university of groningen / faculty of science and engineering / artificial intelligence

Confidence-based Early Classification for Motor Imagery Brain-Computer Interfaces

Bachelor's Project Thesis

Mohamed Hassan
Supervisor: I.P. (Ivo) de Jong, MSc

Abstract: Brain-Computer Interface (BCI) research has seen advancements in recent years with regard to early classification and confidence. This project explores a confidence-based early classification method to improve the earliness and performance of Motor Imagery (MI) BCI. A dynamic early classification model was implemented using a stopping criterion based on prediction confidence to determine the optimal timing for classification decisions. This was compared to a static classification model by providing the optimal early stopping times found by the dynamic model. The models were tested on the BCI Competition IV dataset 2a (Brunner et al., 2008), using Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM) as classifiers. The models were evaluated on Accuracy, Cohen's Kappa, and Information Transfer Rate (ITR). Results showed increased or maintained performance for the confidence-based early classification models as compared to the static model.

1 Introduction

Brain-Computer Interface (BCI) systems are a growing and advancing technology, aiming to connect and create collaboration between the brain and computer. Interest in this technology stems from the hope that it could help individuals suffering from severe motor disabilities, affecting movement and communication. Many of these systems have been developed with vastly differing performances, architectures, and goals. This research presents a method that accounts for earliness, accuracy, and confidence.

1.1 Motor Imagery BCI

BCI systems allow individuals with neuromuscular impairments the ability to interact with the world by translating neurophysiological signals into control commands, bypassing the neuromuscular pathway (Wolpaw et al., 2000). The steps involved in such a system consist of the collection of brain data, followed by a set of preprocessing, feature extraction, and classification methods as can be seen in Figure 1.1.

By recording and translating a user's brain activity, such as with an electroencephalogram (EEG), the user can control neuroprosthetics. A Motor Imagery (MI) BCI relies on the similarity in the activation of the primary sensory-motor areas, between imagined motor movement and actual movement. Furthermore, the neuronal activity during MI results in the amplitude suppression or enhancement of the mu (7–13 Hz) and beta (13–30 Hz) frequencies known as event-related desynchronization (ERD) and event-related synchronization (ERS) respectively (Pfurtscheller & Neuper, 2001). The MI BCI then aims to translate these changes into discernible features and classify the movements into the intended user control commands. EEG data therefore allow for non-invasive analysis of the motor imagination of a user, allowing MI BCI to be used.

1.2 Confident systems

MI BCI systems rely on machine learning to interpret the motor intentions due to its role in automatically analyzing, modifying, and classifying EEG signals (Craik et al., 2019). However, despite its potential, the widespread adoption of machine learning models in clinical settings has been hindered due to skepticism surrounding their black-

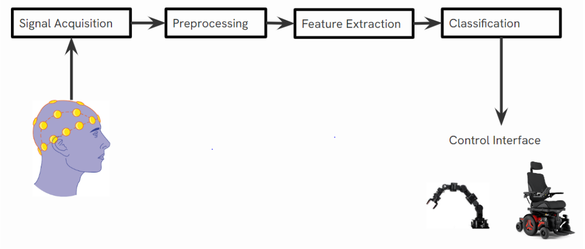

Figure 1.1: Schematic of a general MI BCI system.

box nature and the challenges posed by e.g. EEG noise sensitivity. This could lead to unpredictable and potentially harmful outcomes when predictions are made without any insight into how they were obtained or due to the uncertainty present in those predictions (de Jong et al., 2023). As machine learning models are being widely used for decision-making and inference, there is a need to evaluate their confidence before they are deployed. In this regard, there has been little but growing research into understanding when a classifier's prediction should and should not be trusted (Jiang et al., 2018).

There is, therefore, a need for confident and uncertainty-aware systems that are capable of withholding classification when they are unsure. Quantifying the uncertainty of the model prediction could help prevent harmful and unintended actions or commands from being executed by the BCI. This would enhance reliability and improve user trust and efficacy of BCI applications (de Jong et al., 2023).

1.3 Early classification

Real-time classification is crucial in BCI systems, as BCIs are largely aimed at mobility assistance or neurorehabilitation (Xu et al., 2014). To provide effective rehabilitation and assistance, classifications must be made with little delay. Therefore, in addition to a need for uncertainty-aware systems, one also needs classifications to have low latency.

It is important to distinguish between a dynamic and static model, as they are of focus in this research. When only considering earliness, a dynamic model is defined as an early classification method that is able to classify time-series data as early as possible (Akasiadis et al., 2022). This means that a dynamic model can classify without requiring the full time-series observations. It should find the earliest time points for which a reliable prediction can be made. A dynamic model should also maximize the tradeoff between accuracy and earliness. This model can further be combined with confidence-based classification.

A static model, however, is not an early classification model. It is unable to classify early, and can only classify either given the fully-observed time series or at pre-defined time points. While both models can classify early, the main distinction is that the dynamic model can perform early classification without predefined stopping times, meaning it can find the optimal early prediction times dynamically.

Early classification in general time-series data has been reviewed by A. Gupta et al. (2020). They describe models based on conditional probabilities to optimize the trade-off between earliness and reliability. They mention several early time-series classification models that implement reliability thresholds and stopping rules based on those thresholds, such as in M. R. Gupta et al. (2012).

Dynamic stopping methods, with respect to BCI, have been mostly researched in other BCI systems such as event-related potential (ERP) based or visual evoked potential (VEP) BCI. One such example is a VEP BCI system that works using a stopping criterion that is considered iteratively at each epoch (Spüler, 2017). Furthermore, Schreuder et al. (2013) evaluated several early stopping methods in ERP based BCI that had positive results. Among these, there is a method by Jin et al. (2011), in which a parameter J_{min} is used. The parameter set a minimum number of consecutive iterations which had to result in the same class prediction. This paper considers a similar early classification method, which is discussed further in Section 2. Another early classification method relying on a number of consecutive and similar predictions is the TEASER model described in Akasiadis et al. (2022). Renardi (2024) implemented confidence-based dynamic early stopping in the context of the P300 speller and found positive results with regard to both time and accuracy compared to non-dynamic early stopping models. However, dynamic stopping in MI, particularly when incorporating uncertainty quantification through confidence threshold-based dynamic stopping, remains relatively unexplored.

1.4 Project aim

There is, therefore, a need to consider the efficacy of confidence-based early classification in the context of MI BCI. This research aims to compare a dynamically classifying model, with a statically classifying model. The dynamic model uses uncertainty quantification to compare the classifier's prediction confidence to a confidence threshold for dynamic early stopping and utilizing similar methods such as in Jin et al. (2011) and Akasiadis et al. (2022). This research explores whether the described method maintains accuracy while improving time efficiency for MI BCI users. The project, therefore, aims to answer the following: Can a confidence-based dynamic classifying model classify as accurately as a static model for MI, given the same time windows?

As early stopping methods allow the BCI to adapt to the current state of the user, it allows the BCI to reduce the time it needs for a selection. Similarly, as they also allow the BCI to adapt to changes in the data, there is an increase in the robustness of the system (Schreuder et al., 2013). There have also been several early stopping methods that showed positive results in other BCI systems. It can, therefore, be assumed that such results could transfer to MI BCI. The expectation is that a dynamically classifying model will maintain classification accuracy comparable to that of a static model when evaluated over identical time windows.

In the following sections, the general methods used in the BCI model and the confidence-based early classification method are described in Section 2. Section 2 also includes the evaluation procedure and a description of the performance metrics. That is followed by the results and their analysis in Section 3. Finally, the findings are reflected on and recommendations for future research are made in Section 4.

2 Methodology

2.1 Dataset

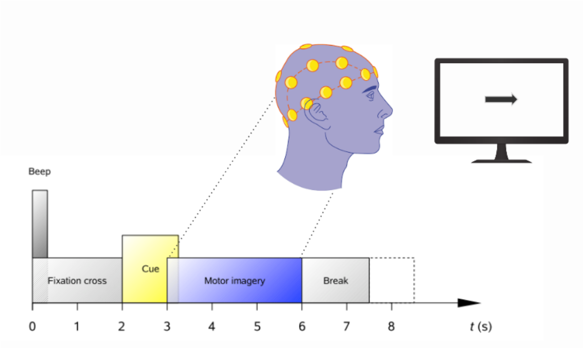

Figure 2.1: Timing of a trial. Adapted from Brunner et al. (2008).

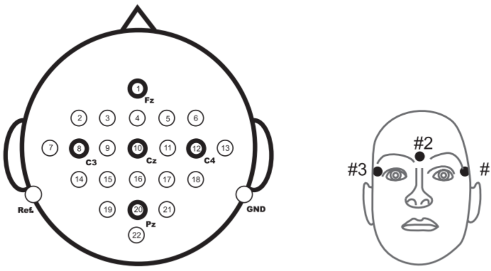

Figure 2.2: Electrode placement. Left: electrode montage corresponding to the international 10-20 system. Right: electrode montage of the three monopolar EOG channels for artifact detection (Brunner et al., 2008).

The BCI Competition IV data set 2a from Brunner et al. (2008) contains EEG data from 9 healthy subjects who were asked to perform four different MI tasks. Based on a given cue, the subjects performed the associated imagined movement of either the left hand (class 1), right hand (class 2), both feet (class 3), or tongue (class 4). They were recorded over 2 sessions. There were 6 runs in each session separated by short breaks. Every run had 48 trials, where each of the four classes had 12 trials, and therefore, 288 trials in each session. The subjects sat in comfortable chairs, with a computer screen in front of them.

As seen in Figure 2.1, at the start of each trial ($t = 0s$) a fixation cross appeared on a black screen with a short auditory warning tone. At $t = 2s$, a cue appeared on the screen in the form of an arrow, the direction of which (left, right, down, up) corresponded to one of the four classes (left hand, right hand, feet, or tongue). The subjects then performed the given task until $t = 6s$, after which there was a short break.

This paradigm was recorded using 22 Ag/AgCl electrodes, with a 3.5 cm distance between each electrode, shown in Figure 2.2. The EOG channels were omitted from the provided dataset and were not to be used for classification. The EEG signals were then sampled at 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz (Brunner et al., 2008). This dataset was obtained from Mother of All BCI Benchmarks (MOABB, v1.0.0) (Aristimunha et al., 2023).

Due to the two-session setup, the evaluation will consider the first session for training and the second session for testing in a session-to-session manner. This approach is a closer reflection of a practical BCI scenario where data is gathered during an initial session on which the model is trained and calibrated, and the system is expected to perform on subsequent data. Furthermore, during training, the entire trial is used, and during inference, the model evaluates window-wise.

2.2 Data preprocessing

2.2.1 Bandpass filtering

EEG signals suffer from artifacts and low signal-to-noise ratio (SNR), making the separation between the desired signal and the undesired background noise difficult (Wolpaw et al., 2002). MI BCIs also use multichannel EEG recordings to find MI patterns. Therefore, one needs to preprocess the recordings by removing undesired signals that are unrelated to MI to enhance SNR. This can be done by applying a band-pass filter within the desired frequency bands of mu (7-13 Hz) and beta (13-30 Hz) due to those bands being the ones related to motor imagination (Pfurtscheller & Neuper, 2001).

2.2.2 Window segmentation

After bandpass filtering, the data was segmented into either sliding or expanding windows. Hwang et al. (2023) showed that using sliding window segmentation increased performance, and helped to extract more discriminable features. In the case of this research, segmenting the data into windows is also desired in order to introduce early classification. This is so that the model can classify as soon as the early stopping criterion has been reached, without seeing the full-length data. An additional comparison in this paper is the between sliding and expanding windows, which will be discussed in Section 3.

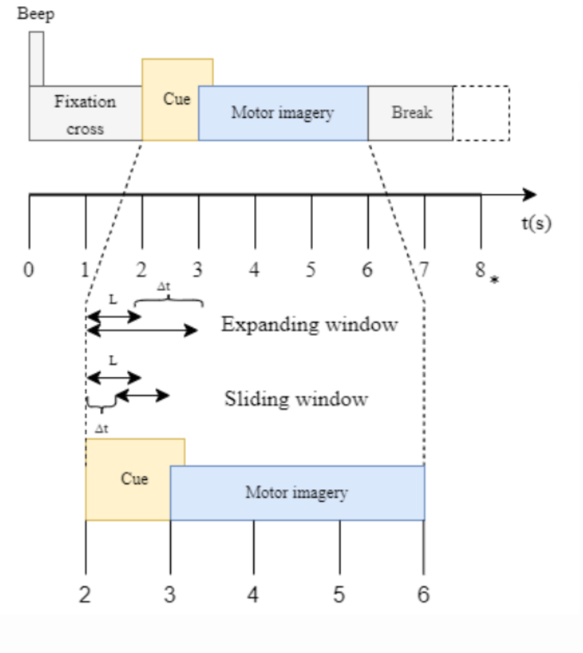

Figure 2.3: Sliding and expanding window segmentation. Adapted from: Hwang et al. (2023).

2.3 Feature extraction

Furthermore, due to the high dimensionality of the EEG data caused by the number of electrodes, feature extraction is employed as a form of dimensionality reduction. An effective feature extraction method in providing distinguishable features for MI BCIs is the Common Spatial Pattern (CSP) method (Vavoulis et al., 2023). CSP has also been used by many winners of BCI competitions (Tangermann et al., 2012). By utilizing feature extraction methods one can reduce the feature space, and increase the model's ability to discriminate the MI patterns. The goal of the CSP algorithm is to learn the optimal spatial filters that can transform the EEG data to maximize the variance of one class, while simultaneously minimizing the variance of the other classes based on band-power features (Padfield et al., 2019).

The problem in this research is solving a multiclass classification task while CSP was designed and generally utilized for two-class BCIs. The implementation of multiclass CSP that is used is one described in Grosse-Wentrup & Buss (2008). They describe a version of multiclass CSP that performs better than extended two-class CSP methods. As a combined feature extraction and feature selection process, they describe a method that uses Joint Approximate Diagonalization (JAD) to find the set of potential spatial filters, as well as use the mutual information between the class labels and the extracted components to choose the most optimal filters.

2.4 Classification

There are two main groups of machine learning algorithms: supervised, which relies on known outputs or labels, and unsupervised, which builds patterns without instructions. While both supervised and unsupervised methods have been deployed and assessed for EEG classification, supervised methods have had a higher accuracy on average (Hosseini et al., 2021). Furthermore, there were no substantial differences between traditional machine learning algorithms such as Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM), and deep-learning methods in terms of classification accuracy for MI (Vavoulis et al., 2023). Therefore, classic machine learning classifiers remain effective for MI EEG data classification and will be used in this paper. Additionally, this paper compares the performance of LDA and SVM classifiers, which is discussed further in section 3.

2.4.1 Linear discriminant analysis

The Linear Discriminant Analysis classifier aims to find the axes that maximize the separation between multiple classes through the projection of the provided data onto a lower dimensional space (Tharwat et al., 2017). Firstly, the model needs to calculate the separability between classes, called the between-class variance. Secondly, the model is to calculate the distance between the mean and samples of each class, which is called the within-class variance. Lastly, the model constructs the lower dimensional space, seen in Figure 2.4, that maximizes the between-class variance and minimizes the within-class variance.

In this project, learned CSP spatial filters are used to transform the MI EEG data features, which are then used as input for the LDA classifier to train against the known target values. The classifier can then predict the class labels for new unseen data points. The trained classifier can also provide class probabilities for new data points. To determine these probabilities, the LDA model first calculates a set of scores for each class, given the input data and the trained model. These scores are then converted into probability estimates for each class using the softmax function, which normalizes the scores to ensure that they sum to one across all classes. Each probability then represents the likelihood of the input data point belonging to a particular class (Pedregosa et al., 2011).

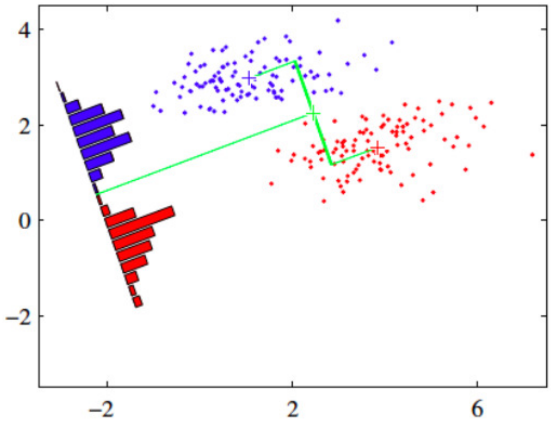

Figure 2.4: Example illustration: LDA projection (Li & Wang, 2014).

2.4.2 Support vector machines

Support Vector Machines map the data points into a higher dimensional space where a hyperplane is used to separate the classes with a maximum distance in between (Garg & Mago, 2021).

The plane $w * x - b = 0$, as seen in Figure 2.5, separates the data points into two classes. The goal of the model is maximizing the margin, to increase the separability, while incurring a penalty through misclassification or when a data point is within the margin boundary. This implements the soft maximum margin formulation, as problems usually are not perfectly separable. A slack variable is included to allow data points to be a certain distance away from their respective margin boundary. A small enough slack is essentially the hard maximum margin formulation. SVM can be extended to a multi-class solver by using one-vs-one (or one-vs-rest),

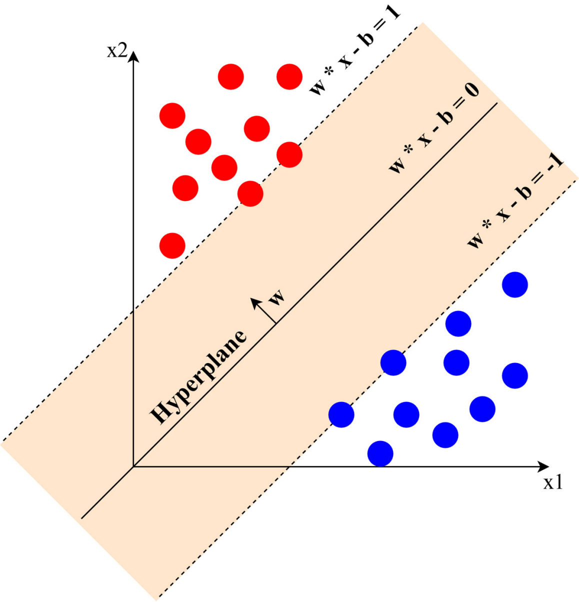

Figure 2.5: Example illustration: SVM hyperplane and margins (Garg & Mago, 2021).

resulting in one classifier for each pair of classes (or one for each class)(Pedregosa et al., 2011).

Similar to the LDA, the SVM classifier is provided with the transformed MI data to train on to find the optimal separating hyperplane and margins, which can then be used to classify and create probability predictions for new data points. Once the SVM classifier is trained, it calculates decision function scores for each class based on a data point's distance from the hyperplane. To estimate probabilities, SVMs employ Platt scaling, where a logistic regression model is fit to the decision function scores. The probability estimates indicate the likelihood of a data point belonging to each class based on its distance to the hyperplane (Chang & Lin, 2011; Wu et al., 2003).

2.5 Confidence-based dynamic classification

Definitions:

Window

A segmented section of time x_t of a trial. Denoted as:

$$x = [x_1, x_2, ..., x_t] \tag{2.1}$$

for time window $t \in \{1, 2, ..., T\}$.

Probability estimate

The probability output of the classification model at time window t for class c is denoted as:

$$p_{t_c} = p_\theta(y = c|x_t) \tag{2.2}$$

where θ is the trained model parameters, and y the predicted class label. The probability vector output of the classification model at time window t for all the classes is therefore:

$$\vec{p}_t := [p_{t_1}, p_{t_2}, ..., p_{t_C}] \tag{2.3}$$

$\vec{p}_t$ is a probability vector containing each estimated probability for class $c \in \{1, 2, ..., C\}$.

Dynamic model

Given the class probabilities from the classification model at a specific time window, we can estimate the uncertainty of the model prediction by using predictive entropy. Predictive entropy, given class probability estimates, measures the total amount of uncertainty over all the classes. Given the class probability estimate vector $\vec{p}_t$ and p_c being the probability for class c, the function is denoted as:

$$\mathbb{H}_{pred}(\vec{p}_t) = -\sum_C p_{t_c}\, log\, p_{t_c} \tag{2.4}$$

This allows one to capture the uncertainty of the model that arises due to a data point's distance to a decision boundary. The uncertainty can then be turned into confidence through:

$$Confidence(\vec{p}_t) = 1 - \mathbb{H}_{pred}(\vec{p}_t) \tag{2.5}$$

After measuring the confidence of the model's predicted probability estimates for a window, it is compared to the confidence threshold hyperparameter τ. The following indicator function is used for this comparison:

$$I(Confidence(\vec{p}_t)) = \begin{cases} 1 & if\ Confidence(\vec{p}_t) > \tau \\ 0 & otherwise \end{cases} \tag{2.6}$$

As previously mentioned, Jin et al. (2011) utilizes a parameter J_{min} that sets the minimum number of consecutive iterations with the same prediction.

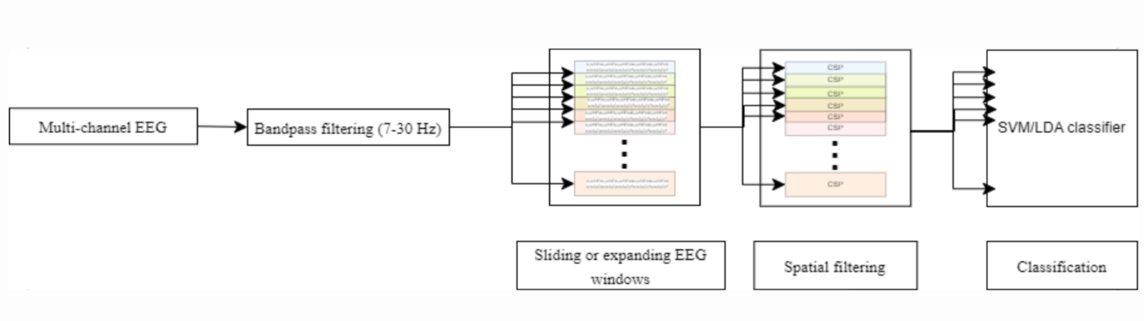

Figure 2.6: Overview of the processing of the MI EEG data.

Algorithm 2.1 Dynamic early classification

```
$x \Leftarrow$ set of time windows in trial
$Patience \Leftarrow$ number of times confidence should surpass the threshold in a row
$sum \Leftarrow 0$
$maxC \Leftarrow$ None
$currentMaxC \Leftarrow$ None
for $x_t \in x$ do
    $\vec{p}_t \leftarrow$ predicted class probability vector given $x_t$
    $currentMaxC \leftarrow$ max probability class in $\vec{p}_t$
    if $I(Confidence(\vec{p}_t)) == 1$ then
        if $maxC ==$ None OR $maxC \neq currentMaxC$ then
            $maxC \leftarrow currentMaxC$
            $sum \leftarrow 1$
        else
            $sum \leftarrow sum + 1$
        end if
    end if
    if $sum == Patience$ then
        return $maxC$
    end if
end for
```

Similarly, the proposed dynamic model utilizes an optimized parameter *Patience*, that sets the number of consecutive windows in which the model has had a confidence above a given threshold. The dynamic model predicts the highest probability class c as soon as the number of times the confidence of the model's predicted probability estimates is above the confidence threshold in a row is equal to the optimal *Patience* value. The pseudocode in algorithm 2.1 shows how this early dynamic classification is implemented.

2.6 Metrics

To evaluate the dynamic and static models, Accuracy, Cohen's Kappa, and Infomation Transfer Rate (ITR) are considered.

Averaged accuracy measures the mean ratio of correct predictions over the total number of predictions of the classes as seen in Equation 2.7 (Salauddin Khan et al., 2023):

$$Averaged\ accuracy = \frac{\sum_{i=1}^{C} \frac{tp_i+tn_i}{tp_i+fp_i+tn_i+fn_i}}{C} \tag{2.7}$$

where:

- tp - true positive
- tn - true negative
- fp - false positive
- fn - false negative
- C is the number of classes

As the dataset is balanced, accuracy provides a useful and interpretable metric for model performance.

Cohen's Kappa is a measure for the level of agreement between two annotators on a classification task, it is used here to measure the agreement between the true label and predicted labels of the model.

$$k = \frac{(p_0 - p_e)}{(1 - p_e)} \tag{2.8}$$

p_0 is the relative observed agreement among raters, and p_e the expected agreement when both annotators assign labels randomly (Pedregosa et al., 2011; Landis & Koch, 1977). It is commonly used in the comparison of models in BCI research (Tangermann et al., 2012).

Information Transfer Rate (ITR) is the most commonly applied metric to assess the overall performance of BCIs (Yuan et al., 2013). It is particularly useful to assess BCI performance as it takes into account both classifier accuracy P and earliness in terms of prediction time T. It measures the amount of information communicated per unit of time (Wolpaw et al., 2002).

ITR is computed as the product of two components: B, the information transfer rate in bits per trial, and Q which scales B to bits per minute and allows ITR to also consider time. B is defined by:

$$B = \log_2 N + P \log_2 P + (1-P) \log_2 \frac{1-P}{N-1} \quad (2.9)$$

where N is the number of classes. Q adjusts B to bits per minute based on prediction time T:

$$Q = \frac{60}{T} \quad (2.10)$$

ITR is, therefore, given by:

$$ITR(\frac{Bit}{Min}) = B * Q \quad (2.11)$$

ITR allows one to find the optimal *Patience* value by balancing between earliness and accuracy. It is also, therefore, a comprehensive measure of BCI system performance, as BCIs need to achieve high accuracy while also delivering timely predictions.

2.7 Hyperparameter tuning

A random search with 60 iterations and uniform distribution was conducted on the training data using the static model. All windows were utilized to optimize window size, CSP, and classifier hyperparameters, with a focus on maximizing accuracy.

Additionally, a *Patience* value was determined through another random search of 60 iterations, but using the dynamic model and optimizing for Information Transfer Rate (ITR). This search, therefore, did not use all windows, as earliness was also taken into consideration. Only considering accuracy would likely mean that the optimal *Patience* would be at a vastly later point. Finding an optimal balance between the two is therefore important. 10-fold cross-validation was also applied to both searches.

2.7.1 Window parameters

As seen in the window segmentation illustration in Figure 2.3, the sliding window had two tuned parameters. The first was the length of the window of size L_s and step size Δt_s in samples across the trial, separating the trial into equal-length windows where each window had Δt_s samples between the start of one window and the start of the next window. The sliding window parameters L_s and Δt_s had both a possible size between 0.1 and 1 seconds.

The expanding window had tuned parameters initial window length of size L_e, and an expansion rate of size Δt_e, separating the trial into iteratively longer windows. The expanding window parameter L_e had a possible value between 0.1 and 1 seconds, and an expansion rate Δt_e between 0.1 and 0.5 seconds.

2.7.2 CSP and model parameters

The number of CSP filers which denoted the number of spatial filters to be used by the method to maximize the difference in variance was also tuned. The set possible values were either 4 or 8, with one or two filters for each class in the dataset. Any more resulted in the unwanted fitting of electrodes placed above the occipital lobe which is responsible for visual processing and not MI.

The LDA and SVM models also had several parameters tuned to improve accuracy. See Table 2.1 for the tuned model parameters and their respective possible values.

The dynamic model had an additional tuned parameter *Patience* which had the possible values of the number of windows that the trial is split into, given the optimal window parameters.

2.8 Dynamic and static model evaluation

As specified in 1.4 and in 1.3, the aim of this research was to evaluate the difference in performance between a static and a dynamically classifying model. The dynamic model is the model that includes the defined dynamic stopping function described in Section 2.5, and can find the average prediction time it took to classify early for a given condition. The static model is then the model without this early stopping method. As such, it is not able to find prediction times and needs specific time points to predict.

To evaluate the dynamic model, the confidence thresholds were set to a range of values between 0 and 1, specifically: [0.001, 0.01, 0.1, 0.2, 0.3, 0.4,

Table 2.1: Tuned parameters and their values for the models.

Model	Parameter description	Values explored
LDA	Algorithm Solver	['svd', 'lsqr', 'eigen']
	Tolerance Level	[1.0e-2 to 1.0e-15]
	Number of Components	[None, 1, 2, 3]
	Regularization	[None, 'auto', 0.0 to 1.0]
SVM	Regularization Strength	[0.1, 1, 10, 100, 1000]
	Kernel Type	['linear', 'rbf', 'poly', 'sigmoid']
	Kernel Flexibility	[1, 0.1, 0.01, 0.001, 0.0001]
	Polynomial Degree	[2, 3, 4, 5]

0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.999]. The changing behavior of the dynamic early classification can then be seen when the strictness of the required confidence is varied. Given the optimal *Patience*, for each threshold, average prediction time, average accuracy, average kappa, and average ITR are calculated across all subjects and all of the subjects' trials.

To then evaluate the static model similarly, it is provided with the average prediction times found by the dynamic model for each confidence threshold. Average accuracy, average kappa, and average ITR are also calculated for the static model at those prediction times. The performances for both the static and dynamic models are plotted together to see to which extent having a confidence-based early classification model maintains performance while simultaneously classifying early. This would show how the performance of the models differ.

3 Results

The dynamic and static models are compared on Accuracy, Kappa, and ITR. The performance of SVM and LDA, as well as, the two different window segmentation methods are also compared. Table 3.1 shows the performance of all the models on accuracy, kappa, and ITR. Finally, the plots below show the performance of the dynamic and static models. For the dynamic model, each point represents a confidence threshold, where it generates an average prediction time and performance across subjects, where each subject's performance was the average across the test trials. These prediction times were then given to the static model to also assess performance. The error bars in the plots indicate the standard error of the mean (SEM), where the difference in the means can be seen.

3.1 Accuracy and Kappa

Table 3.1 shows that the performance is improved by the use of the dynamic model, irrespective of the classifier. This is reflected in Figures 3.1 and 3.2 for accuracy, and in Figures 3.3 and 3.4 in terms of kappa. Both the LDA sliding and SVM sliding models maintain or improve with the confidence-based dynamic classification method.

Smaller thresholds, while providing earlier prediction times, have less performance difference between the dynamic and static models. However, as the thresholds increase, so does the difference in performance with the dynamic model performing better. When the confidence threshold increases, so does the model's ability to leverage the confidence threshold and make more reliable predictions. For the sliding models there also seems to be a reduced accuracy for later time points. This is likely due to the proximity to the offset, as the participants could be fatigued or end their motor imagination early as the subjects may be expecting the offset to be approaching. The performance difference could likely be affected by this as well.

The expanding window models, however, see less difference between the dynamic and static models, as seen in Figures 3.5 and 3.6 for accuracy and Figures 3.7 and 3.8 for kappa. This results in equal and maintained performance between the static and dynamic models.

Additionally, the expanding models have a higher average accuracy than the sliding models. Their accuracy and prediction time also seem to be linearly related as the threshold increases. This is because the models were trained on the entire trial. As the threshold increases, so does the number of

Table 3.1: Model performances on Accuracy, kappa, and ITR

	Sliding window				Expanding window			
	LDA		SVM		LDA		SVM	
Metric	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static
Accuracy (%)	56.4	54.7	52.9	51.5	58.0	58.0	57.5	57.5
Kappa	0.42	0.40	0.37	0.35	0.44	0.44	0.43	0.43
ITR (bits per min)	9.27	8.48	7.56	7.11	7.41	7.42	7.09	7.10

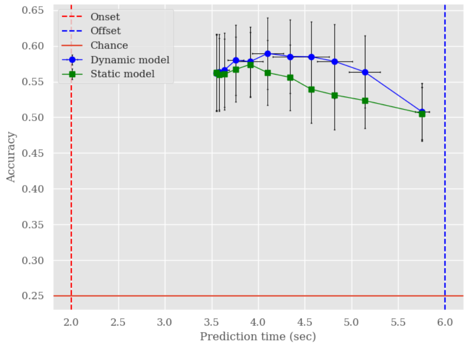

Figure 3.1: Accuracy over prediction time for LDA sliding window models: Dynamic versus Static. Blue line: Dynamic model. Green line: Static model. Each point is a confidence threshold, lower threshold values generate earlier prediction times, and larger threshold values generate later prediction times. Error bars: standard error of the mean. Dashed lines: start and end of trial. Bottom red line: chance level accuracy

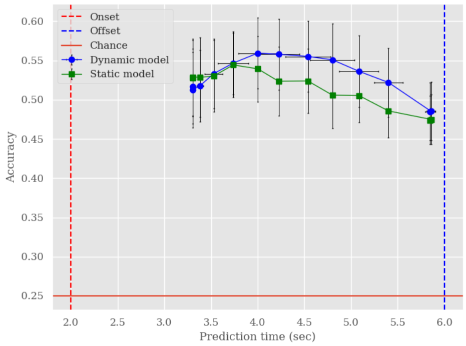

Figure 3.2: Accuracy over prediction time for SVM sliding window models: Dynamic versus Static. Similar to Figure 3.1 but for SVM.

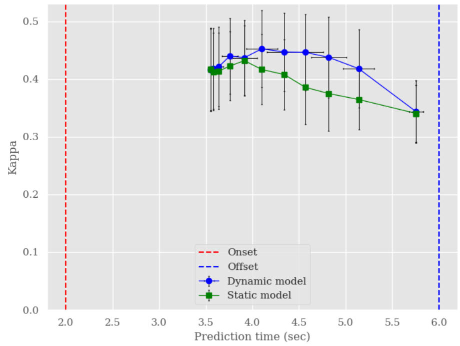

Figure 3.3: Kappa over prediction time for LDA sliding window models: Dynamic versus Static. Similar to Figure 3.1 but using kappa.

windows visited to reach *Patience*. As the number of windows visited increases so does its similarity to the training data, which explains this relation. This could also explain the minimal performance difference. The increasing window size means more data and more alignment with the training data. This could reduce the advantage of the early classification for the dynamic model.

The best performance, in terms of accuracy and kappa, is shown for sliding window models at thresholds or time points in the middle range. For expanding window models, the best performance is seen at thresholds closer to 1 or at the end of the full trial time.

For all models, there is a relatively large gap between when the cue is given at the onset and the first average prediction time, even with very low threshold values. This has two likely reasons. Firstly, in the earliest time points the model is generally unable to accurately classify the MI. This

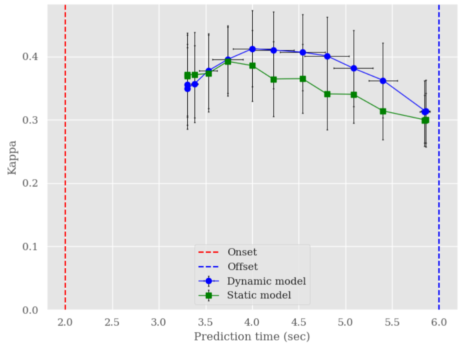

Figure 3.4: Kappa over prediction time for SVM sliding window models: Dynamic versus Static. Similar to Figure 3.1 but for SVM and using kappa.

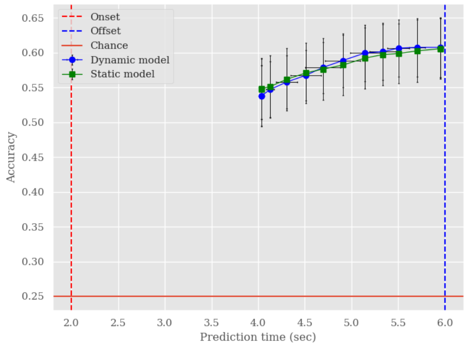

Figure 3.5: Accuracy over prediction time for LDA expanding window models: Dynamic versus Static. Similar to Figure 3.1 but using expanding windows.

could be due to the time taken after a cue by the subject to process and initiate the MI. This could also be compounded by the time it takes to initiate MI to where there is reliable ERD/ERS. Secondly, the models need to wait for a certain amount of time for which the model's confidence should be above the threshold, i.e. *Patience*, and optimal patience is likely more than one window.

Overall, the accuracy for all models hovers between 50% and 60%, which is considerably above the chance level, indicating that the models are capturing relevant patterns in the EEG data and can classify correctly. The general performance of kappa falls within the range of fair to moderate agreement which signifies that the models are reasonably reliable in their predictions (Landis & Koch, 1977).

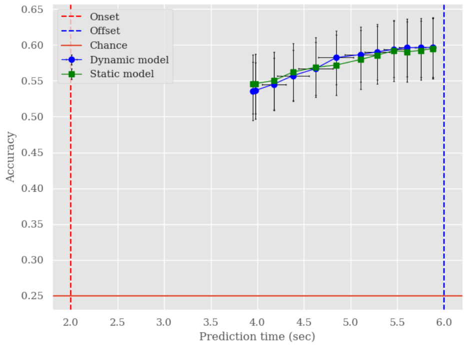

Figure 3.6: Accuracy over prediction time for SVM expanding window models: Dynamic versus Static. Similar to Figure 3.1 but using expanding windows and SVM.

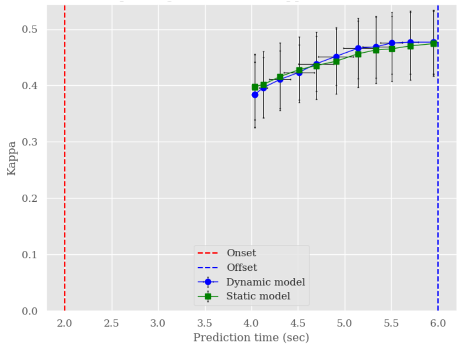

Figure 3.7: Kappa over prediction time for LDA expanding window models: Dynamic versus Static. Similar to Figure 3.1 but using expanding windows and kappa.

3.2 Information transfer rate

The ITR plots in Figures 3.9 to 3.12 show similar results to the previous figures, keeping the same performance differences, and gaps, as discussed above. The dynamic models have better performance, across the thresholds for the sliding models and the performance difference between expanding models is minimal. The difference between the static and dynamic models are the same as in the accuracy and kappa plots, as ITR takes into account accuracy as well. The ITR plots also show that an increase in time is a decrease in ITR with all the ITR plots decreasing over time. ITR, therefore, seems to favor earliness. The optimal ITR, unlike for accuracy and kappa, is at the earliest time

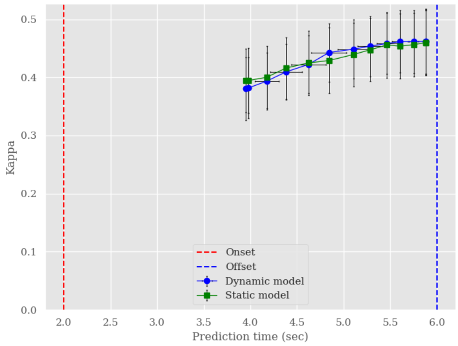

Figure 3.8: Kappa over prediction time for SVM expanding window models: Dynamic versus Static. Similar to Figure 3.2 but using expanding windows, SVM, and kappa.

points for all models. At these time points, static and dynamic models have little difference in performance. While accuracy may improve at later time points such as for the expanding window models, earliness is of more importance for the ITR as seen in Figures 3.11 and 3.12.

Looking at Table 3.1, while accuracy is higher for the expanding models, the sliding models have higher ITR. This likely means that they can classify earlier than the expanding models while still maintaining relatively similar accuracy. The sliding dynamic stopping models also have higher ITR than the sliding static models, making sliding dynamic models the better models overall.

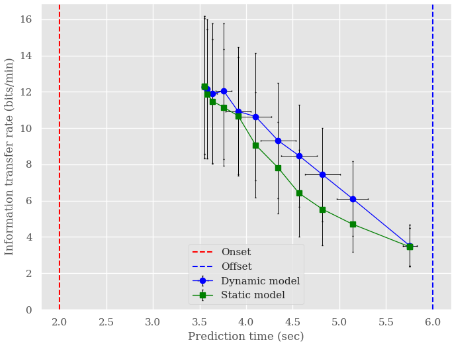

Figure 3.9: ITR over prediction time for LDA sliding window models: Dynamic versus Static. Similar to Figure 3.1 but using ITR.

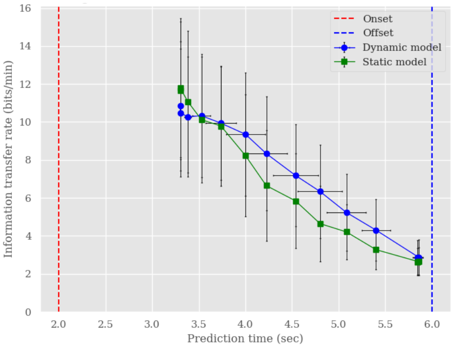

Figure 3.10: ITR over prediction time for SVM sliding window models: Dynamic versus Static. Similar to Figure 3.1 but using SVM and ITR.

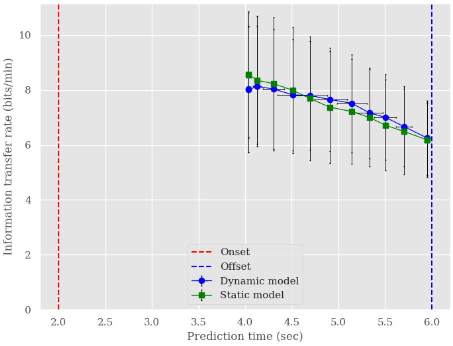

Figure 3.11: ITR over prediction time for LDA expanding window models: Dynamic versus Static. Similar to Figure 3.1 but using expanding windows and ITR.

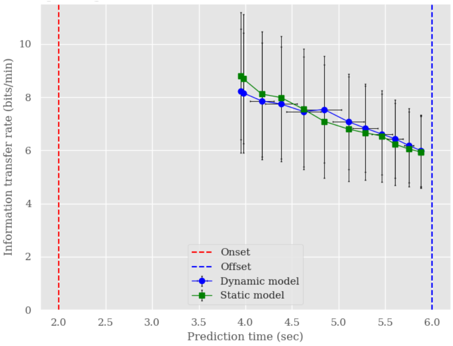

Figure 3.12: ITR over prediction time for SVM expanding window models: Dynamic versus Static. Similar to Figure 3.1 but using expanding windows, SVM, and ITR.

4 Discussion

This research explored whether a confidence-based early classifying model would maintain or improve performance when compared to a static model given the same time windows. The expectation was that this would indeed be the case, due to previous methods in other types of BCI applications seeing improved performance (Jin et al., 2011; Renardi, 2024; Schreuder et al., 2013; Spüler, 2017).

The results confirmed that a confidence-based early classification model was able to maintain and even improve classification accuracy, kappa, and ITR. Notably, sliding windows showed a larger performance difference between the dynamic and static models, whereas expanding windows exhibited little to no difference. This trend was consistent for both SVM and LDA classification models. ITR favored sliding dynamic models, which also showed large performance differences between the dynamic and the static models.

4.1 Limitations and future work

While the findings confirmed the expectations, there are several limitations to this research. This method was solely evaluated on the BCI Competition IV data set 2a from Brunner et al. (2008). It is, therefore, important to verify these findings on additional MI datasets. This method was also only evaluated on synchronous data, raising the question of generalization to real-world usage. It is, therefore, needed to assess whether the application of asynchronous data, without cues, affects the model performance and whether the current findings hold.

Additionally, while effective, this research used predictive entropy, which is a relatively simple uncertainty quantification method. The LDA model, additionally, used softmax for its probability estimation. Softmax outputs are known for being overconfident, making them imperfect for accurate uncertainty estimation (Pearce et al., 2021). A first improvement could be to implement probability calibration, where the models predicted probabilities are adjusted to better reflect true probabilities. While this may have been implicitly done for the SVM, as it internally uses Platt scaling (Chang & Lin, 2011), it was not verified whether the predicted probability estimates of the models match the true probability of the target classes.

The models themselves are also unable to capture epistemic uncertainty. Epistemic uncertainty arises from a lack of knowledge and can be observed when the model is applied to data that is different from the training data (Hüllermeier & Waegeman, 2021). The model may be overconfident and less reliable when encountering new data without the ability to capture this type of uncertainty. Addressing these limitations by incorporating an uncertainty quantification method capable of capturing epistemic uncertainty and calibrating the probability estimated to represent true probability, could increase the performance difference between the dynamic and static models. The ensemble method discussed by de Jong et al. (2023) could serve this task.

Another potential limitation is the optimal ITR being at the earliest time points at which point the dynamic and sliding models had little difference in performance. It could be argued that one could choose to find this optimal time and provide this to the static model for inference for new subjects. While true that the static model may seem like an alternative to the dynamic model, it is unable to balance earliness and accuracy. Rest periods between trials, while not used in this research, could serve to highlight the strengths of the dynamic model. Assume an infinite rest period prior to a trial during which the model is evaluated on ITR, when the trial begins, the dynamic model should still be able to classify as soon as patience is reached during the trial. The ITR analysis for the static model would then not be possible or at least be very limited and brittle.

The expanding window models as seen in Section 3, had interesting results. As previously mentioned, due to the training data being on the whole trial, as the windows get longer, so does their closeness to the training data, increasing accuracy. The extent to which this is desirable is questionable. However, the performance with thresholds in the middle range did provide better accuracy than for the sliding window models, even while including data from the beginning of the trial. The assumption would be that including time points from this early period in the trial where classification is difficult would affect the performance for larger windows negatively. As seen in the accuracy and kappa plots for the expanding window models, this is not the case. The performance difference is, however, still affected by this.

As BCIs have inter-subject variability, the CSP, model (including patience), and window hyperparameters, should all be optimized for each subject separately. This was not done in this research. Implementing subject-specific optimizations could improve performance for the subjects. It would also show whether per-subject confidence-based early MI classification is more effective. This would increase the practical application of this system, and further show the improved abilities of dynamic MI BCI.

References

Akasiadis, C., Kladis, E., Michelioudakis, E., Alevizos, E., & Artikis, A. (2022). Early Time-Series Classification Algorithms: An Empirical Comparison. *arXiv*. doi: 10.48550/arXiv.2203.01628

Aristimunha, B., Carrara, I., Guetschel, P., Sedlar, S., Rodrigues, P., Sosulski, J., … Chevallier, S. (2023). Mother of all BCI Benchmarks. doi: 10.5281/zenodo.10034223

Brunner, C., Leeb, R., & Müller-Putz, G. (2008). BCI Competition 2008–Graz data set A. doi: 10.21227/katb-zv89

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3). doi: 10.1145/1961189.1961199

Craik, A., He, Y., & Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of Neural Engineering*, *16*(3), 031001. doi: 10.1088/1741-2552/ab0ab5

de Jong, I. P., Sburlea, A. I., & Valdenegro-Toro, M. (2023). Uncertainty Quantification in Machine Learning for Biosignal Applications – A Review. *arXiv*. doi: 10.48550/arXiv.2312.09454

Garg, A., & Mago, V. (2021). Role of machine learning in medical research: A survey. *Computer Science Review*, *40*, 100370. doi: 10.1016/j.cosrev.2021.100370

Grosse-Wentrup, M., & Buss, M. (2008). Multiclass Common Spatial Patterns and Information Theoretic Feature Extraction. *IEEE Transactions on Biomedical Engineering*, *55*(8), 1991-2000. doi: 10.1109/TBME.2008.921154

Gupta, A., Gupta, H. P., Biswas, B., & Dutta, T. (2020). Approaches and Applications of Early Classification of Time Series: A Review. *IEEE Transactions on Artificial Intelligence*, *1*(1), 47-61. doi: 10.1109/TAI.2020.3027279

Gupta, M. R., Parrish, N., & Anderson, H. S. (2012). Early time-series classification with reliability guarantee. *Sandia National Laboratories*. doi: 10.2172/1051704

Hosseini, M.-P., Hosseini, A., & Ahi, K. (2021). A Review on Machine Learning for EEG Signal Processing in Bioengineering. *IEEE Reviews in Biomedical Engineering*, *14*, 204-218. doi: 10.1109/RBME.2020.2969915

Hwang, J., Park, S., & Chi, J. (2023). Improving Multi-Class Motor Imagery EEG Classification Using Overlapping Sliding Window and Deep Learning Model. *Electronics*, *12*(5). doi: 10.3390/electronics12051186

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, *110*(3), 457–506. doi: 10.1007/s10994-021-05946-3

Jiang, H., Kim, B., Guan, M. Y., & Gupta, M. (2018). To Trust Or Not To Trust A Classifier. *arXiv*. doi: 10.48550/arXiv.1805.11783

Jin, J., Allison, B. Z., Sellers, E. W., Brunner, C., Horki, P., Wang, X., & Neuper, C. (2011). An adaptive P300-based control system. *Journal of Neural Engineering*, *8*(3), 036006. doi: 10.1088/1741-2560/8/3/036006

Landis, J. R., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, *33*(1), 159–174. doi: 10.2307/2529310

Li, C., & Wang, B. (2014). Fisher linear discriminant analysis. *CCIS Northeastern University*, *6*.

Padfield, N., Zabalza, J., Zhao, H., Masero, V., & Ren, J. (2019). EEG-Based Brain-Computer

Interfaces Using Motor-Imagery: Techniques and Challenges. *Sensors*, *19*(6). doi: 10.3390/s19061423

Pearce, T., Brintrup, A., & Zhu, J. (2021). Understanding Softmax Confidence and Uncertainty. *CoRR*. doi: 10.48550/arXiv.2106.04972

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pfurtscheller, G., & Neuper, C. (2001). Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, *89*(7), 1123-1134. doi: 10.1109/5.939829

Renardi, B. (2024). *Improving P300 BCI Speller Using Uncertainty Quantification* (Unpublished master's thesis). University Of Groningen.

Salauddin Khan, M., Nath, T. D., Murad Hossain, M., Mukherjee, A., Bin Hasnath, H., Manhaz Meem, T., & Khan, U. (2023). Comparison of multiclass classification techniques using dry bean dataset. *International Journal of Cognitive Computing in Engineering*, *4*, 6-20. doi: https://doi.org/10.1016/j.ijcce.2023.01.002

Schreuder, M., Höhne, J., Blankertz, B., Haufe, S., Dickhaus, T., & Tangermann, M. (2013). Optimizing event-related potential based brain–computer interfaces: a systematic evaluation of dynamic stopping methods. *Journal of Neural Engineering*, *10*(3), 036025. doi: 10.1088/1741-2560/10/3/036025

Spüler, M. (2017). A high-speed brain-computer interface (BCI) using dry EEG electrodes. *PLOS ONE*, *12*(2), 1-12. doi: 10.1371/journal.pone.0172400

Tangermann, M., Müller, K.-R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., ... Blankertz, B. (2012). Review of the BCI Competition IV. *Frontiers in Neuroscience*, *6*. doi: 10.3389/fnins.2012.00055

Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A. E. (2017). Linear discriminant analysis: A detailed tutorial. *AI Communications*, *30*, 169-190. (2) doi: 10.3233/AIC-170729

Vavoulis, A., Figueiredo, P., & Vourvopoulos, A. (2023). A Review of Online Classification Performance in Motor Imagery-Based Brain–Computer Interfaces for Stroke Neurorehabilitation. *Signals*, *4*(1), 73–86. doi: 10.3390/signals4010004

Wolpaw, J. R., Birbaumer, N., Heetderks, W., McFarland, D., Peckham, P., Schalk, G., ... Vaughan, T. (2000). Brain-computer interface technology: a review of the first international meeting. *IEEE Transactions on Rehabilitation Engineering*, *8*(2), 164-173. doi: 10.1109/TRE.2000.847807

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, *113*(6), 767-791. doi: 10.1016/S1388-2457(02)00057-3

Wu, T.-F., Lin, C.-J., & Weng, R. (2003). Probability estimates for multi-class classification by pairwise coupling. *Advances in Neural Information Processing Systems*, *16*. doi: 10.5555/1005332.1016791

Xu, R., Jiang, N., Lin, C., Mrachacz-Kersting, N., Dremstrup, K., & Farina, D. (2014). Enhanced Low-Latency Detection of Motor Intention From EEG for Closed-Loop Brain-Computer Interface Applications. *IEEE Transactions on Biomedical Engineering*, *61*(2), 288-296. doi: 10.1109/TBME.2013.2294203

Yuan, P., Gao, X., Allison, B., Wang, Y., Bin, G., & Gao, S. (2013). A study of the existing problems of estimating the information transfer rate in online brain–computer interfaces. *Journal of Neural Engineering*, *10*(2), 026014. doi: 10.1088/1741-2560/10/2/026014