Online Classification of Movement Intention: An end-to-end EEG-based BCI Pipeline

Christoffer Hansen*, Fredrik De Frène†, and Simon Park Kærgaard‡
Technical Faculty of IT and Design, Aalborg University
Denmark
Email: {*ch17, †fdefre16, ‡skarga17}@student.aau.dk

***Abstract*—Brain-computer interfaces (BCIs) based on electroencephalography (EEG) allow users to control external instruments with brain signals and can be used to aid in the rehabilitation of patients with loss of motor function. An online application of such a system requires a robust classifier, but only few studies focus on evaluating BCI systems for online use. This paper proposes an end-to-end BCI pipeline for online application, which encompasses extensive preprocessing and automatic labeling of EEG signals and either a state-of-the-art deep-learning feature extraction method and classifier or a gradient-based classifier. An online simulation suite is applied to evaluate the system's performance in an online setting. We collect 220 traces, each consisting of 20 self-paced motor task executions, from 10 subjects for use in evaluations. We construct the pipeline to make the process as hands-off as possible. The pipeline automatically labels training data for the classifiers by cross-referencing EEG and electromyography (EMG) data recorded simultaneously. Our automatic labeling method shows that it can contend well with noisy data compared to an earlier method. The pipeline includes a ViTransformer deep-learning classifier and XGBoost, a gradient-based classifier. We evaluate the classifiers online by simulating the classifiers' rate to receive input data. Our experiments show that a well-performing offline classifier does not necessarily translate to a well-performing online classifier. From online simulation experiments, XGBoost achieves the best results where it predicts 66.7% of the movements and achieves an 81.5% $\pm$ 4.2 precision, a 0.45 s $\pm$ 0.31 mean prediction latency, and a 2.33 s $\pm$ 1.20 mean prediction latency false positives on a single subject.**

***Index Terms*—Brain-Computer Interface (BCI), Movement Related Cortical Potential (MRCP), Electroencephalogram (EEG), Electromyography (EMG), Onset Detection, Automatic Labeling, Machine Learning, Data Acquisition, Self-paced, Simulation, Online Evaluation**

I. Introduction

For some individuals who have lost motor function due to neurological injuries or disease, the brain is the only place where the intention of body movement can be detected. Thus, many studies are invested in decoding motor intentions in the human brain [1]. Brain-computer interfaces (BCI) aim to translate brain activity into commands that capture movement intentions, thereby enabling partial or complete control of external devices. BCI systems can thus help patients suffering from loss of motor abilities by, e.g., enabling control of a wheelchair [2] or enabling the control of a robotic glove, thus helping restore motor abilities [3]. The latter application was the focus of our previous study [4].

Individuals' motor intentions, being either imagery or execution, can be decoded using electroencephalography (EEG), an inexpensive and non-invasive method of measuring brain activity. The resulting electrograms capture voltage fluctuations from the brain on the scalp's surface and are obtained from electrodes placed on the scalp. The brain's motor cortex is essential for planning and executing motor commands [1].

Traditionally, BCI studies adopt cue-based approaches, where test subjects follow a timed auditory or visual cue from an external device to execute a movement or motor imagery (MI). This makes it possible to determine the time points of the signals that signify movement or MI and label the data accordingly. However, it can also lead to an association between the afferent stimulus from a timed cue and the motor-cortical activity [5], which does not map well to a real-world online-BCI, where patients do not follow a visual or auditory cue to initiate a movement. Also, patients with cognitive and perceptual deficiencies that hinder their ability to attend to the task and predict the cue occurrence may have difficulty with the cue-based method [5], which requires initial training in reacting to the cue. In contrast, a self-paced approach allows subjects to execute movements at their own initiative and pace, i.e., in a manner closer to a real-world scenario.

Different paradigms of self-paced motor-related signals within EEG can be used for detecting motor intention, such as sensorimotor rhythms (SMR) and movement-related cortical potentials (MRCPs) [6]. MRCPs are naturally occurring patterns in EEG signals that appear approximately two seconds before the point of motor imagery or execution [7]. Unlike SMR, no specific neurofeedback training of individuals is required to elicit MRCPs [8], and MRCPs can be recorded from both healthy and non-healthy individuals performing self-paced movements.

In the self-paced approach, electromyography (EMG) signals can be recorded alongside EEG signals from a subject to take over the role of timed cues when having to label movements in EEG signals. EMG signals are recorded from skeletal muscle activity using non-invasive electrodes placed on the subject's skin on top of the activated muscles during movement execution. The resulting signals capture the muscle activity during the subject's movements. These signals can then be cross-referenced with the EEG signals to distinguish between rest and movement within EEG signals, thus enabling labeling of the data to obtain a ground-truth labeled training set

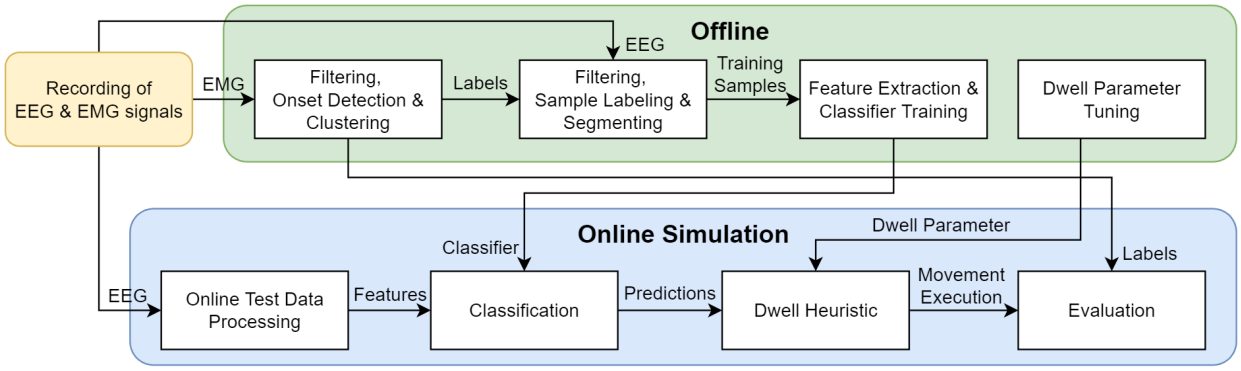


Fig. 1: Diagram of our proposed pipeline. EEG and EMG signals are recorded and used in the pipeline's offline and online simulation parts. In the offline part, we use EMG signals to create labels before training samples, and features are extracted from the EEG signals and fed to a classifier. The trained classifier is fed EEG data continuously in the online simulation part. A dwell heuristic assesses the predictions before we execute a movement execution. Finally, we evaluate the movement executions.

for a movement intention classifier. This process is currently done manually [3], [4], making it prone to human errors and challenging to apply to a larger amount of data.

Online experimentation and evaluation are more labor-intensive and challenging in terms of consistency between online experiments and how to assess the outcome. It also poses the challenge of processing signals in real-time [9]. These challenges are why few BCI studies focus on online predictions when evaluating system performance. Furthermore, many studies are based on datasets from BCI Competitions, which limits the ability to translate the results into a real-world scenario [10]. In contrast to offline classification, where the entirety of the training data is available for training ahead of time, online classification implies that data must be processed in real-time, i.e., only past and present data points are available, not future data points. Validation metrics also differ between offline and online studies. A BCI with high accuracy in an offline study does not necessarily have high performance or usability in real, online use cases. A well-performing online BCI system must offer not only a high true positive (TP) rate but also a low false positive (FP) rate. For example, in the use case of a robotic soft glove, false positive predictions can lead to undesired movements, which, in turn, may cause unwanted and potentially dangerous situations.

The BCI literature features a variety of algorithms for extracting features from EEG signals. A recent survey [11] states that the common spatial pattern (CSP) algorithm is the most used feature extraction algorithm for EEG-BCIs. The combination of CSP and a support-vector machine (SVM) is the most common approach to classification. In general, machine learning is adopted widely for classification tasks in EEG-BCIs. XGBoost is a machine learning algorithm based on the gradient boosting framework that employs a linear model solver and tree learning concepts. XGBoost is commonly used to achieve state-of-the-art results on real-world scale problems [12] but has yet to be applied widely in EEG-BCIs. We propose an implementation of XGBoost for use in the EEG-BCI field. Deep learning classifiers have achieved state-of-the-art performance in various domains, including in speech recognition, object detection, and beyond [13]. In EEG-BCI, classifiers based on convolutional neural networks (CNNs) are used most frequently [14], [15]. Transformers [16] have recently shown high performance across a multitude of tasks [17]–[19]. However, they have yet to see a widespread application in EEG-BCI. Unlike XGBoost, a transformer relies entirely on self-attention to compute its input and output representations. We implement the ViTransformer (ViT) [17] classifier, a version of the original transformer classifier that works on images, and we compare it to existing EEG-BCI deep-learning solutions.

This report presents an end-to-end pipeline for online prediction of hand opening and closing movements, as shown in Figure 1 for control of a robotic soft glove. The pipeline involves four crucial steps: automatic labeling, feature extraction, classification, and online evaluation through simulation. We evaluate the pipeline in a series of experiments to gain insight into how well the XGBoost and ViT classifiers perform in an online setting on self-paced data. We propose an online simulation method to get closer to a standardized method of evaluating BCIs that reflects their performance during online use and can be applied to all classifiers.

All experiments are conducted on new self-paced EEG and EMG data recorded from ten subjects in the BCI-lab at Aalborg University in collaboration with the university's Health Science and Technology Department.

In summary, our study makes the following main contributions:

- We contribute a new dataset, recording EEG, EMG, and electrooculography (EOG) data from ten healthy subjects as they perform self-paced hand opening and closing movements mimicking the control of a robotic soft glove. Rather than following a specified cue, the subjects move at their own pace and will. We record a large number of movements from each subject during a single session. From each subject, we have acquired 440 movement executions.
- We propose a threshold-based clustering algorithm to detect movement onsets within EMG signals and de-

fine several proximity heuristics to cluster the detected movement onsets automatically. We cross-reference the detected movement onsets with the recorded EEG signals, thus enabling automatic training sample labeling.
- We evaluate the classification performance for movement detection of the XGBoost and ViT algorithms, as these machine learning algorithms have shown great performance in other fields and have yet to be applied widely in the EEG-BCI field. We compare the feature extraction and classification methods we propose with already established methods.
- We propose an online simulation suite to conduct empirical studies, which can be used on any classifier configuration. We differentiate the evaluation of online BCI-systems from the evaluation of offline systems by utilizing different evaluation metrics that not only focus on accuracy. Our online evaluation metrics include the mean prediction latency (MPL), i.e., the time between a prediction and its associated movement onset.

The remainder of the paper is organized as follows: Section II presents existing related methods. Then, Section III introduces the data acquisition process and the subjects involved. Section IV describes our proposed automatic labeling method, and Section V. In Section VI, we present our experimental findings. Section VII encompasses discussion and future direction. Lastly, Section VIII concludes on our findings.

II. Related Work

This study's main contributions cover different areas of BCIs. Therefore, this section is twofold, where the first part encompasses related work that proposes labeling self-paced data using EMG signals and evaluating the feasibility of a BCI-system's use in real-time. The latter describes related studies concerning machine learning applications for classifying various EEG events.

A. Labeling of Self-Paced Data & Online Evaluation

In a previous study [4], we focus on minimizing the manual labor required of experimenters in the aforementioned procedure [3]. Specifically, we propose a new procedure to label EEG signals detected movement onsets from EMG signals automatically. This includes a threshold-based clustering algorithm that detects the individual points of movement in EMG data using size and distance measures. However, the algorithm is sensitive to noisy signals and has difficulty encapsulating the critical parts of a movement, such as the onset and end of a movement, resulting in issues in determining the correct number of moments. We then use the movement onsets to find peak negativity of MRCPs, which is a naive approach because it is inconsistent on a sample-by-sample basis, which can lead to poor labeling and hence can impair the final evaluation. The evaluation phase includes an online simulation suite that mimics a real-time procedure, which includes new metrics for evaluating the performance of our BCI under real-time constraints.

Rashid, U. et al. [20] study the feasibility of automatically labeling MRCPs using only EEG signals. The proposed methods derive the onsets, amplitudes at the onsets, the slopes of two distinct MRCP features, and the time and amplitude of a third distinctive MRCP feature. They evaluated the method on a dataset of simulated MRCPs and a dataset of EEG signals and found that the performance of the proposed method was comparable to that of manual labeling by domain experts. They conclude that the method can be used to obtain robust *estimates* for the MRCP features automatically.

Mondini et al. [21] propose a modular EEG-based BCI system to train subjects to elicit MI in EEG signals for the control of a virtual arm. They use CSP for feature extraction and an SVM for classification. The system is tested on ten subjects but an online evaluation is only performed on subjects that reach a criterion level of 70% accuracy during earlier training phases, which was 7. They report that the most significant limitation of their system is EEG artifacts, particularly muscular artifacts. A cue-based approach is applied when collecting data during their training phases, and the system does not contain any online artifact rejection method, but an experienced inspector manually removes signals that exhibit systematically occurring artifacts after each training session.

Savić et al. [3] collect data in self-paced experiments to predict MRCPs for online control of an assistive active glove. They utilize a manual threshold for EMG signals collected simultaneously with EEG signals to estimate movement onsets. The movement onsets are manually annotated and used to label training samples in the EEG signals, where an experimenter manually inspects and selects which training samples to use in classifier training. They evaluate the performance of an online MRCP-based BCI by adopting a sliding window approach, which simulates a real-time input feed. They utilize a combination of metrics to evaluate online performance; however, detection latency is not registered during their online simulation assessment but is manually estimated by the experimenter post-hoc.

Valenti et al. [22] present a novel combination of 3D-convolutional and recurrent deep neural networks for multi-class prediction in an online setting. They introduce an input representation of EEG data that preserves the spatio-temporal dependencies of the different channels, capturing more information. They use an efficient deep-learning method with simpler input signal preprocessing to reduce the computational overhead, making it better suited for real-time application. Their model can perform classification at every time step of an epoch, predicting a correct prediction 0.25 s before the movement onset. They explain that this allows their model to process a constant input stream in real-time, making it suited for online application. However, the paper does not discuss a method of online evaluation for their system.

Rashid, M. et al. [23] outline existing challenges and limitations for online EEG-based BCI systems. Firstly, existing EEG caps are non-standardized and contain anywhere from 3–64 electrodes. The increased number of electrodes causes the user discomfort and introduces many sources of errors.

They recommend standardizing EEG caps with the minimum number of electrodes for a specific task. Another key challenge is the need for cross-study evaluation metrics. They report that current research uses different performance and often task-specific metrics to evaluate their systems, making it challenging to set uniform benchmarks.

In summary, current methods of labeling training samples from self-paced data require a manual inspection from an experimenter with domain knowledge. Additionally, the necessity of standardized cross-study evaluation is to be considered. In this study, we investigate improving the automatic labeling of self-paced data to reduce manual labor and make online evaluation more consistent and transparent by continuing and improving our work from the previous semester.

B. Classifiers

Only few studies exist in the field of EEG-BCI involving XGBoost for classification, and only very few are related to the classification of movements. In one study [24], CSP features are extracted from every non-overlapping 4 Hz band within the 4–40 Hz range and given to the XGBoost classifier, eliminating the need for subject-specific frequency band selection. The study achieves an average accuracy of 69.2% across nine subjects of the BCI Competition 2008-Graz dataset A's [25] evaluation set.

Bressan et al. [26] use a CNN to classify fine hand movements from EEG signals, specifically MRCP. They report similar or better results than their baseline soft linear discriminant analysis (sLDA) and random forest (RF) classifiers. Gao et al. [27] report substantially improved accuracy when predicting wide-range frequency EEG signals using a CNN-LSTM compared to their baseline Fisher's linear discriminant analysis (FLDA) and SVM classifiers. Zhang et al. [28] present a deep learning-based EEG decoding method to predict mental states. Their novel 1D CNN combined with a residual neural network-like structure to promote feature extraction achieves higher accuracy than traditional classifiers. SessionNet [29] presents a method for improving between-session generalization for movement classification, using ensemble learning and feature similarity. Schirrmeister et al. [15] explore the use of CNNs for predicting imagined or executed tasks from EEG signals and is the first study using deep learning to report better performance than FBCSP [30]. They propose a shallow and a deep CNN that we refer to as ShallowConv and DeepConv, respectively. EEGNet [14] introduces a compact CNN for EEG-based BCIs, using depthwise and separable convolutions to establish an EEG-specific model encapsulating EEG-BCI feature extraction concepts. They show that it can generalize well across different BCI paradigms, even with limited data, by comparing it to the state-of-the-art approach for MRCP, event-related potential (ERP), and oscillatory-based BCIs across four EEG datasets. In this study, we apply EEGNet, DeepConvNet, and ShallowConvNet as our reference models on our self-paced datasets.

In summary, deep learning methods display good overall performance in the BCI-field, outperforming non-deep learning classifiers. However, most of the literature focuses on CNNs, and all papers mentioned in this section are tested on cue-based datasets and only perform an offline evaluation. In contrast, we evaluate system performance in an online setting on data collected in a self-paced manner.

III. Data Collection

This section describes the setup and procedure used to collect data from subjects for classifier training and online evaluation. Figure 2 gives an overview of the individual parts of the procedure.

A. Instrumentation

We recorded EEG signals from nine electrode channels using a g.USBamp (g.tec GmbH, Austria) amplifier and a g.GAMMAbox (g.tec GmbH). We positioned each channel in the g.GAMMAcap2 (g.tec GmbH) and placed the cap over the subject's scalp on locations T7, C5, C3, C1, Cz, C2, C4, C6, and T8 following the international standard 10–20 system, as seen in Figure 2(C). We placed the ground electrode at AFz, and the reference electrode on the left earlobe at A1. We used FP1 to record EOG signals for detecting eye-movement artifacts. EMG activity was recorded alongside the EEG signals using an EMG electrode placed on the forearm with a reference electrode on the wrist and a ground electrode on the lateral epicondyle of the elbow on the subject's dominant arm, which is also shown in Figure 2(C). We connected the EMG electrodes to the amplifier. We selected these positions as they capture the movement of the activated muscles when performing the motor tasks. All electrodes were filled with conductive gel to ensure a connection between the skin and the electrode. We recorded all signals with a sampling rate of 1200 Hz using no embedded filtering from the amplifier.

By placing electrodes on either side of the longitudinal fissure, we can record signals from the opposite hemisphere of the dominant part of the motor cortex. Our goal with this EEG electrode layout is to increase the variance between the left and right hemispheres, making them easier to distinguish [31]. The symmetrical electrode placement along the mid-scalp allows us to record from both left and right-handed subjects without changing the layout.

B. Participants

We recorded data from 10 subjects (eight male, two female, 24 ± 1.25 years, nine right-handed, one left-handed). All subjects were healthy and without any known neurological disorders. None of the subjects were acquainted with BCI systems or had experience with the data collection procedure prior to the experiment. All subjects are in agreement with the collection of data.

C. Data Recording

From each subject, we initially recorded 20 *traces*. A trace consists of 20 movements of a single type, either closing or opening of the hand. Additionally, we recorded two traces of test data to validate our models and a rest trace with one

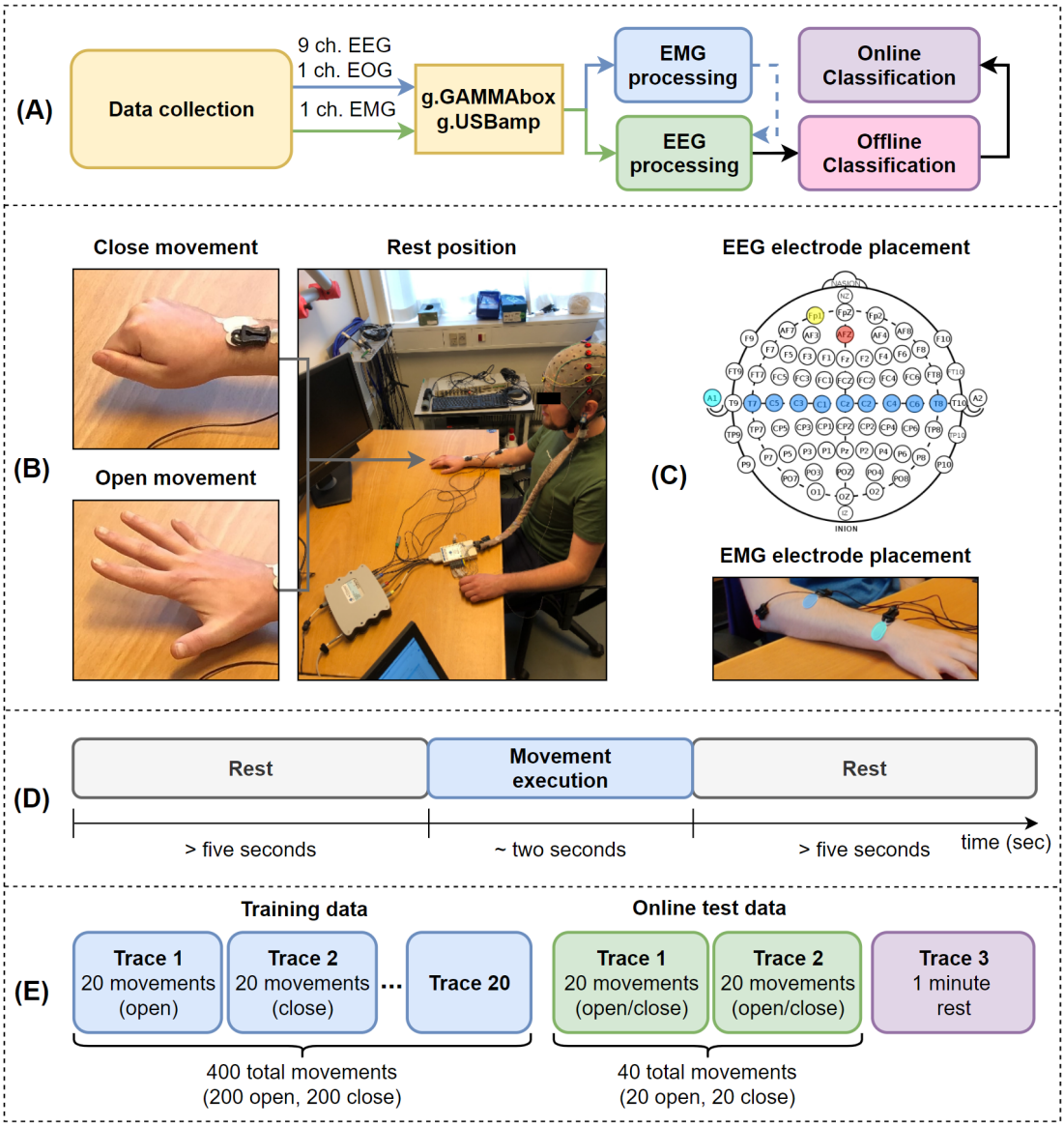


Fig. 2: Overview of the experimental setup for data collection. A) The collected data's flow. The signals from nine EEG channels, a single EOG channel, and a single EMG channel are processed. We use the EMG data to label the EEG training samples, which we use for offline classification training. We use the trained models for online simulation. B) Illustration of the hand's open and close movement execution while we record EMG, EEG, and EOG data. C) EEG electrode placement over the scalp following the international standard 10–20 system. The nine EEG electrodes are marked with blue, the reference electrode with cyan, the EOG electrode with yellow, and the ground electrode with red. EMG electrode layout is also shown with the ground electrode on the elbow, the main electrode on the forearm, and the reference electrode on the wrist, following the same color scheme. D) Timeline of the execution of a single movement during data acquisition. E) We record 20 traces of training data for each subject, which in total is 400 movements. Additionally, we record an online block consisting of two traces of movements and one minute of rest.

minute of data recording while the subject was not performing any movements. The test data traces consist of 20 movements, but we instruct the subject to alternate between opening and closing the hand. Thus, we recorded a total of 200 opening and 200 closing movements plus 40 test movements from each subject. We express the shape of a trace as: $C_{eeg} \times Trace_T$, where C_{eeg} denotes the number of EEG channels and $Trace_T$ denotes the number of data points. We recorded data from each subject during a single session in which the electrodes remained on the subject for the duration of the session.

Each subject was positioned comfortably in a chair in front of a table with the dominant arm resting on the table. This position is referred to as the rest position and is shown in Figure 2(B). The subject executed a motor task correctly if the subject initialized the movement from the rest position and then opened or closed the hand and returned to the resting position. During data recording, no excess communication with the subject was allowed, and all instructions were given

prior to recording. We instructed the subjects to execute the movements in a self-paced manner with a minimum of five seconds between movement executions, but without making a conscious effort to count the seconds or focus on maintaining a constant pace. We monitored the pace and number of movements executed by the subject and would indicate verbally when the halfway mark was reached and when the trace was completed. As explained in Section I, a self-paced approach allows individuals to execute movements at their own pace and initiative, in contrast to a cue-based approach. The timeline of a single movement execution is depicted in Figure 2(D). We also instructed the subjects not to blink, swallow, or exercise other facial muscles to minimize artifacts within the EEG signals while executing the movement. We instead asked the subjects to perform such motions while being in the rest position between motor task executions. The subjects were allowed to hold a small break of 1–5 minutes between each trace. An extended break is held after 12 traces before continuing with the rest of the traces.

To sum up, the dataset includes three classes: opening of the hand, the closing of the hand, and rest. However, considering opening and closing as a single movement class and performing binary classification between movement and rest, it is suitable for a possible use-case, such as opening and closing a robotic soft glove. This method can be considered a brain switch that reflects opposing states of an external device, like the robotic soft glove. The brain switch idea reflects the same implementation of similar BCI studies [3], [7].

IV. Automatic Labeling

This section explains the steps to prepare the raw EMG signals to detect muscle activity and label training samples automatically. Our proposed clustering algorithm and movement detection consist of four iteratively traversed steps. We show the overall process for movement detection in Figure 3. Following this, we illustrate how we use movements detected in the EMG signals to label the EEG signals automatically.

A. EMG Preprocessing

The EMG signals are central in the labeling process of our training samples. Cue-based studies utilize timed cues to annotate the movements, enabling labeling of where in the EEG signals movements are present. Data is labeled at the cue's position regardless of whether the individual has executed the movement at the specified time cue, which can lead to misalignment of the movement in training samples. Our data acquisition technique does not include timed cues; instead, the subjects initiate movements in a self-paced manner as described in Section III. We record EMG signals from skeletal muscle activity, which allows us to detect when the movements are executed and align them with the EEG signals.

The movements are characterized as localized bursts in the EMG signal. We use a threshold-based technique proposed by Solnik et al. [32] to distinguish movement from rest. We enhance the detection of movements by conditioning the raw EMG signals before applying the threshold. We apply a

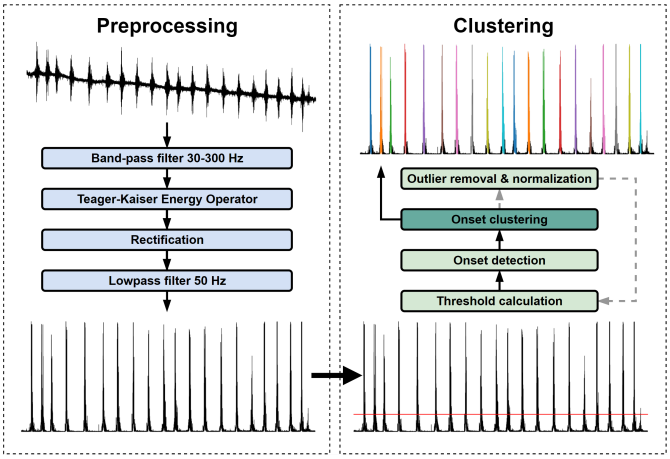


Fig. 3: Preprocessing and clustering of the EMG signal. Initially, the raw EMG signal is bandpass filtered at 30–300 Hz using a sixth-order Butterworth filter before being conditioned with TKEO. Then the signal is rectified and lowpass filtered at 50 Hz before being passed on to the clustering algorithm. The clustering algorithm takes the processed EMG data and calculates a linear threshold to extract the data points considered movements. We cluster each group of movements based on proximity to each other. We iteratively detect and remove outlier clusters with the first set of clusters, whereas we apply normalization in the end. This process, denoted by the gray dashed arrows, continues until it identifies 20 clusters.

sixth-order Butterworth bandpass filter at 30–300 Hz to the raw EMG signals to eliminate baseline drift and movement artifacts. Hereto, we use the FILTFILT and BUTTER implementation from SciPy [33], [34]. Butterworth filtering, an important component of our EMG and EEG preprocessing, is a non-causal filtering method. This means it relies on future input, i.e., the entire trace, thus we cannot apply it in real-time [35]. We then apply Teager-Kaiser energy operator (TKEO) conditioning to enhance the signal-to-noise ratio and rectify the resulting signal. Finally, we apply a second-order Butterworth lowpass filter at 50 Hz to reduce high-frequency noise.

Solnik et al. [32] gives the equation for TKEO as:

$$\Psi_t = y_t^2 - y_{t+1} \cdot y_{t-1}, \tag{1}$$

where y_t is the EMG amplitude at time t.

On the preprocessed EMG signal, we apply a dynamic threshold from BioSPPy's onset detection package [36] called $thld$, which it calculates as follows.

$$thld = 1.2 \cdot \frac{1}{n} \sum_{i=1}^{n} |emg_i| + 2 \cdot \sigma(|emg|), \tag{2}$$

where emg is the EMG signal, n is the number of data points in emg, and σ is the standard deviation. We use this threshold to determine which data points should be considered a movement. For all the data points in the rectified EMG signal, we check if the data point exceeds the threshold. If

it does, we append it to the set of movement candidates. This process is described by the formula:

$$\forall\, y \in emg(|y| > thld \Rightarrow movements \leftarrow movements \cup \{y\}), \quad (3)$$

where emg is the EMG data, y is a data point in emg, $thld$ is the threshold, and $movements$ is the resulting set of all y larger than $thld$. The symbol "$\leftarrow$" denotes assignment.

We execute an iterative process with four steps to properly group movement candidates that are in proximity. We define sets of data points grouped together by this process as clusters.

Step 1: First, we apply our clustering algorithm. For every data point in the set of movement candidates found in Equation 3, we check if the time difference to the previous data point is less than a distance parameter $distance$, which we seed with a low value relative to the data's sampling rate. If this is the case, we add the data point to the cluster currently being formed. If it exceeds the distance parameter, we instead consider the data point the first data point in a new cluster. We capture this in the equation:

$$\forall\, movement \in movements \left(movement - movement_{\text{prev}} < distance \Rightarrow cluster \leftarrow cluster \cup \{movement\}\right), \quad (4)$$

where $movements$ is the list of all movements, $movement$ and $movement_{\text{prev}}$ are data points in $movements$, $distance$ is the maximum allowed distance for adding a movement to the cluster currently being formed, and $cluster$ is a movement cluster. Since we expect a trace to consist of 20 movements, we iteratively increase the distance parameter until we detect 20 or fewer clusters. If there are fewer than 20, we decrease the distance parameter in smaller steps until we detect 20 or more clusters. However, the detected clusters are not necessarily correct, so further processing is required, which is handled by the following steps.

Step 2: We apply a proximity heuristic to refine the clusters detected in the initial step. We merge clusters that are close to each other but not close enough so that the clustering heuristic in *Step 1* creates a single cluster. The distance between the end of a cluster and the start of the next is calculated for every cluster in the trace. The process merges any two clusters that are less than 50% of the median apart with regard to the distance value. We determine this value by the data acquisition settings and empirical testing. We define the equation for this heuristic by:

$$\forall\, cluster \in cluster\_list \left(|cluster_{\text{end}} - cluster_{\text{next.start}}| < \text{MEDIAN}(allDist) \cdot 0.5 \Rightarrow cluster \leftarrow cluster \cup \{cluster_{\text{next}}\}\right), \quad (5)$$

where $cluster$ and $cluster_{next}$ are clusters, $cluster\_list$ is the list of all clusters in the trace, MEDIAN is the function that returns the median value of a list, and $allDist$ is a list of the distances between each cluster and its next cluster in the trace.

Step 3: Outlier clusters are removed based on their size relative to the rest of the trace. The process removes a cluster with fewer data points than 10% of the median cluster size. This value is determined empirically through testing and may need tuning for other datasets. We convert the $cluster\_list$ to a $cluster\_set$. We show the outlier removal with the formula:

$$\forall\, cluster \in cluster\_set \left(cluster_{\text{size}} < \text{MEDIAN}(allSize) \cdot 0.1 \Rightarrow cluster\_list \leftarrow cluster\_set \setminus cluster\right), \quad (6)$$

where $cluster$ is a cluster, $cluster\_set$ is the set of all clusters in the dataset, and $allSize$ is a list of the sizes of each cluster in the dataset.

Step 4: We then normalize the clusters by reducing any cluster whose peak, the highest rectified data point in terms of amplitude, is larger than the median value of the first quartile (Q1) of all the clusters' peaks. We do this until all the cluster's data points are less than Q1, as we capture in this equation:

$$\forall\, cluster \in cluster\_set \left(cluster_{\text{peak}} > Q1 \Rightarrow cluster \leftarrow cluster \cdot 0.5\right), \quad (7)$$

where $cluster$ is a cluster, $cluster\_set$ is the set of all clusters in the dataset, and $Q1$ is the first quartile's median calculated from all the clusters' peaks. We perform normalization to reduce the amplitude discrepancy in the data. This is especially beneficial in the test traces, where opening and closing movements are present in the same trace. These movements present different amplitudes on the muscle from which we record EMG signals. We then use the normalized EMG signals to perform clustering again, but we skip outlier removal and normalization in this iteration. This adjusts the threshold calculated by Equation 2 and allows the clusters to encapsulate more of a movement, usually more of the movement's start and end.

Algorithm 1 Movement Candidate Detection

Input: emg ▷ EMG signals
1: **for** $i \leftarrow 1, emg.length - 1$ **do**
2: $\quad emg[i] \leftarrow emg[i]^2 - emg[i+1] \cdot emg[i-1]$ ▷ TKEO
3: **end for**
4: $thld \leftarrow 1.2 \cdot \frac{1}{emg.length}\text{SUM}(|emg|) + 2 \cdot \text{STD}(|emg|)$ ▷ Threshold calculation
5: $movements \leftarrow [\,]$
6: **for** y **in** emg **do**
7: $\quad$ **if** $y > thld$ **then**
8: $\quad\quad movements.insert(y)$ ▷ Find movement candidates
9: $\quad$ **end if**
10: **end for**
11: **return** $movements$

Algorithm 1 combines Equations 1, 2, and 3 in abstract pseudocode. The variable names used are consistent with those used in the individual equations, except for $emg.length$,

which is n in Equation 2. Lines 1–3 encapsulate Equation 1, line 4 corresponds to Equation 2, and lines 6–10 mirror Equation 3.

Algorithm 2 Movement Candidate Clustering

Input: $distance, movements$ ▷ distance value and movement candidates
1: $cluster\_list \leftarrow [\,]$
2: $cluster \leftarrow [\,]$
3: **for** $movement$ **in** $movements$ **do**
4: **if** $movement - movement_{prev} < distance$ **then**
5: $cluster.insert(movement)$ ▷ form cluster by proximity
6: **else**
7: $cluster\_list.insert(cluster)$
8: $cluster.clear$
9: $cluster.insert(movement)$ ▷ start new cluster
10: **end if**
11: **end for**
12: $median \leftarrow 0.5 \cdot \text{MEDIAN}(cluster\_list.distances)$
13: $size \leftarrow 0.1 \cdot \text{MEDIAN}(cluster\_list.lengths)$
14: **for** $cluster$ **in** $cluster\_list$ **do**
15: **if** $cluster_{start} - cluster_{prev.end} < median$ **then**
16: $cluster.append(cluster_{prev})$
17: $cluster\_list.delete(cluster_{prev})$ ▷ outlier detection by proximity
18: **end if**
19: **if** $cluster.length < size$ **then**
20: $cluster\_list.delete(cluster)$ ▷ outlier detection by size
21: **end if**
22: **while** $cluster_{peak} > Q1$ **do** ▷ Q1 = median value of first quartile of all peaks
23: $cluster.values \leftarrow cluster.values \cdot 0.5$ ▷ normalization by first quartile
24: **end while**
25: **end for**
26: **return** $cluster\_list$ ▷ set of clusters

Algorithm 2 combines Equations 4, 5, 6, and 7. All variable names are as in the equations presented earlier in this section, apart from $cluster\_list$, which we do not convert to a set. Lines 2–10 represent Equation 4, and lines 11–24 cover Equations 5,6, and 7.

B. EEG Preprocessing & Labeling

Now that we have found the movements in the EMG signals, we can annotate and label the EEG signals to create training samples with movement and rest otherwise. We use the *movement onset* to denote a movement label, which we define as the first data point in a cluster of movement candidates. As described in Section I, MRCP appears naturally related to movement, making the movement onset the desirable annotating point. Shibasaki et al. [37] describe the signature of an MRCP as a slow negative shift in the low frequencies (0–5 Hz) of the EEG signal known as the bereitschaftspotential (BP), a negative peak around 0.4 s prior to the movement onset, followed by an amplitude increase (rebound phase) lasting around 1 s after the negative peak. Our preprocessing of the raw EEG signals includes segmentation, band-pass filtering, and normalization, as depicted in the initial part of Figure 4.

Instead of filtering the entire trace, we filter the signals within a data buffer with a size of 20 s with a second-order Butterworth band-pass filter at 0.5–4 Hz. This frequency range corresponds to the Delta-band [38], where the dominant frequency range of MRCP lies. Again, we utilize SCIPY'S FILTFILT and BUTTER implementation [33], [34]. The data buffer moves in steps of 100 ms across the trace of length $Trace_T$. The data buffer is required to meet the constraints of our online simulation environment, where the entirety of the data is unknown beforehand but arrives in real-time and can potentially be infinite. In Section V-B, we go over the data buffer in more detail.

For segmenting, we introduce a sliding window denoted SW_T, with a size of 2 s. This size ensures we can cover the defining parts of MRCP at the movement onset. We always retain the sliding window in the most recent part of the data buffer, which means that the sliding window moves with the data buffer. Subsequently, we apply z-score normalization across all training samples.

Whenever the sliding window intersects with a movement onset, we label the entire sliding window as a movement, as opposed to rest, and consider it a training sample. Thus, we create a total of 20 movement samples per detected movement.

V. Online Simulation Pipeline

As shown in the rightmost part of Figure 4 and explained in further detail below, XGBoost and ViT require different input. Thus, we first propose two separate pipelines for feature extraction and classification: one for XGBoost and one for ViT. Then, we present the overall online simulation test suite depicted in Figure 5; and finally, we present the online evaluation metrics utilized in post-analyses of each online simulation.

A. Feature Extraction and Classification

As explained in Section I, MRCPs are patterns in EEG signals that encapsulate movement intention and execution. In Section IV-B, we presented the defining signature of MRCP, which are the features we aim to extract. We present distinct feature extraction and classification pipelines for XGBoost and ViT below. Both feature extraction methods are applied to the training samples.

1) XGBoost: We extract three statistical features and two temporal features from each channel of the training samples. The statistical features are the *mean*, *kurtosis*, and *skewness*. The temporal features are the *peak-to-peak time window* and *peak-to-peak slope* proposed by Kalatzis et al. [39]. We alter these features to not only consider the P600 component, where features are extracted in the $500\text{ms} \leq t \leq 800\text{ms}$ time interval, but to use the entire sliding window instead. A movement intention can be located anywhere in a sliding window in an online environment, and the classifier will have to find and

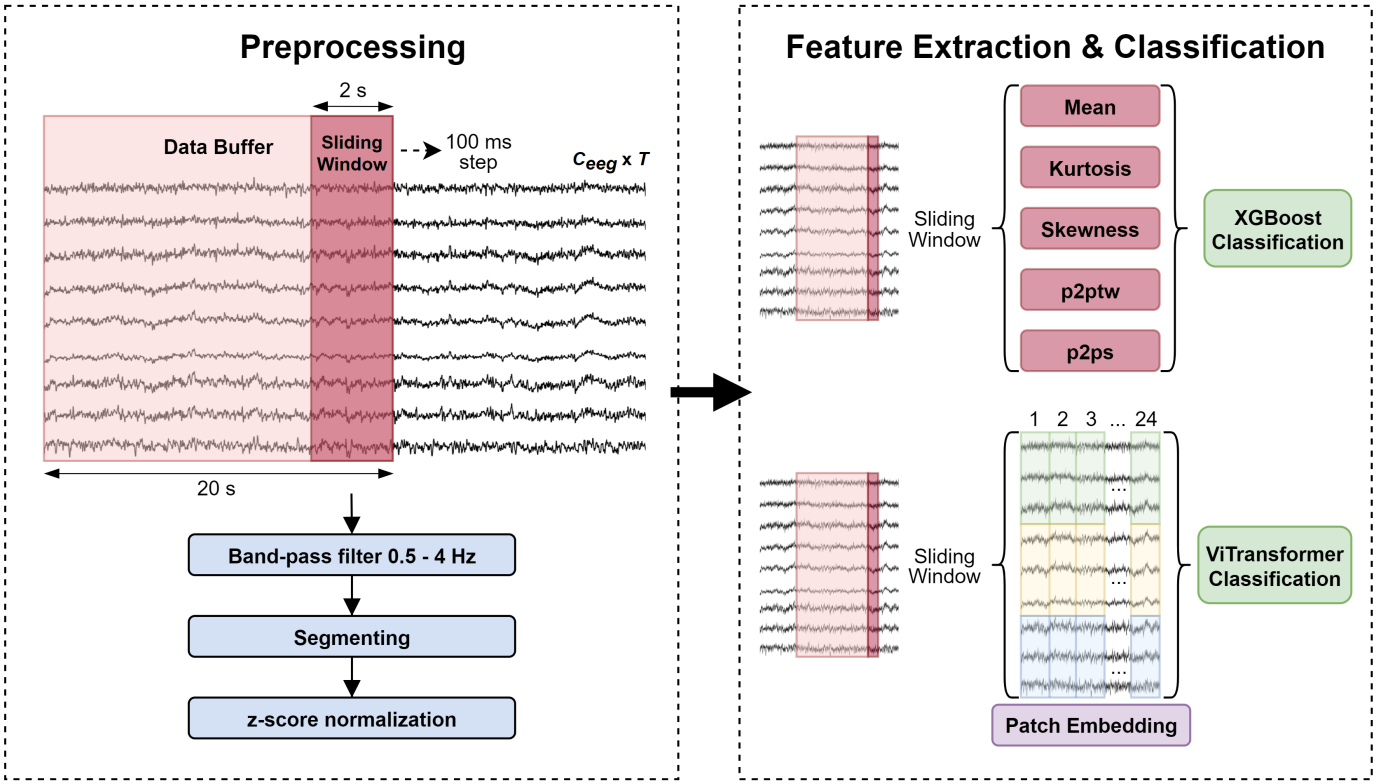


Fig. 4: Preprocessing and feature extraction of EEG signals. We build a 20 s data buffer to continuously filter the EEG signals with a second-order Butterworth band-pass filter at 0.5–4 Hz as it moves in 100 ms steps across each trace. The last 2 s of the buffer are considered the sliding window from which we extract training samples. From each sample, we extract features by applying patch embedding for the transformer and temporal feature extraction for our XGBoost model.

classify movements in various situations. We define the three statistical features as:

$$SW_{mean} = \frac{1}{n}\sum_{i=i}^{n} SW_i \tag{8}$$

$$SW_{kurtosis} = \frac{1}{n}\frac{\sum_{i=1}^{n}(SW_i - \overline{SW})^4}{\sigma^4} \tag{9}$$

$$SW_{skewness} = \frac{1}{n}\frac{\sum_{i=1}^{n}(SW_i - \overline{SW})^3}{\sigma^3}, \tag{10}$$

where n the number of data points in a sliding window.

We define the first temporal feature as:

$$SW_{p2ptw} = t_{max(SW)} - t_{min(SW)}, \tag{11}$$

where $t_{max(SW)}$ is the time point within in the sliding window that has the highest amplitude value. The other temporal feature is defined as a slope given by:

$$SW_{p2ps} = \frac{max(SW) - min(SW)}{SW_{p2ptw}} \tag{12}$$

This results in an input feature vector of $C_{eeg} \times \{mean, kurtosis, skewness, p2ptw, p2ps\}$. We select and tune the parameters of XGBoost through a combination of empirical tests and Bayesian optimization. As a result, we find that the parameter settings depth = 13, subsample = 0.5, learning rate = 0.015, gamma = 1, and column subsample by tree = 0.9 to yield the best performance.

2) ViTransformer: We make the original ViT configuration [17] adhere to the dimensions of the EEG data from our data collection. ViT, in its original implementation, works on two-dimensional images. Initially, ViT performs patch embedding to divide the input into smaller patches. This allows the ViT to identify the defining features within the image by capturing similarities between patches. We can apply patch embedding to our training samples since a sliding window has two dimensions, as visualized in the right part of Figure 4. Our purpose is to perform image recognition to identify the defining signatures of MRCPs.

We first create the necessary patch embedding of a sliding window. For each sliding window, we create patches of size 3×100 and flatten the sequence for the linear projection input of the transformer. Formally, this means that we reshape input $x \in \mathbb{R}^{C_{eeg} \times SW_T \times K}$, where K is 1, into a flattened sequence of 2D patches $x_{patch} \in \mathbb{R}^{N \times (P_W \times P_H \times K)}$, where $P_W = 100$ and $P_H = 3$. The patch resolution $P_W \times P_H$ is relative to the data's sampling rate of 1200 Hz. This results in N number of patches, where $N = \frac{C_{eeg} \cdot SW_T}{P_W \cdot P_H}$, as there are no overlapping patches, meaning that the stride is equal to the patch resolution. To conclude, we extract features from 72 patches.

Using the self-attention mechanism of the transformer, we expect the classifier to learn the EEG signal's spatial and

temporal dependencies. We add position embeddings to retain positional information captured in the patches in the EEG signal. This helps the classifier learn the defining signatures of MRCPs. The multihead self-attention is implemented as described by Dosovitskiy et al. [17]. The input is first linearly transformed into vectors, queries (Q), keys (K), and values (V). We map the parameters *Q, K, V* of our patch input sequence:

$$[Q, K, V] = X_{patches} \in \mathbb{R}^{N \times D}, \tag{13}$$

where we map the flattened patches to D dimensions of a linear trainable projection.

The process of the attention mechanism is to compute a weighted sum for all values in the input sequence. We base the attention weights on the pairwise similarity of all elements in the Q and K representations. We express this as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{d_k}\right)V \tag{14}$$

For our implementation, this corresponds to calculating the similarities between pairs of patches capturing varying parts of the EEG signal. The extension to this approach is the multihead self-attention, where we calculate additional scores for each pair of patches. This results in the classifier having multiple similarity scores to which it can compare. We express the multihead self-attention as:

$$MSA(X_Q, X_K, X_V) = [head_1, \ldots, head_n] \tag{15}$$

$$head_i = Attention(Q_i, K_i, V_i) \tag{16}$$

When training the transformer, we use a learning rate of 0.001, a weight decay of 0.0001, 4 multihead self-attention, 64 projected dimensions, and a number of hidden units that is dependent on the input size. We use layer normalization between each transformer layer and apply dropout to the output.

B. Online Simulation

We propose a simulation suite that mimics a real-time procedure, where movement predictions are made on continuous data to control an external device. The online simulation builds upon our previous work [4]. Traditional online experiments depend on subjects undergoing additional testing and require more labor by the experimenters. Additionally, studies based on publicly available datasets may not have access to the same equipment used to collect the data; hence, they cannot evaluate their BCI in an online setting. Our simulation suite can help overcome such shortcomings. We show the steps of our online simulation in Figure 5.

We give raw data to the classifier in increments of 100 ms, in the same manner as we acquire data from a subject. We apply the same sliding window approach, as presented in Section IV-B, with a sliding window SW_T with a length of 2 s. The increments help to determine whether real-time inferencing is achievable, i.e., if processing a single sliding window takes longer than 100 ms, the system cannot be used in real-time environments and must be optimized. Because of the incremental approach, there is no way to preprocess the entire trace ahead of time. As in offline training, we must preprocess the raw data during the simulation before extracting features and supplying them to the classifier. We apply the same preprocessing steps to the data buffer as presented in Section IV-B. To meet these constraints, we implement a data buffer to allow preprocessing on a segment of the input trace. We can view the data buffer length as a trade-off between filtering the signal uniformly and processing the signal within our time constraint of 100 ms. A data buffer that is too short may not capture enough of the signal to be representative of the entire trace. A data buffer that is too long increases the computational complexity, which can make the process take longer than 100 ms at each step. Through empirical evaluation, we found that the size of $SW_T \cdot 10$, which in our implementation is 20 s, handles both sides of the trade-off sufficiently. Input data is accumulated in the buffer until it reaches its maximum capacity of 20 s. It is updated with each increment like a queue as long as the simulation runs. We start extracting features from the final two seconds of the data buffer once we build it, which corresponds to the sliding window.

Before we extract features and feed them to the classifier, we check for two things: whether we detect a blink within the sliding window and whether the system has executed a movement. We utilize NEUROKIT'S EOG module [40] for blink detection to clean our EOG signal. We use the built-in cleaning method, as proposed by Agarwal et al. [41]. When we detect a blink in the sliding window, we stop the feature extraction and prediction process until the blink is no longer in the sliding window. Blinks are known artifacts that distort the EEG signals, making it challenging for the classifier to detect MRCPs [42]. We also halt the prediction process for the remaining duration of a sliding window once the system executes a movement. This is referred to as the freeze time, a period during which the system can reset.

1) Dwell Parameter: The classifier makes a movement prediction at every 100 ms step, making the system too sensitive if every movement prediction triggers a movement execution. Therefore, we introduce a dwell heuristic, inspired by Savić et al. [3], to reduce the number of FPs and determine when the system should execute a movement. In a previous study, our dwell heuristic was static and checked if four of the five previous predictions were movement predictions before the system executed a movement.

Our new dwell heuristic is more dynamic and considers a prediction queue (PQ) of size $\frac{SW_T \cdot 0.5}{step}$, where step is 100 ms, which equates to ten. Hence, the PQ corresponds to the ten most recent predictions made at each step, where a single prediction is 0 for no movement and 1 for movement. We calculate the sum of the predictions in the PQ, which can be between 0 and 10, and apply a numeric threshold called the *dwell parameter* to the sum to determine if the system should execute a movement. However, as identifying a subject-specific dwell parameter is non-trivial, we propose an algorithmic solution. We refer to this process as dwell

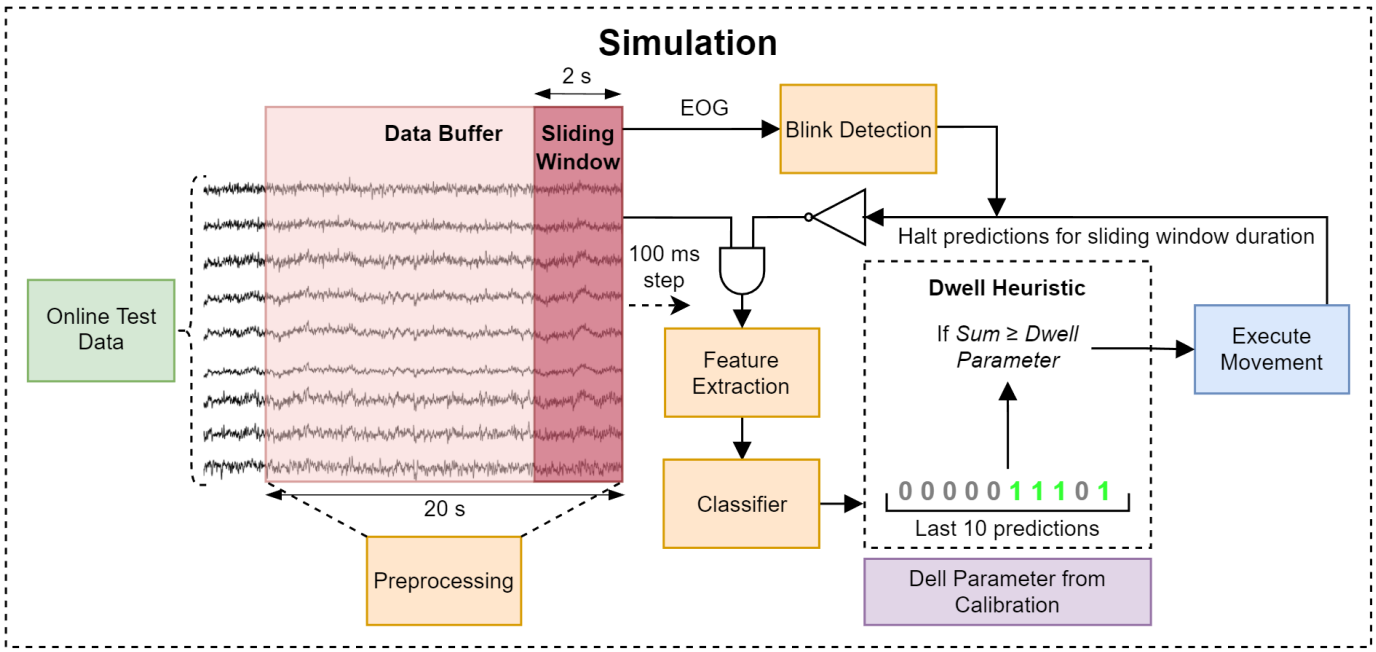


Fig. 5: Simulation overview. We give signals from the two online test traces to the system in 100 ms increments. We build a data buffer from the incoming data, similar to what we do in the offline EEG preprocessing steps. The last two seconds of the data buffer are considered the sliding window for preprocessing the data before extracting features. For each sliding window, we perform blink detection. If we detect a blink in a sliding window, we halt the process until the sliding window has moved past the blink. We append the classifier's predictions to our dwell heuristic that considers the last ten predictions from the previous sliding windows. We compare the sum of all predictions to the dwell parameter determined by the dwell parameter calibration performed beforehand. If the sum exceeds the dwell parameter, the system executes a movement, and we halt the process for the duration of a sliding window.

parameter calibration; a simulation step performed prior to the actual online simulation. During dwell parameter calibration, we run a short simulation on two unseen recorded training traces from the specific subject, meaning that we omit the two training traces from the offline training. We tune the dwell parameter on two different traces to aggregate an average performance. We disable the freeze time in the calibration to map the behavior of the classifier throughout the entire trace. For each PQ, we have an equivalent label queue (LQ), that holds the ground truth labels for each step. The LQ will be filled with movement labels when the sliding window crosses a movement onset.

Algorithm 3 presents the dwell parameter calibration steps. The input parameters $\hat{Y}$ and Y correspond to the chronological instances of the PQ and LQ, respectively, collected during each simulated trace for a single subject. In line 2 of Algorithm 3, we make a pairwise iteration of the PQs and LQs. In line 3 we check whether the sum of LQ is equal to ten, which means the LQ is intersecting a movement onset. If that is the case, we insert the corresponding PQ instance into $dwell\_\hat{y}$. Subsequently, we sort $dwell\_\hat{y}$ and find the median of all PQs' sum. The function EVALUATE, used in line 9 and 13, returns the number of movement onsets detected during a full traversal of a trace with a given dwell parameter, i.e., the number of triggered movement executions.

We see the introduction of the dwell parameter as a means of adjusting the trade-off between sensitivity and specificity.

Algorithm 3 Dwell Parameter Calibration

Input: $\hat{Y}, Y$

1: $dwell\_\hat{y} \leftarrow [\,]$
2: **for** $\hat{y}, y$ **in** zip$(\hat{Y}, Y)$ **do** ▷ iterate both lists pairwise
3: **if** SUM$(y) = \frac{SW_T \cdot 0.5}{step}$ **then** ▷ find movement onset
4: $dwell\_\hat{y}.insert(\text{SUM}(\hat{y}))$ ▷ insert corresponding prediction
5: **end if**
6: **end for**
7: QUICKSORT$(dwell_\hat{y})$
8: $dwell \leftarrow \text{MEDIAN}(dwell\_\hat{y})$
9: $initial\_clusters\_hit \leftarrow \text{EVALUATE}(dwell)$
10: $clusters\_hit \leftarrow initial\_clusters\_hit$
11: **while** $initial\_clusters\_hit = clusters\_hit$ **do**
12: $clusters\_hit \leftarrow \text{EVALUATE}(dwell + 1)$
13: **end while**
14: **return** $dwell - 1$

A low dwell parameter triggers a movement execution more frequently, which can increase the number of TP movement executions at movement onsets but also makes the system susceptible to a higher number of FPs. On the other hand, a high dwell parameter may reduce the number of FPs but will result in more missed movement onsets. We try to find an attractive dwell parameter setting by increasing the dwell parameter until we begin to trigger fewer total movements in the trace. This retains the median performance of the classifier while reducing the total false positives.

C. Online Evaluation Metrics

In a previous study [4], we introduced an evaluation metric for online simulation called mean time to nearest MRCP (MTNM). Its purpose was to measure the mean time from a movement execution to the nearest instance of MRCP, which was labeled using the previous automatic clustering and shared local minimum algorithms. However, we concluded that this approach did not enable a robust evaluation metric with high certainty, as it was difficult to correctly label MRCPs due to the noise and artifacts of EEG signals. We also had to locate the MRCP placement in all nine EEG channels to create the final label, which was a non-trivial task. As a consequence, we have taken a new approach in this study.

The new metric measures the mean prediction latency (MPL) from a movement execution to the nearest movement onset. As a result, we base our evaluation on the labels created from the EMG signals instead of the EEG signals, as done previously. EMG signals contain an explicit representation of movement, as they are measured directly from the activated muscle. Therefore, they make more accurate and concise movement labels than EEG signals.

The MPL metric is beneficial for online evaluation, as movement executions considered FP may still occur close to a movement onset, either slightly before or after. In a real-world application, we could use MPL as a measure to deem FP's as an approved movement execution if it is within an acceptable, user-defined threshold. We adopt the same metric for all our predictions as well as only FP predictions in our online evaluation.

We define the mean prediction latency as:

$$MPL = \frac{1}{n}\sum_{i=1}^{n} \min_{x \in X} |\hat{y}_i - y_x|, \tag{17}$$

where, for each prediction, $\hat{y}$, we find the minimum distance in time to the nearest movement onset, y. Our clustering algorithm provides the movement onset labels as described in Section IV. X is the set of ground truth variables present in the trial.

Combined with the MPL, we also use precision defined as $Precision = \frac{TP}{TP+FP}$ as an evaluation metric. The precision metric lets us monitor the amount of FP predictions.

VI. Experiments & Results

We conduct a series of offline and online experiments on our recorded dataset to evaluate the proposed clustering algorithm and the XGBoost and ViT classifiers. We perform the online experiments using our proposed online simulation test suite. The experiments also cover the topic channel selection, where we apply statistical t-tests to determine the significance of using a different number of channels for classification.

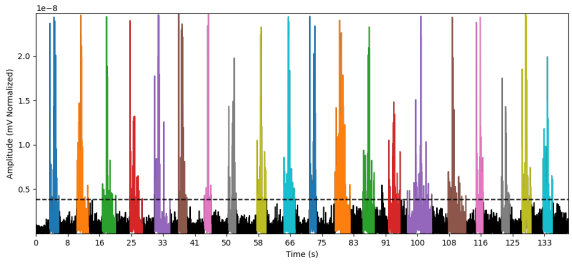
A. Labeling

We compare our automatic labeling method to the method used in our previous study [4]. In Table I, we show a comparison between the previous and the proposed method.

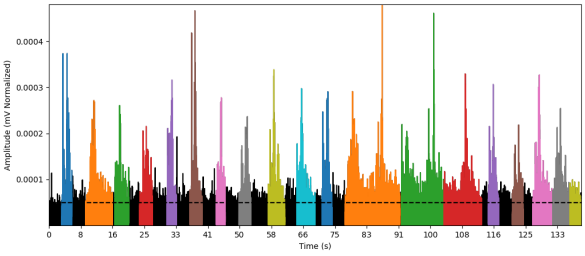
TABLE I: The number of movement clusters detected for each subject using the proposed and previous methods for grouping movements. The expected result is 20 movements per trace, of which there are 12. "*" denotes a left handed subject

	New method	Previous method	Expected result
Sub 0	440	440	440
Sub 1	440	426	440
Sub 2	440	440	440
Sub 3	440	434	440
Sub 4*	440	436	440
Sub 5	440	428	440
Sub 6	440	439	440
Sub 7	440	440	440
Sub 8	440	438	440
Sub 9	440	440	440

We observe that the previous method can detect the expected number of movements for Sub 0, 2, 7, and 9 but misses 1–14 clusters in the other cases. The proposed method achieves the expected result for all subjects. Although a method detects the expected number of clusters, this comparison does not reflect the created clusters' effectivity; in this case, that they each fully encapsulate exactly one EMG burst.



(a) Automatic labeling method for the grouping of movements.



(b) Earlier method for the grouping of movements.

Fig. 6: Comparison between (a) the automatic labeling method for movement detection and clustering, and (b) the method used previously [4]. Both methods were applied to the same trace from Sub 5.

Figure 6 compares the two methods' movement detection and clustering on a single trace. The specific example shows that our new labeling method can detect 20 distinguishable clusters, which is the desired result, whereas the previous method can only detect 19 clusters where some clusters encompass multiple EMG bursts. The automatic labeling method contends better with the noise in EMG signals due to the extra steps taken to condition the signals, yielding better movement

grouping. In other traces, the difference is less noticeable. The previous method performs well in most cases and often comes close to the total expected result in Table I, but fails when the data is noisy. The new method's preprocessing steps and iterative process can abstract well over different artifacts in EMG signals. This enables it to more accurately label noisy data, making it increasingly automatic and hands-off.

B. Experiment Parameters

We compare the performance of our XGBoost and ViT classifiers with the well-documented and established EEG-BCI classifiers mentioned in Section II, namely EEGNet [14], ShallowConvNet [15], DeepConvNet [15], and a default implementation of CSP-SVM [43], [44]. We implement the signal processing steps outlined by EEGNet [14], where the signals are filtered using a 0.1–40 Hz bandpass filter and downsampled to 256 Hz. In particular, since we record our data at 1200 Hz, we downsample it to 240 Hz, mimicking EEGNet's setup as closely as possible. To train and fit the deep learning and gradient boost trees, we utilize 5–fold cross-validation and use the hold-out data to determine early stopping. This helps us minimize the classifier's overfitting. The deep learning classifiers are all set to run for 300 epochs, but due to the validation accuracy not increasing, they stop early.

TABLE II: The number of trainable parameters for each given deep learning classifier.

	EEGNet	DeepConv	ShallowConv	ViT
T Parameters	1,233	143,351	17,521	12,228,033

EEGNet, DeepConvNet, and ShallowConvNet all use the same configuration for training. They use a batch size of 16 and the Adam optimizer [45] with a learning rate of 0.001. We train ViT using a batch size of 128 and with the epoch and early stopping configuration as the aforementioned classifiers and show the number of trainable parameters for each classifier in Table II. We use 159 as the seed for randomization in all implementations. CSP-SVM follows a default implementation from MNE [46] and SCIKIT-LEARN [47]. Lastly, XGBoost has $10,000$ estimators with an early stopping patience of 10 epochs. Additional hyper-parameters for the XGBoost and ViT are presented in Section V-A1 and Section V-A2, respectively.

C. Channel Selection

In an offline experiment, we evaluate the accuracy of using all available channels versus using only the channels placed over the primary movement hemisphere (PMH). For a right-handed subject, the PMH is the left hemisphere (C5, C3, C1), and for a left-handed subject, it is the right hemisphere (C2, C4, C6). Studies show that channel selection is relevant to consider [9], [23]. Reducing the number of channels makes the setup process less time-consuming and reduces computational complexity, hereby minimizing the system's overhead, which is important in a real-time system. We show results using 5-fold cross validation for each subject in Table III.

We see that the accuracy of XGBoost that uses handcrafted features is leading over all the deep learning approaches. However, XGBoost tends to overfit, and with early stopping, this is still a common occurrence. The accuracy achieved by XGBoost is consistent across all subjects, which is not the case for the other classifiers. We observe that classifiers with more trainable parameters, such as DeepConv and ViT, provide better results among deep learning classifiers. Referring to Table II, we see DeepConv and ViT have the most trainable parameters.

We also observe that the accuracy of the classifiers is consistently lower when predicting using the PMH channels compared to using all channels. We conduct a statistical t-test to compare the results of each classifier with a significance level set to $p < 0.05$. We present p-values in Table IV, where DeepConv, ShallowConv, ViT, and XGBoost show a statistical significant difference in accuracy. This means that in offline training, most classifiers can achieve better performance when predicting using data from all channels rather than from only the PMH channels. We hypothesize that the classifiers benefit from the increased information across the movement cortex from all channels to get a slight advantage during offline training. However, we are not certain whether these results translate to equal online performance and whether the PMH channels can capture sufficient information to achieve comparable results. We will investigate this in the next experiment.

D. Online Experimentation

We present an example of an online simulation conducted on the XGBoost classifier in Table V to explain each metric used to estimate the performance of a BCI in an real-time setting. We do so to show the intermediate results of a single classifier before presenting the average results of multiple experiments presented later in Table VII.

We have two test traces for each subject, as presented in Section III-C. The movements category in Table V denotes how often the system correctly triggered a movement execution, when the sliding window intersects a movement onset. The reported number of movements is slightly lower than the expected number of 40 because the data buffer is built from the first 20 s of data, making the classifier unable to predict any movements contained within that period. We also consider the precision metric and our MPL metric presented in Section V-C. We differentiate between two types of MPL: MPL(Total) indicates the mean prediction latency, i.e., the mean distance of all triggered movement executions to their nearest movement onset, and MPL(FP) shows the mean prediction latency for the FP movement executions. A low MPL indicates how far the movement executions are from their associated movement onsets, which can be either slightly before or after we expect them to be. Dwell is the last presented value, obtained through subject-specific offline calibration using Algorithm 3. In this specific instance, we observe values ranging from 4 to 8, in the dwell parameter between the subjects, indicating that subject-specific dwell tuning is necessary. However, in other cases, we observe dwell parameters ranging from 1 to 10.

TABLE III: Offline comparison of accuracy based on the primary hemisphere for hand movement. For a right-handed individual, the primary hemisphere is the left hemisphere, and for a left-handed individual, it is the right hemisphere. All numbers in the table are percentages, and we highlight the best results in bold. "*" denotes a left handed subject.

5×5 Fold	All Channels									
Subjects	Sub 0	Sub 1	Sub 2	Sub 3	Sub 4*	Sub 5	Sub 6	Sub 7	Sub 8	Sub 9
CSP-SVM	59.2 ± 0.8	53.6 ± 1.5	55.7 ± 0.7	55.7 ± 0.7	59.4 ± 0.4	56.6 ± 0.1	55.7 ± 0.4	59.5 ± 0.1	57.1 ± 0.3	54.3 ± 0.7
DeepConv	61.1 ± 0.1	71.3 ± 0.3	74.2 ± 0.3	63.3 ± 0.3	66.7 ± 0.2	66.8 ± 0.2	56.3 ± 0.2	71.3 ± 0.1	74.4 ± 0.1	72.2 ± 0.1
ShallowConv	58.8 ± 0.3	56.0 ± 0.1	59.1 ± 0.1	52.0 ± 0.1	60.3 ± 0.1	56.3 ± 0.2	55.1 ± 0.2	63.3 ± 0.1	61.2 ± 0.2	56.0 ± 0.3
EEGNet	62.1 ± 0.9	62.1 ± 0.7	66.2 ± 0.9	63.3 ± 3.1	65.1 ± 0.7	64.1 ± 1.0	55.1 ± 2.9	68.8 ± 0.4	70.1 ± 0.5	69.4 ± 0.3
ViT	64.5 ± 0.1	58.4 ± 0.7	66.1 ± 1.0	58.5 ± 0.2	67.9 ± 0.9	63.6 ± 0.6	65.4 ± 0.7	69.8 ± 0.7	74.8 ± 1.1	63.4 ± 0.7
XGBoost	$\mathbf{86.9 \pm 0.1}$	$\mathbf{85.9 \pm 0.1}$	$\mathbf{88.1 \pm 0.8}$	$\mathbf{86.3 \pm 0.4}$	$\mathbf{88.8 \pm 0.5}$	$\mathbf{84.1 \pm 0.6}$	$\mathbf{86.5 \pm 0.1}$	$\mathbf{87.1 \pm 0.1}$	$\mathbf{88.9 \pm 0.6}$	$\mathbf{86.4 \pm 0.9}$
5×5 Fold	Primary Movement Hemisphere									
CSP-SVM	58.8 ± 0.8	51.9 ± 0.1	55.1 ± 0.4	52.3 ± 0.6	58.1 ± 0.3	57.1 ± 0.2	55.6 ± 1.0	59.2 ± 0.8	57.1 ± 0.5	53.2 ± 0.9
DeepConv	59.7 ± 1.0	56.1 ± 0.8	64.1 ± 0.1	54.8 ± 2.0	61.8 ± 2.0	59.0 ± 6.4	55.0 ± 1.0	62.7 ± 1.0	64.9 ± 0.1	63.4 ± 0.7
ShallowConv	53.3 ± 1.2	50.9 ± 0.8	55.9 ± 2.0	49.0 ± 1.1	55.1 ± 1.0	54.2 ± 4.1	52.1 ± 2.0	56.3 ± 2.1	61.6 ± 2.5	51.9 ± 1.2
EEGNet	58.9 ± 0.1	61.3 ± 0.9	64.2 ± 0.7	58.4 ± 0.9	61.6 ± 0.5	64.1 ± 0.7	54.1 ± 1.8	62.2 ± 1.7	70.6 ± 1.4	63.6 ± 1.4
ViT	62.9 ± 1.1	53.1 ± 1.5	58.9 ± 0.6	52.1 ± 0.7	62.2 ± 0.8	57.7 ± 0.3	58.1 ± 0.6	67.7 ± 0.4	66.1 ± 1.3	59.4 ± 1.3
XGBoost	$\mathbf{82.2 \pm 0.4}$	$\mathbf{78.3 \pm 2.1}$	$\mathbf{83.3 \pm 0.3}$	$\mathbf{80.7 \pm 0.5}$	$\mathbf{84.3 \pm 0.6}$	$\mathbf{76.7 \pm 1.5}$	$\mathbf{80.3 \pm 1.3}$	$\mathbf{83.8 \pm 0.9}$	$\mathbf{84.6 \pm 0.7}$	$\mathbf{80.1 \pm 1.3}$

TABLE IV: A t-test with a significance level of 0.05, determining if there is a statistical significance between using all channels or channels located over the primary movement hemisphere. We highlight significant *p*-values in bold.

Classifier	CSP-SVM	D-Conv	S-Conv	EEGNet	ViT	XGBoost
p-values	0.222	**0.001**	**0.011**	0.127	**0.012**	**<0.001**

We show a visual representation of the predictions of a simulation of a single test trace in Figure 7. TP predictions are shown as green rectangles when they intersect a movement onset. The red rectangles denote FP predictions. We can interpret the MPL as the distance between each prediction and the nearest movement onset. We show detected blinks as small squares above the predictions. As described in Section V-B, we disallow predictions where blinks are detected and halt subsequent predictions for 2 s, hence why no predictions overlap with blinks.

TABLE V: Example of metrics used for online evaluation. We aggregate the results of the XGBoost classifier on two test traces for each subject. "*" denotes a left handed subject.

XGBoost	Primary Movement Hemisphere				
	Movement	Precision(%)	MPL(Total)(s)	MPL(FP)(s)	Dwell
Sub 0	21 / 33	43.75	0.84	1.50	5
Sub 1	28 / 34	40.57	0.96	1.62	4
Sub 2	18 / 33	30.51	1.27	1.82	7
Sub 3	17 / 34	30.91	1.14	1.66	6
Sub 4*	17 / 34	36.17	0.97	1.53	6
Sub 5	20 / 33	35.71	0.93	1.45	8
Sub 6	19 / 32	40.42	0.96	1.55	6
Sub 7	22 / 33	81.48	0.45	2.33	7
Sub 8	18 / 34	38.29	0.93	1.55	6
Sub 9	18 / 32	32.14	1.00	1.55	6

In the offline training of each classifier using all channels versus a subset of electrodes over the PMH, we see a consistent decrease in accuracy when using fewer channels. In Table VII, we show the results of the online evaluation and investigate if the advantage of using all channels in offline classification also carries over to online performance. We refer to the metrics aggregated from the evaluation as *All* channels and the *PMH* channels. For most classifiers, we observe that fewer electrodes result in a higher number of movements correctly detected by an average of 1.1%. In Table VI, we present a t-test that compares the outcome of using all channels versus the PMH channels for the online results. We do not observe any statistical significance across the movement or the precision metric for any classifier apart from ViT's MPL(FP), where the PMH channels lower the metric, reducing the latency of FPs. These results signify that similar or slightly better results are achievable using a smaller amount of electrodes during online use. Reducing the number of electrodes can also minimize the sources of noise and failures. This is an important advantage for practical use, as fewer electrodes make a BCI easier to manage [9], [23].

In addition to the observations mentioned above, we also observe that the difference in the dwell value between the two approaches is minor on average but not uniform across the same classifier, illustrating the need to tune a dwell-specific parameter for each subject and classifier. We also note that the performance gap between XGBoost and the other classifiers diminishes in the online results.

TABLE VI: A t-test with a significance level of < 0.05 to determine if any of the online results are significant. The values are calculated for each classifier based on Appendix A. We highlight significant p-values in bold.

	Movement	Precision	MPL(Total)	MPL(FP)
CSP-SVM	0.367	0.398	0.296	0.283
DeepConv	0.386	0.428	0.326	0.297
ShallowConv	0.475	0.287	0.433	0.361
EEGNet	0.329	0.348	0.091	0.099
ViT	0.343	0.450	0.131	**0.031**
XGBoost	0.119	0.316	0.367	0.168

Table VIII shows the average results of each subject across all the classifiers' performance. We observe a variance in performance between the subjects. Data from subjects 0 and 6

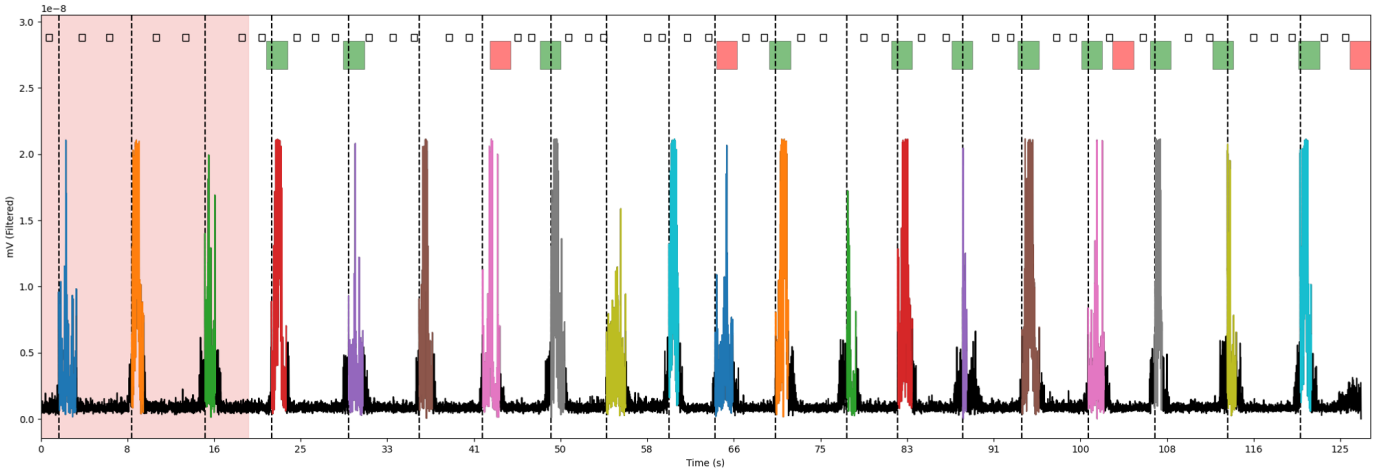


Fig. 7: Visualization of a simulation run on a single test trace from Sub 7, with a dwell value of 7, using the XGBoost classifier. We show each movement cluster, where a dashed line denotes the movement onset. The opaque red field shows the data buffer, which we build initially. The red and green rectangles denote predictions. Their size is relative to their respective sliding windows. We visualize the detected blinks as small squares above the predictions.

TABLE VII: The average results of our online evaluation comparing all channels to using only the electrodes placed above the PMH. All channels are referred to as *A-** and the PMH test as *P-**.

Channels Metric	All Mov(%)	PMH Mov(%)	All Prec(%)	PMH Prec(%)	All MPL(Total)(s)	PMH MPL(Total)(s)	All MPL(FP)(s)	PMH MPL(FP)(s)	All Dwell	PMH Dwell
CSP-SVM	52.7	56.1	33.3	32.9	0.99 ± 0.28	0.94 ± 0.23	1.55 ± 0.42	1.47 ± 0.25	4.6 ± 3.5	6.7 ± 2.9
DeepConv	53.1	54.1	38.2	38.0	$\mathbf{0.77 \pm 0.28}$	$\mathbf{0.82 \pm 0.27}$	$\mathbf{1.28 \pm 0.38}$	1.37 ± 0.38	5.8 ± 2.6	7.1 ± 2.0
ShallowConv	53.6	49.4	35.4	32.4	0.90 ± 0.29	0.88 ± 0.33	1.46 ± 0.44	1.41 ± 0.51	5.8 ± 3.1	4.4 ± 2.9
EEGNet	48.4	51.5	**39.4**	36.8	0.75 ± 0.21	0.88 ± 0.20	$\mathbf{1.28 \pm 0.40}$	1.46 ± 0.39	8.2 ± 1.9	6.7 ± 3.2
ViT	51.5	50.0	34.7	33.8	0.95 ± 0.26	$\mathbf{0.82 \pm 0.27}$	1.50 ± 0.32	$\mathbf{1.31 \pm 0.31}$	6.3 ± 1.3	6.8 ± 1.6
XGBoost	**54.5**	**59.6**	35.9	**38.7**	0.91 ± 0.18	0.94 ± 0.24	1.55 ± 0.41	1.65 ± 0.42	6.3 ± 0.9	6.3 ± 1.2

TABLE VIII: Average online simulation results of each subject across all classifiers for each online evaluation metric. "*" denotes a left handed subject.

	Mov(%)	Prec(%)	MPL(Total)(s)	MPL(FP)(s)	Dwell
Sub 0	64.1	47.6	0.76 ± 0.13	1.46 ± 0.31	7.4 ± 1.9
Sub 1	52.2	30.7	0.94 ± 0.12	1.37 ± 0.20	5.8 ± 1.8
Sub 2	55.8	28.8	1.22 ± 0.14	1.73 ± 0.23	5.8 ± 2.3
Sub 3	48.6	27.4	1.02 ± 0.15	1.41 ± 0.21	6.1 ± 2.9
Sub 4*	49.5	42.8	0.80 ± 0.29	1.41 ± 0.42	4.7 ± 2.2
Sub 5	55.8	32.5	0.92 ± 0.13	1.39 ± 0.21	8.3 ± 1.8
Sub 6	60.9	38.4	0.85 ± 0.13	1.43 ± 0.30	4.2 ± 2.7
Sub 7	47.7	63.1	0.40 ± 0.20	1.19 ± 0.71	7.6 ± 1.3
Sub 8	49.5	36.2	0.95 ± 0.12	1.62 ± 0.51	6.5 ± 1.9
Sub 9	50.9	30.5	0.94 ± 0.21	1.39 ± 0.34	6.1 ± 3.2

achieve the highest number of correctly identified movements, both above 60%. The most noteworthy is subject 7, achieving the highest precision of 63.1%. A high precision metric is desirable, and the value achieved approaches real-world usability. We deem precision more important than the lower number of movements. False negatives (FN) may inconvenience the user by making the system unresponsive, but they have less severe consequences than FPs that triggers unsolicited movements. A subject requiring a high dwell across all classifiers can indicate high certainty near the onset. However, a high dwell combined with a low precision clearly indicates highly noisy data. We show an extended table in Appendix A of all the results for each classifier. The best results of each metric are highlighted when comparing using all channels to those on the PMH.

Upon further investigation of subject 7, visualized in Figure 7, we discover that the subject blinked routinely 2–3 times between each movement. This may indicate that a subject can indirectly improve the system's performance by blinking, as a classifier might learn to expect blinking before or between movements. Nonetheless, this could also be a helpful observation in using the system, as long as it can detect movements that have no blink prior to it. If the incorporation of other movements can improve the detection of others, then that should be considered.

To sum up, our clustering algorithm method improves the performance of the previously used method on noisy EMG data. Our offline experimental results show that using all channels yields a performance increase, and the best performing and most consistent classifier in terms of accuracy is XGBoost. A noteworthy observation from the offline results is that XGBoost is the only classifier using handcrafted feature extraction. Our significance tests demonstrate a statistical significance in accuracy between using all channels or the PMH channels in our DeepConv, ShallowConv, ViT, and XGBoost classifiers. The online experiments indicate that only using the PMH channels improves performance on average and that good offline performance does not guarantee good online

performance. Our significance test on the online results further confirms that the difference between using all channels and the PMH channels is insignificant. Additionally, no recorded simulation run has exceeded the 100 ms step size constraint, confirming that real-time inferencing is achievable for all classifiers in our BCI pipeline.

VII. Discussion & Future Direction

We have proposed a new BCI pipeline focusing on MRCP detection during opening and closing hand movements, and we presented experimental results in the previous section. In this section, we reflect upon the pipeline's individual parts as well as our study's final outcome. We highlight certain aspects that have influenced our decisions and share experiences and observations that can guide future research.

A. Data Collection

In the initial part of this study, we collected self-paced data from ten healthy subjects. We have shown that our data and electrode layout can be used in a BCI pipeline and allow for hand movement predictions on both left and right-handed individuals. Furthermore, we managed to record a substantial number of movements from each subject during a single session without any significant fatigue reported from the subjects. Comparably, well-known EEG-BCI datasets [25], [48] available online have sessions on different days, which requires more manual labor from the experimenters and more time from the subjects.

We designed the data recording process with self-paced movements to reflect online use, allowing subjects to initiate movements at their own will, as opposed to based on a timed cue, to generate data that is more realistic of an online setting. Our only instruction to the subjects in terms of time was a mandatory 5 s or longer rest before and after each movement execution, without a conscious effort to count the seconds or focus on maintaining a constant pace. Some subjects, however, had difficulties following these instructions. We observed that a subject occasionally would, despite our best efforts of instructing them not to, fall into a pattern, e.g., following a blinking routine or not blinking at all, when executing movements. Figure 7 reflects this behavior, where we observe very even spacing between movements in addition to two or three blinks. We report this observation for future studies to ensure a greater variance of pause lengths between movements.

B. Automatic Labeling

The iterative process mentioned in Section IV, specifically *Step 1*, takes advantage of the fact that we know the expected number of movements in a trace. However, the process can operate without this knowledge and performs well under this condition. The process checks whether the previous iteration's movements are identical to the current iteration's movements and ends if it meets this condition. This method of grouping movements is identical to the previously used method but without the iterative process.

The automatic labeling process contains manually tuned parameters, i.e., the outlier detection (Equations 5, 6). The constants are tuned empirically for the data we use. However, the parameter settings may not generalize well to other data or to other types of data.

A shortcoming of self-paced datasets using EMG signals to label ground truths, especially automatically, is that there is no guarantee that labels are correct. A signal artifact, an accidental movement, or an erroneous cluster made by the method could be detected as a movement and labeled as such. Therefore, we consider the automatic labeling method to provide an estimate.

We believe that our labeling approach, where a 2 s sliding window with a 100 ms step size dictates our training samples, lets our classifiers learn very diverse features. This means that we get 20 individual training samples from a single movement onset. As a result, the classifiers can identify different parts of the signal denoting movement, but also increases the risk of false positives, as the classifier may become too sensitive. For example, training samples that only capture the first or last part of an MRCP, without the entire BP, negative peak, and rebound phase, could be confused as a rest sample with minor noise. However, there are other ways to dictate the sampling process. A simpler approach, where we only create a single training sample centered at each movement label, could reduce the sensitivity because a classifier would then only recognize a movement by the full signature of an MRCP. This, however, would come at the cost of having fewer training samples. In summary, we believe that our current labeling approach creates a more robust classifier, especially when combined with our dwell heuristic, which handles the sensitivity and specificity trade-off.

In a different study, Savić et al. [3] stopped labeling data for five seconds after annotating a movement in coherence with the freeze time introduced in their online experiments. This means they do not train their classifiers on the data that occurs after a movement onset. If their classifiers miss a movement, the subsequent input will be regarded as unseen data with respect to the classifier, as the classifier is not trained on data after a movement. In contrast, we do not consider the freeze time during sampling and continuously label the entire trace that we then use to train our classifier. Thus, if our classifier misses a movement, it can potentially infer that the next input succeeds a movement reducing the possibility of a false positive.

C. Classification

In the offline evaluation, cf. Table III, we see classifiers achieving accuracy scores in the range 52.0–88.9% using all channels and in the range 49.0–84.6% using the PMH channels, depending on the subject. We can compare our results to similar studies with binary self-paced movement intention recognition. Hernández et. al [49] have a data collection process similar to ours. They test binary classification of a subject performing both rotating and pulling motions with the same hand. They achieve 59–73% accuracy across seven

subjects. A major difference between our study and theirs is the electrode setup, which uses a 64-channel setup, compared to our 9-channel setup. We also concluded that having more channels is beneficial for offline evaluation. We can attribute their marginally higher mean accuracy to their data sampling approach. They only include samples of their two types of movements. In contrast, our study includes samples of the entire trace.

We decided not to apply direct artifact removal other than band-pass filtering during offline labeling and training to make our classifiers more robust to an online environment. We could have achieved a higher offline accuracy if we applied active artifact removal. However, it would be detrimental to online classification. Like the freeze time approach, removing artifacts would leave the classifier untrained on EEG artifacts, much like not including training samples after a movement.

D. Online Simulation & Evaluation

Only few studies focus on online evaluation, and even fewer consider MRCP detection. Our online simulation evaluation method allows for estimating a BCI's performance during online use and reduces the overhead involved when conducting online experiments. This is a step forward with respect to addressing the lack of online evaluation in BCI studies. However, our simulation test suite cannot entirely replace real online assessments. Instead, we should interpret the simulation results as resembling real-use performance in future research. Real online experiments introduce additional cognitive load, making the task more complex for the subjects. This is known to cause significant changes in MRCP signatures and thus to affect movement prediction [50], [51]. Nonetheless, our simulation test suite is especially beneficial for studies conducted on BCI datasets available online, which may not have access to the data recording equipment, and thus cannot conduct online experiments.

We have shown that offline results do not map directly to good online results. Our proposed pipeline implements the data buffer, dwell heuristic, and freeze time, which heavily influences how we should evaluate the BCI. As a result, the accuracy score utilized in most offline studies cannot adequately capture the performance of a BCI when used in an online setting and may make it more challenging to achieve high performance results. This is a compromise, which may not be well-received by all from an offline-study perspective, but we deem it necessary to advance towards a more common and realistic evaluation process, which is currently non-existing. Current online evaluation methods often do not provide sufficient details or make fair evaluations, e.g., using offline evaluation for online use. The evaluation is affected by the representation of the prediction labels and the data handling. This aspect must also be taken into consideration when evaluating the results.

E. Dwell Parameter & Freeze Time

As mentioned above, the online performance is heavily influenced by the simulation's dwell heuristic and freeze time. However, we deem using a dwell parameter necessary since working in an online or real-life environment requires ways to reduce FPs, as they are detrimental to using a BCI. We argue that our dwell parameter implementation is a good starting point for future researchers who wish to capture the median performance near a movement onset and use that as a threshold for movement execution to reduce sensitivity. However, the duration of the system's freeze period is arbitrary and we should adjust it according to the use case in which we deploy the BCI system.

We terminate calibration of the dwell value once it finds fewer movement onsets than the previous iteration. An alternative implementation is to increase the dwell value as long as it finds fewer FPs than the previous iteration. However, with this approach, the dwell value will often reach 10 and, in turn, introduce a high level of specificity, resulting in more FN movement executions. Our approach to calibration differs from that of Savić et al. [3], as they adjust the dwell manually by observing a subject. Instead, our algorithmic implementation focuses on historical data and determines a value using statistical observations, which we argue is not only more consistent and robust but also reduces the manual labor required.

F. Data Buffer

Real-time signal processing and artifact removal are recognized challenges in online studies [3], but existing studies often fail to present the essential details of overcoming these challenges. We use a data buffer to meet the challenges of applying a non-causal Butterworth filter in a real-time simulation. To the best of our knowledge, other studies do not apply this approach. During preliminary offline testing, we only observed a small decrease in performance from applying filtering on the data buffer as opposed to the entire trace. In Section V-B, we explain that 20 s of a trace is sufficient to capture enough signal variance to represent the remaining signal while still being able to process the data within our 100 ms step size constraint and get representative results. When we increased the data buffer size, the time complexity increased, and the computation failed to complete within 100 ms. However, we note that Butterworth filtering yields the best results when applied to the entire trace. We leave this observation open as a potential future direction on reducing online computational complexity to allow for better filtering and thus better performance. Another direction can be to explore the possibilities of extracting features from raw data, which would avoid this complication.

VIII. Conclusion

This study proposes an end-to-end brain-computer interface (BCI) pipeline based on electroencephalography (EEG) signals for online movement-intention classification. We collect EEG, electrooculography (EOG), and electromyography (EMG) signals from 10 healthy people who perform hand movements in a self-paced manner. We apply a proposed clustering algorithm to the EMG signals. The algorithm iteratively detects and

groups data points that exceed a threshold into movements that are then automatically cross-referenced with the EEG signals to label them as capturing movement or rest. Experiments show that the proposed clustering algorithm for movement onset detection in EMG signals performs better on noisy signals than a previously used method. We also show that our online simulation test suite allows for evaluation closer to real-time usage, and we provide evidence that good offline performance does not translate to good online performance in some cases. The proposed pipeline enables any classification method, but we propose two configurations using XGBoost and ViTransformer. An empirical comparison with other deep-learning baselines shows that XGBoost achieves the best overall performance. We provide detailed explanations of the designs of each of our proposed methods.

In summary, the pipeline composed of our methods, as well as the individual methods, has good potential for use in future studies of how to advance towards a more common evaluation process for BCIs.

ACKNOWLEDGMENTS

This master thesis project was conducted in collaboration with Aalborg University's Health Science and Technology Department. We would like to thank Strahinja Dosen for domain knowledge and guidance as well as access to the data recording equipment in the BCI-Lab. A special thanks goes to Dalin Zhang and Christian S. Jensen for their excellent supervision during this master thesis and previous semester project.

REFERENCES

[1] W.-k. Tam, T. Wu, Q. Zhao, E. Keefer, and Z. Yang, "Human motor decoding from neural signals: a review," *BMC Biomedical Engineering*, vol. 1, no. 1, p. 22, Sep 2019. [Online]. Available: https://doi.org/10.1186/s42490-019-0022-z

[2] A. Cruz, G. Pires, A. Lopes, C. Carona, and U. J. Nunes, "A self-paced bci with a collaborative controller for highly reliable wheelchair driving: Experimental tests with physically disabled individuals," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 2, pp. 109–119, 2021.

[3] A. M. Savić, S. Aliakbaryhosseinabadi, J. U. Blicher, D. Farina, N. Mrachacz-Kersting, and S. Došen, "Online control of an assistive active glove by slow cortical signals in patients with amyotrophic lateral sclerosis," *Journal of Neural Engineering*, vol. 18, no. 4, p. 046085, jun 2021. [Online]. Available: https://doi.org/10.1088/1741-2552/ac0488

[4] C. Hansen, F. DeFrène, and S. P. Kærgaard, "Automatic label creation of mrcp for online control of a robotic soft glove," Jan. 2022. [Online]. Available: https://projekter.aau.dk/projekter/files/460346716/cs_21_mi_9_13.pdf

[5] M. Jochumsen, M. S. Navid, R. W. Nedergaard, N. Signal, U. Rashid, A. Hassan, H. Haavik, D. Taylor, and I. K. Niazi, "Self-paced online vs. cue-based offline brain-computer interfaces for inducing neural plasticity," *Brain sciences*, vol. 9, no. 6, p. 127, Jun 2019, brainsci9060127[PII]. [Online]. Available: https://doi.org/10.3390/brainsci9060127

[6] R. Abiri, S. Borhani, E. Sellers, Y. Jiang, and X. Zhao, "A comprehensive review of eeg-based brain-computer interface paradigms," *Journal of Neural Engineering*, vol. 16, 11 2018.

[7] S. Aliakbaryhosseinabadi, S. Dosen, A. M. Savic, J. Blicher, D. Farina, and N. Mrachacz-Kersting, "Participant-specific classifier tuning increases the performance of hand movement detection from EEG in patients with amyotrophic lateral sclerosis," *J Neural Eng*, vol. 18, no. 5, Sep. 2021.

[8] I. K. Niazi, N. Jiang, M. Jochumsen, J. F. Nielsen, K. Dremstrup, and D. Farina, "Detection of movement-related cortical potentials based on subject-independent training," *Medical & Biological Engineering & Computing*, vol. 51, no. 5, pp. 507–512, May 2013.

[9] T. Alotaiby, F. E. A. El-Samie, S. A. Alshebeili, and I. Ahmad, "A review of channel selection algorithms for EEG signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2015, no. 1, p. 66, Aug. 2015.

[10] P. Wierzgała, D. Zapała, G. M. Wojcik, and J. Masiak, "Most popular signal processing methods in Motor-Imagery BCI: A review and Meta-Analysis," *Frontiers in neuroinformatics*, vol. 12, pp. 78–78, Nov. 2018.

[11] M. Saeidi, W. Karwowski, F. V. Farahani, K. Fiok, R. Taiar, P. A. Hancock, and A. Al-Juaid, "Neural decoding of eeg signals with machine learning: A systematic review," *Brain Sciences*, vol. 11, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2076-3425/11/11/1525

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[14] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016. [Online]. Available: http://arxiv.org/abs/1611.08024

[15] R. Schirrmeister, J. Springenberg, L. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for eeg decoding and visualization: Convolutional neural networks in eeg analysis," *Human Brain Mapping*, vol. 38, 08 2017.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[18] Y. Song, X. Jia, L. Yang, and L. Xie, "Transformer-based spatial-temporal feature learning for eeg decoding," 2021. [Online]. Available: https://arxiv.org/abs/2106.11170

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] U. Rashid, I. K. Niazi, M. Jochumsen, L. R. Krol, N. Signal, and D. Taylor, "Automated labeling of movement- related cortical potentials using segmented regression," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 6, pp. 1282–1291, 2019.

[21] V. Mondini, A. L. Mangia, and A. Cappello, "Eeg-based bci system using adaptive features extraction and classification procedures," *Computational Intelligence and Neuroscience*, vol. 2016, p. 4562601, Aug 2016. [Online]. Available: https://doi.org/10.1155/2016/4562601

[22] A. Valenti, M. Barsotti, D. Bacciu, and L. Ascari, "A deep classifier for upper-limbs motor anticipation tasks in an online bci setting," *Bioengineering*, vol. 8, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2306-5354/8/2/21

[23] M. Rashid, N. Sulaiman, A. P. P. Abdul Majeed, R. M. Musa, A. F. Ab. Nasir, B. S. Bari, and S. Khatun, "Current status, challenges, and possible solutions of eeg-based brain-computer interface: A comprehensive review," *Frontiers in Neurorobotics*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2020.00025

[24] M. Vijay, A. Kashyap, A. Nagarkatti, S. Mohanty, R. Mohan, and N. Krupa, "Extreme gradient boosting classification of motor imagery using common spatial patterns," in *2020 IEEE 17th India Council International Conference (INDICON)*, 2020, pp. 1–5.

[25] C. Brunner and R. Leeb, "Bci competition 2008 – graz data set a," 2008.

[26] G. Bressan, G. Cisotto, G. R. Müller-Putz, and S. C. Wriessnegger, "Deep learning-based classification of fine hand movements from low frequency eeg," *Future Internet*, vol. 13, no. 5, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/5/103

[27] Z. Gao, T. Yuan, X. Zhou, C. Ma, K. Ma, and P. Hui, "A deep learning method for improving the classification accuracy of ssmvep-based bci," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3447–3451, 2020.

[28] D. Zhang, D. Cao, and H. Chen, "Deep learning decoding of mental state in non-invasive brain computer interface," in *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, ser. AIIPCC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3371425.3371441

[29] B.-H. Lee, J.-H. Jeong, and S.-W. Lee, "Sessionnet: Feature similarity-based weighted ensemble learning for motor imagery classification," *IEEE Access*, vol. 8, pp. 134 524–134 535, 2020.

[30] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan, "Filter bank common spatial pattern (fbcsp) in brain-computer interface," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2390–2397.

[31] E. Pitsik, N. Frolov, K. H. Kraemer, V. Grubov, V. Maksimenko, J. Kurths, and A. Hramov, "Motor execution reduces eeg signals complexity: Recurrence quantification analysis study," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 2, 2020.

[32] S. Solnik, P. Rider, K. Steinweg, P. DeVita, and T. Hortobágyi, "Teager–kaiser energy operator signal conditioning improves EMG onset detection," *European Journal of Applied Physiology*, vol. 110, no. 3, pp. 489–498, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s00421-010-1521-8

[33] SciPy. scipy.signal.filtfilt. Accessed: 11-06-2022. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.filtfilt.html#id1

[34] ——. scipy.signal.butter. Accessed: 11-06-2022. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html

[35] F. Gustafsson, "Determining the initial states in forward-backward filtering," *IEEE Transactions on Signal Processing*, vol. 44, no. 4, pp. 988–992, 1996.

[36] BioSPPy. Biosppy biosignal processing in python. Accessed: 01-06-2022. [Online]. Available: https://biosppy.readthedocs.io/en/stable/biosppy.signals.html#biosppy.signals.emg.find_onsets

[37] H. Shibasaki and M. Hallett, "What is the bereitschaftspotential?" *Clinical Neurophysiology*, vol. 117, no. 11, pp. 2341–2356, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138824570600229X

[38] N. Bajaj, "Wavelets for eeg analysis," in *Wavelet Theory*, S. Mohammady, Ed. Rijeka: IntechOpen, 2021, ch. 5. [Online]. Available: https://doi.org/10.5772/intechopen.94398

[39] I. Kalatzis, N. Piliouras, E. Ventouras, C. Papageorgiou, A. Rabavilas, and D. Cavouras, "Design and implementation of an svm-based computer classification system for discriminating depressive patients from healthy controls using the p600 component of erp signals," *Computer Methods and Programs in Biomedicine*, vol. 75, no. 1, pp. 11–22, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260703001305

[40] NeuroKit2. Introduction. Accessed: 24-05-2022. [Online]. Available: https://neurokit2.readthedocs.io/en/latest/examples/eog.html

[41] M. Agarwal and R. Sivakumar, "Blink: A fully automated unsupervised algorithm for eye-blink detection in eeg signals," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 1113–1121.

[42] W.-D. Chang, H.-S. Cha, K. Kim, and C.-H. Im, "Detection of eye blink artifacts from single prefrontal channel electroencephalogram," *Comput. Methods Programs Biomed.*, vol. 124, pp. 19–30, Feb. 2016.

[43] Z. J. Koles, M. S. Lazar, and S. Z. Zhou, "Spatial patterns underlying population differences in the background EEG," *Brain Topography*, vol. 2, no. 4, pp. 275–284, Jun. 1990.

[44] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[46] MNE. mne.decoding.csp. Accessed: 13-06-2022. [Online]. Available: https://mne.tools/stable/generated/mne.decoding.CSP.html

[47] scikit learn. sklearn.svm.svc. Accessed: 13-06-2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[48] R. Leeb and C. Brunner, "Bci competition 2008 – graz data set b," 2008.

[49] L. G. Hernández and J. M. Antelis, "Self-paced movement intention recognition from eeg signals during upper limb robot-assisted rehabilitation," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019, pp. 69–72.

[50] S. Aliakbaryhosseinabadi, E. N. Kamavuako, N. Jiang, D. Farina, and N. Mrachacz-Kersting, "Classification of EEG signals to identify variations in attention during motor task execution," *J. Neurosci. Methods*, vol. 284, pp. 27–34, Jun. 2017.

[51] ——, "Influence of dual-tasking with different levels of attention diversion on characteristics of the movement-related cortical potential," *Brain Res.*, vol. 1674, pp. 10–19, Nov. 2017.

[52] H. Yuan and B. He, "Brain-computer interfaces using sensorimotor rhythms: current state and future perspectives," *IEEE transactions on bio-medical engineering*, vol. 61, no. 5, pp. 1425–1435, May 2014, pMC4082720[pmcid]. [Online]. Available: https://doi.org/10.1109/TBME.2014.2312397

[53] A. Tiwari and A. Chaturvedi, "A multiclass eeg signal classification model using spatial feature extraction and xgboost algorithm," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4169–4175.

[54] N. Mammone, C. Ieracitano, and F. C. Morabito, "A deep cnn approach to decode motor preparation of upper limbs from time–frequency maps of eeg signals at source level," *Neural Networks*, vol. 124, pp. 357–372, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360802030037X

APPENDIX A

TABLE IX: The results of all the online tests. The metric across each test is compared, and the highest value is highlighted. "*" denotes left handed subject.

Subjects	Sub 0	Sub 1	Sub 2	Sub 3	Sub 4*	Sub 5	Sub 6	Sub 7	Sub 8	Sub 9
All Channels - Movement										
CSP-SVM	45.5	47.1	**78.8**	52.9	**73.5**	24.2	81.3	54.5	**58.8**	**65.6**
DeepConv	57.6	38.2	69.7	61.8	41.2	51.5	59.4	60.6	52.9	37.5
ShalConv	63.6	35.3	63.7	55.9	35.3	60.6	56.3	39.4	**58.8**	62.5
EEGNet	69.7	47.1	60.6	29.4	41.2	51.5	53.1	42.4	41.1	50.0
ViT	63.6	55.9	57.6	29.4	55.8	54.5	50.0	45.5	50.0	53.1
XGBoost	63.6	61.8	54.5	38.2	61.8	60.6	50.0	54.5	41.1	59.4
Primary Movement Hemisphere - Movement										
CSP-SVM	60.6	58.8	45.4	52.9	58.8	**69.7**	71.9	45.5	50.0	46.9
DeepConv	69.7	41.2	63.7	58.8	44.1	63.6	53.1	48.5	44.1	28.1
ShalConv	69.7	55.9	27.3	**85.3**	29.4	63.6	62.5	33.3	50.0	50.0
EEGNet	**72.7**	47.1	51.5	23.5	50.0	60.6	**87.5**	33.3	50.0	40.6
ViT	66.7	55.9	39.4	50.0	50.0	48.5	46.9	45.5	44.1	53.1
XGBoost	63.6	**82.4**	54.5	50.0	50.0	60.6	59.4	**66.7**	52.9	56.3
All Channels - Precision										
CSP-SVM	41.0 ± 8.5	31.9 ± 3.8	27.5 ± 2.7	27.7 ± 0.1	43.1 ± 0.3	15.8 ± 2.1	34.4 ± 6.2	61.5 ± 4.5	33.1 ± 7.8	30.1 ± 5.2
DeepConv	38.5 ± 5.3	36.9 ± 10.6	30.7 ± 0.7	$\mathbf{33.8 \pm 3.1}$	39.6 ± 4.5	30.8 ± 8.2	45.7 ± 12.5	70.7 ± 13.5	35.8 ± 14.1	$\mathbf{42.8 \pm 10.1}$
ShalConv	43.6 ± 6.4	28.9 ± 8.6	31.6 ± 2.3	25.0 ± 9.0	45.0 ± 7.0	36.1 ± 1.5	37.3 ± 12.4	65.0 ± 7.1	46.6 ± 18.8	28.3 ± 2.8
EEGNet	$\mathbf{54.7 \pm 0.3}$	33.0 ± 9.8	$\mathbf{38.4 \pm 0.4}$	29.4 ± 13.5	42.9 ± 13.6	31.9 ± 4.0	41.6 ± 11.8	55.7 ± 8.1	33.6 ± 26.4	40.3 ± 9.9
ViT	42.0 ± 2.8	27.3 ± 2.3	35.0 ± 6.0	20.3 ± 6.5	42.0 ± 2.8	29.1 ± 7.7	40.0 ± 21.1	60.2 ± 9.0	35.9 ± 5.7	34.0 ± 2.8
XGBoost	50.0 ± 10.1	33.4 ± 14.0	27.8 ± 3.0	27.8 ± 13.6	$\mathbf{46.6 \pm 1.6}$	32.8 ± 5.9	32.3 ± 8.7	67.0 ± 13.9	28.3 ± 27.7	33.6 ± 13.6
Primary Movement Hemisphere - Precision										
CSP-SVM	48.9 ± 8.5	26.0 ± 1.4	20.5 ± 2.3	27.6 ± 0.1	42.7 ± 3.9	38.3 ± 5.6	40.6 ± 5.4	61.6 ± 11.7	30.9 ± 0.1	23.5 ± 3.2
DeepConv	52.3 ± 0.1	30.8 ± 17.9	34.5 ± 3.6	30.2 ± 9.1	46.4 ± 5.1	$\mathbf{39.6 \pm 1.6}$	41.9 ± 14.7	60.7 ± 15.1	37.0 ± 25.4	22.5 ± 10.6
ShalConv	50.0 ± 3.1	27.5 ± 1.4	19.8 ± 12.3	27.7 ± 7.9	45.0 ± 7.0	35.4 ± 4.3	35.4 ± 2.9	57.2 ± 18.1	35.3 ± 17.8	23.7 ± 7.9
EEGNet	54.4 ± 2.9	28.5 ± 2.1	33.7 ± 7.2	22.0 ± 0.9	44.7 ± 11.1	33.2 ± 8.8	34.7 ± 8.2	45.8 ± 0.4	$\mathbf{54.3 \pm 20.3}$	32.4 ± 1.2
ViT	48.1 ± 9.1	28.1 ± 8.7	20.1 ± 1.7	25.9 ± 2.3	38.7 ± 10.0	31.4 ± 0.5	$\mathbf{46.0 \pm 18.0}$	69.4 ± 27.5	31.9 ± 11.4	33.3 ± 3.6
XGBoost	43.7 ± 2.9	$\mathbf{40.5 \pm 0.5}$	30.1 ± 4.5	31.1 ± 3.0	36.1 ± 1.9	36.4 ± 7.3	40.1 ± 5.3	$\mathbf{81.5 \pm 4.2}$	38.1 ± 10.8	32.1 ± 3.4
All Channels - MPL (Total)										
CSP-SVM	0.77 ± 0.04	1.01 ± 0.04	1.39 ± 0.18	1.11 ± 0.07	1.39 ± 0.00	0.97 ± 0.22	0.91 ± 0.05	0.52 ± 0.26	1.01 ± 0.07	0.96 ± 0.07
DeepConv	0.77 ± 0.16	0.85 ± 0.10	1.16 ± 0.04	0.92 ± 0.14	0.65 ± 0.40	0.80 ± 0.24	0.73 ± 0.02	$\mathbf{0.20 \pm 0.02}$	1.03 ± 0.14	$\mathbf{0.58 \pm 0.10}$
ShalConv	0.80 ± 0.05	1.07 ± 0.05	1.24 ± 0.02	1.11 ± 0.07	0.77 ± 0.11	0.91 ± 0.17	0.89 ± 0.08	$\mathbf{0.20 \pm 0.11}$	0.93 ± 0.18	1.15 ± 0.14
EEGNet	0.81 ± 0.21	0.90 ± 0.05	$\mathbf{0.99 \pm 0.08}$	$\mathbf{0.67 \pm 0.14}$	0.73 ± 0.02	0.89 ± 0.10	$\mathbf{0.68 \pm 0.02}$	0.25 ± 0.03	0.88 ± 0.15	0.73 ± 0.17
ViT	0.77 ± 0.26	1.03 ± 0.13	1.28 ± 0.19	1.15 ± 0.14	0.84 ± 0.24	0.97 ± 0.24	0.94 ± 0.01	0.48 ± 0.12	0.87 ± 0.16	1.16 ± 0.22
XGBoost	0.75 ± 0.14	0.92 ± 0.07	1.23 ± 0.00	0.90 ± 0.03	0.88 ± 0.00	0.97 ± 0.02	1.02 ± 0.22	0.61 ± 0.00	0.93 ± 0.07	0.92 ± 0.24
Primary Movement Hemisphere - MPL (Total)										
CSP-SVM	0.86 ± 0.12	0.94 ± 0.14	1.22 ± 0.03	1.11 ± 0.07	1.06 ± 0.02	0.97 ± 0.08	0.86 ± 0.02	0.40 ± 0.19	1.09 ± 0.04	0.94 ± 0.17
DeepConv	0.61 ± 0.16	0.97 ± 0.17	1.01 ± 0.24	1.13 ± 0.04	0.55 ± 0.08	0.96 ± 0.04	0.77 ± 0.01	0.28 ± 0.03	0.98 ± 0.04	0.97 ± 0.26
ShalConv	0.69 ± 0.21	0.96 ± 0.18	1.38 ± 0.05	1.05 ± 0.04	$\mathbf{0.41 \pm 0.51}$	1.00 ± 0.06	0.86 ± 0.14	0.49 ± 0.26	1.07 ± 0.15	0.94 ± 0.21
EEGNet	$\mathbf{0.66 \pm 0.16}$	$\mathbf{0.81 \pm 0.09}$	1.32 ± 0.09	1.05 ± 0.09	0.70 ± 0.02	0.88 ± 0.12	0.96 ± 0.05	0.69 ± 0.09	$\mathbf{0.80 \pm 0.02}$	0.92 ± 0.17
ViT	0.75 ± 0.14	0.87 ± 0.25	1.12 ± 0.04	1.02 ± 0.02	0.65 ± 0.04	$\mathbf{0.87 \pm 0.12}$	0.69 ± 0.09	0.28 ± 0.25	0.97 ± 0.00	1.04 ± 0.03
XGBoost	0.84 ± 0.04	0.96 ± 0.07	1.27 ± 0.21	1.14 ± 0.08	0.97 ± 0.08	0.93 ± 0.25	0.91 ± 0.22	0.45 ± 0.31	0.93 ± 0.16	1.02 ± 0.09
All Channels - MPL (False Positive)										
CSP-SVM	1.33 ± 0.26	1.49 ± 0.15	1.94 ± 0.14	1.54 ± 0.12	2.44 ± 0.01	$\mathbf{1.15 \pm 0.23}$	1.40 ± 0.22	1.34 ± 0.52	1.52 ± 0.29	1.39 ± 0.01
DeepConv	$\mathbf{1.25 \pm 0.14}$	1.36 ± 0.07	1.68 ± 0.09	1.38 ± 0.14	1.07 ± 0.57	1.18 ± 0.48	1.39 ± 0.36	0.79 ± 0.43	1.68 ± 0.60	$\mathbf{1.05 \pm 0.37}$
ShalConv	1.42 ± 0.07	1.51 ± 0.09	1.82 ± 0.03	1.48 ± 0.07	1.39 ± 0.02	1.42 ± 0.30	1.47 ± 0.42	$\mathbf{0.55 \pm 0.20}$	1.93 ± 1.02	1.58 ± 0.19
EEGNet	1.79 ± 0.45	1.37 ± 0.28	1.61 ± 0.14	$\mathbf{0.95 \pm 0.03}$	1.32 ± 0.28	1.31 ± 0.07	$\mathbf{1.18 \pm 0.19}$	0.58 ± 0.02	1.49 ± 0.82	1.22 ± 0.10
ViT	1.33 ± 0.38	1.42 ± 0.14	1.97 ± 0.12	1.44 ± 0.05	1.47 ± 0.51	1.36 ± 0.18	1.66 ± 0.56	1.22 ± 0.02	$\mathbf{1.38 \pm 0.37}$	1.76 ± 0.42
XGBoost	1.57 ± 0.60	1.41 ± 0.18	1.71 ± 0.08	1.27 ± 0.28	1.65 ± 0.04	1.45 ± 0.09	1.54 ± 0.53	2.02 ± 0.86	1.44 ± 0.66	1.46 ± 0.68
Primary Movement Hemisphere - MPL (False Positive)										
CSP-SVM	1.65 ± 0.04	1.26 ± 0.16	1.57 ± 0.05	1.54 ± 0.12	1.85 ± 0.16	1.57 ± 0.00	1.46 ± 0.17	1.01 ± 0.21	1.58 ± 0.07	1.23 ± 0.18
DeepConv	1.29 ± 0.34	1.42 ± 0.11	1.56 ± 0.45	1.63 ± 0.14	1.04 ± 0.25	1.59 ± 0.11	1.37 ± 0.37	0.81 ± 0.41	1.65 ± 0.61	1.29 ± 0.51
ShalConv	1.41 ± 0.52	1.32 ± 0.27	1.75 ± 0.33	1.48 ± 0.07	$\mathbf{0.82 \pm 1.01}$	1.55 ± 0.00	1.33 ± 0.16	1.41 ± 1.22	1.76 ± 0.72	1.22 ± 0.14
EEGNet	1.45 ± 0.26	$\mathbf{1.12 \pm 0.09}$	2.01 ± 0.36	1.35 ± 0.10	1.30 ± 0.22	1.35 ± 0.36	1.49 ± 0.27	1.28 ± 0.16	1.95 ± 0.94	1.37 ± 0.22
ViT	1.49 ± 0.54	1.24 ± 0.49	$\mathbf{1.41 \pm 0.04}$	1.38 ± 0.07	1.08 ± 0.25	1.27 ± 0.16	1.32 ± 0.25	0.92 ± 0.00	1.45 ± 0.25	1.56 ± 0.65
XGBooost	1.50 ± 0.15	1.62 ± 0.11	1.81 ± 0.19	1.66 ± 0.19	1.53 ± 0.18	1.45 ± 0.23	1.55 ± 0.50	2.33 ± 1.20	1.55 ± 0.54	1.55 ± 0.00