university of groningen

faculty of science and engineering

# Automatic detection of myoclonus bursts in EMG data

August 18, 2023

Thomas Heerdink s3823997
Bachelor's Project Computing Science
First supervisor: Prof. Dr. Michael Biehl
Second supervisor: Prof. Dr. Kerstin Bunte
Daily supervisor: Elina van den Brandhof
External Supervisor: Dr. Jelle Dalenberg

# Contents

# 1 Introduction

## 1.1 Myoclonus

Myoclonus is a hyperkinetic movement disorder characterised by myoclonic bursts - brief, involuntary muscle contractions [1]. It is normal for everyone to experience involuntary muscle contractions occasionally, but if these bursts occur frequently or interfere with daily activities, it could be a sign of an underlying and potentially serious condition [2]. There are many known causes of myoclonus, including neurological conditions, medications, and genetic factors [2].

Myoclonus can take different forms and affect different body parts in each patient [3]. Because the symptoms of myoclonus bursts can vary so much, there exist multiple methods for classifying these bursts. From a physician's perspective, the most practical way to distinguish myoclonus subtypes is by their physiological origin, as this can provide guidance for treatment options [2]. This means the bursts are classified on where the impulse for the involuntary muscle contraction originates [3]. Usually, this physiological classification is divided into four categories; cortical, subcortical, spinal or peripheral. The classification of myoclonic bursts can be crucial to the diagnosis and with that, the way of treatment of the illness [4]. For example, the way clinicians treat cortical myoclonus is very different from the way clinicians treat subcortical myoclonus [2].

To classify myoclonus bursts, a clinician typically uses surface electromyography (EMG) measurements [3]. EMG is measured by placing small electrodes on the skin overlying the muscles of interest, detecting the electric signals produced by the muscle fibres. Surface EMG provides information about the timing and intensity of muscle activation. Surface EMG data is represented by a time-series plot with the voltage (in millivolts) recorded on the y-axis and the time (in milliseconds) on the x-axis [5].

For burst classification, the clinician first identifies the myoclonus bursts in the data and then classifies the bursts in one of the physiological groups. The amount of data per patient can be extensive, as demonstrated by our own data set, which consists of 36 tasks of 30 seconds per patient, during which surface EMG is measured using 16 sensors attached to the patient. Handling these large data sets can pose significant challenges for human experts. Fortunately, machine learning techniques offer a potential solution to speed up diagnostics.

Machine learning leverages training data to enable algorithms to make predictions based on inputs [6]. With the decreasing cost of computer power, machine learning is becoming more viable and is being applied to an increasingly diverse range of fields. Many successful implementations of machine learning techniques have been demonstrated in problems with similar data sets [7, 8, 9, 10], but not yet for the automatic detection of myoclonic bursts.

The goal of the current study was to make the first step in developing a computer-assisted diagnosis tool that can accurately classify movement disorders using an Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM).

## 1.2 Research questions

Our main research question is: **Can machine learning be used to automatically detect myoclonus bursts?**

To answer this main question we defined the following sub-questions:

- Which machine learning model is best suited for detecting myoclonus bursts automatically?
- How can we efficiently train the model using existing labelled data?
- Does data preprocessing help the model's accuracy?
- Which set of parameters yields the best performance from our machine learning model?

# 2 Problem analysis

## 2.1 Data set

### 2.1.1 NEMO

The Next Move in Movement Disorders (NEMO) study is a cross-sectional study at Expertise Centre Movement Disorders Groningen, University Medical Centre Groningen (UMCG) [11], which aims to develop a computer-aided classification tool for hyperkinetic movement disorders.

### 2.1.2 Participants

The dataset used in this study was collected by the NEMO team. It consists of approximately 200 patients diagnosed with movement disorders, selected from the (UMCG) hyperkinetic movement disorders database and recruited at the UMCG outpatient clinic. We use a subset of this dataset consisting of 11 patients with cortical myoclonus and 22 patients with myoclonus-dystonia, which is a disorder that has the characteristics of both myoclonus and dystonia, another movement disorder [12].

### 2.1.3 Procedures

To obtain the EMG data, the NEMO team attached 16 electrodes to the patient's arms, hands, and face (see Figure 1 for the placement of these sensors).

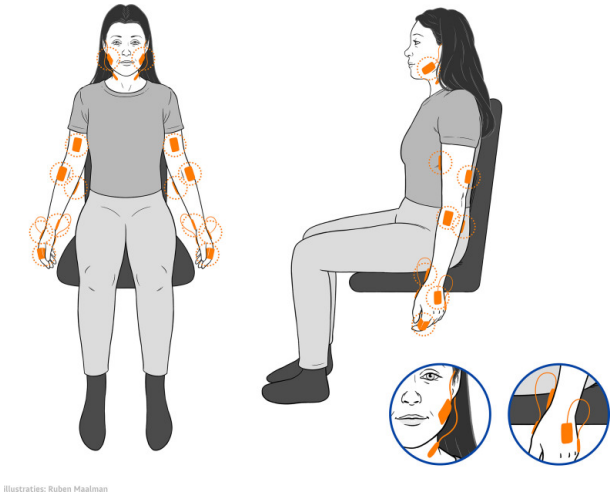



Figure 1: *Sensors used during the tasks*

The patients were instructed to perform a set of 36 distinct tasks, each lasting approximately 30 seconds. These tasks involved various movement exercises selected from the Unified Myoclonus Rating Scale [13]. The electrodes record data at a frequency of approximately 2000 Hz, generating an average of 60,000 data points for each 30-second task. In order to enhance precision, each data point in the dataset is accompanied by a timestamp indicating the precise moment it was recorded.

### 2.1.4 Previous work

The dataset also includes around 3000 myoclonus bursts that have been manually labelled, complete with timestamps. The labelling process specifically focused on four designated tasks (1, 17, 22, and 23) and involved six specific sensors: right biceps, left biceps, right underarm, left underarm, right index finger, and left index finger. These tasks were selected to represent three distinct task types: resting (1), postural (17), and dynamic (22, 23).

Task 1 involved keeping the hands rested on the legs with palms facing upward, task 17 required keeping the hands straight in front of the body with vertical palms, and tasks 22 and 23 required extending the hands sideways and subsequently touching the nose (left arm for task 22 and right arm for task 23).

Labelled myoclonus burst data consisted of a timestamp indicating the onset of the burst and another timestamp indicating its offset. The labelling was performed using video-polymyography, a method that combines video footage of a patient with their corresponding EMG data. An expert goes through this data by hand and manually selects when a burst starts and ends. At the moment this is standard practice when classifying myoclonus [1].

Of all labelled bursts, 24% were identified in rest (task 1), 51% were identified during the postural task (17) and 25% were identified during the dynamic tasks (22 and 23). Since task 17 has the highest number of identified bursts and is the task during which most patients experienced bursts, we will focus on this task for this project.

## 2.2 Problem definition

Our first sub-question was to find which machine learning model is best suited for our problem. To answer that we take a look at the problem itself and at the solutions used in similar problems. Our model is the first step within a multi-step plan focused on the detection and classification of myoclonus bursts. The objective at this stage is to identify where, if any, myoclonus bursts are present in EMG data. Following this, the next step is to classify these bursts into distinct myoclonus types. This second step was not within the scope of the current project.

Previous studies have delved into the automation of diagnoses using EMG data, employing a wide range of machine learning techniques to process and analyze the EMG data. [7]
These techniques include classical machine learning algorithms, such as support vector machines [14] or a double threshold algorithm [8]. In addition to these classical machine learning algorithms, more advanced deep learning models have been proposed to automate the analysis of this type of data [9, 10, 15, 16].

It is important to not only look at similarities in related work but also look at the differences. A double threshold algorithm could possibly be viable if used on resting tasks like task 1. In this resting task, the myoclonus bursts are relatively easy to detect as they are reflected in a clear high voltage signal compared to the rest of the task within the EMG measurement. As you can see in figure 2 and figure 3, the bursts are (relatively) easy to identify by eye. The left and right pictures are the same EMG data, of task 1 and the sensor on the left underarm, with the right picture having the bursts highlighted by the red lines.

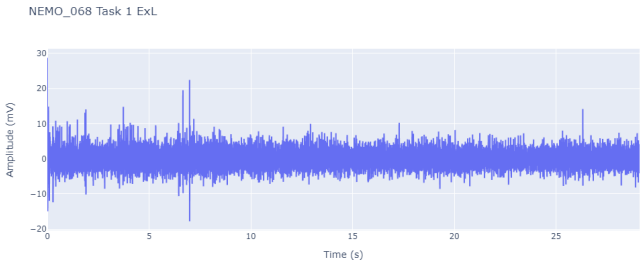



Figure 2: *EMG data of the left underarm of a patient during task 1*

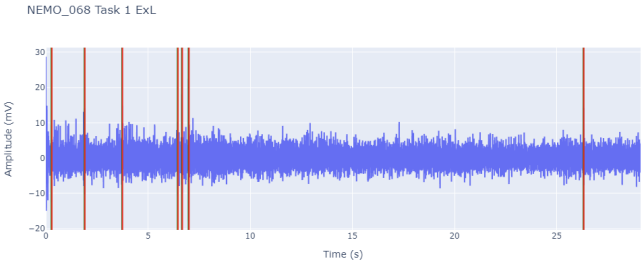



Figure 3: *The same EMG data with the bursts marked*

But not all myoclonus patients have bursts during resting tasks, some only show bursts during postural or dynamic tasks, as we have seen in our own data set. However, EMG measurements are much more chaotic and difficult to interpret during these tasks. Figures 4 and 5 show data from the same patient and sensor as depicted in Figures 2 and 3 but now during task 17, holding the hands in front with the palms vertical. Here, a double threshold solution would probably not be sufficient and requires a more sophisticated solution.

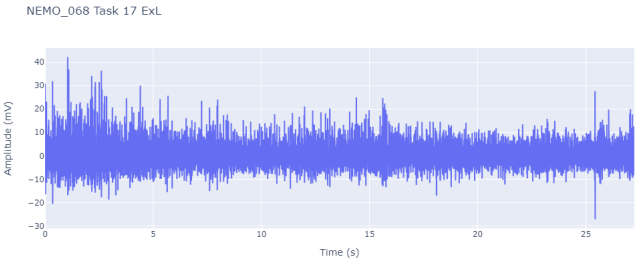



Figure 4: *EMG data of the left underarm of a patient during task 17*

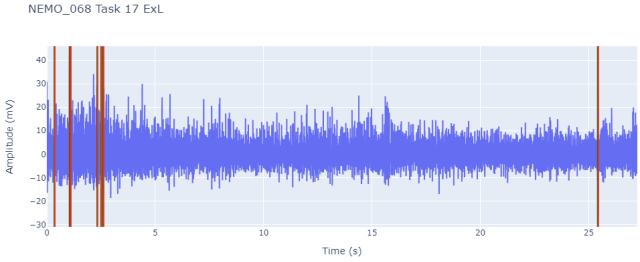



Figure 5: *The same EMG data with the bursts marked*

Given that the data is sequential and the contextual information of each data point is crucial, models that incorporate the context of the data points are more appropriate. Furthermore, the position of the bursts within the data is unknown, the model needs to be able to identify bursts regardless of their location within

the input sequence. Since we have a data set of labelled bursts, we are able to apply supervised learning techniques [17].

The EMG data is very noisy, so it is possible the model could benefit from some preprocessing, like a denoising function. Determining the precise onset and offset of myoclonus bursts is a complex task, and our dataset is made up of hand-labelled data. Since the labelling is subjective and may vary between individual labellers, it is crucial to acknowledge the potential bias introduced by this manual approach. To ensure the model's understanding is not influenced by human biases, we need to develop a method that can handle the inherent uncertainty in burst start and end times during training.

We currently focus on analyzing one sensor at a time, out of the 6 sensors with annotations per task. This approach helps us avoid unnecessary complexity in addressing the problem, allowing for a simpler starting point to tackle the issue. This complexity can be added later if necessary.

# 3 Methods

Our first question in the current study was to identify the most appropriate model to detect myoclonus bursts in EMG measurements. Since a recurrent neural network with long short-term memory has been proven effective before in similar problems [9]. We, therefore, decided to also implement a layered recurrent neural network (RNN) with long short-term memory (LSTM). An RNN is a machine learning algorithm specifically designed for sequential data [18]. Unlike traditional models that process the entire input sequence at once, an RNN analyzes inputs one by one, incorporating the output from the previous input as an additional input for the next data point. This inherent memory allows RNNs to capture temporal relations in time-series data. To enhance the modelling of long-term dependencies in RNNs, an extension called Long Short-Term Memory (LSTM) was introduced [19]. LSTM enhances the basic RNN architecture by incorporating specialized memory cells and gating mechanisms. These enable LSTM to incorporate a value into the LSTM cell, which is then used as an input for each data point. Unlike the standard RNN cell, the LSTM cell lacks a direct weight connection, using only a bias. As a result, the content of the LSTM memory cell is not multiplied by a scalar at every data point. This helps the LSTM retain information from a more distant past, facilitating better memory of long-term dependencies.

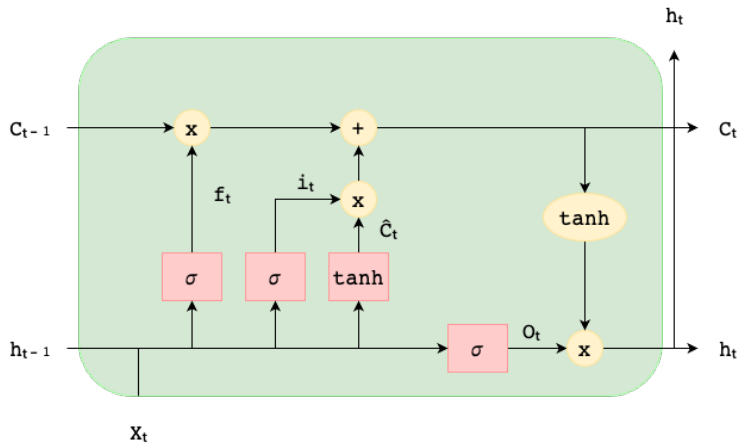



Figure 6: *LSTM cell architecture*

Figure 6 provides an illustration of the LSTM cell's architecture, which is applied to each data point. $X_t$ denotes the current data point. $h_t$ is the output of the LSTM cell. And $C_t$ is the value of the memory unit of the cell.

Within the cell, three gates are present:

- The forget gate, $f_t$, determines the proportion of information that should be forgotten from the LSTM memory unit.
- The input gate, $i_t$, controls the new information to be incorporated into the LSTM memory unit.
- The output gate, $O_t$, generates a filtered version of the internal cell state.

$$
\begin{aligned}
f_t &= \sigma(W_f * [h_{t-1}, X_t] + b_f) \\
i_t &= \sigma(W_i * [h_{t-1}, X_t] + b_i) \\
O_t &= \sigma(W_O * [h_{t-1}, X_t] + b_O) \\
\hat{C} &= tanh(W_C * [h_t - 1, X_t] + b_C)
\end{aligned}
$$

Where $W_f, b_f, W_i, b_i, W_O, b_O$ are the weights and biases associated with their corresponding gate. These weights and biases have to be learned during training [19].

The circles with x in them mean the two inputs get multiplied, the circle with the + means the two inputs get added together.

This gives:

$$
\begin{aligned}
C_t &= i_t * \hat{C}_t + f_t * C_{t-1} \\
h_t &= o_t * tanh(C_t)
\end{aligned}
$$

These cells are chained together using the output and the memory unit from the previous cell ($h_{t-1}$ and $C_{t-1}$) as the part of the input for the current cell.

# 4 Implementation

## 4.1 Data preprocessing

Our second research subquestion focuses on the effective training of the model using our labelled data. To address this, we delve into the challenges posed by our labelled data and explore how preprocessing techniques can help overcome these obstacles.

### 4.1.1 Windowing

Given the human bias in the precise start and end times of myoclonus bursts, training a model on the exact onset and offset labels poses challenges. To mitigate these potential human biases from the training data, we implemented a different approach. Rather than asking the model to identify exact start and end times, we partitioned the data into windows and trained the model to classify each window as either a burst or a non-burst window. Bursts in our dataset are typically 50-70 ms or 100-140 data points. The selection of an appropriate window size involves a delicate balance. On one hand, the window must be large enough to contain an entire burst, encompassing both the burst itself and contextual information around it. On the other hand, caution must be exercised to avoid excessive window size, as this could make it more difficult for the model to identify the precise location of the burst within the window. We have chosen a window size of 256 data points as this is the first power of two that is large enough to fit all individual bursts in our data set. This enables the model to capture the entirety of each burst, contributing to more comprehensive training. Ensuring the model observes a substantial portion of the burst may be crucial for accurate identification and classification. Given the uncertainty about the exact locations of bursts in the EMG data, it is possible for bursts to be split across windows. To address this, we introduced a 50% overlap between consecutive windows to ensure that each burst is significantly present in at least one window.

### 4.1.2 Filtering

Due to the high level of noise present in the EMG data, incorporating a denoising function, similar to previous approaches applied to EMG data [10], could be advantageous for the model. We choose a Butterworth denoising filter [20] to test if denoising could be beneficial for the model. We decided to use a butterworth denoising filter with a bandwidth of {10, 450} and an order of 6, as it has yielded good results in the past with EMG data [10]. This means the filter decreases the impact of the frequencies outside of the 10 to 450 Hz range.

### 4.1.3 Augmentation

The size of our dataset is relatively small, especially when considering that we train the model on one task and one sensor at a time. This means we can only train our model on a subset of the total bursts at a time. To address this limitation and provide the model with more examples for training and testing, we have augmented our original dataset. Data augmentation involves the addition of supplementary samples to the dataset. In our case, we have expanded the dataset by including ten different windows, each consisting of 256 data points, for every labelled burst. These windows are created such that the burst is located in ten different positions within each window. This augmentation strategy enables the model to learn and detect bursts regardless of their specific location within the time window. To balance the augmented training set with an equal amount of burst and non-burst windows, we additionally added random windows without bursts.

For simplicity of the model, we allocate two bursts from each patient to the validation set and two bursts to the test set. The rest of the bursts are added to the training set. This guarantees that every patient's bursts are being represented and evaluated during testing. This is potentially easier for the model because it is not looking at entirely new patients, only new bursts in patients that it already has seen. In a later stage this can be changed to more accurately represent the real world where the model will look at patients it has never seen before, we will come back to this point in the discussion.

## 4.2 Model

### 4.2.1 Architecture

We have implemented a recurrent neural network (RNN) with LSTM architecture using Python 3.10.6 and the Keras 2.13.1 framework. Keras provides the advantage of utilizing tensors, which are optimized multi-dimensional arrays designed for GPU processing [21]. Models were trained on an NVIDIA RTX3090 using CUDA version 11.8 [22]. Our model has the following architecture:

**1.** An input sequence layer for preprocessing the data;

**2.** One or more LSTM layers used to learn the time dependencies within the sequential data;

**3.** (optional) Dropout layer to prevent overfitting

**4.** A fully connected layer used to convert the output size of the previous layers into the number of classes to be recognized;

**5.** A softmax layer used to compute the belonging probability to each class;

The input sequence layer preprocesses the data, converting the EMG data into a balanced training set. Each tensor within this set consists of 256 data points and includes a label indicating the presence or absence of a burst within the window. Depending on the specific parameter configurations we explored, the EMG data may undergo optional preprocessing with a Butterworth denoising filter before entering the input sequence layer.

The LSTM layer learns the time dependencies within the sequential data, as explained in the methods section.

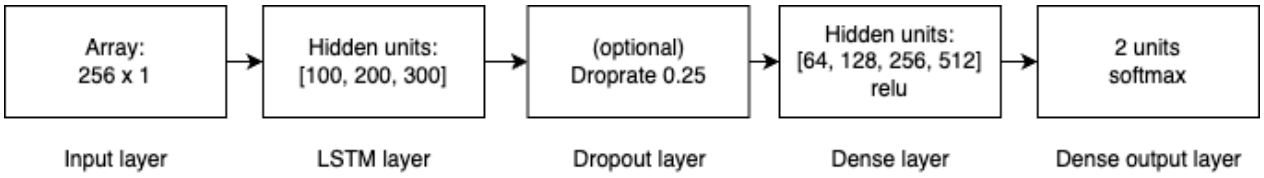
We also include an optional dropout layer in our model to assess whether it improves the model's performance on the validation set. Dropout regularization works by training only a random subset of neurons during each step, introducing a regularization effect that limits the network's ability to fully exploit its complexity. This prevents overfitting of the model to the training data [23].

The fully connected layer receives the output from the previous layer and converts it through a hidden layer into the number of classes to be recognized [21], which, in our case, is 2 classes. Each input node is connected to every hidden state node with adjustable weights and biases. The sum of these connections forms the cell state of the hidden node. All hidden nodes are connected to the output nodes with their respective weights and biases, and their contributions are combined before being sent to the final layer. The Rectified Linear Units (ReLU) function [24] is used as the activation function for this layer.

The softmax layer produces the probabilities of each class using the softmax function [25]. The softmax function takes an input vector of scores and transforms them into probabilities by exponentiating each score and then normalizing the results. The output probabilities represent the likelihood of each class being the correct class.

### 4.2.2 Parameters

These layers all have parameters that influence the accuracy of the model. Determining the optimal parameters in advance is difficult. Therefore, we conduct extensive training by iterating through all combinations of parameters for 250 epochs. Through testing, we identified the combination that yields the best performance for our specific model.
The parameters are:
batch sizes = [8, 16, 32, 64, 128]
Number of LSTM layers = [1, 2, 3]
Number of hidden units in the LSTM layers = [100, 200, 300]
Number of hidden nodes in dense layer = [64, 128, 256, 512]
Learning rates = [1-e5, 1-e4, 0.001, 0.01, 0.1]
Use Butterworth denoising = [yes, no]
Use a dropout layer = [yes, no]



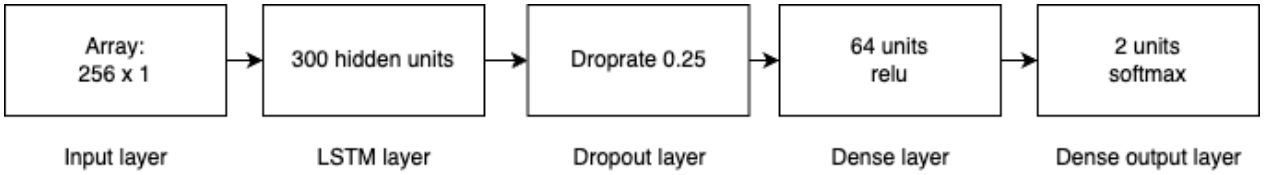

We used the ADAM (adaptive moment) optimization function [26] to train all the models.

At each epoch, the model was assessed using the validation set, allowing us to determine its accuracy in classifying the validation data. Once the training for a specific set of parameters was completed, we saved the weights and biases from the epoch that achieved the highest accuracy on the validation set. Then, we evaluated the performance of this model on the test set, to assess its performance on bursts it has never seen before, to evaluate its ability to generalize to new data.

# 5 Results

## 5.1 Results on test set

The testing showed us that the best combination of parameters are batch sizes = [64] Number of LSTM layers = [1] Number of hidden units in the LSTM layer = [300] Number of hidden units in the Dense layer = [128] Learning rate = [1-e5] If we use Butterworth denoising = [yes] If we use a dropout layer = [yes]




To evaluate the model's performance, we generated a Receiver Operating Characteristic (ROC) curve (figure 7), which helps determine the optimal threshold for classifying a window as a burst. This threshold is applied to the output of the last layer, represented by a two-dimensional vector, containing probabilities for both burst and non-burst classifications. The ROC curve shows the trade-off between the true positive rate and the false positive rate at different classification thresholds. A larger area under the ROC curve indicates better model performance.

In our case, the Area Under the Curve (AUC) measures 0.82, demonstrating the model's ability to distinguish between bursts and non-bursts effectively. A perfect AUC score would be 1, representing flawless classification, while an AUC of 0.5 indicates performance equivalent to random guessing. Interpreting the ROC curve can give valuable insights into the model's sensitivity and specificity, helping in the threshold selection. We generated a confusion matrix (figure 8) with a threshold of 0.5 to give an idea of the level of the model at the moment. Which gives us a sensitivity of 80% a specificity of 0.69% a precision of 0.83% and an accuracy of 76%. However, we will leave the responsibility of determining the appropriate threshold for classification to the developer of the classifier, which is the next step for this project. As the developer, he or she can consult with the clinicians of NEMO to select the threshold that best suits their model's requirements and desired trade-offs between sensitivity and specificity

The AUC value of 0.82 indicates promising results, but further optimization and fine-tuning of the model could potentially lead to even better performance. We will address these optimizations in the discussion.

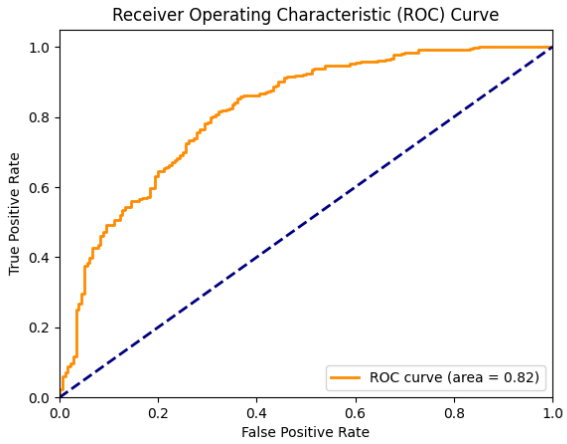



Figure 7: *ROC curve*

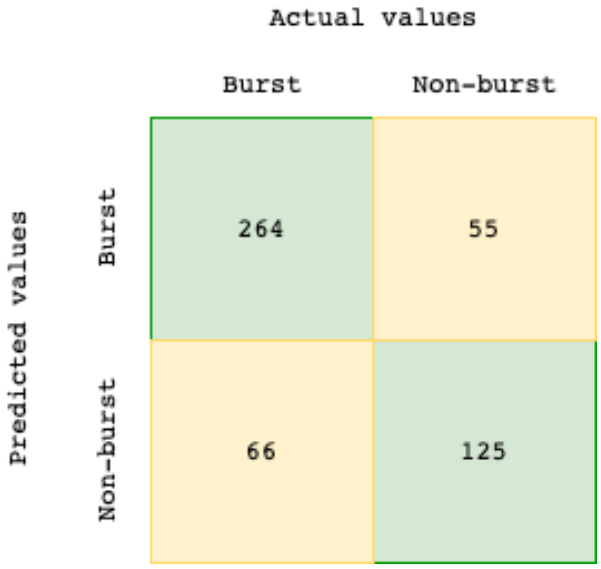
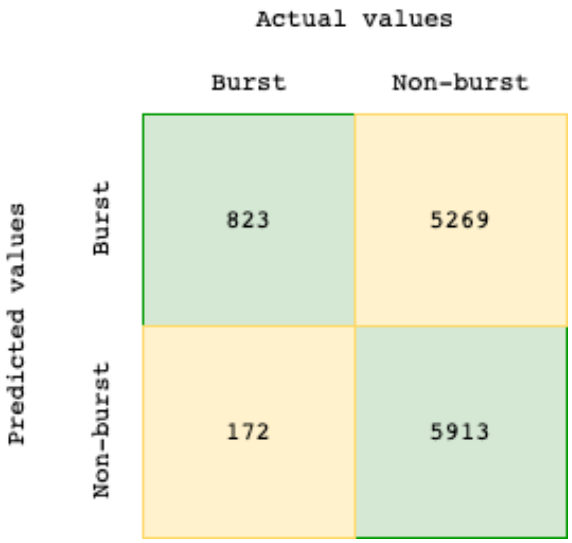



Figure 8: *Confusion matrix*

## 5.2 Results on the whole dataset

After we got the results of the model on the validation and test set. We wanted to test our model on the whole data set. So we fed in the EMG data of all whole 30-second tasks of patients, in windows of 256 at a time with a 50% overlap. Subsequently, we generated this confusion matrix with the results (see Figure 11).




This corresponds to a sensitivity of 0.83%, a specificity of 0.53%, a precision of 0.14% and an accuracy of 55%.
A precision rate of 14 percent may initially appear modest; however, an examination of the data necessitates further discussion. We reviewed approximately 20 bursts that were "erroneously" predicted by the model and cross-referenced them with the video footage accompanied by the EMG data (video-polymyography). And in most cases, we determined that a burst did occur during those instances. And this burst was simply missed during the labelling of our data set.

Notably, more severely affected myoclonus patients often experience almost constant involuntary muscle contractions, including subtle ones. Our training data however, consists of prominently noticeable contractions, our model accurately flags all contractions, including both significant and subtle ones.

# 6 Discussion

## 6.1 Research questions

In this study, our primary research question was to explore the feasibility of using machine learning algorithms for the automatic detection of myoclonus bursts. However, before delving into this main question, we addressed several sub-questions to better understand the problem and refine our approach.

### 6.1.1 Which machine learning model is best suited for detecting myoclonus bursts automatically?

Initially, we conducted a literature search to identify relevant studies and similar problems in the field. This allowed us to select the most appropriate model architecture tailored to our problem.

### 6.1.2 How can we efficiently train the model using existing labelled data?

To address the uncertainty surrounding the exact start and end points of myoclonus bursts, we employed two key techniques in our approach to making our training set. First, we utilized a window-based approach, which allowed us to focus on burst detection within specific segments of the data, making the exact start and end of a burst less critical. Second, we implemented data augmentation techniques to expose the model to various burst positions within the window, ensuring it could effectively handle bursts occurring at different locations.

### 6.1.3 Does data preprocessing help the model's accuracy?

Additionally, we explored the application of a Butterworth pass filter as a data preprocessing technique to assess its potential to enhance the model's performance.

### 6.1.4 Which set of parameters yields the best performance from our machine learning model?

Lastly, we conducted hyperparameter testing to identify the optimal set of parameters for our model.

This gave us a model that could accurately predict if a window contains a burst 82% of the time, but maybe more importantly this gave us a model that already has corrected numerous faulty labels in our training data.

Comparing the results to similar studies [10, 9], we observe a slightly lower level of performance in our results. Since this is the first time that machine learning algorithms are used to automatically detect myoclonus bursts, these results are not unexpected. Nonetheless, it substantiates a significant foundational advancement in establishing the feasibility of this study.

## 6.2 Limitations of the approach

Our data shows how hard it is for clinicians to accurately detect all the myoclonus bursts in a large amount of data. This means the data the model trained on was not clean and had a few faults.

Furthermore, we have yet to validate our model's performance on entirely novel patient cases. Although we anticipate a minimal deviation in accuracy, the extent of this variation remains uncertain.

Lastly, our model's current simplicity may restrict its potential. It has the capacity to incorporate inputs from additional sensors and data sources simultaneously.

In the subsequent chapter, we delve into these limitations and elaborate on our strategies to address them.

# 7 Future work

The results presented in the last chapters have left plenty of questions open for future work. We will discuss these here.

## 7.1 Iterative data cleaning

It is not clear what the actual accuracy of the model is if trained and tested on clean data. So firstly the data has to be inspected again, as the model has already uncovered numerous faulty labels. After this, the model can be trained again on the cleaner data. These steps can be repeated until the data has the desired accuracy. The parameters selected for this project may not remain optimal after reassessing the data. With each iteration of data cleaning, it is crucial to reevaluate and choose the appropriate parameters for the new data.

## 7.2 Model Generalization

Once the data has achieved sufficient accuracy, we can enhance model generalization by modifying the training, validation, and test set composition. Currently, at least one burst from every patient is present in all three sets. To simulate real-world scenarios and improve the model's ability to handle unseen data, we can test the model on bursts from patients it has not seen during training. This approach allows the model to adapt better to novel data and enhances its overall performance.

## 7.3 Increase model complexity

If the model is not accurate enough after training, the model could benefit from more complexity. Sometimes in similar problems, an extra CNN layer is added for feature extraction [10]. This approach could be considered for our problem as well.

A different way to increase the complexity of the model is to use multiple inputs simultaneously. Currently, we only use the EMG data of one sensor at a time, but incorporating data from multiple sensors in close proximity may enhance the model's performance. The NEMO team also possesses accelerometry and video data of patients during tasks, which could serve as additional inputs to the model. However, it's important to consider that the hospital planning to deploy this model would require the necessary equipment to gather and provide this extra data.

## 7.4 Classification

If the model has accurate enough results following these changes, we can address the next stage of our problem: classifying these bursts into myoclonus subcategories. This would involve building a separate model that takes 256-data-point windows as input and predicts the corresponding category for each burst or for each patient.

# 8 Conclusion

The application of machine learning for automatic myoclonus burst detection shows promising results. The model's accurate correction of our training data demonstrates its proficiency in managing the vast amount of complex data encountered in clinical settings. But to get a more definitive answer we first need to refine our data. After refining the data, the model can be accurately assessed, providing a conclusive evaluation of machine learning algorithms for myoclonus detection.

# References


[1] R. Zutt, J. Elting, and M. Tijssen. *Tremor and myoclonus*, volume 161 of *Handbook of clinical neurology*, pages 149–165. 2019.

[2] M. Kojovic, C. Cordivari, and K. Bhatia. Myoclonic disorders: A practical approach for diagnosis and treatment. *Therapeutic Advances in Neurological Disorders, 4(1), 47-62.*, 2011.

[3] J. Caviness. Myoclonus. *CONTINUUM: Lifelong Learning in Neurology*, 25, 2019.

[4] J. Caviness. Treatment of myoclonus. *Neurotherapeutics*, 11:188–200, 2014.

[5] H. Tankisi, D. Burke, L. Cui, M. de Carvalho, S. Kuwabara, S. D. Nandedkar, S. Rutkove, E. Stålberg, M. J. A. M. van Putten, and A. Fuglsang-Frederiksen. Standards of instrumentation of emg. *official journal of the International Federation of Clinical Neurophysiology*, 2020.

[6] T. Mitchell. *Machine learning*. 1997.

[7] Y. Wei, J. Zhou, Y. Wang, Y. Liu, Q. Liu, J. Luo, C. Wang, F. Ren, and L. Huang. A review of algorithm hardware design for ai-based biomedical applications. *IEEE Transactions on Biomedical Circuits and Systems*, 14(2):145–163, 2020.

[8] U. Rashid, I. Niazi, N. Signal, D. Farina, and D. Taylor. Optimal automatic detection of muscle activation intervals. *Journal of Electromyography and Kinesiology 48:103-111*, 2019.

[9] M. Ghislieri, G. L. Cerone, M. Knaflitz, and V. Agostini. Long short-term memory (lstm) recurrent neural network for muscle activity detection. *Journal of NeuroEngineering and Rehabilitation*, 18, 2021.

[10] A. Vijayvargiya, B. Singh, N. Kumari, and R. Kumar. sEMG-based deep learning framework for the automatic detection of knee abnormality. *Signal, Image and Video Processing*, 2022.

[11] A. M. M van der Stouwe, I. Tuitert, I. Giotis, J. Calon, R. Gannamani, J. R. Dalenberg, S. van der Veen, M. R. Klamer, A. C. Telea, and M. A. J. Tijssen. Next move in movement disorders (nemo): developing a computer-aided classification tool for hyperkinetic movement disorders. *bmjopen*, 2021.

[12] D. O. Doheny, C. E. Brin, M. F.and Morrison, C. J. Smith, R. H. Walker, S. Abbasi, B. Müller, J. Garrels, L. Liu, P. De Carvalho Aguiar, K. Schilling, P. Kramer, D. De Leon, D. Raymond, R. Saunders-Pullman, C. Klein, S. B. Bressman, B. Schmand, M. A. Tijssen, L. J. Ozelius, and J. M. Silverman. Phenotypic features of myoclonus-dystonia in three kindreds. *Neurology*, 2002.

[13] S. J. Frucht, S. E. Leurgans, Hallett M., and S. Fahn. The unified myoclonus rating scale. *Advances in neurology*, 2002.

[14] A. Subasi. Classification of EMG signals using pso optimized svm for diagnosis of neuromuscular disorders. *Computers in Biology and Medicine*, 43(5):576–586, 2013.

[15] M. Atzori, M. Cognolato, and H. Müller. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in Neurorobotics, 10, 219975*, 2016.

[16] Abotabl A. Akef Khowailed, I. Neural muscle activation detection: A deep learning approach using surface electromyography. *Journal of Biomechanics, 95, 109322.*, 2019.

[17] A. C. Müller and S. Guido. *Introduction to Machine Learning with Python*.

[18] M. I. Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. 1986.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[20] S. Butterworth. On the theory of filter amplifiers. *Experimental Wireless and the Wireless Engineer*, 1930.

[21] F. Chollet. Keras. https://keras.io, 2015.

[22] NVIDIA, P. Vingelmann, and F. H. P. Fitzek. Cuda, release: 10.2.89, 2020.

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

[24] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 1975.

[25] J. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.