or. The exploding and vanishing gradients problem in time series. Last accessed 29 July 2024.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[34] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000.

[35] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, abs/1412.3555, 2014.

[36] Medium. Understanding gated recurrent unit (gru) in deep learning. Last accessed 3 July 2024.

[37] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2020.

[38] Purva Huilgol. Precision and recall in machine learning. Last accessed 10 July 2024.

[39] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.