Towards Mental Workload Time Series Classification and Interpretation for Real-Time Feedback in Brain-Computer Interfacing Video Games

By

Shreya Milind Mhalgi

MS Thesis submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

EXAMINED AND APPROVED ON:

12/14/2022

APPROVED BY:

Advisor: Professor Rodica Neamtu, Professor of Teaching

Reader: Professor Erin Solovey, Associate Professor

Professor Craig Shue, Department Head Computer Science Department

Abstract:

One of the important factors contributing to the creation of engaging and pleasurable video game experiences is immersion. New Brain-Computer Interfaces (BCIs) enable the computer interfaces to engage with the player's mind such as detecting the player's real-time mental status (high/low mental workload) and using it as a real-time input for an immersive game experience. First, this research aims to find the most suitable classifier with the objective of classifying high and low mental workload by tuning numerous machine learning algorithms on the mental workload time series (fNIRS) dataset using our customized classification tool NaML. Second, this research aims to explore high and low workload intensity clusters using our novel brain signal exploration tool BrainEx to enhance our understanding of the dataset. Contributions to this research introduce a dashboard, NeuroHub, which will enable researchers who do not have an extensive Computer Science or Data Science background to conduct data parsing, data mining, and machine learning efficiently on the Functional Near-Infrared Spectroscopy (fNIRS) data. The results from this research lay a foundation for using exploratory insights to improve the classification of time series brain data.

Keywords: Brain-Computer Interfaces, fNIRS, Machine Learning on time series, cognitive workload, user experience

Acknowledgments

I acknowledge my advisor, Professor Rodica Neamtu, and my reader Professor Erin Solovey, for their encouragement, patience, and support throughout this research. I am grateful to Alicia Howell-Munson (Ph.D. student in BCB at WPI), Christopher Micek (Ph.D. student in CS at WPI), and Max Chen (Ph.D. student in IMGD at WPI) for helping and mentoring me. Lastly, I would like to thank my family and friends for their constant support and for celebrating every accomplishment along the way with me.

Table of Contents

1. Introduction

The amount of mental work that an individual performs compared to the amount of mental work that an individual is capable of doing is referred to as cognitive load [1]. The cognitive load is connected to task difficulty which means, increasing task complexity leads to an increase in cognitive load [2]. According to research in ergonomics, the mental workload may essentially be a measurement of mental activity from such complex tasks [3]. In some safety-critical operating contexts, one or a few operators may be in charge of the safety, and even the lives, of a large number of people. For instance, the Aviation Safety Network recorded 19 incidents with 960 casualties and in most of the cases problems were linked to a higher mental workload as one of the contributing factors [4][5]. The potential for errors due to higher workload: thus it necessitates the study of cognitive workload.
Apart from studies in ergonomics and research in other safety-critical contexts, the mental workload is present in the applications enhancing the usability of interactive systems [14]. Brain-Computer Interfaces (BCIs) were and continue to be utilized in medical applications [7] but also have applications used for recreational purposes like brain-controlled installations [8], brain-controlled movies [9], as well as BCI games [10]. BCI shows the potential of determining the player's current mental state, which may be utilized as a unique input for an immersive gaming experience.

In computer systems and data visualization, visual embellishments are the design elements that support the information that needs to be delivered [30]. These embellishments can be typically used to engage the audience, and reduce emotional response by producing confusion, creating distracting visual artifacts, and causing the loss of meaningful information. In immersive video gaming experiences, these effects do not directly affect the gameplay but affect the player’s perception of the game world [31]. Traditionally, this process of adjusting rendering effects is done carefully by the game development team, while the players trigger the effects when a certain activity happens. BCI shows the prospect for detecting the player’s real-time mental status and the game generates personalized content based on the player’s experience model. Existing BCI games are designed to test a BCI paradigm, which has oversimplified gameplay mechanics[32]. It is possible to bridge the gap between gameplay mechanics and BCI with the use of noninvasive technologies like fNIRS.

Declining costs, availability of brain signal datasets, and technological advances in non-intrusive brain monitoring systems outside clinical settings are promoting research and applications of Brain-Computer interfaces [11, 12]. The tools available for brain signal analysis are moreover directed toward engineers, scientists, and technicians where the data is gathered in a controlled experimental setting. Although the research data analyzed by such a team of experts in experimental settings are the key to discoveries in BCI, the same practices do not work that well in real-world contexts. Exploring the brain signals with a data-driven approach to find similar patterns will be valuable in real-world contexts to get a better understanding of the dataset and discover patterns that might indicate changes in cognitive workload [13]. Our novel web-based, brain data analytics platform - BrainEx [13] is developed with data mining approaches for interactively exploring similar patterns in large fNIRS datasets and built with a core design policy “exploration at every stage”.

A hypothetical neuroscientist could be only interested in knowing the final classification output of a time series (fNIRS) dataset which can be obtained by exploiting some machine learning approaches on that dataset. Although python toolboxes like sktime [14] have introduced state-of-art libraries for implementing machine learning algorithms on time series data, neuroscientists will have to spend some time understanding how to preprocess the data to apply these algorithms to their data and then derive the required insights. Our customized tool for BCI classification named NaML eases this process of applying

machine learning approaches for obtaining classification results without any prior knowledge of Computer Science or Data Science fundamentals.

BrainEx and NaML are two separate tools hosted in different containers on a server. They both need input in a specific format, which is provided by preprocessing tools. We introduce our novel tool NeuroHub which incorporates state-of-the-art tools for brain signal exploration, namely BrainEx and our customized classification tool named NaML along with the other supporting tools like BCI data parser and data converter for input format matching. NeuroHub is a web-based interactive dashboard that integrates data parsing, data exploration, and machine learning tools to facilitate fNIRS research and analysis. This tool enables the user to access all the tools from a single platform that is equipped with instructions for fNIRS data processing and it makes NeuroHub efficient and user-friendly.
This research presents a two-fold case study using this novel tool NeuroHub and the contributions to this research are as follows:

- Training and tuning multiple machine learning algorithms implemented in our customized BCI classification tool NaML to achieve better classification results on the “fNIRS to Mental Workload dataset” from Tufts University [15]. The classification results from this experiment have the potential to improve the BCI-driven mechanics for a real-time video game.
- Using the interactive visualization and sequence similarity exploration tool - BrainEx, this research aims to discover patterns and similar sequences in the fNIRS to Mental Workload datasets from Tufts University [15] and then use the exploration to create clusters of high and low mental workloads. These representative clusters can then be used to gain insights into the patterns and similarities in the data representing either workload intensity.

2. Background

The two-fold case study proposed in this document utilizes a novel tool NeuroHub that brings together the tools required for brain signals data parsing, exploration, visualization, and machine learning that focus on functional near-infrared spectroscopy (fNIRS) research and analysis. This section provides background on fNIRS, Sktime, NaML, Machine Learning approaches for BCI classification, BrainEx, and other analytic tools for fNIRS.

2.1 Functional Near Infrared Spectroscopy (fNIRS)

Functional near-infrared spectroscopy (fNIRS) is a noninvasive form of neuroimaging that provides multivariate time series data by observing brain activity and measuring the changes in concentration of micromolar hemoglobin in the brain. These time series represent oxygenated and deoxygenated hemoglobin. fNIRS is typically gathered using a montage affixed to a cap on the participant’s head for data measuring brain activity that detects changes in hemoglobin species in the brain via optical absorption differences [16] at the location where the channel is placed on the head. The montage consists of multiple channels, detectors, and emitters that use near-infrared light that can penetrate biological tissues and get absorbed chromophores, such as oxyhemoglobin and deoxyhemoglobin [16].

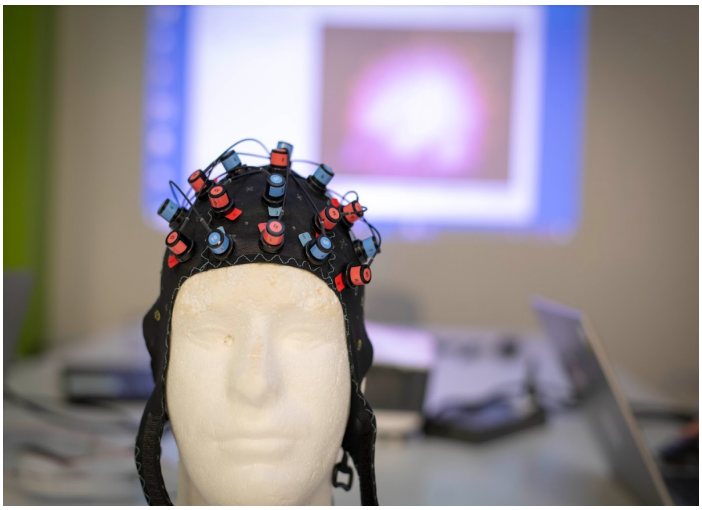


Figure 1. A picture of an fNIRS montage on an fNIRS cap. The montage is the series of red and blue optodes on top of the helmet/skullcap which is the fNIRS cap.

The oxy-deoxy measures enable researchers to compare brain activity across different regions at the same time [27]. fNIRS are useful because of their accuracy, non-invasiveness, mobility, and potential for long-term monitoring [20][13]. Usually, fNIRS datasets are associated with other data that describe sensor locations, the study participant identifier, demography of the participants, activity during the events, etc [13][15]. Researchers mainly look for patterns in brain data that reflect a certain cognitive state using these multivariate time series and the associated data mentioned above. This may be accomplished by using statistical techniques to detect significant variations between two sets of labeled sequences- high mental workload and low mental workload. Machine Learning models commonly use such labeled data for training to predict unknown labels in the future. It is especially difficult to apply the general machine learning approaches to fNIRS time series brain data since the brain signal waves could have different lengths and scales. There are, however, more recent prospects for using machine learning algorithms and performing analytics on brain data.

2.2 Sktime

Sktime [14] provides an open-source unified framework for machine learning on time series data. This is based on a python library that provides a toolbox that extends current machine learning capabilities to work with univariate and multivariate time series data. Univariate time-series data refers to a time series consisting of observations of a single variable recorded in equal time increments[17]. Multivariate data is composed of several observations gathered over time[18]. Sktime offers a single API for processing any machine learning time series required and extensive parameter customization for more control over data analysis. In addition, sktime has numerous features like forecasting and regression. Multivariate time series data is generated using fNIRS. Measurements from many channels measuring different areas of the brain are recorded at each time period. Since data points inside a channel must keep both sequence order and order of values within each sequence, standard machine learning methods are not suitable to classify fNIRS data. Because of the data's intricacy, a tool like sktime is required. A representation of fNIRS data can be seen below.

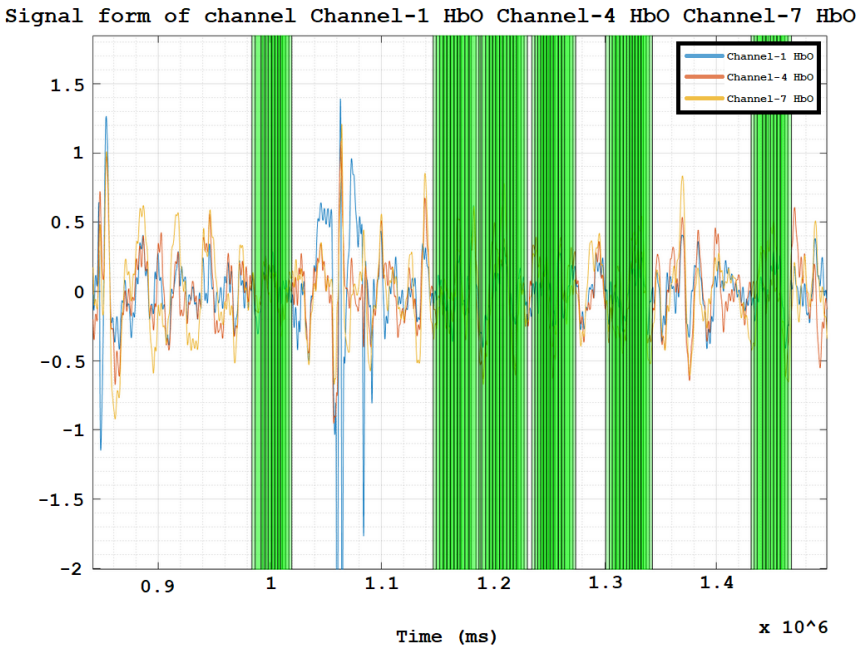


Figure 2. The three colored lines represent time series measurements in blood oxygenation variations seen in three channels. The green highlighted bars represent a five-second interval before and after the occurrence of an event. There is an overlap between the event periods due to the test framework, as indicated by the areas of greater opacity green coloring.

Sktime has various functions for time series data, such as forecasting and regression; nevertheless, this research focuses particularly on time series classification and transformation. A machine learning classifier uses dataset features as input to identify which class a target feature belongs to. Classifiers are categorized according to their type composition, dictionary, distance, feature, hybrid, interval, kernel, and shapelet [14].
Selecting the appropriate classifier for a particular scenario is not an easy task. This is because there is a rising number of algorithms from various families available and a lack of approaches that may assist in selecting a certain family of algorithms in advance for a specific type of dataset [19]. Our customized classification tool- NaML assists to select the best classifier for any particular brain data and gives the user the ability to change the default parameters such as random_states, n_splits, and n_jobs to provide better classification metrics.

2.3 NirsAutoML (NaML) - a Machine Learning Framework

A framework developed using React, Django and Postgres is designed to make use of the sktime python library more efficiently. NaML's user interface is created as a single-page application in React (Figure 3) and is implemented on a remote server using a Docker container that allows NaML to work in a separate environment, avoiding conflicts with other tools or services hosted on the same server. NaML is intended to make better use of the sktime python library and reduce the barrier to entry for researchers by providing them with more advanced tools for analyzing and interpreting the fNIRS time series data. NaML has 12 out of 27 sktime classifiers implemented that allow for classification to run on a channel-to-channel basis using a ColumnConcatenator or a ColumnEnsembleClassifier. ColumnConcatenator can be used to split multivariate time series data columns and concatenate them into a single univariate channel before

applying a classifier to the concatenated data. On the other hand, ColumnEnsembleClassifier works by applying univariate methods to singular channels/columns rather than concatenating several columns into a single long univariate variable [20][21]. In the case of NaML, the latter method is preferred since the ensemble methods maintain the data structure closer to the original format. The following table shows the classifier families and the classifiers implemented in NaML -

Classifier family	Classifiers
Composition	ColumnEnsembleClassifer (CEC)
Dictionary-based	IndividualBOSS, BOSSEnsemble, ContractableBOSS, WEASEL, IndividualTDE, TemporalDictionaryEnsemble
Distance-based	KNeighborsClassifier
Interval-based	TimeSeriesForestClassifier, RandomIntervalSpectralForest, SupervisedTimeSeriesForest
Shapelet-based	ShapeletTransformClassifier

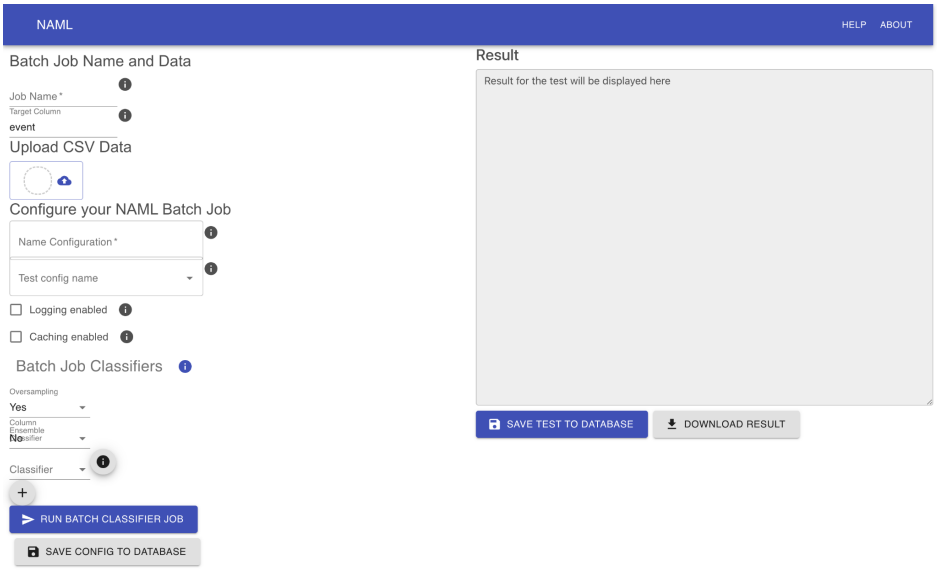


Figure 3. NaML user interface that enables the user to run tests using sktime machine learning classifiers.

2.4 Machine Learning approaches for BCI classification

Classification pipelines can be categorized into two parts, hand-designed or learned representations and cross-subject pipelines that use multiple data sources for the training data [15]. Hand-designed features refer to the properties derived from the information using various algorithms. The common approach in BCI classification is *feature-then-classify* (hand-designed) where the first feature selection stage utilizes each univariate time series and transforms it into a fixed-size feature vector through the use of

manually-designed summary functions like mean, variance, slope, etc. The second classification stage employs a standard classifier that utilizes the features gathered from stage one. However, recent BCI makes use of learned features automatically extracted through the Neural Networks models and do not have to be supplied with hand-crafted features as they are able to learn the best features from the data. These models are more flexible and overcome the limitations of the common approaches that have a limited capacity [15][22]. Some other works from the BCI literature focus more on improving generalization on multiple subjects by utilizing approaches based on contrastive learning [23] and domain generalization [24].

2.5 BrainEx

BrainEx [13] is a novel visual exploratory tool for brain data that combines brain sensing research with time series data mining and visualization research to address issues in brain signal analytics, with a particular emphasis on functional near-infrared spectroscopy (fNIRS) utilized in BCI settings.
Similarity search is widely used in many other applications, for example, search engines and weather predictions also rely on discovering the most similar sequences given a query. In BCI, finding similar fNIRS data sequences is an essential step for finding brain signals that may be indicative of a type of cognitive workload. The BrainEx website interface which the user accesses and is developed in HTML, CSS, React, and D3.js. The BrainEx Engine Server is developed in Python and usable with Linux OS, and the RESTAPI preprocesses datasets, performs similarity searches, and clusters data which is implemented with Django. BrainEx makes use of algorithms that allow for fast exploration of complicated, large time series data. It is also an easy-to-use visual tool, that allows researchers to do similarity searches, study feature data and natural grouping, and select sequences of interest for future searches and exploration.[13]

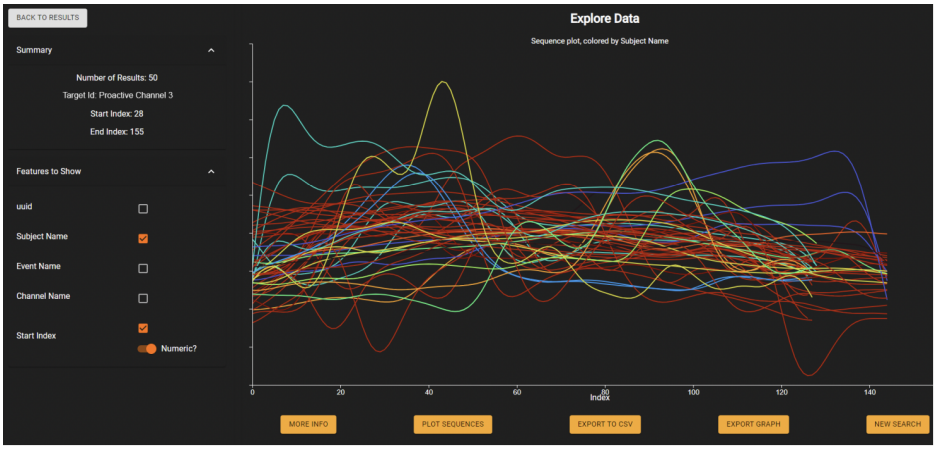


Figure 4. BrainEx [13] web-based analytic tool displaying 50 similar sequences.

Cluster Exploration: All clusters may be filtered based on the number of sequences in the cluster and the sequences of equal lengths in each cluster.
Furthermore, the user can select clusters based on the values of specified user-customized properties. Participant name, event name, and channel name are typical characteristics in brain signal datasets that can be used to filter the clusters. [13]

Similar sequences functionality: The similarity search feature allows the end user to provide a search query in the form of a time series, and BrainEx checks the representative of each cluster. If a representative falls under the similarity threshold of the time series used for the query, our system will query every other time series within that cluster to determine the similarity between them. In the end, BrainEx returns comparable sequences from the dataset. [13]

2.6 Analytic tool for fNIRS

A lot of specialized tools such as HomER, NIRS-KIT, and Turbo Satori [25-27] have been developed for fNIRS brain data analysis. HomER is a MATLAB-based graphical user interface application that supports concepts like generic linear modeling and enables easy visual interactions with the fNIRS data. Usually, in neuroscience experiments, the stimuli have precisely timed onsets and are of the same length. HomER works on these assumptions from the experimental settings which is restrictive for a lot of real-world applications. with a requirement that stimuli have perfectly timed onsets and are all the same length [13][25]. NIRS-KIT is another offline, MATLAB toolbox used for analysis that enables researchers to analyze resting-state fNIRS data. It also supports task-driven data similar to HomER [26]. In general, both of these tools support numerous filtering and signal processing techniques along with the calculation of oxy-deoxy hemoglobin. Turbo Satori on the other hand is a real-time analytics software that includes most of the features of the other two tools but offers real-time visualization and classification of brain signals [27]. All of these tools are useful and frequently employed in specific use cases, but they are all intended for researchers with specialized knowledge. Making brain data exploration less tedious is especially important today, to enable more robust analysis across larger datasets collected in various contexts so that researchers may examine brain signal data collected by other researchers to be able to gain valuable insights across diverse contexts. However, there are not many tools available that allow you to browse the relationships and structures inside fNIRS datasets or to get a general understanding of the same. However, our novel brain signals exploration tool BrainEx overcomes these limitations.

3. System Overview - NeuroHub

The NeuroHub dashboard is a web-based tool created to facilitate data parsing, data exploration, and machine learning on the fNIRS data. In Software Engineering terms, Neurohub is a frontend/user interface built in ReactJS and utilizes Rest API through the Node.js(Express.js) server, for the BCI Data Parser and format matcher scripts.
NeuroHub is an interactive platform along with some instruction and feedback that currently provides a gateway to access BCI Data Parser, BrainEx, NaML, an input format converter with just a button click, and directions to use the MATLAB GUI for raw data processing.
The following workflow represents two pipelines of NeuroHub that enables data processing on the data collected at WPI and also any public datasets. Steps 1 and 2 are data collection steps and steps 3,4,5 represent preprocessing and exploration/classification steps. The use of raw data workflow for the NeuroHub is designed considering the brain data collection process in WPI's BCI lab. NeuroHub focuses on steps 3,4 for raw data processing (figure 5).

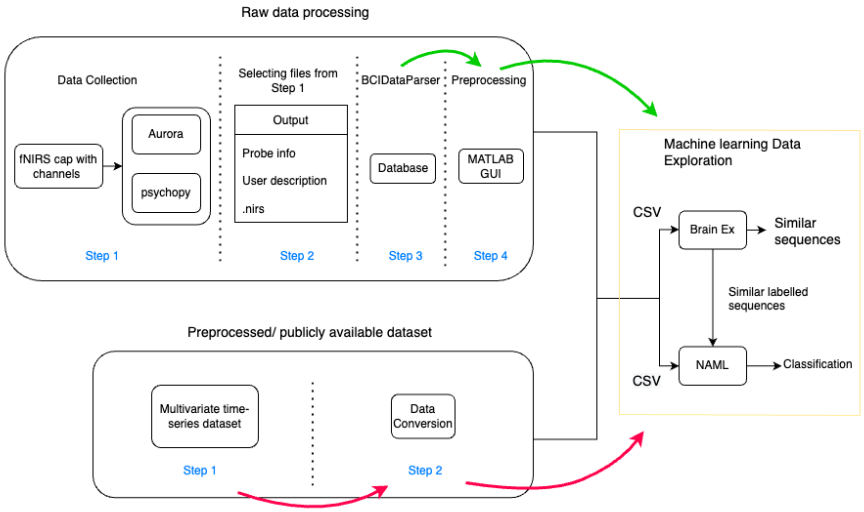


Figure 5. NeuroHub workflow shows the processing of raw and preprocessed data

BCI Data Parser, MATLAB GUI, BrainEx, and NaML function independently. However, the inputs for BrainEx and NaML are dependent on the output from MATLAB GUI. And the MATLAB GUI takes the parsed data files uploaded to the database by the BCI Data parser as input. All of these components were not integrated and required manual intervention and monitoring at every stage. The NeuroHub dashboard is created to integrate this workflow, minimize manual intervention and make all the tools easily accessible.

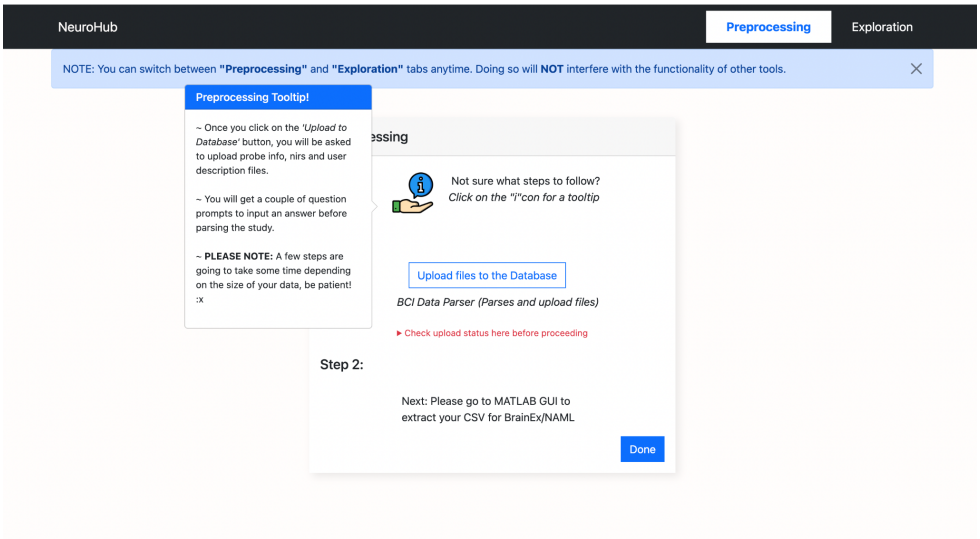


Figure 6. Screen 1 of the NeuroHub User Interface, hosted locally. This image displays the steps represented as ‘Raw Data Processing’ in the workflow above

If researchers are working on a publicly available dataset or already have preprocessed data and are solely interested in conducting some similarity search and classification with BrainEx and NaML, NeuroHub has a feature to make this easier. NeuroHub also allows working with open access/public

datasets/preprocessed data with its input format conversion feature. Other datasets may require conversion into a tool-friendly format specifically for using BrainEx and/or NaML.
For instance, NaML requires datasets to include at least one value column and five feature columns. Name, event, channel, start time, and finish time are the feature columns in sequence. When identifying rows, NaML is solely concerned with different start timings, thus as long as the chunks begin at different times, the dataset will be appropriately structured. The value columns are a set of columns sorted chronologically that show the measurement of variations in micromolar hemoglobin in that place at that time.

name	event	channel	start time	end time	1	2	3	4	5	6
sub_1	0	AB_I_O	0	1	-0.19829	-0.19565	-0.19279	-0.18994	-0.1874	-0.18548
sub_1	0	AB_PHI_O	0	1	0.335677	0.29721	0.275041	0.271573	0.288043	0.32442
sub_1	0	AB_I_DO	0	1	-0.56079	-0.58071	-0.60045	-0.61975	-0.63838	-0.65609
sub_1	0	AB_PHI_D	0	1	-0.05604	-0.03217	-0.01237	0.002192	0.010818	0.013304
sub_1	0	CD_I_O	0	1	0.460231	0.524553	0.585149	0.641092	0.691434	0.735207
sub_1	0	CD_PHI_O	0	1	0.271119	0.208284	0.162781	0.133816	0.119983	0.119547
sub_1	0	CD_I_DO	0	1	-0.64589	-0.68137	-0.71663	-0.751	-0.78375	-0.81415
sub_1	0	CD_PHI_D	0	1	0.198804	0.202136	0.194758	0.176938	0.149429	0.113429
sub_1	0	AB_I_O	1	2	-0.18994	-0.1874	-0.18548	-0.1845	-0.18472	-0.18633
sub_1	0	AB_PHI_O	1	2	0.271573	0.288043	0.32442	0.37938	0.450381	0.533808

Figure 7: An example dataset used with NaML demonstrating the required feature columns and a subset of value columns. This dataset is from the “fNIRS to Mental Workload” datasets from Tufts University [15]

NeuroHub offers a format conversion feature available via a button click for this purpose, which is used to package the data for usage by these tools (BrainEx/NaML). NeuroHub’s input format converter is a python script integrated with a REST API call to the script from the Node.js backend.

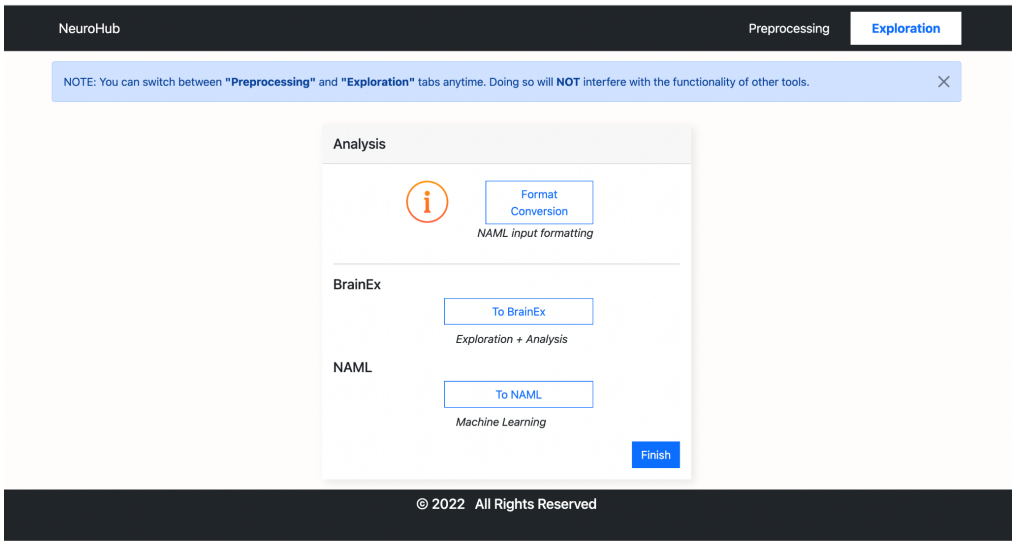


Figure 8. Screen 2 of the NeuroHub User Interface. This image displays step 2 from ‘Preprocessed data’ from the workflow above and the gateway to data exploration and machine learning components.

4. Proposed Approach

4.1 Dataset

fNIRS to Mental Workload dataset [15], an open-access dataset from the researchers at Tufts University will be used for this project. The dataset consists of brain activity recordings from 68 healthy participants (no reported neurological, psychiatric, or other brain-related diseases that might affect the result), during a series of controlled n-back experimental tasks designed to induce working memory workloads of varying relative intensity. The dataset intended for benchmark classifiers can be used to distinguish between high and low mental workloads with higher accuracy. There are two versions of the data available namely, raw and preprocessed. This research plans to use the preprocessed slide window data which is offered in window sizes and timestamps of 10/15/50/100/150/200 and 2/5/10/20/30/40 seconds respectively. The window stride of the sliding window is given as 3. The readings for each subject in the trial, with a sequence time of five seconds and sequence length of 25 measurements. This provides 2143 evenly distributed events across four labels, 0, 1, 2, and 3. This dataset comprised 10 columns, eight of which were permutations of the brain region, measurement type, and blood oxygenation concentration of interest. These correspond to various channels in the available NaML datasets. The following columns are chunk and labeled. A NaML dataset's label column correlates to the event column. Chunk denotes a series, and the sequence length is 25. The readings for each channel are appended as a new column for each chunk of 25 rows, and the chunk number leads to the generation of start and end timings.

AB_I_O	AB_PHI_O	AB_I_DO	AB_PHI_DO	CD_I_O	CD_PHI_O	CD_I_DO	CD_PHI_DO	chunk	label
-0.1982913131	0.3356768324	-0.5607903739	-0.05604172201	0.4602313113	0.2711194025	-0.6458922179	0.19880432	0	0
-0.1956510344	0.2972101412	-0.5807118249	-0.03216954136	0.524553193	0.2082838723	-0.6813699608	0.202136283	0	0
-0.1927904087	0.2750414061	-0.6004456399	-0.01236980865	0.5851489867	0.162781122	-0.7166322466	0.194758224	0	0
-0.1899431036	0.2715729057	-0.6197505492	0.002191503045	0.6410919133	0.1338160253	-0.7509975044	0.1769383358	0	0

Figure 9. Screenshot of de-identified data sample [15]

4.2 Task Design

A set of 40 numbers appeared on the screen, one after another. Some of the numbers were targets. For a 1-back task, a number is only a target if it is identical to the number 1 step back. The participants were asked to press the left arrow key for all the targets and the right arrow key for all the non-targets. To identify the correct target numbers, the participants had to remember the numbers that were n-steps back, which induces a mental workload in the working memory. The following figure represents a 2-back task inducing a low mental workload. Green cards represent targets, and the rest represent non-targets.

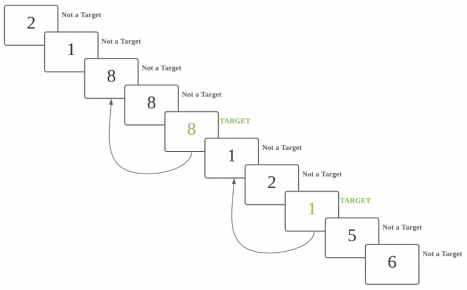


Figure 10. Representation of mental workload inducing 2-back task [15]

A total of 16 rounds were conducted to complete n-back tasks where n could be either 0,1,2 or 3 and before each round, the participants were given instructions to find the target numbers for the current n. The mental workload induced on the participants increases with n (0,1,2,3).

In the experiment conducted by z. Huang et. al [15], for each subject, the data collected during the tasks produces fNIRS recordings lasting over 20 minutes. Researchers in [15] used a sliding window approach to extract the exact duration windows from the n-back tasks. Z. Huang et al. extracted overlapping windows with a stride of 0.6 seconds with regular predictions every 0.6 seconds. Classifiers such as Logistic Regression, Deep ConvNet, and EEG Net were used with hyperparameters and tested for fixed-duration multivariate time series windows such that each classifier consumes an fNIRS window and each n-back was assigned to 1 split.
Conclusions from their experiments suggest that the 30-second window combined with the generic paradigm for training data gives better accuracy.

According to Z. Huang et al.,[15] the generic paradigm for training the classifier that uses the available labeled fNIRS recordings is dividing all the subjects into 17 buckets of four subjects each. And from these 17 buckets, 17 train/test splits were formed with each split using one bucket of subjects for the test and all the other buckets combined to form a training data size of 64 subjects (out of a total of 68 subjects). The training dataset was then randomly split into 75% training and 25% validation datasets. It was observed that this generic paradigm for training the classifier outperformed the other two subject-specific and fine-tuning paradigms. Below is the description of the other two paradigms used and evaluated by Z. Huang et al.[15] -
1: Subject-Specific: In the process of finding the best classifier, [15] used only one participant's data for model training and no data from the other subjects. This paradigm trains a separate model for each subject and each hyperparameter.
2: Generic + Fine-tuning: This paradigm aims to leverage both sources of training the model by first pre-training the model on a large generic pool and then fine-tuning it on the target subject. The model that has the best validation accuracy is selected to fine-tune the target subject data.[15]

5. Experimental Evaluation

The research question presented in this document is a two-part case study in which first we aim to find the most suitable classifiers for a well-known dataset by running a comparative study of multiple classifiers and then tuning their parameters to improve their performance, with the goal of finding high workload vs low workload. Second, I use BrainEx to deepen our understanding of the dataset, by finding clusters with the highest number of high and low workloads and exploring the metadata associated with these sequences. To conduct this two-fold case study, I will use our novel tool NeuroHub to find the classifier that best classifies high/low mental workload (NaML) and explore the mental workload dataset [15] using the sequence similarity exploration tool (BrainEx).
Part one of this case study aims to improve the cross-subject generalization for classifying high mental workload (2-back task) and low mental workload (0-back task) from the mental workload dataset by utilizing the Sktime machine learning libraries implemented in NaML. According to the experiment by Z. Huang et al.,[15], the thirty-second window with the generic paradigm (Task Design section 4.2) gave the best classification results with the classifiers they used. In this research, I will train the initial pass on the 30-second window dataset across all the subjects using the default parameters of the classifiers. Based on the classification results from the first pass, I will tune the parameters like max_ensemble_size for the BOSSEnsemble classifier and n_estimators for the TimeSeriesForestClassifier [28], to analyze the

performance of the top three best-performing classifiers with the changed parameters. In this multivariate time series dataset, the goal of part one of this experiment is to find which classification models work the best for classifying cognitive workloads with the highest accuracy.
The second part of the case study will use cluster exploration and similarity search to find the clusters that have the sequences with the highest workload and lowest workloads. This data will be further analyzed to gain some insights into the data representing high or low workload intensity. To do so, I will utilize our novel exploratory tool BrainEx for clustering the sequences having a high concentration of events (high mental workload). Next, I will select the clusters to obtain the sequences across participants such that several clusters will have the highest number of sequences with a high workload, and another set of clusters will have the highest number of sequences with a low workload. These subsets of data containing clusters with high and low mental workloads will be used to explore the mental workload dataset[15] and discover the patterns in high and low mental workloads.

6. Experimental Results and Discussion

6.1 Classification using NaML

Workload classification on the 30-sec sliding window mental workload dataset [15] was started by running all the Sktime classifiers implemented in NaML on this dataset using the default parameters (Appendix section B). The preliminary results indicated Shapelet Transform as the best performer. ShapeletTransform(69.67%), TimeSeriesForest (67.1%)TemporalDictionaryEnsemble(66.18%) are the classifiers that performed better than the other classifiers while using the default parameters. Out of the three best classifiers, ShapeletTransform returned a higher accuracy of approximately 70%. According to L Ye et al., the Shapelet Classifier can perform well on binary data, since classification with a shapelet and its corresponding split point produces a binary decision [29]. The mental workload dataset in this experiment produces binary results (a low (0-back) or a high (2-back) mental workload) and that could be a reason why the Shapelet Classifier performed well on this data.

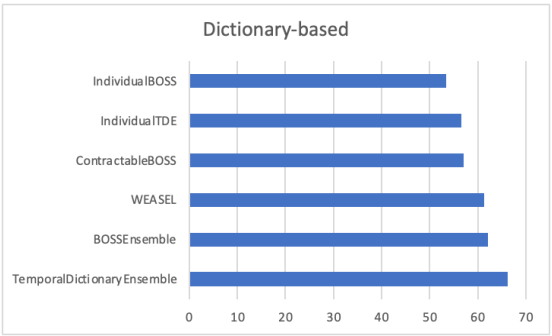


Figure 11. Out of the six dictionary-based classifiers implemented in NaML, TemporalDictionaryEnsemble gave the highest accuracy of 66.18%

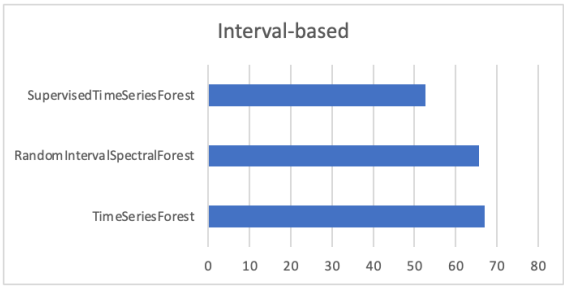


Figure 12. Of the three interval-based classifiers implemented in NaML, TimeSeriesForestClassifier achieved the highest accuracy of 67.1% with the default parameters.

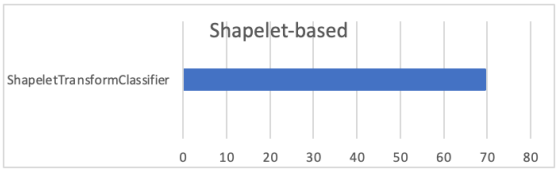


Figure 13. ShapeletTransform is the only classifier from the Shapelet family implemented in NaML which gave the best results for the iteration of workload classification. Accuracy = 69.67%

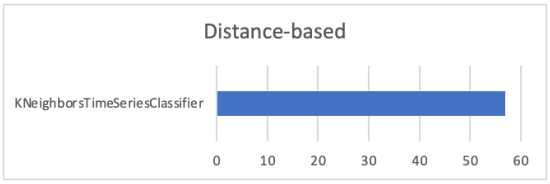


Figure 14. KNeighborsTimeSeriesClassifier is the only classifier from the distance-based family, implemented in NaML.

Further, in phase 2, better classification results were achieved by tuning the parameters of each of these sktime classifiers implemented in NaML. Since the NaML GUI does not have support for running classifiers with parameters other than the default ones, I used config files to run the classifiers on the NaML backend. A config file is a JSON file used to run NaML jobs. Although NaML GUI does not provide support for parameter tuning yet, the NaML server still makes it convenient to work with time series classification. The NaML server is equipped with standardized reformatting scripts which enable researchers to classify any balanced time series brain data using Sktime classifiers. Figure 15 shows a sample config file used for running TimeSeriesForestClassifier on the NaML backend.

```
 GNU nano 4.8                                                    TSF.json
{
        "filepath": "~/NAML/naml_backend/naml_django/naml/data/c2.csv",
        "logging": true,
        "target_col": "event",
        "percentTrain": 0.75,
        "jobs": [
            {
                "classifier": "TimeSeriesForestClassifier",
                "parameters": {
                    "n_estimators": 680,
                    "n_jobs": 2,
                    "random_state": 32
                }
            }
        ]
    }
```

Figure 15. TimeSeriesForest Config file enables users to tune the parameters such as n_estimators, n_jobs, and random_state to achieve a better model performance on classification

In the phase 2 classification with parameters tuning (Appendix section C), the shapelet classifier along with the interval-based sktime classifiers - TimeSeriesForest classifier (TSF), and SupervisedTimeSeriesForest classifier (STSF), gave a better classification accuracy compared to the first iteration. TSF was found to be the best performer after tuning the parameters, resulting in workload-intensity classification accuracy of 73.71%. TSF combines the advantage of calculating temporal features over time series intervals, random feature sampling and uses Entrance (entropy and distance) gain measures to evaluate high-quality splits. This makes the algorithm robust to handle the distortion of the time axis and has a complexity of linear time in the time series length [33].
Comparing the results from phase 2 (parameter tuning iteration) of mental workload classification, with the results from the experiments of Z. Huang et al., on a large pool of participants, the forest classifiers beat all the other classifiers. Z. Huang et al. did not use any Sktime classifiers in their experiments. However, among the classifiers used - Logistic Regression (LR), Random Forest (RF), Deep ConvNet, and EEGNet, RF achieved the highest accuracy of 72% on a dataset of size 64 participants [15]. For this research, I used all the Sktime classifiers implemented in NaML toward mental workload classification on

a pool of 68 participants. As shown in Figure 11, interval-based and shapelet-based families had better average accuracy than all the other classifier families implemented in NaML. Among the interval-based classifiers, TimeSeriesForestClassifier and SupervisedTimeSeriesForest resulted in an average of 72% accuracy for certain parameters in the algorithm and the ShapeletTramsform classifier from the shapelet-based family resulted in a higher accuracy of 70.9% after tuning the parameters. N_estimators for TSF and STSF and n_shapelet_samples for ShapeletTransform were the parameters impacting the classifier accuracy. Results after tuning the parameters for each of these Sktime classifiers are shown in the chart below -

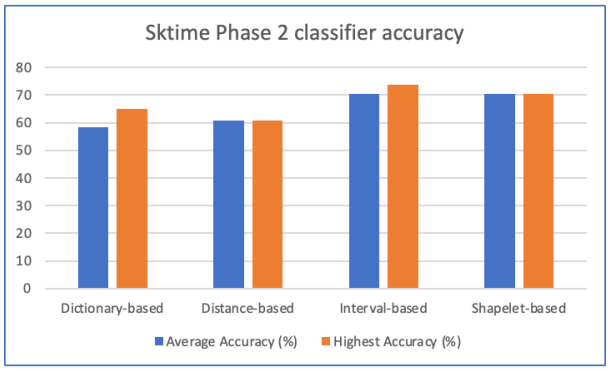


Figure 11. The highest and the average accuracy with Sktime classification in this research on a pool of 68 participants following the usual 75-25 train-test split.

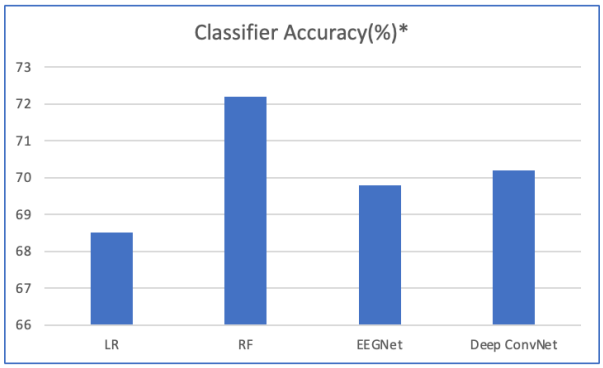


Figure 12. The average accuracy with non-Sktime algorithms from the experiments of Z. Huang et al., with the generic paradigm (Task Design section 4.2) on a pool of 64 participants.

Although Sktime provides state-of-art libraries for time series classification, there is nevertheless a lot of preprocessing of fNIRS brain data required before using the dataset with the classifiers (Sktime libraries). For instance, the NaML server reorders the channels, drops excess columns, and applies the from_long_to_nested function to the dataset, so the table goes from long to nested and returns a sktime-formatted dataset with individual dimensions represented by columns of the output data frame. In addition, the NaML server can also validate the dataset for any null values/missing values that prevent errors and unreliable results. These features of the NaML server let the user run a classifier on a brain signals dataset (fNIRS) without concern for the Sktime-formatting. With these features, using NaML can help to make the classification task uncomplicated and encourage researchers to focus more on the areas like finding the parameters affecting the model generalization on brain data.
From the second iteration and the comparative study of Sktime classification with the results of Z. Huang et al, we can conclude that forest classifiers perform well on this data. I was able to achieve these comparative results in the second iteration of classification after tuning the parameters, which also led to the determination of hyperparameters for the top three winner classifiers. Gaining insights into the data can, however, help improve the accuracy of the model.

6.2 Cluster Exploration and similar sequence search in BrainEx

For the second part of the case study, For the second part of the case study, I utilized our novel exploratory tool BrainEx [13] to find the clusters with the highest number of high workloads and low workloads sequences. To investigate the patterns in the dataset that might give some insights into the workload classification, I used the same 30-sec window dataset for cluster exploration and similarity sequence search in BrainEx.

6.2.1 High Workload Cluster Exploration

To find the patterns in the sequences representing high workload, I explored clusters that had a larger number of event 2 sequences. These were found in clusters of sizes ranging from 10 to 500. I explored all the clusters of variable sequence length but having a percentage of high workload event sequences greater than 66% (Refer to Figure 13). All the larger clusters had a 50-50 distribution of event 0 and event 2 sequences not making the cluster high/low workload cluster (Refer to Figure 14). The high workload clusters contained sequences either from sub_86 or sub_91. For a large pool of 68 participants like this, a cluster could have more participants involved in the high workload representation. However, sequences only from two participants over a variety of clusters might indicate that a smaller cluster size could be directly proportional to the association of lesser participants in the cluster. Furthermore, it was found that the majority of the sequences in these clusters came from the CD_PHI_DO channel (with majority event 2 labels) while there were a few from CD_PHI_O (Refer to Figure 16).

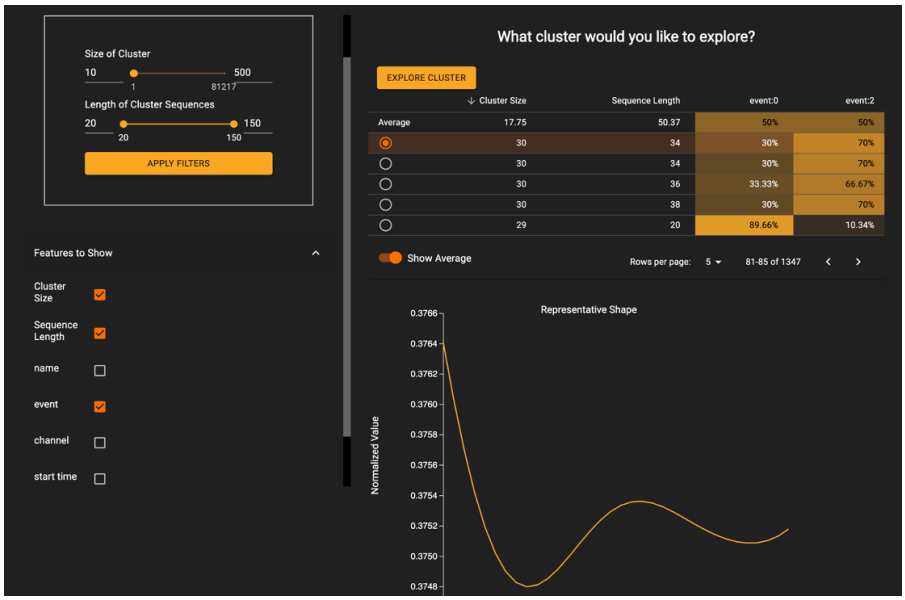


Figure 13. Representing BrainEx cluster explorer, filtering the clusters with size set between 10-500 on the left and displaying the clusters grouped by the sequence length and the class label (event 0/event 2) on the right. The representative shape for the selected cluster suggests that the cluster contains sequences having a similar shape.

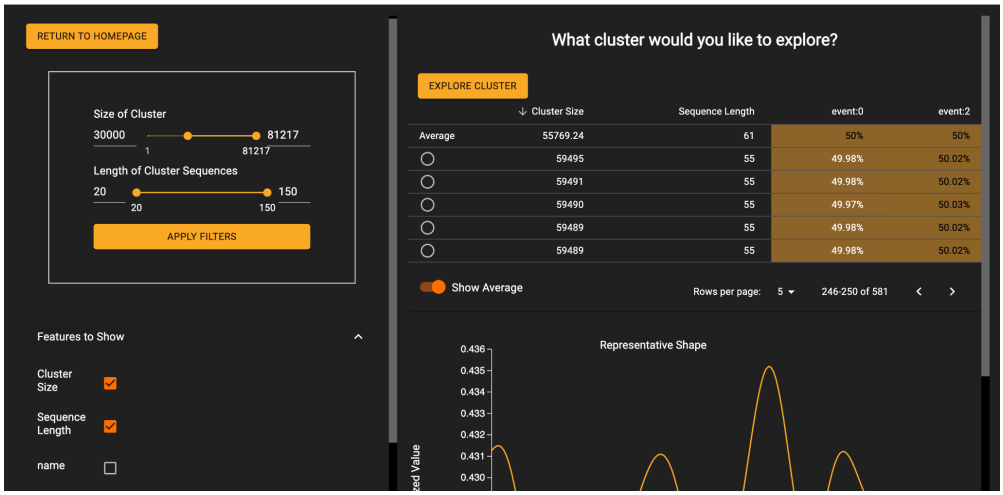


Figure 14. This image shows that the larger clusters are not representative of either of the workloads (event 0/ event 2).

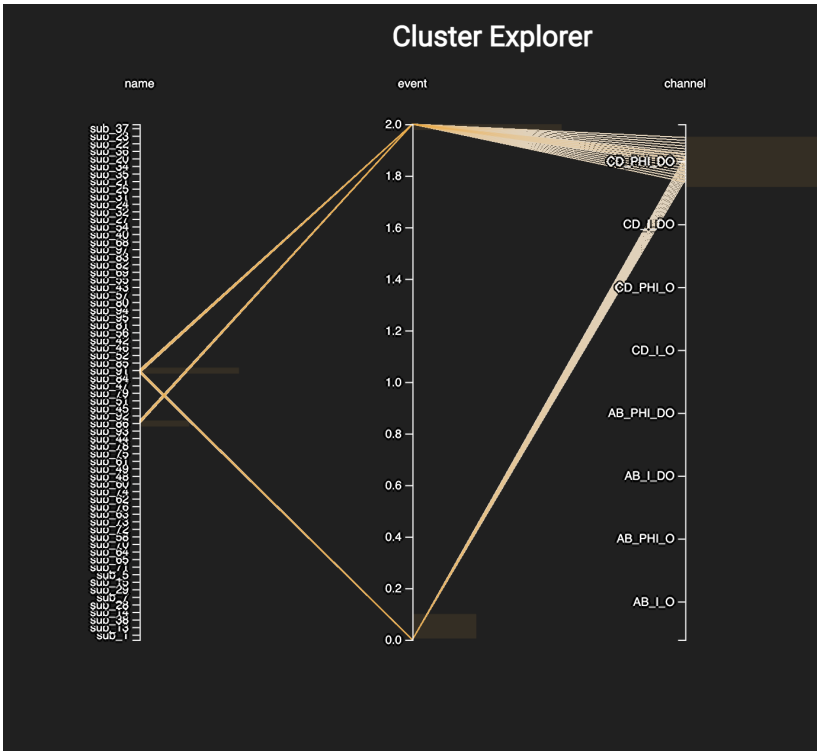


Figure 15. Results from a high workload cluster representing subjects, events, and channels

6.2.2 Low workload cluster exploration

To find the patterns in the sequences representing high workload, I explored clusters that had a larger number of event 0 sequences. These were found in the cluster size range of 10 to 1500 and 50 to 2000. For low workload, I explored all the clusters of variable sequence length but had a percentage of low workload event sequences greater than 74%. The low workload representative clusters were larger

compared to the event 2 clusters. These low workload clusters have a large number of sequences from sub_52, however, there are also sequences from sub_13, sub_49, sub_58, sub_75, sub_86, and sub_91 (Refer to Figure 15). The tendency in the low workload clusters is that the channels and the labels are common across all the clusters. Figure 16 portrays channels and labels for each subject that appeared in the low workload cluster.

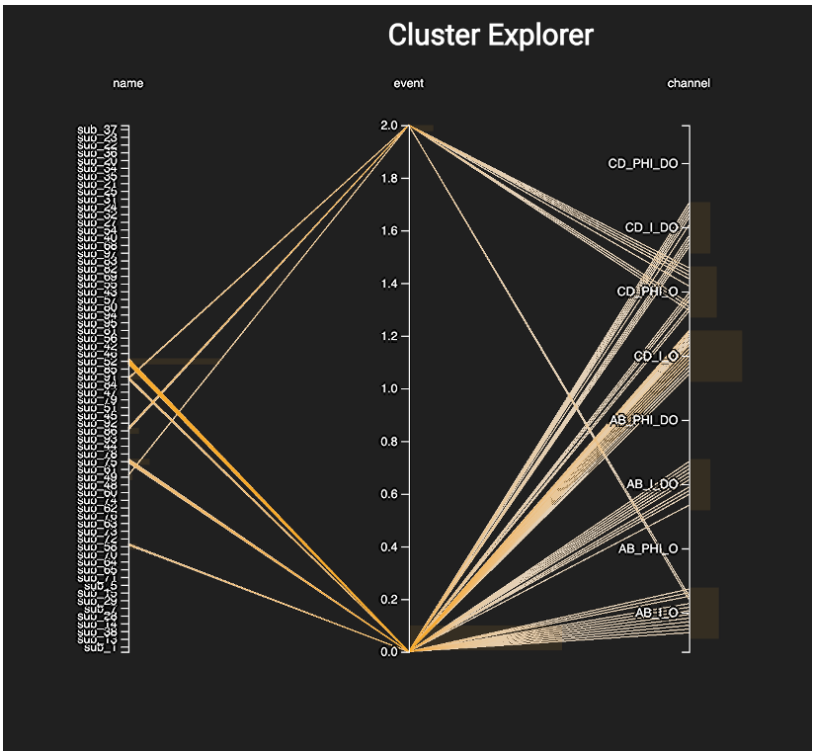


Figure 16. Results from a low workload cluster representing subjects, events, and channels

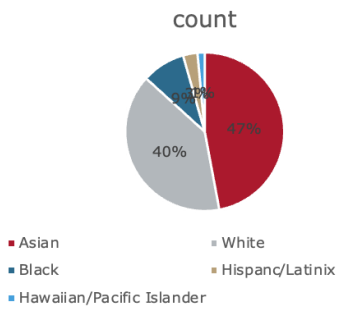
Low workload clusters had sequences from sub_86 and sub_91 however, a larger number of sequences were from the channel CD_PHI_O. Interestingly, a pattern was found in the number of oxygenated hemoglobin (oxy) and deoxygenated hemoglobin (deoxy) sequences in the clusters. For larger low workload clusters (size = 72 and sequences from a couple of subjects), there were 52 oxy sequences and 20 deoxy sequences. And for smaller low workload clusters (size = 28 and sequences from only two subjects 86, 91), there were 19 oxy sequences and 9 deoxy sequences. Another trend in the low workload clusters was observed- though the smaller clusters of event 0 consisted of sequences only from sub_91 and sub_86, most of the sequences were from the CD_PHI_O channel and the ones from CD_PHI_DO were labeled event 0 (Refer to figure 18). This is contrary to the pattern from the high workload clusters where almost all the sequences from sub_86 and sub_91 are deoxy from the channel CD_PHI_DO.

Figure 17. This image shows the distribution of sequences over the channels and participants in a low-workload cluster of size 72.

Figure 18. Represents the trend in the event of 0 clusters of smaller size (cluster size = 28)

6.2.3 Dataset Trends

The mental workload dataset [15] used in this research is limited in terms of racial diversity. Results from the experiments of Z. Huang et al. suggest that a model trained on a dataset consisting of diverse racial groups can generalize well compared to the models trained on homogenous data. Gaining inspiration from these findings, I connected the demography of the participants in either workload cluster. It was observed that the participants representing high workload clusters in this dataset belonged to the white racial group while most of the participants representing low mental workload in this dataset were Asian. The table below shows the racial origins of the participants from the respective intensity cluster.

High workload clusters	Low workload clusters	
Sub_91: White female Sub_86: White male	Sub_91: White female Sub_86: White male Sub_75: Asian male Sub_58: Hawaiian Sub_52: Asian male Sub_49: Asian male Sub_13: Asian female	 Figure 19. Racial distribution in the pool of 68 participants [15]

Results from the BrainEx exploration has sequences from only two participants in the high-workload clusters, while the clusters representing low-workload intensity had sequences only from seven participants. These sequences have similarities in channels and oxy/deoxy levels and establish that further investigation on a larger and less skewed dataset might give some promising results which might be effective toward workload classification. Also, the fact that the original data of 68 participants were skewed on the White and Asian population might be impacting cluster exploration and the link with demographics. Since these sequences representing workload intensities have only a few subjects from a pool of 68 participants, the link between demography and workload representation is also non-conclusive and needs further investigation.

7. Limitations of our existing tools

To benefit from the insights from BrainEx, I attempted to merge the low and high workload clusters. The clusters obtained through cluster exploration in BrainEx were small compared to the pool of participants. However, these clusters were representatives of the two workload intensities. Using the data from the respective clusters, we are removing the sequences that are not representative of the intensities. With such data, we can expect to get a better classification performance. However, in the process of using the data from BrainEx in NaML, the following limitations were encountered -

7.1 Duplicates in NaML

The first two sequences in figure 20 are from the same subject with the same event label, same channel, and different start times. This is a file exported from BrainEx, and it represents one of the low-workload clusters. It is interesting to note that for sub_49 the signals coming from channel AB_I_O were prominent, having the start times in a similar range. It is also important to note that, BrainEx chunks the sequences into sub-sequences, and that is the reason we get sequences and clusters of all lengths. These two signals could be sub-sequence of the same sequence from the master dataset, however, they are representing two different time series of the same length in the low-workload cluster. While the first five columns of these series are the same, the two columns relevant to BrainEx - Start Index (start index is a starting point relative to the sequence) and End Index are different. Although these series are not duplicates, NaML catches those as duplicates with respect to the first five columns. One way to address this problem can be by removing one of the entries manually. But there can be multiple such sequences

and removing all of them to get rid of the null value error will lead to significant data loss. Another way to address this problem can be jittering the start time values - manually adding the value of the start index to the start time. Modifying the start times is one of the ways to deal with this problem but moreover, it is an adjustment to make this data work in NaML and not a solution to this problem.

	A	B	C	D	E	F	G	H	I	J	K
1	name	event	channel	start time	end time	Start Index	End Index	1	2	3	4
2	sub_49	2	AB_I_O	40817.308	40845.962	15	74	0.4503536	0.4499885	0.4496299	0.4492796
3	sub_49	2	AB_I_O	40817.308	40845.962	16	75	0.4499885	0.4496299	0.4492796	0.4489391
4	sub_49	2	AB_I_O	40834.308	40845.962	17	76	0.4496299	0.4492796	0.4489391	0.4486092
5	sub_49	2	AB_I_O	40835.308	40845.962	18	77	0.4492796	0.4489391	0.4486092	0.4482905
6	sub_49	2	AB_I_O	40837.308	40845.962	20	79	0.4486092	0.4482905	0.4479835	0.4476886
7	sub_49	2	AB_I_O	40839.308	40845.962	22	81	0.4479835	0.4476886	0.4474063	0.4471368
8	sub_49	2	AB_I_O	40840.308	40845.962	23	82	0.4476886	0.4474063	0.4471368	0.4468804
9	sub_49	2	AB_I_O	40841.308	40845.962	24	83	0.4474063	0.4471368	0.4468804	0.4466371
10	sub_49	2	AB_I_O	40842.308	40845.962	25	84	0.4471368	0.4468804	0.4466371	0.4464066
11	sub_49	2	AB_I_O	40844.308	40845.962	27	86	0.4466371	0.4464066	0.4461881	0.4459802
12	sub_52	0	CD_I_O	576.92308	605.57692	0	59	0.4607366	0.4606425	0.460557	0.4604864
13	sub_52	0	AB_I_DO	576.92308	605.57692	0	59	0.4470399	0.446974	0.4469083	0.4468417

Figure 20. A screenshot of the data from a low-workload cluster CSV file exported from BrainEx.

7.2 Null values in Sktime

The last two sequences in figure 20 are from sub_52, and the sequences above those are from sub_49. Here, the signals from sub_52 are coming from 2 channels - AB_I_DO and CD_I_O. However, all the signals for sub_49 are from AB_I_O. All the sequences for sub_49 are 2 and all the labels for sub_52 are 0. So, again we have some interesting outcomes from the cluster exploration BrainEx. Looking only at this low-workload cluster, we observe that 0 & 2 event signals are from different channels which could be a valuable insight into the workload intensities. While attempting to classify this in NaML, Sktime is assuming null values for the channels not having any data for that subject in that range of time. There can be an adjustment to include the sequences from other channels in this dataset. And we can do this by adding the sequences from the master dataset to this dataset.

7.3 Curating the data manually

The data exported from BrainEx cannot be used in NaML directly due to the stated limitations. However, in order to make this data work NaML needs these manual adjustments. Reducing the task of curating the data manually needs further investigation.

8. Conclusion and Future Work

This research builds a foundation for using exploratory insights with machine learning models to improve classification accuracy on brain data. Using state-of-art libraries such as Sktime makes machine learning on time series data efficient, especially for researchers without a background in Computer Science or Data Science. Tuning parameters is worth spending time since it has the potential to improve the classifier performance and can lead to the determination of hyperparameters for the classifiers. Finally, insights from sequence metadata have the potential to improve model generalization by using the data from BrainEx in NaML and running classification iterations on this dataset.

In this research, I used the 30-sec slide window data for the classification as well as cluster exploration. The classification results were compared with the results from the researchers of Tufts University [15] after parameter tuning. Although this 30-sec slide window data had a large pool of 68 participants, the clusters obtained after cluster exploration were quite small and also had fewer participants in comparison to the pool. Second, this benchmark dataset had the limitation of being skewed on two racial groups which are influential for the behavior of the classifiers. Hence, using a large balanced dataset for cluster exploration & classification is worth considering for future research.

This research used all the 12 Sktime classifiers implemented in NaML. an avenue for future work would be to expand NaML to include the other Sktime classifiers. This will expose the researchers to more classifiers to test their data and might get better classification models in the process.

In an attempt to improve the model generalization, this study met some limitations related to the tools, especially in the area of data exported from BrainEx to NaML. Finding solutions to the problems (# Limitations 1, 2) and figuring out a way to use the BrainEx insights into NaML has the potential to introduce new perspectives to time series classification.

References

[1] Parasuraman, R., Sheridan, T. B., and Wickens, C. (2008). Situation awareness, mental workload, and trust in automation: viable, empirically supported cognitive engineering constructs. J. Cogn. Eng. Decis. Mak. 2, 140–160. DOI: 10.1518/155534308x284417

[2] McKendrick R and Harwood A (2019) Cognitive Workload and Workload Transitions Elicit Curvilinear Hemodynamics During Spatial Working Memory. Front. Hum. Neurosci. 13:405. DOI: 10.3389/fnhum.2019.00405

[3] Babiloni, F. (2019). Mental Workload Monitoring: New Perspectives from Neuroscience. In: Longo, L., Leva, M. (eds) Human Mental Workload: Models and Applications. H-WORKLOAD 2019. Communications in Computer and Information Science, vol 1107Springer, Cham. https://doi.org/10.1007/978-3-030-32423-0_1

[4] Arico, P., et al.: Human factors and neurophysiological metrics in air traffic control: a critical review. IEEE Rev. Biomed. Eng. PP(99), 1 (2017)

[5] Borghini, G., Aricò, P., Di Flumeri, G., Babiloni, F.: Industrial Neuroscience in Aviation: Evaluation of Mental States in Aviation Personnel. BIOSYSROB, vol. 18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58598-7

[6] Longo L. Experienced mental workload, perception of usability, their interaction and impact on task performance. PLoS One. 2018 Aug 1;13(8):e0199661. DOI: 10.1371/journal.pone.0199661. PMID: 30067747; PMCID: PMC6070185.

[7] Y. Yu et al., "FlyingBuddy2: a brain-controlled assistant for the handicapped," presented at the Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, Pennsylvania, 2012. [Online]. Available: https://doi.org/10.1145/2370216.2370359.

[8] Z. Chen et al., "Paint with Your Mind: Designing EEG-based Interactive Installation for Traditional Chinese Artworks," presented at the Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction, Salzburg, Austria, 2021. [Online]. Available: https://doi.org/10.1145/3430524.3442455.

[9] R. Ramchurn, S. Martindale, M. L. Wilson, and S. Benford, "From Director's Cut to User's Cut: to Watch a Brain-Controlled Film is to Edit it," presented at the Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019.

[10] D. Marshall, D. Coyle, S. Wilson, and M. Callaghan, "Games, Gameplay, and BCI: The State of the Art," IEEE Transactions on Computational Intelligence and AI in Games, vol. 5, no. 2, pp. 82-99, 2013, DOI: 10.1109/tciaig.2013.2263555.

[11] M Ferrari and V Quaresima. 2012. A brief review on the history of human functional near-infrared spectroscopy (fNIRS) development and fields of application. NeuroImage 63 (2012), 921–935

[12] Erin T. Solovey and Felix Putze. 2021. Improving HCI with Brain Input: Review, Trends, and Outlook. Now Publishers Inc

[13] Alicia Howell-Munson, Christopher Micek, Ziheng Li, Michael Clements, Andrew C. Nolan, Jackson Powell, Erin T. Solovey, and Rodica Neamtu. 2022. BrainEx: Interactive Visual Exploration and Discovery of Sequence Similarity in Brain Signals. Proc. ACM Hum.-Comput. Interact. 6, EICS, Article 162 (June 2022), 41 pages. https://doi.org/10.1145/3534516

[14] Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J. (2019). sktime: a unified interface for Machine Learning with Time Series. LearningSys.

[15] Zhe Huang, Liang Wang, Giles Blaney, Christopher Slaughter, Devon McKeon, Ziyu Zhou, Robert J. K. Jacob1, and Michael C. Hughes, “The Tufts fNIRS Mental Workload Dataset & Benchmark for Brain-Computer Interfaces that Generalize”.

[16] Chen, W., Wagner, J., Heugel, N., Sugar, J., Lee, Y., Conant, L., Malloy, M., Heffernan, J., Quirk, B., Zinos, A., Beardsley, S. A., Prost, R., & Whelan, H. T. (2019). Functional Near-Infrared Spectroscopy and Its Clinical Application in the Field of Neuroscience: Advances and Future Directions. *Frontiers in Neuroscience*. https://doi.org/10.3389/fnins.2020.00724

[17] “6.4.4 Univariate Time Series Models”, “Engineering Statistics Handbook” nist.gov/

[18] Beeram, S.R., Kuchibhotla, S. (2021). Time Series Analysis on Univariate and Multivariate Variables: A Comprehensive Survey. In: Satapathy, S.C., Bhateja, V., Ramakrishna Murty, M., Gia Nhu, N., Jayasri Kotti (eds) Communication Software and Networks. Lecture Notes in Networks and Systems, vol 134. Springer, Singapore. https://doi.org/10.1007/978-981-15-5397-4_13

[19] Laura Morán-Fernández, Verónica Bólon-Canedo, Amparo Alonso-Betanzos, How important is data quality? Best classifiers vs best features, Neurocomputing, Volume 470, 2022, Pages 365-375, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2021.05.107.

[20] Ikram, F. (2019). NirsAutoML: Building an automated classification platform for fNIRS data. : Worcester Polytechnic Institute.

[21] Buntel, E. (2020). Brain Wave Analysis. : Worcester Polytechnic Institute.

[22] Zabihi, Morteza & Ince, Turker & Kiranyaz, Serkan & Gabbouj, Moncef. (2017). Learned vs. Hand-Designed Features for ECG Beat Classification: A Comprehensive Study. 10.1007/978-981-10-5122-7_138.

[23] Y. Cheng, H. Goh, K. Dogrusoz, O. Tuzel, and E. Azemi. Subject-aware contrastive learning for biosignals. arXiv preprint arXiv:2007.04871, 2020

[24] K. Han and J.-H. Jeong. Domain Generalization for Session-Independent Brain-Computer Interface. arXiv:2012.03533 [cs], 2020. http://arxiv.org/abs/2012.03533.

[25] Theodore J Huppert, Solomon G Diamond, Maria A Franceschini, and David A Boas. 2009. HomER: a review of time-series analysis methods for near-infrared spectroscopy of the brain. Applied optics 48, 10 (April 2009), D280—98. https://doi.org/10.1364/ao.48.00d280

[26] Xin Hou, Zong Zhang, Chen Zhao, Lian Duan, Yilong Gong, Zheng Li, and Chaozhe Zhu. 2021. NIRS-KIT: a MATLAB toolbox for both resting-state and task fNIRS data analysis. Neurophotonics 8, 1 (January 2021), 010802. https: //doi.org/10.1117/1.nph.8.1.010802

[27] Michael Lührs and Rainer Goebel. 2017. Turbo-Satori: a neurofeedback and brain–computer interface toolbox for real-time functional near-infrared spectroscopy. Neurophotonics 4, 4 (2017), 041504.

[28] Welcome to Sktime, https://www.sktime.org/en/stable/

[29] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). Association for Computing Machinery, New York, NY, USA, 947–956. https://doi.org/10.1145/1557019.1557122

[30] Hicks, K., Gerling, K., Dickinson, P. and Vanden Abeele, V. Juicy game design: Understanding the impact of visual embellishments on player experience. City, 2019

[31] Hunicke, R., LeBlanc, M. and Zubek, R. MDA: A formal approach to game design and game research. City, 2004.

[32] Marshall, D., Coyle, D., Wilson, S. and Callaghan, M. Games, Gameplay, and BCI: The State of the Art. IEEE Transactions on Computational Intelligence and AI in Games, 5, 2 (2013), 82-99.

[33] Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A Time Series Forest for Classification and Feature Extraction. *arXiv*. https://doi.org/10.48550/arXiv.1302.2277

[34] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2013. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. ACM Transactions on Knowledge Discovery from Data (TKDD) 7, 3 (2013), 1–31.

[35] Link to NeuroHub github repo: https://github.com/WPIHCILab/NeuroHub/tree/version1_code

Appendix

A. Preparing the dataset for NaML

In this study, I used the preprocessed 30-sec sliding window dataset for classification and exploration tasks. This section discusses the step that needs to be completed to make the dataset Sktime and NaML compatible. NeuroHub has a format converter feature for using open-source datasets with BrainEx and NaML. The script accesses a directory on the machine to combine and reformat the required CSVs. The directory path can be changed according to the location. However, the compatible "NaMLCompatible.csv" file gets downloaded under /downloads on the user's machine. This file can be uploaded to NaML for the classification task.

B. Default parameters of classifiers in NaML

This section shows the classifiers implemented in NaML grouped by their classifier families with their default parameters and classification results from the phase 1 classification of this study.

B.1. Dictionary-based classifiers

For time series classification, dictionary-based methods modify the bag of words paradigm, which is frequently used in signal processing, computer vision, and audio processing. A series is traversed by a sliding window of varying length. The real-valued series of length for each window is turned into a symbolic string of length consisting of potential letters using approximation and discretization procedures. Each occurrence of each 'word' from the dictionary in a series is counted, and after the sliding window is complete, the series is turned into a histogram. The histograms of the words derived from the series, rather than the raw data, are used for classification [28].

BOSSEnsemble	ContractableBOSS	IndividualBOSS
threshold:0.92 max_ensemble: 500 max_win_len_prop: 1 min_window: 10 random_state: 0 n_jobs: 1 column: 0 **Accuracy: 62.13%**	n_parameter_samples: 250 max_ensemble_size: 50 max_win_len_prop: 1 min_window: 10 time_limit_in_minutes: 0 save_train_predictions: False n_jobs: 1 random_state: 0 column: 0 **Accuracy: 56.99%**	window_size: 10 word_length: 8 norm: False alphabet_size: 4 save_words: True n_jobs: 1 random_state: 0 column: 0 **Accuracy: 53.49%**
IndividualTDE	TemporalDictionaryEnsemble	WEASEL
window_size: 10 word_length: 8 norm: False levels: 1 lgb: False	n_parameter_samples: 250 max_ensemble_size: 50 max_win_len_prop: 1 min_window: 10 randomly_selected_params:	anova: True bigrams: True binning_strategy: information-gain window_inc: 4

alphabet_size: 4 bigrams: True dim_threshold: 0.85 max_dims: 20 n_jobs: 1 random_state: 0 column: 0 **Accuracy: 56.62%**	50 bigrams:none dim_threshold: 0.85 max_dims: 20 **Accuracy: 66.18%**	chi2_threshold: -1 random_state: 0 column: 0 **Accuracy: 61.4%**

B.2. Interval-based classifiers

Interval-based techniques examine the complete series' phase-dependent intervals, producing summary statistics from selected sub-series for classification. These classifiers are decision tree ensembles built of time-series intervals. Every algorithm's approach to selecting and building ensembles on the intervals varies which makes these algorithms different [28].

TimeSeriesForestClassifier	SupervisedTimeSeriesForest	RandomIntervalSpectralEnsemble
min_interval: 3 n_estimators: 200 n_jobs: 1 random_state: 0 column : 0 **Accuracy: 67.1%**	n_estimators: 500 n_jobs: 1 random_state: 0 column : 0 **Accuracy: 52.76%**	min_interval: 16 n_estimators: 200 n_jobs: 1 random_state: 0 column : 0 acf_lag: 100 acf_min_values: 4 **Accuracy: 65.62%**

B.3. Shapelet-based classifiers

Shapelet is a time-series subsequence representing class membership and measure phase-independent similarity between time series. The shapelet transform focuses on the shape of the data and improves on the usage of shapelets by separating shapelet extraction from the classification, then enabling interpretable phase-independent classification of time series using any conventional classification technique such as random forest [28].

ShapeletTransformClassifier
n_shapelet_samples: 10000 max_shapelets: None max_shapelet_length: None estimator: None transform_limit_in_minutes: 0 time_limit_in_minutes: 0 save_transformed_data: False n_jobs: 1 batch_size: 100 column: 0 **Accuracy: 69.67%**

B.4. Distance-based classifiers

A time series version of the scikit-learn KNeighborsClassifier is the classifier KNeighborsTimeSeriesClassifier in the distance-based family. The KNN Time Series Classifier supports time series distance measurements [28].

KNeighborsTimeSeriesClassifier
n_neighbors: 1 weights: uniform distance: dtw distance_params: None column: 0 **Accuracy: 56.99%**

C. Results for parameter tuning in NaML for each classifier

This section displays results for all iterations of parameter tuning for each Sktime classifier implemented in NaML.

C.1. BOSSEnsemble

Alphabet_size: 8 Min_window: 25 **Accuracy: 50.37%**	Alphabet_size: 8 Min_window: 15 **Accuracy: 50.00%**	Alphabet_size: 8 Min_window: 30 **Accuracy: 50.37%**
Alphabet_size: 10 Min_window: 20 **Accuracy: 50.18%**	Alphabet_size: 5 Min_window: 25 **Accuracy: 52.76%**	Alphabet_size: 12 Min_window: 35 **Accuracy: 50.00%**

C.2. ContractableBOSS

alphabet_size: 5 min_window: 20 **Accuracy: 61.21%**	alphabet_size: 5 min_window: 25 **Accuracy: 61.58%**	alphabet_size: 8 min_window: 35 **Accuracy: 60.85%**
alphabet_size: 8 min_window: 30 **Accuracy: 64.34%**	alphabet_size: 8 min_window: 25 **Accuracy: 65.07%**	alphabet_size: 5 min_window: 12 **Accuracy: 62.13%**

C.3. IndividualBOSS

window_size: 20 n_jobs: 6	window_size: 5 n_jobs: 6	window_size: 2 n_jobs: 8

norms: true random_state: 37 **Accuracy: 52.57%**	norms: true random_state: 37 **Accuracy: 52.76%**	norms: true random_state: 37 **Accuracy: 50%**
window_size: 5 n_jobs: 6 norms: false random_state: 37 **Accuracy: 54.78%**	window_size: 5 n_jobs: 6 norms: false random_state: 37 alphabet_size: 8 **Accuracy: 53.49%**	window_size: 5 n_jobs: 6 norms: false random_state: 37 alphabet_size: 16 **Accuracy: 52.57%**

C.4. IndividualTDE

window_size: 15 word_length: 8 alphabet_size: 4 n_jobs: 1 random_state: 0 **Accuracy: 53.23%**	window_size: 5 word_length: 8 alphabet_size: 4 n_jobs: 6 random_state: 37 **Accuracy: 54.04%**	window_size: 5 word_length: 6 alphabet_size: 4 n_jobs: 6 random_state: 24 **Accuracy: 53.49%**
window_size: 3 word_length: 6 alphabet_size: 4 n_jobs: 6 random_state: 24 **Accuracy: 50.28%**	window_size: 8 word_length: 6 alphabet_size: 4 n_jobs: 6 random_state: 24 **Accuracy: 53.88%**	window_size: 6 word_length: 8 alphabet_size: 4 n_jobs: 6 random_state: 37 **Accuracy: 52.94%**

C.5. TemporalDictionaryEnsemble

n_parameter_samples: 300 max_ensemble_size: 50 save_train_predictions: true time_limit_in_minutes: 30 **Accuracy: 64.71%**	n_parameter_samples: 560 max_ensemble_size: 50 save_train_predictions: true randomly_selected_params:5 **Accuracy: 59.93%**	n_parameter_samples: 350 max_ensemble_size: 10 save_train_predictions: true time_limit_in_minutes: 30 **Accuracy: 61.21%**
n_parameter_samples: 350 max_ensemble_size: 50 randomly_selected_params:5 save_train_predictions: true n_jobs:4 **Accuracy: 61.58%**	n_parameter_samples: 750 max_ensemble_size: 50 randomly_selected_params:25 save_train_predictions: true n_jobs:4 **Accuracy: 61.21%**	n_parameter_samples: 250 max_ensemble_size: 50 save_train_predictions: false n_jobs:1 **Accuracy: 61.21%**

C.6. WEASEL

anova: true window_inc:5 random_state: 37 **Accuracy: 58.82%**	anova: true window_inc:25 random_state: 37 **Accuracy: 59.01%**	anova: true window_inc:30 random_state: 37 **Accuracy: 58.09%**
anova: false window_inc:25 random_state: 25 **Accuracy: 57.9%**	anova: true feature_selection:”random” window_inc:25 random_state: 25 **Accuracy: 58.96%**	anova: true feature_selection:”random” window_inc:25 random_state: 25 n_jobs: 52.94% **Accuracy: 58.96%**

C.7. TimeSeriesForestClassifier

N_estimators: 350 N_jobs: 1 Random_state: 42 **Accuracy: 71.14%**	n_estimators:450 n_jobs:1 random_state:42 **Accuracy:71.88%**	n_estimators:650 n_jobs:1 random_state:25 **Accuracy:72.06%**	n_estimators:650 n_jobs:2 random_state:32 **Accuracy:66%**	n_estimators:670 n_jobs:2 random_state:32 **Accuracy:69.67%**
n_estimators:680 n_jobs:2 random_state:32 **Accuracy:73.53%**	n_estimators:720 n_jobs:2 random_state:32 **Accuracy:72.06%**	N_estimators:770 n_jobs:2 random_state:32 **Accuracy:73.71%**	n_estimators:770 n_jobs:5 random_state:25 **Accuracy:70.96%**	n_estimators:1000 n_jobs:3 random_state:25 **Accuracy:71.32%**
n_estimators:750 n_jobs:4 random_state37: **Accuracy:69.85%**	n_estimators:770 n_jobs:3 random_state:42 **Accuracy:71.69%**	n_estimators:750 n_jobs:2 random_state:32 **Accuracy:72.24%**	n_estimators:100 n_jobs:2 random_state:32 **Accuracy:68.3%**	n_estimators:770 n_jobs:2 random_state:25 **Accuracy:72.61%**

C.8. SupervisedTimeSeriesForest:

n_estimators:770 n_jobs:2 random_state:42 **Accuracy: 67%**	n_estimators:770 n_jobs:4 random_state:42 **Accuracy:68.93%**	n_estimators:820 n_jobs:4 Random_state:42 **Accuracy:65.44%**	n_estimators:750 n_jobs:4 Random_state:37 **Accuracy:69.85%**	n_estimators:735 n_jobs:4 Random_state:42 **Accuracy:68.57%**
n_estimators:735 n_jobs:6 Random_state:37	n_estimators:750 n_jobs:6 Random_state:37	n_estimators:250 n_jobs:6 Random_state:37	n_estimators:740 n_jobs:6 Random_state:37	n_estimators:735 n_jobs:8 Random_state:37

Accuracy:70.59%	**Accuracy:68.38%**	**Accuracy:66.91%**	**Accuracy:70.22%**	**Accuracy:68.75%**
n_estimators:735 n_jobs:2 Random_state:37 **Accuracy:70.96%**	n_estimators:745 n_jobs:6 Random_state:37 **Accuracy:66.18%**	n_estimators:780 n_jobs:6 Random_state:37 **Accuracy:66.54%**	n_estimators:735 n_jobs:7 Random_state:37 **Accuracy:68.01%**	n_estimators:760 n_jobs:4 Random_state:42 **Accuracy:69.01%**

C.9. RandomIntervalForestEnsemble:

n_estimators: 550 n_jobs: 4 Random_state:37 **Accuracy: 63.97%**	n_estimators: 735 n_jobs: 2 Random_state:37 **Accuracy: 64.71%**	n_estimators: 740 n_jobs: 6 Random_state:37 **Accuracy: 66.37%**	n_estimators: 740 n_jobs: 6 Random_state:37 acf_lag:50 **Accuracy: 65.07%**
n_estimators: 740 n_jobs: 6 Random_state:37 acf_lag:150 **Accuracy: 62.68%**	n_estimators: 735 n_jobs: 6 Random_state:37 acf_min_values:16 **Accuracy: 63.6%**	n_estimators: 770 n_jobs: 2 Random_state:37 **Accuracy: 64.15%**	n_estimators: 820 n_jobs: 2 Random_state:37 **Accuracy: 63.6%**

C.10. ShapeletTransformClassifier:

n_shapelet_samples: 10000 max_shapelets: 10 time_limit_in_minutes: 5 n_jobs: 4 batch_size: 20 **Accuracy: 61%**	n_shapelet_samples: 10000 time_limit_in_minutes: 5 n_jobs: 1 **Accuracy: 62.23%**	n_shapelet_samples: 8500 time_limit_in_minutes: 30 n_jobs: 1 **Accuracy: 58.27%**	n_shapelet_samples: 8000 n_jobs: 1 **Accuracy: 69.3%**
n_shapelet_samples: 8000 n_jobs: 4 **Accuracy: 68.38%**	n_shapelet_samples: 12000 n_jobs: 1 **Accuracy: 70.4%**	n_shapelet_samples: 14000 n_jobs: 1 **Accuracy:69.69%**	n_shapelet_samples: 11000 n_jobs: 1 **Accuracy: 67.28%**

C.11. KNeighborsTimeSeriesClassifier

n_neighbors: 2 **Accuracy: 56.92%**	n_neighbors: 5 **Accuracy: 58.82%**	n_neighbors: 10 Distance: “euclidean” wgt: “distance” **Accuracy: 61.95%**	n_neighbors: 35 distance: “euclidean” wgt: “distance” **Accuracy: 59.93%**
n_neighbors: 20 **Accuracy: 62.13%**	n_neighbors: 100 **Accuracy: 58.82%**	n_neighbors: 50 distance: “euclidean” wgt: “distance” **Accuracy: 60.29%**	n_neighbors: 35 wgt: “distance” **Accuracy: 60.85%**