An Efficient Machine Learning Software Architecture for Internet of Things

Mahima Chaudhary

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE
STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2021

Abstract

Internet of Things (IoT) software is becoming a critical infrastructure for many domains. In IoT, sensors monitor their environment and transfer readings to cloud, where Machine Learning (ML) provides insights to decision-makers. In the healthcare domain, the IoT software designers have to consider privacy, real-time performance and cost in addition to ML accuracy. We propose an architecture that decomposes the ML lifecycle into components for deployment on a two-tier cloud, edge-core. It enables IoT time-series data to be consumed by ML models on edge-core infrastructure, with pipeline elements deployed on any tier, dynamically. The architecture feasibility and ML accuracy are validated with three brain-computer interfaces (BCI) based use-cases. The contributions are two-fold: first, we propose a novel ML-IoT pipeline software architecture that encompasses essential components from data ingestion to runtime use of ML models; second, we assess the software on cognitive applications and achieve promising results in comparison to literature.

Acknowledgement

Firstly, I would like to express my gratitude to my supervisor, Professor Marin Litoiu. Under his guidance, I was able to get involved in fields of Internet of Things, Machine Learning and Cloud Computing. I could not have accomplished this thesis without his valuable insights and feedback. Professor Litoiu would constantly provide new perspectives and insights to my research that were invaluable to my progress in the Master's program. I am genuinely appreciative of the assistance of Professor Lauren Sergio. With her combined knowledge of the field of computational neuroscience, her vision and expertise were essential for this accomplishment. I am deeply grateful of the valuable research experience that I gained working under her. In addition, I kindly thank Dr. Sumona Mukhopadhyay for generously providing a large amount of positive feedback and constructive criticism to my work. I want to thank Dr. Meaghan Adams for helping me with the concepts of cognitive neuroscience and for continuously giving me feedback on my work. I give my thanks to the examination committee members Professor Hamzeh Khazaei and Professor Manar Jammal for taking out time to be a part of the examination committee. I would also like to express my gratitude to all the members I worked with at CERAS labs. I would like to thank York University's Human Participants Research Committee for approving all procedures for human participation. I would also like to thank the various people to voluntarily participated in the experiments.

Finally, I want to dedicate this work to my father Mr.Dinesh Chaudhary, my mother Mrs. Sangeeta Chaudhary, my brother Sujay, my sister Mohita and many of my friends for supporting me throughout my studies as a graduate student. All of your presence has brought immeasurable positive energy in my life. Thank you for all the unconditional love and constant support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

IoT is a technological revolution as it connects an enormous amount of heterogeneous devices that are equipped with control, communication, sensing and actuating capabilities [1]. Thereby, IoT results in an avalanche of data and it is therefore one of the main contributors to the Big Data paradigm. Innovative applications can be developed by connecting the IoT devices globally to the Internet, though to obtain economic benefits, the huge amount of generated data must be analyzed [2]. Thus, IoT applications must be equipped with an intelligent processing system that permits to infer, learn and extract information from the IoT data [3]. This capacity is provided by machine learning (ML). The conventional approach for IoT data analytics sends all the data to a single service, where ML algorithms are applied to extract valuable information for the end-user applications after cleaning and preprocessing the data. In this work however, we propose a ML architecture where the workflow resides flexibly in an edge-core architecture, and each part of the pipeline works independently. This facilitates pipeline to be highly maintainable, testable and independently deployable. Thus, each part like feature engineering and application of model can be invoked separately.

1.1 Problem and Motivation

Recently, many IoT applications have been actively exploiting ML technologies, which use Deep Learning (DL) [4] models like deep neural networks to capture and understand their environments. For instance, Amazon Echo, Google Home and Apple Homepod can be categorised as IoT applications as they connect the physical and human world with the digital world, by translating human voice commands using DL. The training of datasets for ML models must be managed across multiple physical environments throughout their lifecycle, from ingestion to transformation, exploration, training and deployment. In a mainstream system design, all of these tasks would run together in a monolithic manner. This means the same script will extract the data, clean and prepare it, model it, and deploy it. Since ML models usually consist of far less code than other software applications, the approach to keep all of the assets in one place makes sense. However, scaling of this monolith architecture can give rise to problems related to volume for instance the ingestion and preparation will be repeated regularly although they are exactly identical. Also the versioning will be difficult, because changing the configuration will require manual update of all the scripts, which is time consuming and error-prone. We therefore intend to organize the ML workflow in a manner that each process can work independently of the rest. In our proposed method, ML benefits from a holistic architecture with data management that extends across two tiers edge and core.

A smart data architecture alleviates the exhaustive data housekeeping tasks so that they can focus on AI applications that deliver business value. This work describes an edge-core ML architecture for IoT applications. We explore the possibility to divide the ML pipeline from ingestion to model deployment, into independent and compact units such that they can be placed on the edge or core. We describe this architecture by applying it

to the most relevant usecases based on Brain-Computer Interfaces (BCI) [5] i.e. applications that integrate computational neuroscience with algorithms to control the environment. BCI is often defined as the communication system that acquires brain signals, monitors and examines them to translate the cerebral activity to certain features, corresponding to user intentions, to control sensing and actuation of devices remotely. The advancements in sensor technologies have facilitated the growth of wearable headband devices for development in BCI applications and therefore it is emerging as a novel alternative for supporting interaction between IoT devices and individuals. The unification of BCI and IoT systems can enable interaction from the brain to smart objects (i.e. sensors) which could allow actuation and sensing to be performed on smart devices through commands sent by people from their controlled brain activity. In terms of application/implementation, this thesis mainly focuses on the potential of Electroencephalogram (EEG) signals from a portable, wearable and non-invasive device to capture the brain activity associated to the user intent. The raw incoming data from the wearable EEG has been cleaned, preprocessed and engineered into meaningful data which can be fed to a ML pipeline and the results can be used to regulate and operate devices and other applications through an integrated edge-core IoT framework.

In this study, a flexible machine learning architecture for edge-core IoT applications has been proposed, using a wearable EEG which can be used for BCI control operations. We have designed an ML-IoT infrastructure that can be used in a variety of domains, from creating smart and accessible homes for individuals with restricted motor abilities that can optimize and automate the use of control appliances [6, 7] for them to applications for baseline testing [8,9], monitoring performance deterioration in workplaces [10] and applications that involve information personalization and adaptive intelligence. The raw EEG data, like any other sensor data, is generally not of much

use if it is not engineered and processed properly before feeding it into an intelligent ML algorithm, to find useful results and inferences. Therefore, in current research we have focused on building a robust preprocessing pipeline that could transform time-series sensor data into desirable form so that it could be utilized by ML models in the later phase. The focus has been on three applications that have been designed using raw sensor data from EEG device for time-series classification. In first application we designed a multiclass classification model to detect the primary colors, red (R), green (G), and blue (B), from the features of raw EEG signals. Such application could be potentially used in scenarios where people with restricted motor ability could switch on/off appliances by looking at a particular color, control driving of a car by following certain color sequences e.g. apply brakes when looking at red color or to control robotic arms. The second application is based on a deep learning-based performance sabotage detection architecture, that can efficiently detect wilful misperformance in workplace safety application for companies that monitor performance impairment, like assembly lines and such, and the insurance industry that is obsessed with 'malingering'. The third application can be potentially used for monitoring of cognitive load conditions in e-learning or by clinicians to monitor behavioural performance by classifying EEG data into high cognitive and low cognitive. All this work has been achieved by using Interaxon's Muse 2 which is a wearable headband commercially used for meditation purpose. The applications have been integrated into a single web application that has integrated data preprocessing, data engineering and machine learning pipeline for the three proposed applications.

1.2 Research Objectives and Questions

The goal of this research is to answer the following research questions (RQ):

1. RQ 1: Can we design robust ML software architecture for IoT applications that can ensure flexibility, extensibility, and, scalability?

2. RQ 2: Can portable applications using data from a wearable device be used to test our proposed architecture ?

3. RQ 3: Can the ML architecture be divided into components that work independently across edge-core for training and inference ?

The RQs are described in details below:

RQ-1: ML does not only revolve around complex models but involves various steps that are important for the models to perform well on the data [11]. All these steps account for a ML workflow that can ingest raw data and transform it into some meaningful feature vector that can be fed to ML models for predictions. These pipelines are of utmost importance in case of raw IoT sensor data, that, if used in its original form would not yield good results. This data needs to be cleaned and preprocessed to extract features before feeding it to the models. We intend to introduce a pipeline that can work for most of the IoT applications and is flexible, extensible and scalable.

RQ-2: We aim to test the proposed architecture with novel applications that are based on BCI and that deal with huge amount of sensor data which is noisy and requires preprocessing before applying ML.

RQ-3: In the big data archetype, the computing is distributed across edge-core [12]. Edge is a smaller cloud, with limited resources and at close proximity to the IoT device. Core refers to private or public cloud (such as IBM or AWS cloud), with many resources. We investigate the possibility to divide the ML pipeline into independent

components that can work efficiently across the edge-core architecture. This can ensure robustness and portability of IoT applications.

1.3 Thesis Contributions

The goal of this thesis is to design a framework that can work with wearable sensor technology by integrating flexible machine learning architecture with IoT applications, that can operate on two tiers, the edge and the core. The research contributions (RC) are:

1. RC 1: In this work, we propose a novel architecture that can accept sensor data from IoT infrastructure, and, systematically address tasks like data ingestion, data cleaning, preprocessing, modeling and deployment in form of computation units that are easy to replace such that it is possible to rework them independently without changing the rest of the system to ensure better implementation. The focus of the architecture has primarily been on the problem of time-series classification of raw sensor data. To this end, we provide various data preprocessing and feature engineering functionalities like; performing imputation of missing values, normalizing the raw data, performing time-frequency analysis using fast fourier transform, generating statistical features and performing feature reduction. Various machine learning algorithms are provided by the architecture that take care of hyperparameter optimization in order to give the best performing model to the user. The user can save the entire pipeline from data preprocessing and engineering to the final machine learning model as a workflow to use it later for prediction.

2. RC 2: A major contribution of this thesis is three BCI applications. Our re-

search identifies use-case applications that are a good fit for edge-core deployed machine learning. We have designed applications that work on brain wave data and utilize the proposed architecture for various BCI tasks. We have worked with wearable headband technology to come up with cost effective, portable and easy-to-use cognitive solutions.

The first application investigates the relationship between the primary colors and brain stimuli. We propose a multiclass classification model to detect red, green and blue (RGB) colors from raw brain wave data. The classification results have shown an improvement of almost 20% from the past studies done in this domain using wearable devices. This application could potentially be used in an integrated IoT environment where it could be used to control and actuate appliances. One such application could be, where people with restricted motor ability could switch on/off appliances by looking at a particular color. The work can also be expanded in the healthcare field where it could be used to detect color blindness.

3. RC 3: The second application that we have developed, works on the idea of classifying cognitive load levels using brain wave data while the participants perform a cognitive motor integration (CMI) task. We have preprocessed and transformed the data using the architecture proposed in RC 1. We have engineered the raw data by using the time frequency analysis functionality and fed this representation to various self attention based multi-channel deep learning models. An improvement of approximately 9% was observed over past approaches that used portable headbands. The technique of transfer learning has been used to perform classification on unseen brain wave data using pretrained model. To our best knowledge, transfer learning on EEG data has not been performed earlier in any cognitive study and it helped us to reuse the existing model in case of

shortage of data and to reduce the cost incurred by retraining the entire model. Our methodology can help an individual identify the optimal level of mental workload and hence enhance one's learning performance. An early automated diagnosis of workload can allow monitoring performance demand levels in work or duty situations so that the employees can perform to their fullest potential. In an academic setting, a regular assessment of student's cognitive engagement can be used to optimize the pace of teaching and enhance the effectiveness of the learning process and can be potentially used for monitoring of cognitive load conditions in e-learning.

4. RC 4: The novel problem of sabotage detection from brain wave data has been studied in the third application. There have been no previous studies that have focused on this problem, thus we propose a model that can detect human performance sabotage from brain wave data. Spectrogram stacks have been generated using time frequency analysis functionality of the architecture . A deep learning model has been trained on this data to perform binary classification into sabotage and no-sabotage classes and promising results were obtained. We have achieved an overall accuracy of 98.71%, as well as low false-positive rates. Our method can be applied in clinical applications such as baseline testing, assessing current state of injury and recovery tracking as well as industrial applications like monitoring performance deterioration in workplaces.

5. RC 5: Our final contribution has been a prototype implementation of the architecture in RC 1 by making the ML pipeline granular in terms of deployment i.e. deploy the independent components across edge-core. We attempted to automate the steps of ML training and deployment. It was observed that proposed distribution of granular tasks over edge-core ensured reusability and provided a

scope of extention. The solution that we put forward follows the microservices architecture and fulfils the requirements of the ML pipeline by addressing the services with API endpoints that are provided by docker containers. Thus, we suggest a cloud-native end-to-end pipeline for ML workflows. We have designed the application in a way such that the user can define customized ML workflow and save them for inference, however for each workflow we do not create separate docker containers instead we use the workflow definition over a predefined container to get the results for online/batch prediction, this makes our proposed application economical to use.

1.4 Thesis Organization

This research work is structured as follows. Chapter 2 presents the background and research related to the field. Chapter 3 presents the framework of the architecture and methodology. Chapter 4, 5 and 6 demonstrate the BCI applications and results achieved. Finally, Chapter 7 concludes our work and presents the next steps to improve the architecture based on our research.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 EEG and Wearable headbands

Electroencephalography (EEG) [13] is a neuroimaging modality for monitoring and recording electrical activity of the brain. The primary measure of EEG activity used in psychophysiological research is in the form of signals which are measured in microvolts. The raw EEG contains the following main frequencies [14]; Delta with frequency 3Hz or below (Deep dreamless sleep), Theta with frequency 3.5 Hz to 7.5 Hz (Deep meditation), Alpha with frequency 7.5 Hz to 13 Hz(Calm relaxed yet alert state), Beta with frequency 14 Hz to 39 Hz (Active, busy thinking) and Gamma with frequency greater than 40 Hz (Higher mental activity). Each type of brainwave controls a variety of states of consciousness ranging from sleep to active thinking. Figure1 shows the different waves in EEG. EEG can be obtained in two ways; invasive, with the electrodes placed along the scalp and non-invasive, from electrodes implanted inside the cranium.

The portable EEG headbands offer a non-invasive way of recording the brain wave signals. Wearable EEG devices generally have lower number of electrodes or channels when compared to traditional medical-grade EEG and are more economical, to name a few, NeuroSky (1 channel), Muse (4 channels), Emotiv (+ 5 or 14 channels), OpenBCI (8 – 16 channels), are in popular culture.

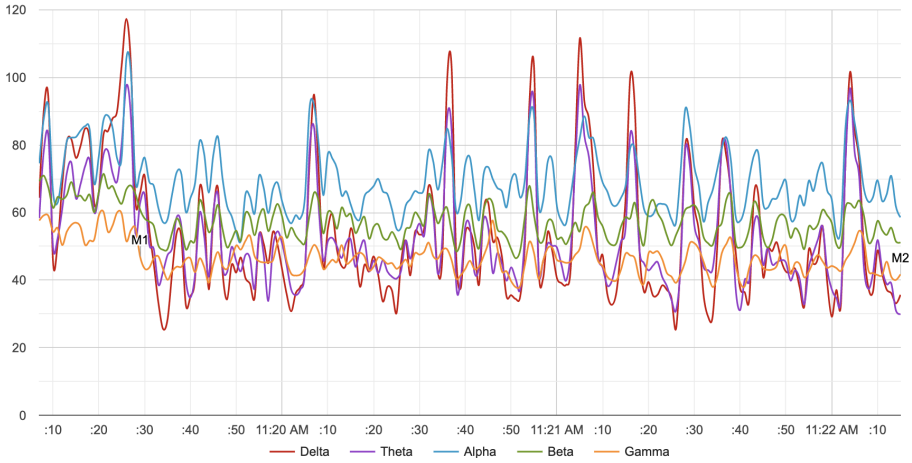


Figure 1: The brain waves present in raw EEG signals.

2.1.2 Digital Signal Processing

EEG signals are time-dependent, non-stationary, and most likely multicomponent signals because of cumulative electrical activity. The EEG from non-invasive source is very prone to noise and artifacts and therefore are not much useful when used in their raw form. This data should be subjected to suitable signal processing techniques to obtaining meaningful signals from them.

2.1.2.1 Time-Frequency Analysis

In signal processing, time–frequency analysis (TFA) consists of techniques that study a time domain signal in both the time and frequency domains simultaneously. By the use

of TFA on EEG one can detect the presence of Delta, Theta, Alpha, Beta and Gamma band signals over all the time points across the signal. TFA basically represents the power intensity of a particular frequency at a particular point of time in a signal. It can be performed using various techniques, we have employed Morlet Wavelet Transform to get the time-frequency representation of the EEG data. For the analysis of data with respect to time, there are many time domain analyses: autocorrelation analysis, cross-correlation analysis, stationarity analysis, seasonal adjustment/decomposition analysis, singular spectrum analysis (principal component analysis for time series), count series analysis, and many others. For the analysis of data with respect to frequency, there are many frequency domain analyses: Fourier analysis, wavelet analysis, Laplace transform analysis, and many others. Most frequency domain analyses assume that the time series is stationary. Time-frequency analysis (TFA) simultaneously analyzes both time and frequency. TFA is particular useful for time series data that are nonstationary with respect to time, frequency, or both: time series that have time-varying trends, time-varying periodic cycles, irregular cycles, and other time-dependent properties.

2.1.2.2 Morlet Wavelet Transform

Complex Morlet wavelets are very widely used in EEG data analysis for time-frequency decomposition. They have a sinusoidal shape which is weighted by a Gaussian kernel, and they can therefore capture local oscillatory components in the time series. Unlike the standard short-time Fourier transform, wavelets have variable resolution in time and frequency. The frequency resolution is high but the time resolution is low for low frequencies. It's the opposite for low frequencies. Thus wavelets provide both frequency and temporal precision.

2.1.2.3 Spectrograms

A spectrogram is a visual way of representing the signal intensity over time at various frequencies present in a particular waveform. Spectrograms are matrix like representations of the data in time-frequency domain where the third dimension represented by colors.

2.1.3 Machine Learning

Machine Learning (ML) focuses on teaching computers how to learn without the need to be programmed for specific tasks by training them on some examples or data. In ML, algorithms can be used for a variety of tasks like classification, clustering, prediction etc. In our particular case we have used ML for time-series classification. Also, four common classes are used to group ML algorithms: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

- Supervised learning uses a set of labelled examples to coach an algorithm. Labelled data implies that each input in the training dataset has a defined output, which is referred to as the class label value. At the end of training, for each input point, the trained algorithm predict an output similar to the label value. Supervised learning can be applied to solve two types of problems: classification problems and regression problems. In classification problems, algorithms are used to provide a class label classifying the input data as part of a category or group. Regression problems involve predicting a continuous value from each input data point.

- Unsupervised Learning is a ML technique in which the users do not need to provide labelled data to the model. Instead, it allows the model to work on its own

to recognise patterns and hidden information that was previously undetected. It mainly deals with the data that has no class labels. Unsupervised learning algorithms enable the users to perform more complicated processing tasks compared to supervised learning.

- Semi-supervised learning trains ML models using datasets that contains both labelled and unlabelled data. In some cases, labelling all the input data points is a time consuming and tiresome task because it involves huge volumes of data, which leads to the use of semi-supervised learning. In semi-supervised learning, data labels are still needed to validate accuracy of algorithms.

- Reinforcement learning is a method of training the ML models to make a sequence of decisions. The agent learns to achieve a goal in an potentially complex environment. The agent employs trial and error method to come up with a answer to the problem. If the algorithm takes action that moves closer to the goal, it receives a reward; otherwise, it is receives a punishment. Its goal is to maximize the total reward.

In our study we have used the supervised learning approch to perform time series classification task using the following models:

K Nearest Neighbor (KNN) KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution and that's why we tested it in our problem. K is a critical hyperparameter. The Euclidean distance is often used as the distance metric.

Logistic Regression (LR) It is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Online

transactions Fraud or not Fraud, Email spam or not spam. LR transforms its output using the logistic sigmoid function as activation function to return a probability value.

Random Forest (RF) The random forest algorithm is an ensemble approach that uses multiple decision trees and makes a classification decision by voting from all the trees. The number of estimators is a hyperparameter for RF that is equivalent to number of trees.

Artificial Neural Network (NN) A neural network consists of neurons, arranged in single or many layers, which apply some activation function to an input vector to convert them into some output. Each unit takes an input, applies an activation function to it, and then passes the output produced on to the next layer. These networks are referred to as feed-forward beacuse a unit feeds its output to all the units on the next layer without any feedback to the previous layer. Weights are applied to the signals passing from one unit to another, and these weights are learned in the training phase to adapt a neural network to the particular problem.

Support Vector Machine (SVM) SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve both linear and non-linear problems and work well for many practical problems. The idea of behind SVM is the algorithm creates a line or a hyperplane which separates the data into classes. SVMs can be categorized as hard-margin and soft-margin SVMs.

Gradient Boosting (GB) Gradient boosting is an ensemble learning approach that produces a prediction model in the form of an ensemble of weak prediction models. Gradient boosting combines weak learners into a single strong learner.

2.1.4 Deep Learning

Deep Learning (DL) is a subfield of ML, that is inspired by the structure of the brain and it uses complex multi-layered neural networks, where the level of abstraction increases gradually by non-linear transformations of input data. We have employed the following models in our study.

CNN A CNN [15] is a kind of artificial neural network that uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. A classical CNN may contain mainly a convolution layer, pooling layer, fully connected layer and normalization layer. Recently, CNNs have shown state-of-the-art results on challenging activity recognition tasks with very less or no data feature engineering, by instead using feature learning on raw data and that is the reason we use these in our case. The full CNN framework and formula derivation can be seen in the literature [16]. In a CNN, the convolutional product between the image and the filter is done and a 2D matrix is obtained where each element is the sum of the elementwise multiplication of the filter and of the given input in matrix form. Often a bias is also added to the the result. For simplicity we only provide the formula for convolutional layer, which works as a filter and is then activated by a non-linear activation function, as follows:

$$\mathrm{g}_{x,y} = \mathrm{f}(\Sigma_{i=1}^{n_H}\Sigma_{j=1}^{n_W}\Sigma_{k=1}^{n_C}\mathrm{K}_{i,j,k}\mathrm{I}_{x+i-1,y+j-1,k} + \mathrm{b}) \tag{2.1}$$

where $g_{x,y}$ is the corresponding activation,$K_{i,j,k}$ denotes the i×j×k weight matrix of convolution kernel, $I_{x+i-1,y+j-1,k}$ indicates the activation of the upper neurons connected to the neuron (x, y), (n_H, n_W, n_C) is the dimension of the input (image)where n_H is the height(64 in our case), n_W is the width(64 in our case) and n_C is the number of channels

(4 in our case b is the bias value, and f is a non-linear function.

LSTM LSTM [17] network models are a type of recurrent neural network that are able to learn and remember over long sequences of input. They are intended for use with data that is comprised of long sequences of data. They are a good fit for time-series problem. The model learns to extract features from sequences of observations (RAW data in our case) and how to map the internal features to different activity types. The benefit of using LSTMs for time-series sequence classification is that they are directly able to learn from the raw time series data, and in turn do not require domain experts to manually engineer input features. The model can learn an internal representation of the time series and can achieve comparable performance to models fit on a dataset with engineered features.

A vanilla LSTM [18] block has three gates (input, forget and output), block input, a single cell, an output activation function, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates. The vector formulas for LSTM layer forward pass are given in [18].

Self-Attention The self-attention [19] mechanism allows the inputs to interact with each other and find out who they should pay more attention to. The attention layer aims to learn the important time points from the sensor time series data that aid in determining the state label. After leveraging both local contextual features and temporal dynamics by fusing CNN layer and LSTM units from the input time series, we used self-attention layer to learn weight coefficients which were the importance of each feature in input data samples. The attention score, s, for a sample is then given by:

$$\mathrm{a}_s = \mathrm{softmax}(V_{att} tanh(U_{att} h_t \text{`}) \tag{2.2}$$

$$s = a_s h_t` \tag{2.3}$$

In the above equations, $U_{att} \in R^{DxE}$ and $V_{att} \in R^{FxD}$ are weight matrices forming the attention module, F represents the attention length, D represents the length of the output and E represents the number of hidden units in the previous layer (LSTM in our case) and $h_t`$ is the encoded input from LSTM. Equation 12 finds the softmax of the compatibility (similarity) of the input and Equation 13 gives the combination of the transformed input.

ConvLSTM ConvLSTM [20] is another type of recurrent neural network that has recently become popular for time-series classification problems. In ConvLSTM the matrix multiplication is replaced by convolution operation in the state-to-state and input-to-state transitions and thus what we get as output from this a ConvLSTM layer is the transformed data in its original dimension or input dimension which is 4 in our case. The advantage of ConvLSTM over traditional LSTM is that it captures spatio-temporal patterns in the data.

2.1.5 Muse Headband

Muse headband is a wearable meditation device. It consists of four channels/electrodes namely AF7 and TP9 on the left and AF8 and TP10 on the right as we can see in Figure 2. These are named and positioned according to the International 10-20 System. The sampling rate of Muse is 256Hz.

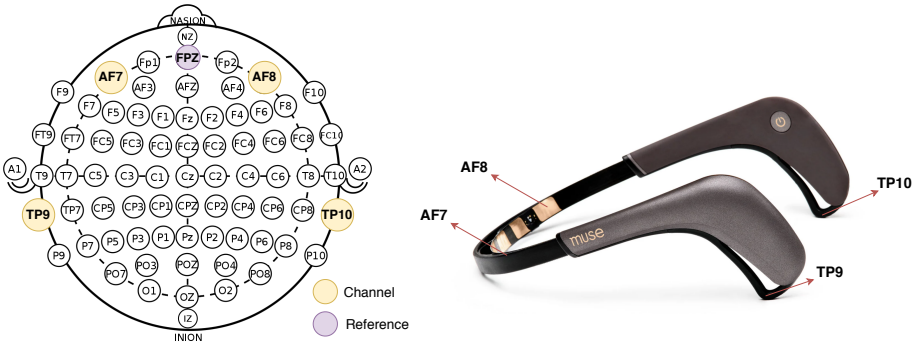


Figure 2: The 10-20 system of electrode placement for Muse.

2.1.6 BrDI Application with Cognitive Motor Integration Task

It is a computer-based visuomotor skill assessment task (BrDI™) that includes one standard and one non-standard conditions where vision and action are decoupled as shown in Figure 3. This latter condition requires the integration of spatial and cognitive rules, and thus required cognitive-motor integration (CMI). The task requires one move the index finger of their dominant hand along the touch screen of the tablet to move a cursor (white dot, 5 mm diameter) from a central location to one of four peripheral targets (up, down, left, or right relative to center) as quickly and as accurately as possible. We used the EEG data collected from Muse while participants performed this task for cognition and sabotage applications.

2.2 Related Work

In recent years, researchers have used wearable headbands to analyze the response of EEG under different stimuli. K. Johannesen et. al. [21] used SVM to derive useful EEG features in order to predict working memory performance in schizophrenia and healthy adults. The authors in [22] used a regression model trained on data gathered from cognitive tasks (collected from a 6-channel EEG headset) in order to model men-

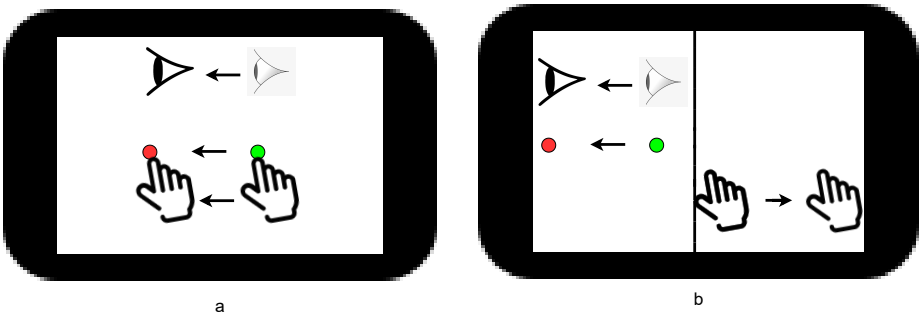


Figure 3: (a): Task 1 (Simple sliding finger along the same direction); (b): Task 2, A CMI task.

tal workload using EEG features for intelligent systems. In [23, 24] the EEG data has been used to examine driver's alertness during driving sessions.

2.2.1 Related Work in Classification of Color Stimuli

Diane Aclo et. al. [25] used a 14 channel EEG device to monitor the effect of color stimuli on people. They used features like power spectral density and waveform length for classification using an Artificial Neural network. In [26] feature selection algorithm has been investigated for EEG signal due to RGB colors using screwable gold EEG electrodes. Arnab Rakshit et. al. [27] proposed the use of a fuzzy space classifier to discriminate colors from EEG by using a 10 electrode device. In [28], an Emotiv headset has been used to study separation and classification of EEG response to color stimuli by using SVM. Zhang et. el. in [29] showed how alpha and beta band powers are affected by stimuli from RGB colors. All the above classification tasks have been conducted using complicated EEG devices in contrast to our work. Furthermore, our proposed method achieves a high accuracy using the Muse headband. Recently, the

use of portable headband devices have gained popularity due to their ease of use and accessibility. The authors in [30, 31] have used four channel G.tec's MOBIlab four channel portable device in a problem similar to ours. However, their method yielded a lower accuracy of 58% in comparison to our proposed approach. A headband from Mindwave Neurosky has also been applied [32] in a task similar to our. However, the authors achieved a lower accuracy of 53% with their method. Our results have shown significant improvement. In many studies, Muse has also been used to acquire EEG signals for various classification tasks. EEG-based excitement detection in immersive environments has been studied by Jason et. al in [33]. Krigolson et. al. [34]used Muse headband to Assess Human visual attention by assigning subjects an "oddball"task wherein they saw a series of infrequently and frequently appearing circles and were instructed to count the number of target circles that they saw. However they did not apply any ML model in their work. In [35] classification task has been performed to classify recreational and instructional video sessions using Muse. They used spectral power and connectivity features from raw EEG in their work and got the best performance with SVM and Logistic Regression model.

2.2.2 Related Work in Classifying Cognitive Load

Recently, there has been a lot of interest in detecting cognitive effort using EEG data and many researchers have studied this problem using data from a variety of tasks. In our work, we have specifically used a CMI task for this purpose. Ronglong Xiong et al. in [36] used conventional ML models like Decision Tree and Support Vector Machine (SVM) for the classification of cognitive load using data from Neurocom EEG 128-channel system. They achieved an accuracy of 97.2% using SVM, comparable

to ours using a 4-channel headband. Cognitive load based on five pre-defined mental tasks: Relax, Letter Composition, Multiplication, Counting, and Rotation has been studied in [37] and the EEG data has been classified using Extreme Machine Learning (ELM) approach. The highest accuracy of 86.70% was obtained. Deep recurrent neural network has been used by Kuanar et al. [38] for cognitive analysis of working memory load and a classification accuracy of 92.5% has been attained. A variation of CNN with large-margin softmax loss function has been used in [39] to classify cognitive load signals from 64 electrodes EEG device and an error rate of 6.43 has been obtained.

In [40] a deep learning methodology using a stacked denoising autoencoder-LSTM-MLP (Multi-layer perceptron) has been proposed for classification of cognitive load estimation and an accuracy of 89.51% has been achieved. Authors in [41] used a fuzzy c-means unsupervised clustering approach for classifying cognitive load. Some studies have also used Muse headband to study cognitive load. Human stress has been studied using Muse by Aamir Arsalan et al. in [42] and an accuracy of 92.85% has been achieved for binary classification using SVM, Naive Bayes, and MLP. In [43] mental states in 2D and 3D virtual environments have been studies using Muse EEG and an accuracy of 66.88% and 59.27% in 3D-VR and 2D-screen group respectively has been achieved using SVM. Richer et al. [44] performed classification using quantiles of the estimated histogram as threshold for mental state recognition using Muse and achieved highest sensitivity of 82.0%. A similar study was done by Bird et al. in [45] using Muse and they achieved an overall accuracy of 87% using Random Forest classifier. In [46] a CNN has been proposed to classify mental workload using 28 Channel EEG and an accuracy of 72.7% has been achieved. DL for health informatics has become quite popular and has been proposed in various previous studies [47, 48]. In our approach we have used various DL models and used data cubes as input to the models.

We have also applied the concept of Self-attention to improve the performance of these models further. The test accuracy that we have achieved (96.1%) is higher than all of these previous studies. The advantage of using attention based multi-channel models for this classification problem is that we were able to take into account the data from all the channels of EEG and yet only focus on the important details using the self-attention technique and this is what has improved the performance of our model over the past approaches.

2.2.3 Related Work in Detecting Performance Sabotage

Mild traumatic brain injuries (mTBI) or concussions have become an increasing public health concern, affecting an estimated 42 million individuals annually [49]. As a brain injury, concussion affects many aspects of function, including sensory, motor, and cognitive domains, and can thus have major implications for participation in activities of daily living. Approximately 15-20% of those who sustain concussions, develop chronic symptoms and functional impairments that persist for months or years [50, 51]. As a result, individuals affected by concussion are seeking clinical and rehabilitation care in greater numbers than ever, highlighting the need for such care to reflect evidence based on objective metrics.

One issue with the current state of injury impact assessment and recovery tracking is its reliance on self-report. By their nature, symptoms are subjective: there is no way to measure a headache other than to record what an individual reports. However, there is evidence that subjective symptoms may resolve before full neurological recovery has occurred, leading to vulnerability to further injury [52,53]. Deficits have been reported in a wide range of laboratory and neurophysiological outcome measures in asymp-

tomatic individuals after concussion [54–59]. This reliance of self-reported symptoms, therefore, means that diagnosis is less precise, intervention targets are more difficult to identify, and determining recovery is less clear than would be possible with more objective measures.

A second issue around assessment and recovery tracking at present is the use of pre-injury assessments. Objective measures applied to the same person pre-post injury would be a useful metric to assess injury effects and monitor recovery, since they would allow for more personalized care. However, because of their unreliability, the utility of 'baseline measures' often collected annually for those in athletics or lines of duty have been called into question [60,61]. A primary contributor to this concern is the potential for individuals to sabotage their baseline tests (i.e., deliberately perform poorly to erode or incapacitate the assessment). For a variety of personal or social factors, such as a desire to return to work, duty, sport, or the pressures of impending litigation, individuals may not perform to the best of their ability. Thus, any injury/recovery assessment approach using baseline measures is only effective if both the pre- and post-injury assessments capture an individual's best effort. To this end, a way to detect and prevent sabotage during baseline testing would add considerable reliability to concussion assessment and recovery metrics, improving care and preventing further injury.

Electroencephalography (EEG) has been used in several previous studies examining objective sabotage detection. [62], used EEG to detect when people were lying by analyzing eye blinks, and were able to detect lies with 95% accuracy. In literature, there are many studies that have worked on the problem of Lie Detection using EEG data obtained by Guilty Knowledge Test (GKT), a psychophysiological questioning technique that can be used to determine whether a person is lying especially in a polygraph test. [63] acquired data from participants by making them answer GKT and extracted

various features from the EEG like morphological features and frequency based features, and then selectively fed features to Linear Discriminant Analysis algorithm for classification. They achieved 86% correct detection of total subjects. Deep Belief Network was used for deceit classification by [64] based on GKT and the highest subject accuracy achieved was 83.4% using 16 channel EEG. A new machine learning method referred to as F-score based Extreme Learning Machine (ELM) was proposed by [65] to classify the lying and truth-telling using the EEG signals (from GKT based task) from a 9 channel medical-grade electrode and they achieved best accuracy of 98.97% was achieved.

A support vector machine (SVM) was used to detect lie by [66] using a 9 channel EEG headband to collect data from a stimulus display task, after preprocessing and feature extraction, SVM gave the best accuracy of 70.83% in this study. [67] classified the two states i.e. lie and deception based on the EEG patterns while the participants were responding to "Pokemon cards" whether the card belongs to them or not. They used Short-time Fourier transform (STFT) and Multi-layer Perceptron (MLP) for classification and achieved the accuracy of around 90%.

However, no studies have examined changes in the time-frequency sepctrograms of EEG signals in response to sabotaging a cognitive-motor integration task. In our approach we propose to use spectrograms of the EEG data as the input to our model. Previous studies examining sabotage detection have all used traditional EEG systems, collecting data from a minimum of 9 scalp electrodes. An important focus when considering the application of our work to clinical concussion care [and workplace/clinic assessment] is the use of technologies that are deployable in clinical environments. To this end we used a portable EEG headband (Muse2™, InteraXon Inc., Toronto, Canada), a commercially available and consumer-grade device, to collect our EEG

data. To our knowledge, this is the first time a portable EEG system has been used to detect sabotage. Here, we use a Deep Learning (DL) [68] approach to analyze EEG spectral data. DL is a subfield of Machine Learning (ML) [69], which is in turn a form of artificial intelligence [70] that focuses on teaching computers how to learn without the need to be programmed for specific tasks by training them on some examples or data. Deep learning is a class of machine learning algorithms inspired by the structure of the brain which uses complex multi-layered neural networks, where the level of abstraction increases gradually by non-linear transformations of input data. We have employed a multi-channel attention based CNN-LSTM [16, 17] model to identify sabotage and the model has been trained on EEG data from a CMI task. The proposed model i.e. CNN-LSTM has also been used in the past [71, 72] for classifying time series data. However in our model is different as it uses Multi-Channel CNN i.e. a CNN that accepts inputs from all the EEG channels at the same time and Self-Attention [19] mechanism.

The objective of the present study was to determine if performance sabotage could be detected using a deep learning analysis of neurophysiological data collected during a visuomotor skill assessment. We hypothesized that these EEG spectral measures of neural activity during intentionally poor performance on a CMI task would be significantly different from a maximal effort performance.

2.2.4 Related Work in ML-IoT architectures

In [73] Zhao et al. proposed an architecture using ML libraries to support the sharing of ML models. This platform enabled one to make Docker images of pre-trained models. Their work also described training and deployment of models using container-based virtualization.

A platform called Rafiki was introduced by Wang et al. in [74] that provided functionality for both training and inference of ML models. However this platform did not let the user construct ML models, tune hyper-parameters or apply preprocessing on data. They used docker containers to implement the workflow. Another platform called Clipper was introduced by Clipper et al. in [75]. In this platform the models were containerized to facilitate isolation.

In [76] a similar approach has been proposed however they mainly focus on limited ML models and also deploy separate container for each model and each functionality which is not needed in most cases and can incur huge costs.

Previous work [73–76] has been main focus has been to provide generic ML platforms, which are more focused on achieving better efficiency during training and deployment of ML models. In this work, we focus more on challenges in handling IoT data and provide an approach that focuses on the complete pipeline of data ingestion, preprocessing, data engineering, model training and inference.

2.3 Summary

In this chapter, we introduced the basic concepts and background for our proposed platform. We then discussed relevant literature related use of ML using EEG data to study color classification, cognitive load and sabotage detection. We also discussed about related work in ML-IoT architecture.

Chapter 3

Architecture and Methodology

Figure 4: End-to-End ML pipeline. The pipeline arrangement proposed by our architecture divides the ML tasks under the following components, namely, Data Ingestion, Data Cleaning, Data Preprocessing, Modelling and Deployment.

3.1 Introduction

The end-to-end ML [77] pipeline that we have used is shown in Figure 4. This is the traditional pipeline that is used for ML workflows. Our proposed architecture comprises of the edge where the raw data is ingested and the core where the data is analysed, preprocessed and engineered to feed it to the model for training. Often a third tier of cloud is also added to the above architecture for data archival. The steps involved in the workflow are often written as a single source code at the time of testing and this often creates dependencies and restricts the use of cross-platform models, however if

each step is treated as independent unit then it becomes easier to scale the pipeline, add new functionalities without affecting the existing ones and also makes it possible to leverage the models offered by different platforms.

ML workflows in our study are considered for two scenarios; first is the training phase and the second one is the inference phase. In the former, we use the data stored over time, generally in form of big data from a data source and we then apply the tasks to prepare the data for training. A variety of models are tried over the dataset and often with different hyperparameters. The best performing model offered from the training is generally used for the realtime inference phase. The inference phase uses the same preprocessing steps as defined in training phase and sends the data to finalised ML model for prediction. In IoT applications, these predictions can be used to monitor applications, actuate control devices and plenty of other tasks. In this work, we have addressed the issue of having a systematic workflow for applications that revolve around timeseries data that is often noisy, has missing values and is the most demanding in terms of data cleaning. In order to achieve an optimal model one needs to apply proper data engineering methods on the data to get the best possible results. The data pipeline followed during the training phase needs to be the same as that for the inference phase. We propose an application that can be used across edge and core to accomplish ML pipeline for different workflows. In our proposed architecture, each process takes place in a different module and independently, only the data is shared across various modules. The different processes in the pipeline follow the microservices architecture and are accessible as restful APIs, which can be deployed as docker containers. This makes it possible to simply call the service in order to perform a task and helps in reusability of code. Our methodology allows users to create a ML workflow without writing any piece of code and use it on the go for inference on test data.

Each workflow created is independent of each other and uses the process modules for tasks like preprocessing and model training in a self-reliant manner. We now explain the two scenarios and their edge-core architecture in detail.

3.2 Architecture for training phase

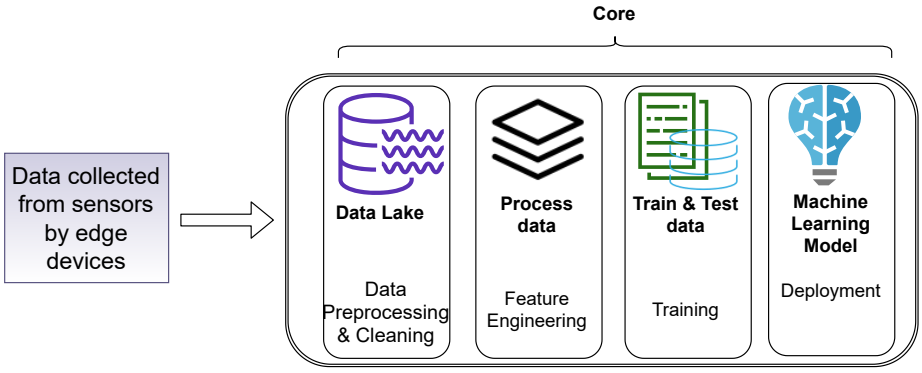


Figure 5: Proposed architecture for Model training. The data is sent from sensors to edge. The data from edge is sent to core where the tasks like data preprocessing, feature engineering, training and model deployment take place.

The training is generally done offline with the data that is collected over time. This phase is very important as it is on the basis of the training that the final model for a problem is decided. The training phase generally has many preprocessing steps after which the data is fed to various models for the final training. During the model training there are various hyperparameters that are to be taken care of. Our approach takes care of every process in the training phase, from data preprocessing to hyperparameter optimization and finally model training. We put forward an architecture that spreads across the edge and the core and addresses the different tasks as microservices that work in an autonomous manner. The advantage of this is that pipeline becomes highly maintainable, testable and independently deployable. The processes like data cleaning and preprocessing can be done in a customized manner using our architecture and

the user can apply models as per their requirements. Our application using the user-specified pipeline gives a final result. Based on the result the user can save and store the pipeline for inference or testing purposes. We now discuss in detail our proposed architecture.

3.2.1 Edge

The edge mainly comprises of data ingestion. The variety of IoT sensors and devices that generate data from a few gigabytes to a petabytes form the edge in a typical edge-core pipeline. In our case the edge can be a device with limited resources (laptop or a mobile device) and the task of the edge is to host the interface for serving data to our architecture. The edge then sends the request to the processes running on core for further computation. Computing on edge is another possibility however, it has certain disadvantages. More storage capacity is required to perform computing on the edge, however it might not be a good idea to have an extra storage for processes that are not always running, rather using a core based solution is a better idea that enables storage on the go. There are often security challenges in edge computing as, if we store or process data on end-user devices, that we don't control, it's difficult to guarantee those devices are free from vulnerabilities that could be exploited by attackers. It is also very cost ineffective, and requires advanced infrastructure. However, it's important to mention that edge regardless of disadvantages, is needed for its proximity especially if we're dealing with mission or data critical applications. The edge in our proposed solutions serves as a host and a proxy between the microservices and the data source and provides an easy medium for the user to interact with the various components of the workflow. It serves the user interface which makes it easy for the user to connect with APIs which serve the request for the users.

3.2.2 Core

The core is the most important part of our architecture. It can possibly be in the cloud with an S3 tier for storage, or on premises with a file store or object store. In our case, the core is served on the cloud which makes it possible to host the various services in form of a microservices architecture. The core is supposed to have large computation power and resources as compared to the edge. It serves the requests which it receives from the edge devices and performs the tasks accordingly. In this work, the core is where the docker containers providing various services run and they serve edge with response of various tasks that have been requested by the user. For the user the core is like a black box that runs all the processes in the backend. It has primarily the following processes running; data cleaning and preprocessing, feature engineering, model training and deployment.

3.3 Architecture for the Real-time Analysis/Inference Phase

In order to get real time result from the proposed pipeline, we use the edge-core architecture such that, the data can be sent to the core from edge devices and, the computation and model prediction can be done on core. This data can be collected over time and can be used to retrain the model in production. The data can either be streamed to the edge device which acts as a proxy or it can be provided in bulk by the user for inference. For the inference, our architecture provides the user with capability to save the final pipeline that was generated in the training phase along with it's metadata and this can later be retrieved by the user to do real-time analysis or batch inference. The inference phase is also similar to the training phase the only difference is that it does

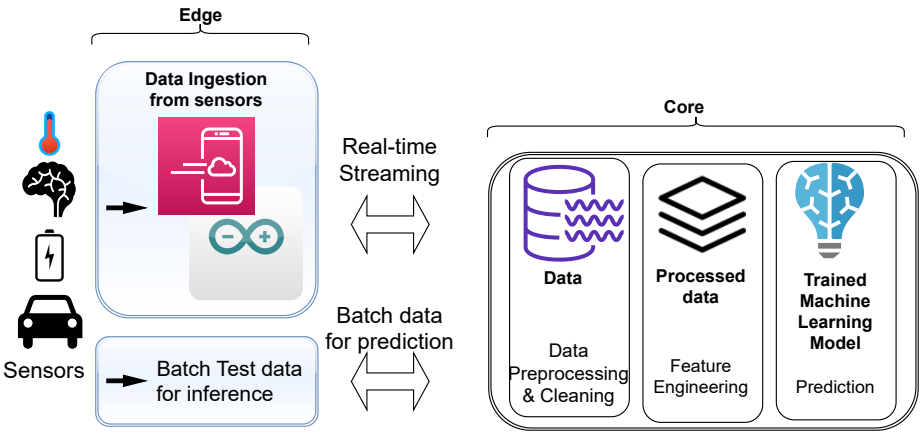


Figure 6: Proposed architecture for Real-time Analysis/Inference phase. The inference phase consists of pretrained models which are saved in form of workflows. The data is streamed or sent in batch to the core from the edge to core.

not require training of the model but it uses the model that was saved during the training phase for predictions.

3.4 Methodology

We shall now explain the methodology we have followed in each stage of the ML workflow.

3.4.1 Data Collection on Edge

The sensors or the IoT devices send/stream the data to the edge layer. This data either collected over some interval (for training) or in real time (for prediction) is sent to the core where the rest of the processing takes place. In our usecase, we used Open Sound Protocol (OSC) to stream the data to the core. This data was sent in its raw form to the subsequent pipeline for further analysis.

3.4.2 Data Cleaning and Preprocessing

It mainly involves forming a unified data lake by normalizing the data and cleaning it by various imputation methodologies. Many real-world datasets may contain missing values for various reasons. They are often encoded as NaNs, blanks or any other placeholders. Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. In our solution, we provide three main ways to deal with the missing value.

- Drop the missing values :The easiest way to deal with missing value problem is to drop the missing values but this leads into a lot of data loss.
- Impute with Mean/Median: The mean/median of non-missing values is calculated in each column and this value is used to impute the missing value. It is easy and fast method and works well with datasets that are small however it is not very accurate.
- Replace with zero : It replaces the missing values with zero. This helps to avoid the loss of the values that are removed by dropping the missing values.

Another important step for preparing the data is normalization. Since the time-series IoT data is from different sensors it might vary in its range for each sensor, in this scenario it becomes important to normalize the missing values. The main goal of normalization is to vary the values of numeric columns within the dataset to a typical scale, without distorting differences within the ranges of values. For ML model, every dataset doesn't require to undergo normalization. It's required only when features have different ranges. There are mainly two types of normalization techniques and we offer them in our architecture, namely Standard scalar and Min-max normalization.

- Standard Scalar normalization (Z-score normalization): It standardizes features

by removing the mean and scaling to unit variance. The standard scaled value for x is given by:

$$z = \frac{(x - \bar{x})}{s} \tag{3.1}$$

where $\bar{x}$ is the mean of the samples and s is the standard deviation of the samples.

- Min-max normalization: The data is scaled to a fixed range, usually 0 to 1, using this approach. In contrast to standardization, the price of getting this bounded range is that we'll have smaller standard deviations, which might suppress the effect of outliers. Thus min-max Scalar is sensitive to outliers and is given by the formula:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.2}$$

3.4.3 Feature Engineering

ML fits mathematical models to data so as to derive insights or make predictions. These models take features as input. A feature could be a numeric representation of a facet of data. Features sit between data and models within the machine learning pipeline. Feature engineering is that the act of extracting features from information, and remodeling them into formats that's suitable for the ML model. It's a vital step within the pipeline [78], because the proper features can ease the issue of modelling, and thus enable the pipeline to output results of upper quality. The process of feature engineering is generally very important when the models being used are the traditional ML models, however in case of DL model one can skip the overhead of feature engineering but the data needed to train a DL model is huge so a potential bottleneck comes in there. Our solution offers a variety of feature engineering techniques to the users. The users just have to specify a few parameters and can easily apply the feature engineering

techniques. Feature engineering encompasses techniques like feature transformation/-generation and feature selection and reduction. We offer a variety of these techniques in our architecture which are mainly aimed for timeseries classification and we shall discuss some of them now.

3.4.3.1 Time Frequency Analysis using Fast Fourier Transform

Many organizations have to analyze large numbers of time series that have frequency-varying or time-varying properties (or both). The time-varying properties can include trends that vary in time, and the frequency-varying properties can include periodic cycles that are time-varying. Time-frequency analysis can simultaneously analyze both time and frequency; it is particularly useful for monitoring time series that contain several signals which differ in frequency [79]. These signals are commonplace in data that are associated with the Internet of Things (IoT). This is a very crucial part of the pipeline and we applied various techniques to process the data. For our proposed applications we used Continuous wavelet transform (CWT) to convert the raw data into time-frequency domain. We used a Morlet wavelet to perform the CWT. A Morlet wavelet was obtained by the convolution of a Gaussian with a sine wave and is described by the following equation.

$$\Psi_\sigma(t) = c_\sigma \pi^{-\frac{1}{4}} e^{-\frac{1}{2}t^2} (e^{i\sigma t} - \kappa_\sigma) \tag{3.3}$$

where $\kappa_\sigma = e^{-\frac{1}{2}\sigma^2}$, σ is duration of the wavelet, and the normalisation constant c_σ is:

$$c_\sigma = \left(1 + e^{-\sigma^2} - 2e^{-\frac{3}{4}\sigma^2}\right)^{-\frac{1}{2}} \tag{3.4}$$

The Morlet wavelet helped to reduce edge artifacts and noise from the data. It also helped to obtain a balance in temporal precision and frequency precision. The technique of Fast Fourier Transform (FFT) has been used to convert the data into time-frequency domain. We first performed FFT on the raw data to convert it into frequency domain as shown in Equation 3.5.

$$\hat{x}(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-i\omega t} dt \tag{3.5}$$

where $x(t)$ is the time series signal and then we performed FFT on the Morlet wavelet (the wavelet was formed from the frequency of interest) following Equation 3.6.

$$\hat{\Psi}_{\sigma}(\omega) = c_{\sigma} \pi^{-\frac{1}{4}} \left(e^{-\frac{1}{2}(\sigma-\omega)^2} - \kappa_{\sigma} e^{-\frac{1}{2}\omega^2} \right) \tag{3.6}$$

After we got the two signals in frequency domain we convolved them by point-wise multiplication (the sampling rate of the signal and the Morlet wavelet must be the same in order to perform convolution). The data was then converted back to time domain using inverse fourier transform using Equation 3.7.

$$\texttt{Cw=}\frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{x}(\omega) \hat{\Psi}_{\sigma}(\omega) d\omega \tag{3.7}$$

The power of this signal was then calculated by finding the magnitude of the complex signal and then squaring it to get the absolute power component of the signal. In Algorithm 1 we summarize the procedure we followed to find band power from the raw data. Figure 7 shows the methodology followed for time-frequency transform.

Algorithm 1: How to extract band powers from Raw data from Morlet wavelet convolution

Input: Raw data
Output: Band power in form of spectrogram
1. Initialize the FFT parameters i.e. the minimum and maximum frequency, the time period of the wavelet which is equal to the sampling rate of the signal, the result matrix.
2. Find the FFT of the Raw data.
3. while *Frequency* $\leq$ *MaxFrequency* **do**
  4. Create a complex morlet wavelet from frequency by convolution of sine wave and gaussian.
  5. Find the FFT of the wavelet.
  6. Find the convolution of FFT of signal and FFT of wavelet by pointwise multiplication.
  7. Find inverse fourier transform of convoluted signal to convert back to time domain.
  8. Extract the magnitude of the complex signal and square it to get the absolute power component of the signal and add it to the result matrix.
end

3.4.3.2 Statistical fetaures using sliding window

Finding features like mean, variance, skewness, kurtosis values for a given raw data can help generate new features for the dataset and this can help in reducing the noise that may be caused by individual data points due to noise [80]. For a given timewindow t where n is the number of timepoints, the mean, variance, standard deviation, skewness, kurtosis can be calculated as:

$$mean = \bar{x} = \frac{\Sigma_{i=T}^{T+t} x_i}{n} \tag{3.8}$$

$$variance = \sigma^2 = \frac{\Sigma_{i=T}^{T+t} x_i - \frac{\Sigma_{i=T}^{T+t} x_i}{n}}{n-1} \tag{3.9}$$

$$\text{standard dev} = s = \sqrt{\frac{\Sigma_{i=T}^{T+t} x_i - \frac{\Sigma_{i=T}^{T+t} x_i}{n}}{n-1}} \tag{3.10}$$

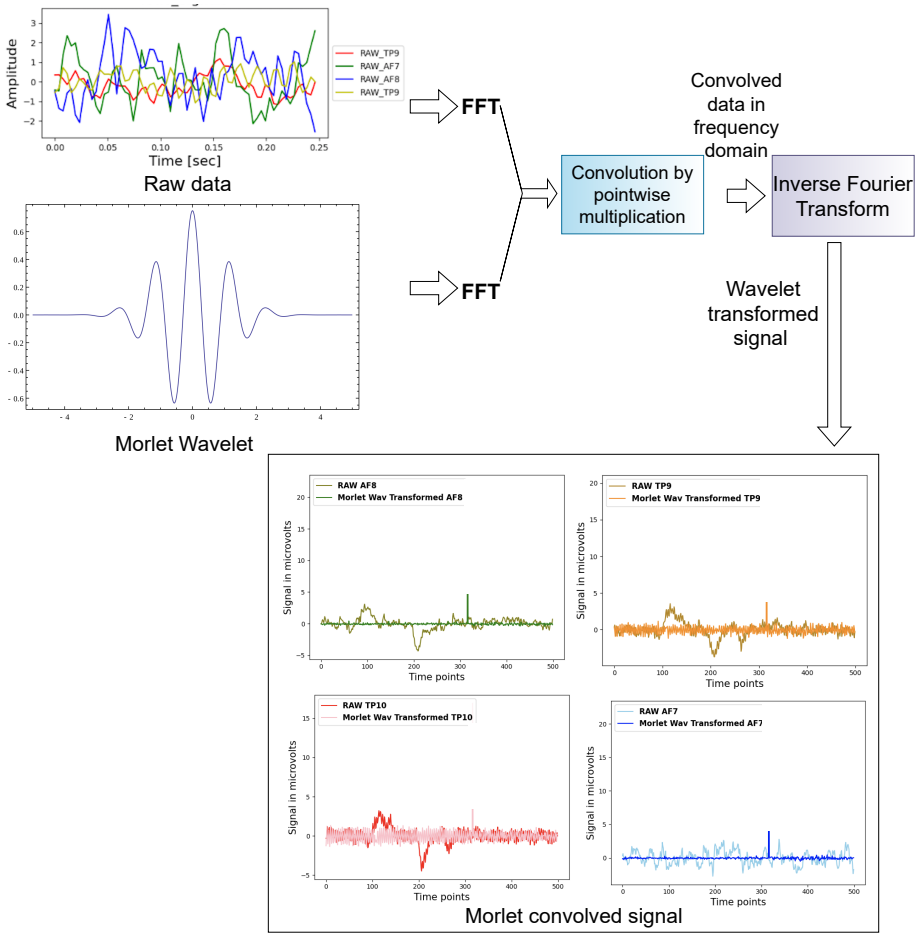


Figure 7: The steps followed for Time-Frequency Transform. The technique of fast fourier transform was used to get time-frequency representation of the data.

$$skewness = \frac{(\Sigma_{i=T}^{T+t} x_i - \frac{\Sigma_{i=T}^{T+t} x_i}{n})^3/n}{s^3} \tag{3.11}$$

$$kurtosis = \frac{(\Sigma_{i=T}^{T+t} x_i - \frac{\Sigma_{i=T}^{T+t} x_i}{n})^4/n}{s^4} \tag{3.12}$$

After preprocessing and feature engineering, we use this module to perform cross validation training of the ML model after dividing the data into train and test. Our solution automatically trains, tests, and validates a variety of ML models using the

features built in feature engineering. Each model is supplied a set of examples, and tasked with learning a general relationship between the features and the goal. The models are then evaluated on a unseen set of data that was not initially used during training, and the best performing model is selected as the winning model. We also use the technique of Grid search to perform hyperparameter optimization for the model. Different ML models are tested on the data with different hyperparameters.

3.4.3.3 Feature reduction

Feature reduction, also called dimensionality reduction, is that the process of reducing the quantity of features during a resource heavy computation without losing important information. Reducing the quantity of features means the quantity of variables is reduced making the computer's work easier and faster. There are many techniques by which feature reduction is accomplished. a number of the foremost popular are generalized discriminant analysis, autoencoders, non-negative matrix factorization, and principal component analysis. Feature reduction leads to the need for fewer resources to complete computations. Less computation time and less storage capacity needed means the computer can do more work. In ML, it removes multicollinearity which leads to improvement of the model in use. Algorithms that rely on Euclidean distance as the measure of distance between two points don't work well. This is referred to as the curse of dimensionality. Models such as K Nearest Neighbors and Linear Regression can easily overfit to high dimensional data and hyperparameter tuning is very important for them. Thus dimensionality reduction can be quite advantageous for any predictive model. We provide the users with two main techniques forward feature selection and autoencoders.

Forward Feature Selection In this method, we start by selecting one feature and calculating the metric value for each feature on the cross-validation dataset. The feature offering the best metric value is selected and appended to a list of features. The process is repeated next time with two features, one selected from the previous iteration and the other one selected from the set of all features not present in the set of already chosen features. The metric value(f-measure) is computed for each set of two features and features offering the best metric value are appended to the list of relevant features. This process is repeated until we have the desired number of features as shown in Figure 8

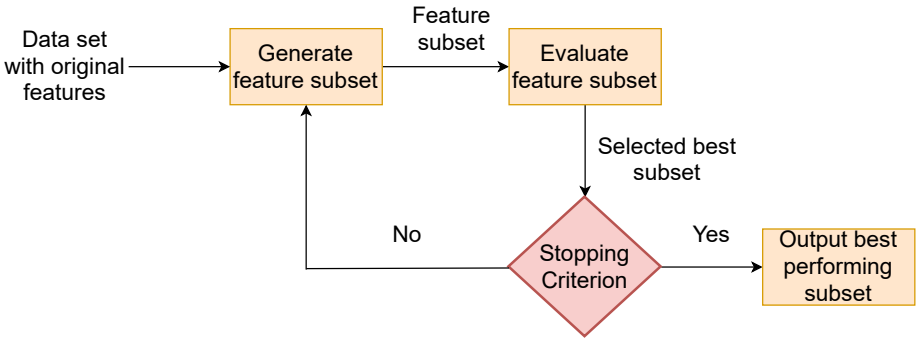


Figure 8: The flowchart for Forward Feature Selection. It gave the top n best features for a model.

Autoencoders Autoencoders are an unsupervised learning technique within which neural networks are leveraged for representation learning. Particularly, a neural network architecture is designed in a manner that it imposes a bottleneck in the network which forces a representation of compressed knowledge of the original input. If the input features are each independent of one another, the compression and reconstruction would be a very tedious task. However, if some sort of correlations exists in the data, it can be learned and consequently leveraged when forcing the input through the network's bottleneck. A stacked autoencoder as shown in Figure 9 is a neural network that consists of several layers of autoencoders where output of each hidden layer is

connected to the input of the next hidden layer.

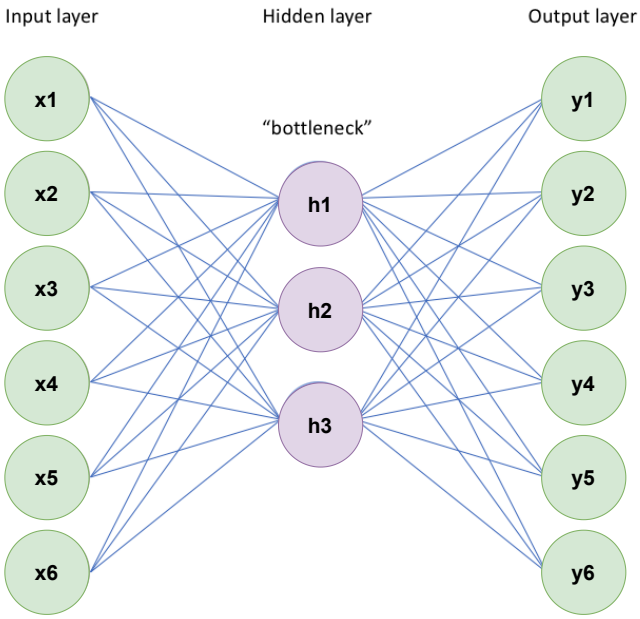


Figure 9: Structure of an autoencoder. The data in the encoded form or latent space from the autoencoder was fed to the classifier for classification.

3.4.4 Modeling

We used a variety of traditional machine learning and deep learning models in this step. We have applied K Nearest Neighbor (KNN), Logistic Regression (LR), Random Forest(RF), Artificial Neural Network (NN), Support Vector Machine (SVM) and Gradient Boosting(GB) as well as deep learning models like convolutional neural networks (CNN), long short term memory (LSTM) and their variants like CNN-LSTM. We provide the best performing models for the final deployment. For deploying a workflow the user has to save the workflow. For a workflow, the best performing model and its parameters and the metadata for preprocessing are saved in pickle, h5 and JSON format.

3.5 Summary

In this chapter, we discussed in detail about the architecture. We explained the implementation of the proposed approach for training and prediction phase. The methodology was then explained in subsequent sections. In the next chapters we discuss in detail about each IoT application that we have implemented in this work. The applications that we have worked on provide a generic usecase which can be replicated and generalized on similar application or application that deal with timeseries data like human activity recognition, price/yield forecasting, anomaly detection in signals and others.

Chapter 4

Understanding Brain Dynamics for Color Perception using Wearable EEG headband

4.1 Introduction

The perception of color is an important cognitive feature of the human brain. The variety of colors that impinge upon the human eye can trigger changes in brain activity which can be captured using electroencephalography (EEG). In this work, we have designed a multiclass classification model to detect the primary colors from the features of raw EEG signals. In contrast to previous research, our method employs spectral power features, statistical features as well as correlation features from the signal band power obtained from continuous Morlet wavelet transform instead of raw EEG, for the classification task. We have applied dimensionality reduction techniques such as Forward

Feature Selection and Stacked Autoencoders to reduce the dimension of data eventually increasing the model's efficiency. Our proposed methodology using Forward Selection and Random Forest Classifier gave the best overall accuracy of 80.6% for intra-subject classification. Our approach shows promise in developing techniques for cognitive tasks using color cues such as controlling Internet of Thing (IoT) devices by looking at primary colors for individuals with restricted motor abilities. We mainly focused on Alpha and Beta frequency bands, as these are most likely to be stimulated when a person is alert, attentive, or concentrating and not performing a high cognitive activity. We employed various linear and non-linear Machine Learning (ML) algorithms namely, K Nearest Neighbors, Support Vector Machine (SVM), Logistic Regression, Random Forest, models like Artificial Neural Networks, and boosting approaches like Gradient Boosting, to perform the three-class classification task. We investigated the classification performance of ML algorithms both on a single person's data (intra-subject) as well as on combining the data from different people (inter-subject). We also applied dimensionality reduction techniques like forward feature selection and stacked autoencoders to increase the performance of the architecture. Our approach shows promise in developing techniques for cognitive tasks using color cues such as controlling IoT devices by looking at primary colors for individuals with restricted motor abilities. The possible research questions we seek to answer using this application are:

1. Is it possible to distinguish EEG signals from a four-channel wearable headband, produced by RGB color exposure, by training ML models on features that account for statistical, spectral and correlation properties of EEG?

2. Can feature reduction techniques like Forward-Feature Selection(supervised) and Autoencoders (unsupervised) make the ML algorithms for EEG classification more efficient?

3. Does the performance of ML algorithms differ for intersubject and intra-subject classification?

4.2 Proposed Methodology

This work has been accomplished in the following five phases: data acquisition, data cleaning and preprocessing, feature extraction, feature selection, and classification of data into red, green, and blue which is shown in Figure 10.

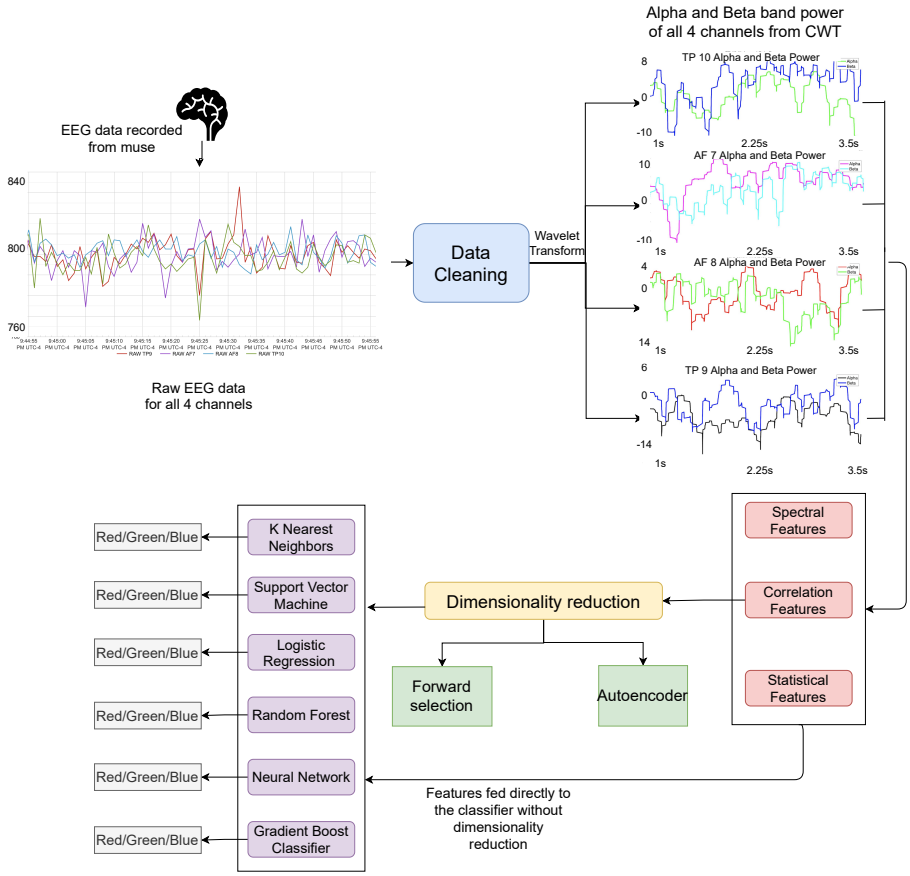


Figure 10: The architecture used in the methodology to classify raw EEG data into primary colors.

4.2.1 Data Acquisition

Muse headband consists of four electrodes/channels.There were eight subjects (aged 18-30yrs) who participated in the visual experiment. The experiment was conducted using the University of Nottingham's Psychopy 3 toolbox [81]. Three trials each four minutes long were conducted for each participant at different times. In each trial, a color from RGB was shown in a random order, twenty times each, for a period of two seconds each, such that a black color was shown for two seconds between each of the RGB colors to provide a baseline to the experiment. The experiment was conducted in a dark room and the subjects were told to do minimum facial movements and eye blinks.

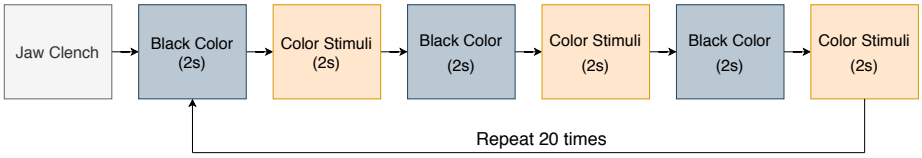


Figure 11: The experiment protocol for data acquisition for the color stimuli application.

4.2.2 Data cleaning and preprocessing

The raw EEG data is generally very noisy and it needs to be cleaned and pre-processed in order to remove artifacts from it. In our methodology we cleaned the data in two steps. Firstly we analyzed the data using Matlab's EEG lab software [82] and labeled any visible unwanted spikes and noise manually from the data. Secondly, we divided the data into small time windows of 50 ms and computed the variance of data in each window, if it was more than an selected threshold then the time window was flagged. We also examined the individual subject's data and used the trial that has a minimum number of jaw clenches and eye blinks for further experimentation. We then passed

this data to the proposed web application for further preprocessing.

4.2.3 Feature Extraction

Feature extraction is a very vital part of our problem. The use of raw EEG data did not give good results in our experiment and so we used Time-Frequency analysis to find frequency band coefficients that were most relevant for our problem i.e. Alpha coefficients(8- 12Hz) and Beta coefficients(13-30Hz). In past works, [35, 83] Discrete wavelet transform(DWT) has been used to extract the frequency bands of interest. However in our case, we are not interested in all the frequency bands, instead, we only consider alpha and beta bands. The use of DWT would have given us an improper breakdown of bands with the Alpha band in the range of 8-16Hz and beta in range of 16-32Hz and therefore to avoid this we used Continuous wavelet transform method to extract the bands of interest.

The mother wavelet that we used is the Morlet wavelet. The morlet wavelet has a peak in the center after which it tapers to the edges. The complex Morlet wavelet can be obtained by the convolution of a Gaussian with a sine wave. We now explain the features that we extracted from the time-frequency spectrograms:

4.2.3.1 Spectral features

These features were calculated by taking into account the average power of each band, the variance in the power of each band, and the hemispherical difference in each band over a time window for each of the four electrodes. Thus we got 18 features(8 average power coefficients for each channel, 8 variance power coefficient for each channel, and 2 hemispheric difference coefficients) for each sample that was formed by a single time window. This method was similar to the one followed in [35]. This set of spectral

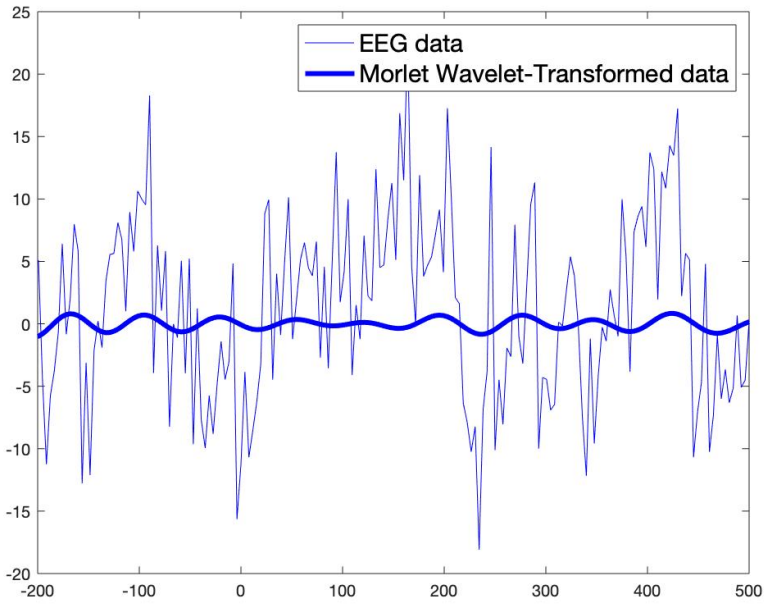


Figure 12: The original EEG signal and its Morlet-convolution version using a wavelet of 30 Hz.

features are the most commonly used feature in many EEG related studies as they allow the model to evaluate any potential changes in the absolute band power due to stimuli.

4.2.3.2 Pairwise Correlation features

In addition to the spectral features, it is also important to study the correlation among different frequency bands from different electrodes.We calculated this using a pairwise correlation in the time windows for each band and each electrode. [35] follows this method too. We got a total of 28 correlation features using this method from a single time window. These features were helpful to find cross-region similarity as some of our data was discontinuous because of artifact removal.

4.2.3.3 Statistical Features

Features that represent the statistical properties of the signal like Kurtosis, Skewness, Shannon Entropy and Hjorth Parameters were also extracted.

Kurtosis, Skewness and Shannon Entropy Kurtosis is a measure of outliers in data. Data with less value of Kurtosis has less number of outliers. The Skewness measures the asymmetry in data. The entropy is a measure of information in data. We calculated each of these parameters both for alpha and beta bands, therefore we got 24 features from these properties.

Hjorth Parameters They are indicators of statistical properties used in signal processing in the time domain introduced by Bo Hjorth in 1970 [81]. We obtained 16 Hjorth parameters for alpha and beta band for all 4 channels. We calculated two Hjorth parameters namely, the mobility parameter as in equation 4.1 and complexity parameter as in equation 4.2 on alpha and beta power bands that we obtain from CWT.

$$\text{Mobility} = \sqrt{\frac{\text{var}\frac{\text{d}y(t)}{\text{d}t}}{\text{var}y(t)}} \tag{4.1}$$

$$\text{Complexity} = \sqrt{\frac{\text{Mobility}\frac{\text{d}y(t)}{\text{d}t}}{\text{Mobility}(y(t))}} \tag{4.2}$$

Here $y(t)$ is the alpha or beta band power for a time window. We got a total of 40 statistical features. Figure 14 shows the visualization of features in 2-D space by applying Linear Discriminant Analysis. It shows that the three classes are almost separable.

Table 4.1: The features obtained from raw data.

Features	Alpha	Beta	Total Features
Avg. Power features	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	8
Var. Power features	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	8
Hem. diff features	Left hem sensors - right hem sensors	Left hem sensors - right hem sensors	2
Correlation features	Cross corr of alpha & beta for all channels	Cross corr of alpha & beta for all channels	28
Kurtosis	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	8
Sknewness	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	8
Shannon Entropy	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	8
Hjorth Parameters	TP9, TP10, AF7, AF8	TP9, TP10, AF7, AF8	16
Total Features			**86**

4.2.4 Feature Selection

The process of feature selection is important because it has many advantages like reduced training times, simplified and interpretable models, reduced chances of overfitting i.e. lesser variance and less impact of the curse of dimensionality.We perform feature selection by two different methods. Firstly we used the Forward Feature selection technique which is a supervised approach and secondly, we used Autoencoders which is an unsupervised approach for feature reduction.

4.3 Classification Task

We applied ML models on the 86 features that we extracted by the procedure explained in Section 4.2.3. The classification task was done in two folds. We first considered the data from individual subjects and applied models to that data to perform intra-subject classification for which we achieved an accuracy of 80.6%. Intra-subject classification

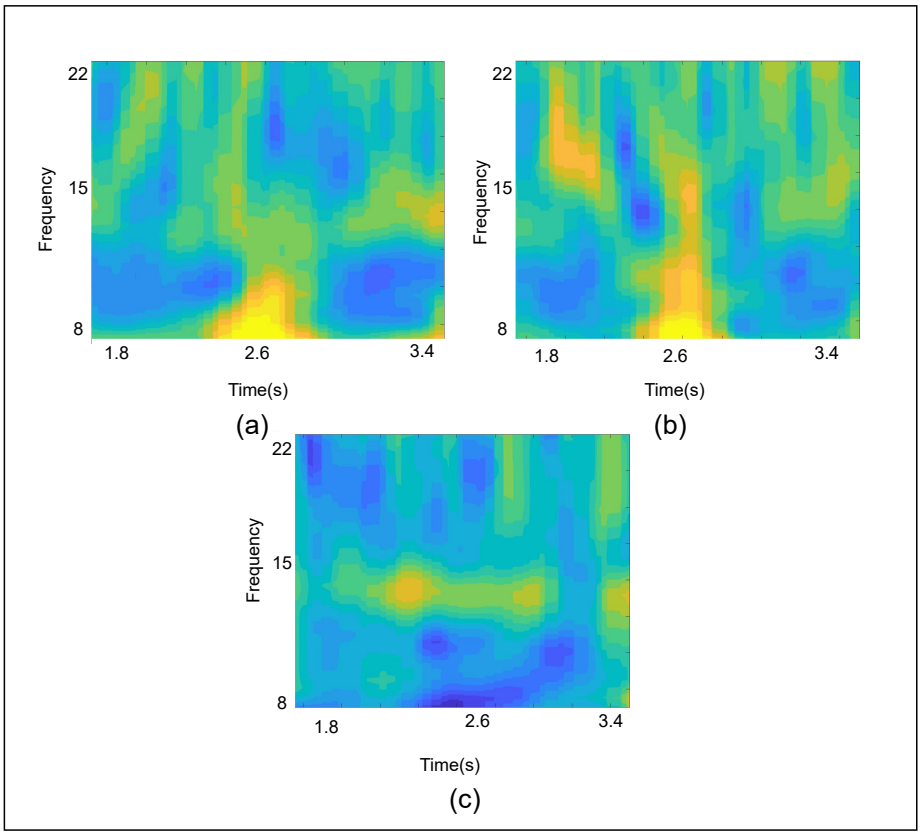


Figure 13: Spectrograms of (a) Red, (b) Green, and (c) Blue respectively. The stimuli for different colors is discriminated in respective spectrograms.

helped us to study subject-specific differences of the EEG reactivity patterns. Then we considered the combined data from all the subjects and performed inter-subject classification and got an accuracy of 58.1%. The inter-subject case helped us to make a more generalized model. The classification was done in two ways and their performances have been compared. We performed classification using the original feature set as well as the reduced feature set from forward selection and autoencoders for both intra-subject and inter-subject. Below we elaborate on the models that we have used along with the chosen hyperparameters. We tuned the hyperparameters using Grid Search and Keras tuner.

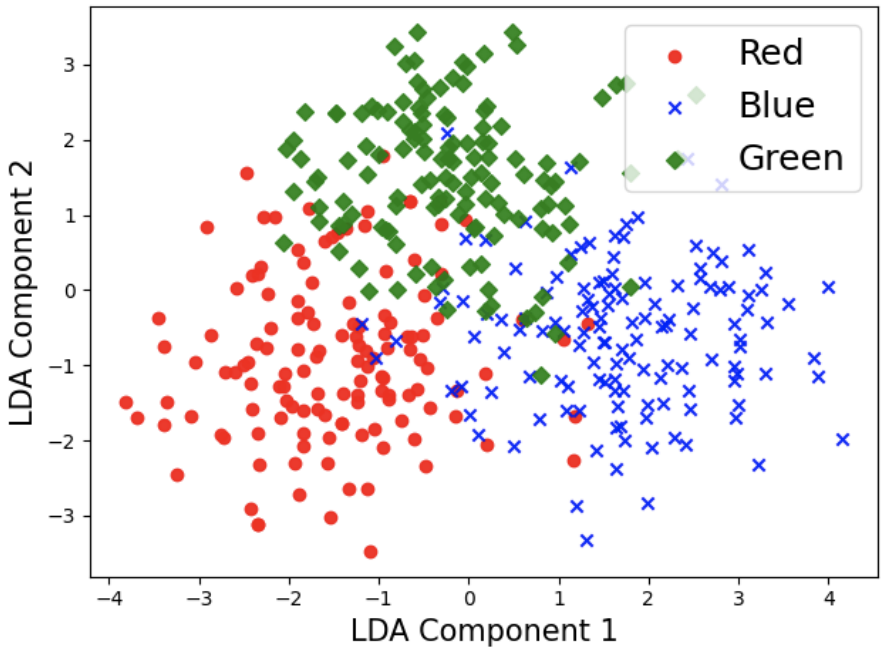


Figure 14: Visualization of data of Subject 1 for a single trial in 2-D space using Linear Discriminant Ananlysis

4.3.1 K Nearest Neighbor (KNN)

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution and that's why we tested it in our problem. K is a critical hyperparameter that we varied in the range 4 to 8 in our experiment. The Euclidean distance was used as the distance metric. As KNN is a lazy learner, therefore it is not advisable to use it in our application, we use it for comparison purposes only.

4.3.2 Logistic Regression (LR)

We used logistic regression model both with ridge and lasso regularization. We varied the parameter C or penalty term in the range 0.01 to 100. We found out that lasso regularization gave better results on our data.

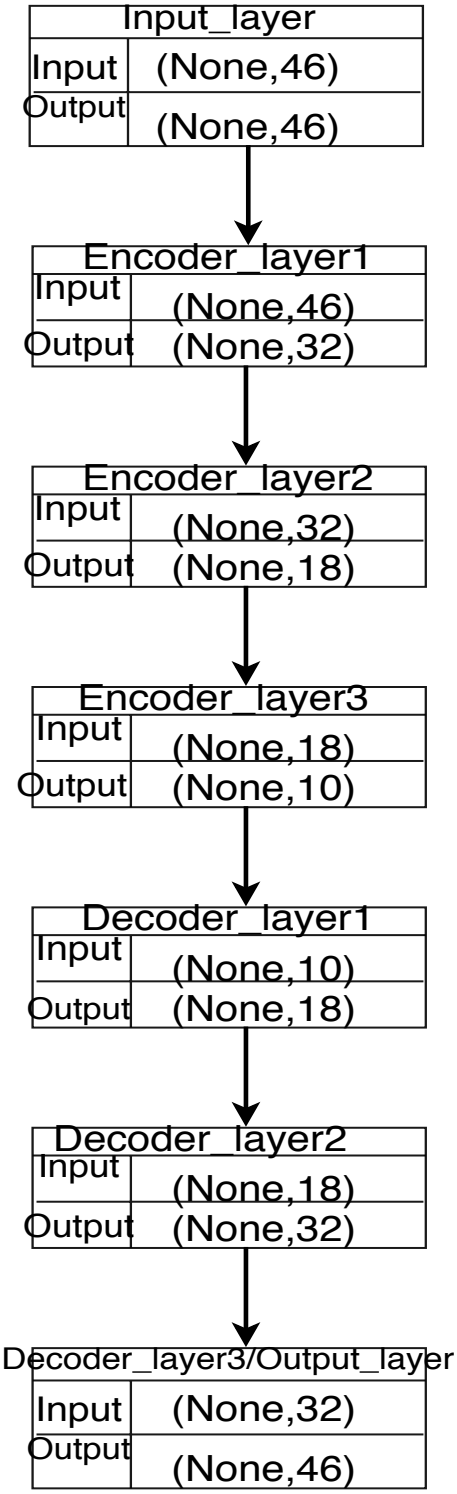


Figure 15: Final Autoencoder Architecture used for color stimuli application.

4.3.3 Random Forest (RF)

The random forest algorithm is an ensemble approach that uses multiple decision trees and makes a classification decision by voting from all the trees. The number of estimators in our problem were varied from 10 to 100.

4.3.4 Artificial Neural Network (NN)

We have used ANN with the following architecture: First hidden layer with 300 neurons and second hidden layer with 100 neurons. The activation function used was sigmoid. L2 regularization had been used to avoid overfitting, with a regularization rate of 0.0001. The hyperparameter tuning was done using grid search and keras tuner.

4.3.5 Support Vector Machine (SVM)

SVM with RBF kernel has been used in our experiment. The hyperparameters C and Gamma were varied between 0.001 and 100 and 0.01 and 10 respectively.

4.3.6 Gradient Boosting (GB)

Gradient boosting is an ensemble learning approach that produces a prediction model in the form of an ensemble of weak prediction models. Gradient boosting combines weak learners into a single strong learner. In our Gradient boosting model we varied the hyperparameter estimators from 10 to 100.

4.4 Evaluation Metrics for Multi-class Classification

Many metrics are used to evaluate ML Models like average accuracy, precision, recall, F-measure, ROC-AUC score, MCC score etc. In our case, we used three metrics for

performance evaluation of our models- Average Accuracy, Average ROC-AUC score, and Average Matthews Correlation Coefficient (MCC). Since our data is balanced i.e. each class has almost equal representation the average accuracy score would have sufficed but we used the other two additional metrics to verify the performance of our models. We used scikit learn [77] library of python to evaluate the models.

4.4.1 Accuracy Score

The accuracy score in our problem was calculated as :

$$\text{Average Accuracy Score}(y,\hat{y}) = \frac{1}{n_{sample}}\Sigma_{i=0}^{n_{samples}-1} 1(y_i = \hat{y}_i) \tag{4.3}$$

In equation 8, $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value and $1(x)$ is the indicator function. $n_{samples}$ is the total number of samples. The accuracy indicates the samples that were correctly classified from all the samples.

4.4.2 ROC-AUC score

ROC-AUC stands for Receiver operator characteristics- Area under the curve, it basically calculates the area under the receiver operator curve.The ROC curve is created by plotting the true positive rate (TPR $= \frac{TP}{TP+FN}$) against the false positive rate (FPR $= \frac{FP}{TN+FP}$) at various threshold settings. We find the area under the curve to evaluate our model. Since our problem is multiclass therefore we computed the average AUC of all possible pairwise combinations of classes using equation 5 as suggested in [82].

$$\text{Average ROC-AUC Score} = \frac{2}{c(c-1)}\Sigma_{j=1}^{c}\Sigma_{k>j}^{c}(\text{AUC}(j|k) + \text{AUC}(k|j)) \tag{4.4}$$

where c is the number of classes and AUC($j|k$) is the AUC with j as the positive class and k as the negative class and AUC($k|j$) is vice versa. In general, AUC($j|k$) $\neq$ AUC($k|j$)) in the multiclass case.

4.4.3 Matthews Correlation Coefficient

The Matthews correlation coefficient [84] is used to evaluate the quality of binary and multiclass classifications. The MCC is a kind of correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. In the multiclass case, Matthews correlation coefficient can be defined in terms of a confusion matrix C for K classes. The MCC for multiclass as suggested in [83] is calculated as follows:

$$MCC = \frac{c \times s - \Sigma_k^K p_k \times t_k}{\sqrt{(s^2 - \Sigma_k^K p_k^2) \times (s^2 - \Sigma_k^K t_k^2)}} \tag{4.5}$$

where $t_k = \Sigma_i^K C_{ik}$ is the number of times class k truly occurred, $p_k = \Sigma_i^K C_{ki}$ is the number of times class k predicted, $c = \Sigma_k^K C_{kk}$ is the total number of samples correctly predicted and $s = \Sigma_i^K \Sigma_j^K C_{ij}$ the total number of samples.

4.5 Experimental Results

We performed two categories of classification namely, intra-subject and inter-subject classification. For intra-subject classification, we applied 5-folds cross-validation for data from five trials of each subject and in the second classification, we applied leave out one subject cross-validation where we trained the model on seven subjects data and validated it using a single subject data and we repeated it for all subjects. The performance metrics that we used to evaluate our model are average cross validation

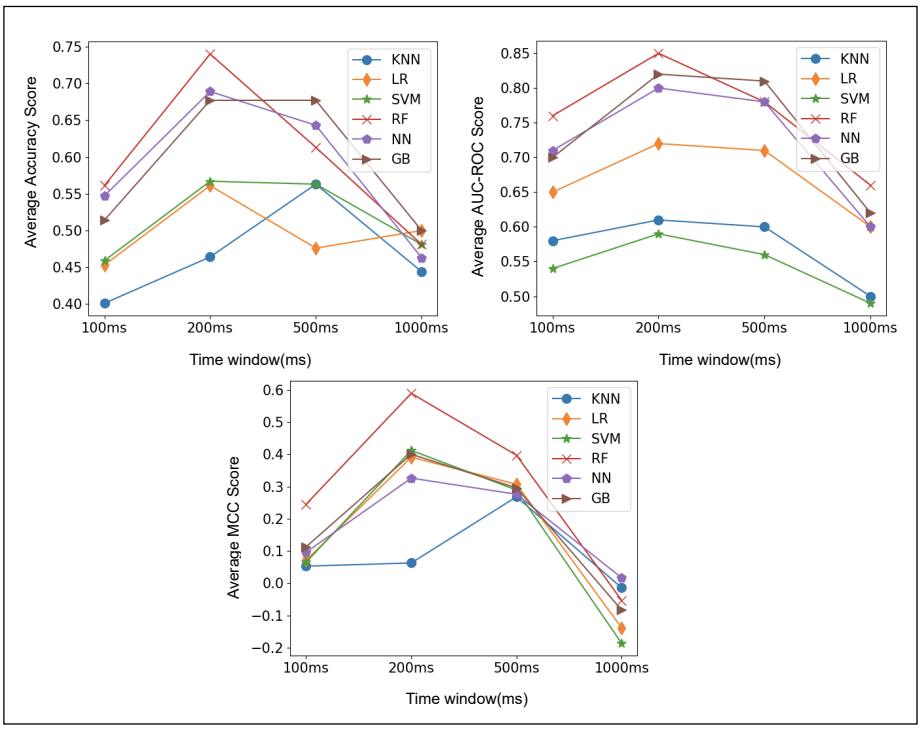


Figure 16: Average Accuracy, Average ROC-AUC score and Average MCC score for different time windows for intra-subject classification.

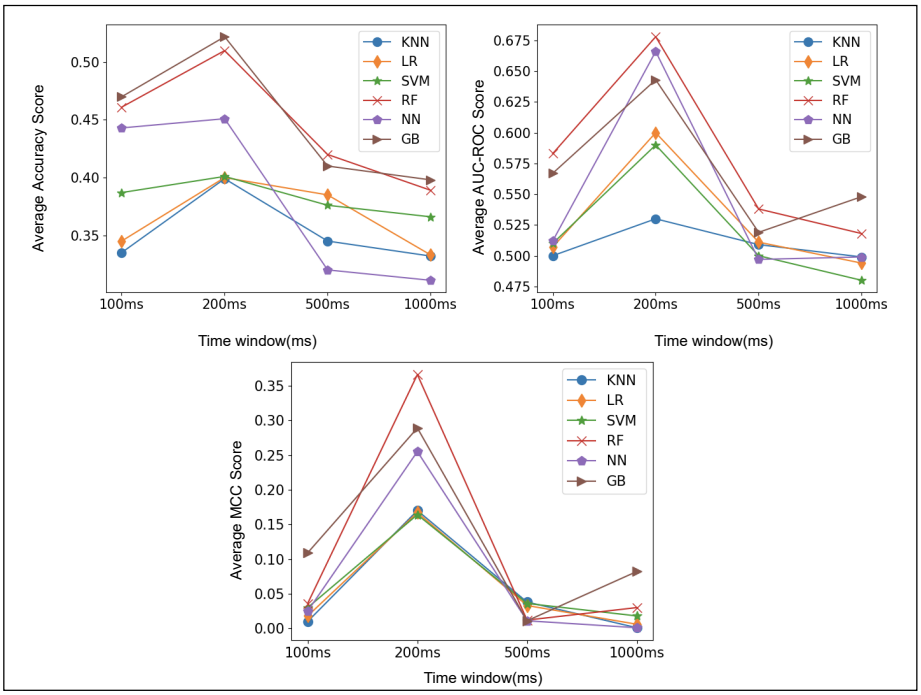


Figure 17: Average Accuracy, Average ROC-AUC score and Average MCC score for different time windows for inter-subject classification.

accuracy, average ROC-AUC score, and average MCC. We report the results on the complete dataset as well on the reduced dataset from dimensionality reduction techniques. The average accuracy, average AUC score, and average MCC score with different time windows for both intra-subject and inter-subject classification are shown in Figure 16 and Figure 17 respectively. We got the best results for a time window of 200ms. We considered the time window of 200ms for further experimentation. We discuss the results in three segments, the results without dimensionality reduction, results after dimensionality reduction from the forward selection algorithm and results after dimensionality reduction from Autoencoder. In the following subsections, we show the results considering the average metric score of all the subjects, the best metric score among all subjects and the inter-subject metric score for the three metrics explained in Section 4 . In all the tables the number in brackets is the standard deviation. We

have highlighted the highest metrics for each case in all tables. The code for all the experiments is available online

4.5.1 Results without Feature Selection

We saw that the Random Forest algorithm performed the best and Neural Network and Gradient Boosting classifier also showed comparable results as shown in Table 4.2. The highest accuracy for an individual was 70.2% which was reasonably better than the accuracy of random guess i.e. 33%. The inter-subject accuracy of 56.8% was also very promising considering the fact that we applied leave one subject out cross-validation in this case. The results were better for intra-class classification which means that a customized model could be trained on an individual's data and then it can be used for predictions for a particular subject rather than using data from different people which might also cause privacy issues.

Table 4.2: Accuracy by using all the features at 200ms time window.

Metrics	KNN	SVM	Logistic Regression	Random Forest	Neural Network	Gradient Boost
Avg Subject Accuracy	0.414 (0.056)	0.474 (0.018)	0.506 (0.028)	**0.625 (0.018)**	0.523 (0.013)	0.608 (0.057)
Best Subject Accuracy	0.513 (0.021)	0.578 (0.031)	0.600 (0.022)	**0.702 (0.000)**	0.700 (0.026)	0.669 (0.039)
Inter-subject Accuracy	0.338 (0.033)	0.377(0.030)	0.408 (0.030)	**0.568 (0.025)**	0.490 (0.033)	0.472 (0.040)

4.5.2 Results with Forward Feature Selection

We saw significant improvement in the results with the use of forward feature selection which is a supervised feature selection technique. The irrelevant and noisy features were removed and the

feature set was reduced to 10. This methodology helped us to curb the overfitting issue too and thus the performance on the validation set improved. There was an increase in average accuracy by nearly 10% and we got the highest accuracy of almost 80.6 which was much better than any other previous approaches that have been used for EEG classification using RGB colors using wearable devices. The average subject accuracy of 72% showed that the classifier performed well for all the subjects. The average accuracy increased by 9.5İn this case, also the results of intra-subject classification were better than that of inter-subject classification. The inter-subject classification accuracy improved by 1.3%. Random Forest algorithm had given us the best results in this case too with Neural network and Gradient Boost with comparable performance. The results are shown in Table 4.3.

Table 4.3: Accuracy by using 10 features by forward selection at 200ms time window

Metrics	KNN	SVM	Logistic Regression	Random Forest	Neural Network	Gradient Boost
Avg Subject Accuracy	0.492 (0.038)	0.487 (0.045)	0.492 (0.028)	**0.720 (0.035)**	0.513 (0.036)	0.597 (0.048)
Best Subject Accuracy	0.615 (0.051)	0.604 (0.028)	0.590 (0.050)	**0.806 (0.041)**	0.766 (0.039)	0.720 (0.035)
Inter-subject Accuracy	0.377 (0.013)	0.366 (0.024)	0.388 (0.012)	**0.581 (0.032)**	0.475 (0.040)	0.411 (0.019)

4.5.3 Results with Autoencoders

We applied autoencoder to observe how an unsupervised feature reduction technique would work on our data. With the autoencoder, a reduced feature set of 10 was obtained. Using this reduced feature set as input to the ML models, we achieved a lower average CV accuracy in comparison to classification using forward feature selection. Therefore autoencoders are not recommended for our application. We show the results in Table 4.4.

The final proposed model for our application is that of Random Forest classifier with forward feature selection. In Table 4.4 we compare our results with previous efforts that have been done

to classify EEG signals on the basis of color stimuli using wearable EEG devices. In Figure 18 we show the ROC curve for the proposed model with AUC-ROC score of individual classes for all the subjects where 0 represents red, 1 represents green and 2 represents blue.

Table 4.4: Accuracy by using 10 features by Autoencoder at 200ms time window

Metrics	KNN	SVM	Logistic Regression	Random Forest	Neural Network	Gradient Boost
Avg Subject Accuracy	0.398 (0.010)	0.362 (0.026)	0.393 (0.022)	**0.430 (0.011)**	0.409 (0.009)	0.417 (0.000)
Best Subject Accuracy	0.417 (0.000))	0.406 (0.000)	0.434 (0.000)	0.489 (0.008)	0.473 (0.024)	**0.510 (0.000)**
Inter-subject Accuracy	0.358 (0.012)	0.348 (0.027)	0.347 (0.023)	0.397 (0.017)	0.355 (0.028)	**0.398 (0.012)**

Table 4.5: Performance of other methods on EEG classification into color stimuli. Our aprroach shows 5-folds average accuracy value for intra-class classification

Algorithms	Best Average Accuracy
Martin Angelovski et.al. [85] using 2 channel portable EEG	53%
Sara Åsly et. al. [35, 86] using 4 channel portable EEG	58%
Kyle Phillips et. al. [87] using 14 channel emotiv EEG	79.6%
Rakshit et. al. [88] using 10 channel medical EEG	81.2%
Our approach using 4 channel portable EEG	80.6%

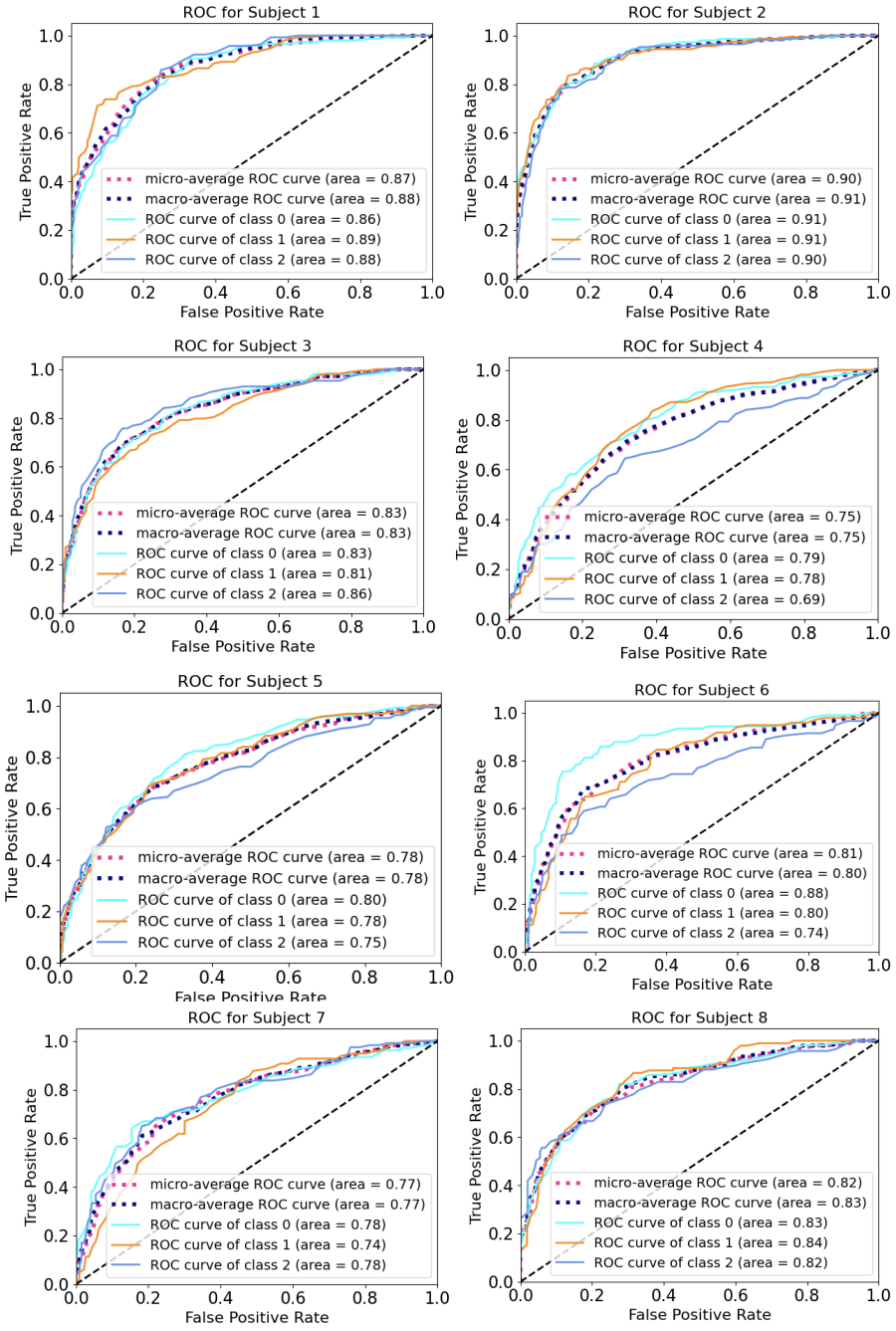


Figure 18: The ROC-AUC curve using our proposed architecture for all the subjects for color classification.

4.6 Summary

We have used EEG signals from a wearable consumer-grade EEG headband to classify the raw EEG data into three classes of colors, red, green, and blue. In our approach, we focussed mainly on Alpha and Beta frequencies and discarded all other lower and higher frequencies which otherwise would have added noise to the data. We extracted various spectral, correlation and statistical features from the data and apply ML models to it. Our proposed model of Random Forest with forward feature selection showed significant improvement when compared to previous approaches. Our methodology achieved an improvement of almost 20% in the average accuracy of classification.

Despite having a fewer number of electrodes Muse performed well in the classification task and gave promising results. The intra-class classification accuracy of 80.6% shows that wearable devices can be used in integrated IoT frameworks where they can be used in various control applications. The IoT pipeline for this application must take into account the data preprocessing and feature extraction in real-time. The time window for our particular application was small to capture the effect of color stimuli only and avoid unnecessary artifacts in data. This time-window might vary for different applications. One drawback of Muse that we encountered during experiments was that it cannot be worn for a long time due to comfort issues and also the connection can become weak sometimes however one can overcome this problem by applying water to the channels. With the advancement in wearable computing, more comfortable devices are now available that would not bother one if used for a longer time like the new Muse S headband. The Muse 2 device is also sensitive to muscle movements but that is not an issue in our application as we are only interested in a small time window of data when a person focuses on a color. Our work has thus highlighted the capability of these wearable devices to detect and classify the EEG signal on the basis of color stimuli and the results are encouraging. This study opens up a new door to integrate these devices in our day to day lives to use brain signals to control various devices.

Chapter 5

A Deep Learning Approach To Classify Cognitive Load Using Wearable EEG

5.1 Introduction

In this work, we have proposed a robust method to identify cognitive load using a wearable device to acquire EEG signals while performing a cognitive motor integration (CMI) task. EEG signals have been extensively used for cognitive load assessment as they produce clear manifestations of cognitive changes by estimating neural arousal in the brain. We have designed various self attention based multi-channel deep learning models, classifying the data into high and low cognitive load classes. Our proposed model has achieved the highest overall cross validation train accuracy of 98.1% and test accuracy of 96.1% for intra-subject classification using attention multi-channel ConvLSTM. The technique of transfer learning was used to train the best performing model for inter-subject classification and we achieved the best cross validation train

and test accuracy of 87.5% and 86.3% respectively. Our study is the first of its kind to use transfer learning for inter-subject classification of EEG signals. The results that we have achieved have shown an improvement of approximately 9%, in terms of accuracy, over the past approaches. Moreover, the fact that we use a wearable device paired with a tablet-based performance task, makes our application portable, economical and easy to use.

Cognitive load refers to the total amount of mental activity imposed on working memory [89,90] in any one instant and it was first proposed by John Sweller in [89] out of a study of problem solving in the late 1980s. It can be understood as the level of mental engagement which has a direct impact on the efficacy and quality of learning [91]. Cognitive overload occurs when a person is mentally overwhelmed and it affects one's memory and decision-making abilities, it also deteriorates performance [92]. There are also physiological reactions that can happen, such as frustration, stress and depression [93]. We have proposed a Deep Learning (DL) based method to estimate mental workload which can help an individual identify the optimal level of mental workload and hence enhance one's learning performance. Our methodology can be used in workplaces where it can monitor stress levels and minimize overwhelming job tasks, so that the employees can perform to their fullest potential. In an academic setting, a regular assessment of student's cognitive engagement can be used to optimize the pace of teaching and enhance the effectiveness of the learning process. The approach that we put forward in this work has outperformed the existing approaches to identify cognitive load. We performed binary classification of EEG signals from a low cognitive task and a high cognitive task using DL models. An EEG detects electrical activity in the brain using small, metal discs (electrodes/channels) attached to the scalp and forehead. EEG is a non-invasive neuroimaging modality which measures the electrical signal changes in the brain induced by cortical activity. The integration of computational neuroscience with algorithms to control the environment is often termed as Brain Computer Interface (BCI). We propose a BCI application for classifying human cognitive effort which can be potentially used for monitoring of cognitive load conditions in e-learning or by clinicians to monitor behavioural performance. We collected data from Muse 2 headband.

We propose a methodology to perform time-series classification on the EEG data using four time-series classification models namely, Attention Multi-Channel Convolutional Neural Network(CNN), Attention Multi-Channel Convolutional Neural Network with Long Short Term Memory (CNN-LSTM), Attention Multi-Channel ConvLSTM and Stacked Bidirectional LSTM. These models have been used in various previous studies [72, 94], where authors have tried to classify time series data. We made few variations in these models like making them Multi-Channel i.e. giving the input from all the (4) channels altogether. The multi-Channel approach that takes data cubes (multi-channel stacked spectrograms) as the input to the model, allowed to capture the correlation and causality between different channels and it gave more importance to spatial and temporal information of EEG data by constructing 2-D spectrograms sequences as multi-channel input. Similar multi-channel models have been lately used to classify time series data [95–97]. Using such representation, we avoid the selection of frequency bands or channels in the process of feature selection. We also applied the concept of Self-attention to improve the performance of the models.

The approach that we put forward in this work has outperformed the existing approaches to identify cognitive load. We performed binary classification of EEG signals from a low cognitive task and a high cognitive task using DL models. An important point to mention is that in this work we mainly focus on mental load and not on mental stress as both are different concepts [98] and therefore we do not compare our results with the work dealing with mental stress. We performed the classification task on intra-subject data i.e. data of a single subject as well as inter-subject data which was done by combining the data from all the subjects together. The inter-subject classification was performed using Transfer Learning and it helped us to test the generalizability of the models. Our contributions are:

- We used the data from a CMI task to classify cognitive load using Deep Learning. The proposed architecture uses a novel technique to classify data based on Multi-Channel approach where the input is in form of spectrogram stack from all the channels of Muse 2.

- We compared state-of-the-art time-series classification DL models on the basis of their performance on our data and achieved the highest test accuracy of 96.1% for intra-subject classification using Attention Multi-Channel ConvLSTM. Our model has surpassed the performance of previously proposed models and hence shows promise in reliable detection of cognitive load.
- Our proposed model obtained the highest inter-subject test accuracy of 86.3% for inter-subject classification using the concept of Transfer Learning on Attention Multi-Channel ConvLSTM. Our study is first of its kind to propose the use of Transfer Learning in EEG cognitive classification problem.

5.2 Methodology

We applied data processing techniques to get the brain waves frequencies and use them as input to our DL models. Continuous wavelet transform was used on the raw data to convert it into time-frequency domain and then we fed this representation of the data to DL models. In our methodology we first acquired data from a CMI task, then we preprocessed and engineered it and finally we applied various DL models for classification. The methodology we followed is shown in Figure 19.

5.2.1 Description of Cognitive Tasks

Participants completed two blocks of a computer-based visuomotor skill assessment task (BrDI™) that included one standard and one non-standard conditions where vision and action were decoupled. This latter condition required the integration of spatial and cognitive rules, and thus required cognitive-motor integration (CMI). Participants sat at a desk so they could comfortably reach a 10.1" tablet (Samsung Galaxy Tab A) placed on the desk in front of them. All hand movements were made on the tablet. The task required participants move the index finger of their dominant hand along the touch screen of the tablet to move a cursor (white dot, 5 mm di-

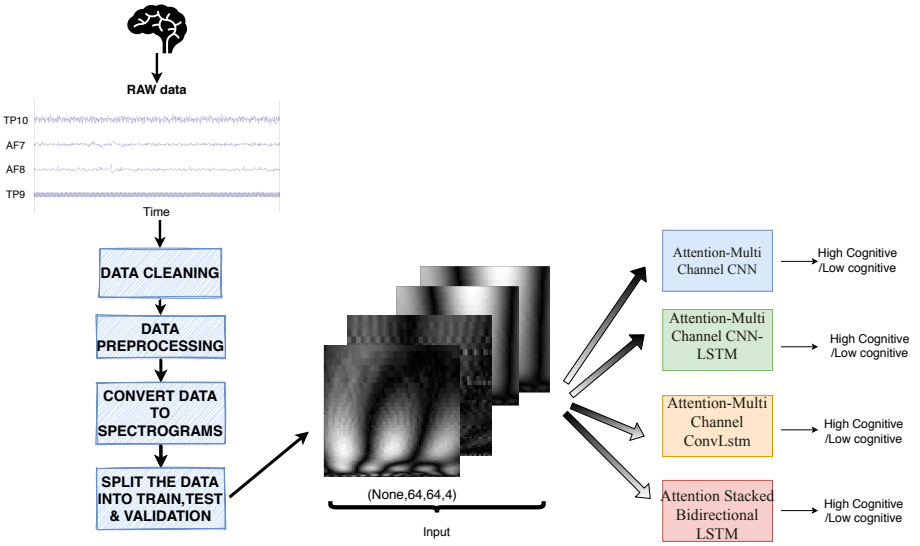


Figure 19: The methodology used in our approach for classifying cognitive load using EEG data.

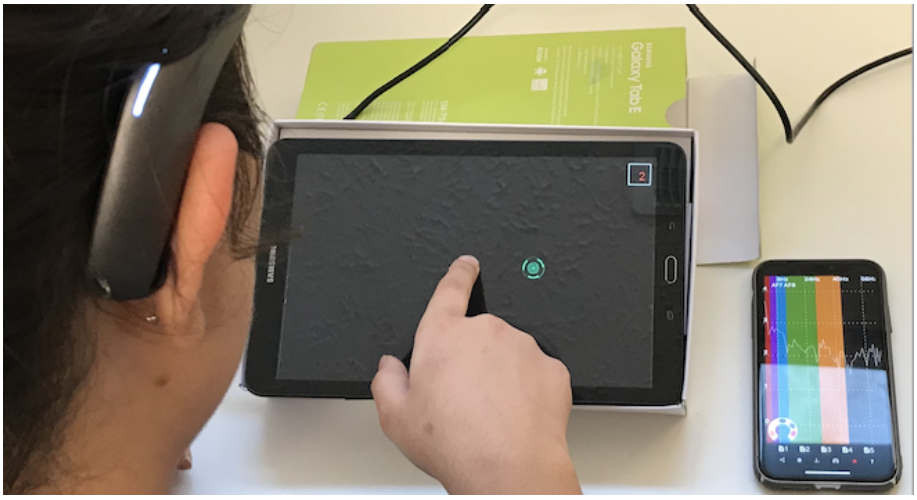
ameter) from a central location to one of four peripheral targets (up, down, left, or right relative to center) as quickly and as accurately as possible. To start a trial, participants guided the cursor to a solid green 8 mm diameter circle in the center of the screen. After a 2000-ms center hold time, an open green peripheral 10 mm diameter target was presented, which served as the "Go" signal for the participant to initiate movement. The participant slid their finger along the touch screen to move the solid green cursor onto the open green target. Once the cursor reached and remained in the peripheral target for 500 ms, it disappeared, signaling the end of the trial. The next trial began with the presentation of the central target after an intertrial interval of 2000 ms. Peripheral targets on the tablet were located 37.5 mm from the central start target (center-to-center distance). There were 20 trials for each task, 5 to each target. In the standard condition, participants looked at the target and used their finger to displace the cursor that was directly under their finger, thereby directly interacting with the targets. In the non-standard CMI condition, the display was split by a horizontal white line. The participant had to view the targets and the cursor presented in the top half of the tablet screen. To displace the cursor in this task, however, they had to slide their finger within the bottom blank half of the tablet screen. Furthermore, the

cursor feedback was 180° rotated from finger motion, such that the participant had to slide one direction to move the cursor in the opposite direction to reach the target. They were instructed to view the targets on the upper screen and not their (extrafoveally located) hand in the blank lower screen. Participants completed two practice trials in each of the four directions before each task was presented for the first time in order to become familiar with the task requirements.

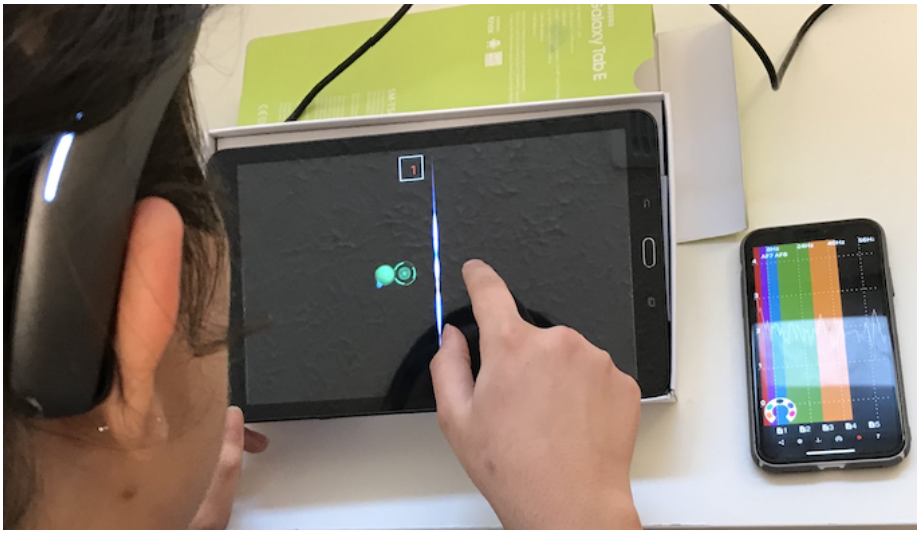
For each condition, they were instructed to complete the task as quickly and as accurately as possible. In summary, participants completed the two tasks (standard task, CMI task), for a total of 40 trials. The entire behavioural task took approximately 6 minutes (2 minutes for each 20-trial of standard task and 3-4 minutes for each 20-trial of CMI task). Both the standard vs. CMI task were randomized for each participant.

5.2.2 Data Collection

EEG data from Muse 2 headband was collected from 12 participants (aged 20-50 years) from all four channels i.e. TP10, TP9, AF7 and AF8. All procedures were approved by York University's Human Participants Research Committee, and all participants provided informed consent to participate.A participant performing the task can be seen in Figure 20. The EEG data was captured in microvolts at a frequency of 256 Hz which is the default frequency of device. The participants were instructed to relax before the start of the experiment and were asked to minimize eye and muscle movements in order to avoid noise in data. A third party application called Mind Monitor was used to collect raw EEG data from Muse. The application has the capability to detect potential eye blinks and jaw clenches and this functionality was later used to clean data. It also provided markers for experiments which facilitated in data collection. The EEG data was collected using Open Sound Control Protocol (OSC) from Mind Monitor application. The default frequency of 256Hz facilitated to perform Wavelet Analysis to extract band power from the data. The brain wave data from both tasks has been plotted in Figure 21.



(a)



(b)

Figure 20: A participant performing the experiment (a) Standard Task (b) CMI task.

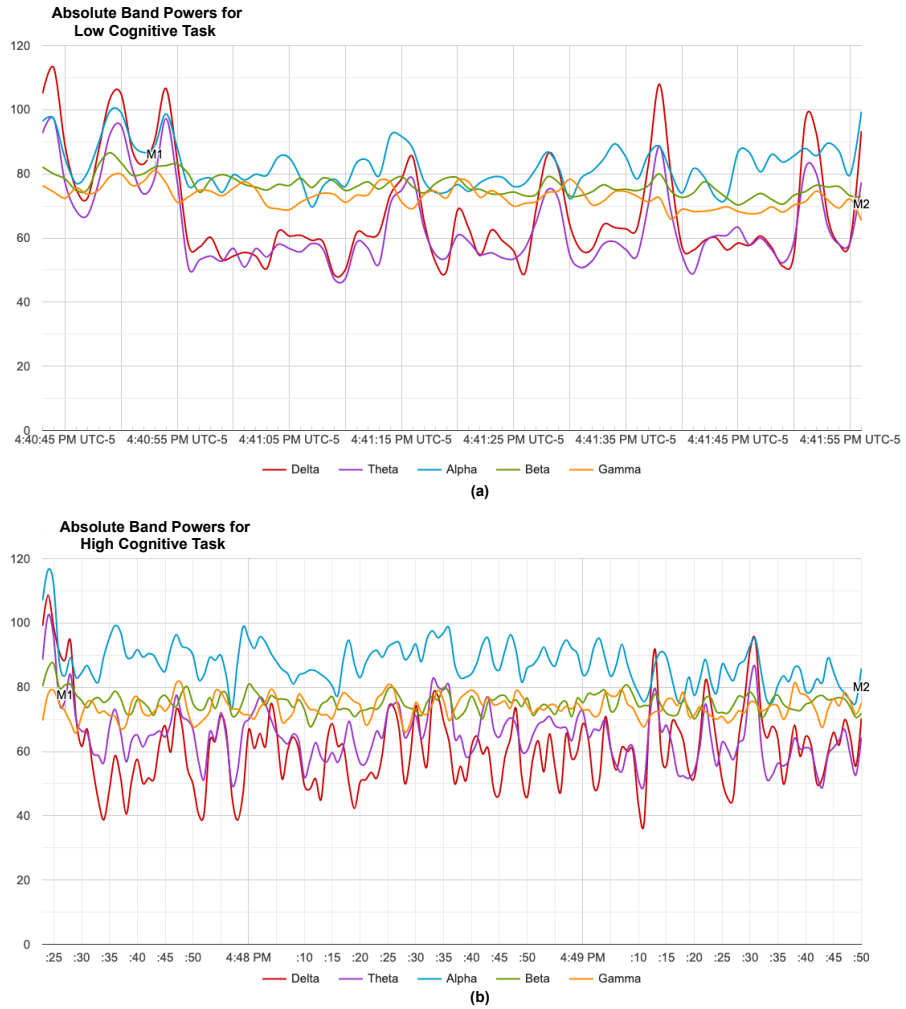


Figure 21: The average absolute power for different frequency bands for (a) low cognitive task and (b) high cognitive task. The average absolute band for gamma and alpha increased in case of high cognitive task when compared to baseline low cognitive task. The makers M1 and M2 show the start and end of stimulus respectively. This plot is for a single subject.

5.2.3 Data Preprocessing

We performed data cleaning and preprocessing in a manner similar to the one described in Section 4.2.2. We then performed time-frequency analysis on the raw data. The conversion of raw EEG signals to time-frequency domain improved the overall performance of the model. We obtained the spectograms of the signals in time frequency domain from all the four channels, this helped us to extract temporal and frequency precision of the raw signals. After the signal was

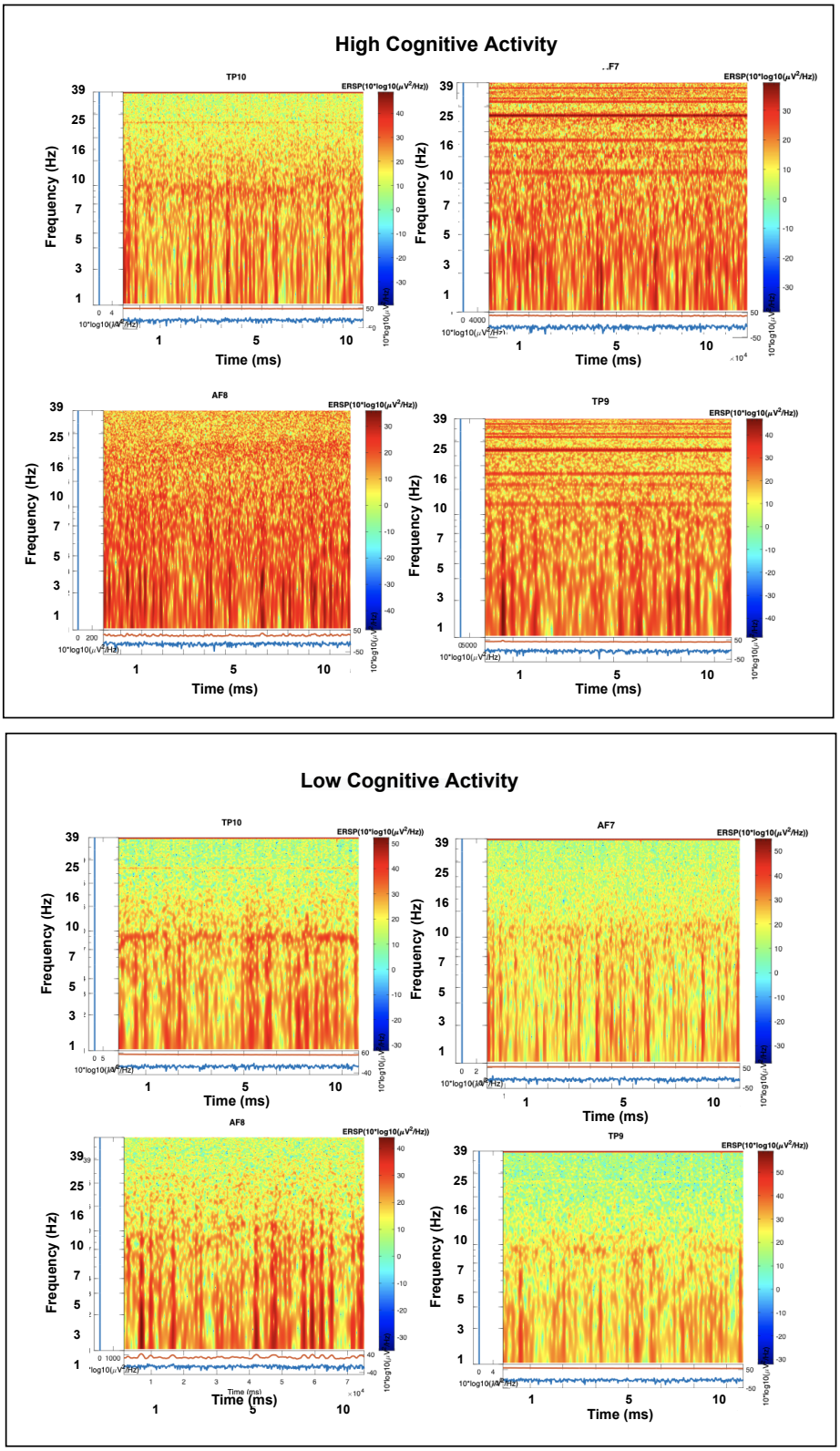


Figure 22: Spectrograms for High Cognitive and Low Cognitive activities for Subject 1 for the four channels TP9, AF7, AF8 and TP10.

converted to the desired form we divided the signal into overlapping windows (with overlap of 50%) of size 64. This was done for the signals from all the four channels. We explain in the results section the reason for taking the time window of 64 with a overlap of 50%. Since the spectrograms we got were of shape N$\times$64$\times$4, where N is the data size, after dividing the data into sliding window of 64 with half overlap, we obtained on the dimension M$\times$64$\times$64$\times$4, where M is the data size after windowing, and this was fed to the DL models. Since we inputted the signals from all the four channels altogether to our models we call our models Multi-Channel. In Figure 22 we show the spectrograms from the 4 channels for the two tasks. The spectrograms for high cognitive task show that the power spectral density (PSD) is of frequencies greater than 30 Hz are higher for high cognitive tasks and lower for low cognitive tasks, thus it shows that the PSD indicated in spectrogram form clearly shows the distinction in two classes. In Figure 23 we can see the spectral map of the various frequencies present in the raw data. Figure 23(a) shows the spectral maps for low cognitive task and Figure 23(b) shows the spectral maps for high cognitive task. In high cognitive spectral maps higher frequencies have more power spectral density which abide by our experiment and show that high frequencies are more prevalent in complex mental tasks.

5.3 Deep Learning Models

5.3.1 Architecture used for Att-MC-CNN

This model comprised of CNN layers to which the input of shape (Nx64x64x4), where N is the data size, was fed followed by a self-attention layer. The first layer was Conv2D layer that comprised of 32(5x5) filters and it was followed by a maxpool (2x2) and a batch normalization layer. The output of first CNN was fed to another Conv2D layer of similar specification and also followed by maxpool and batch normalization layer. The final output from the second CNN was fed to a self-attention layer. The output was then flattened and passed to dense layers of dimension with 1000 neurons with relu activation. This final output was passed to a softmax to

Algorithm 2: Generate 64 by 64 spectrograms from Raw EEG data.

Input: Raw data EEG data
Output: Matrix containing spectrograms of 64 by 64
1. Initialize the FFT parameters i.e. the minimum and maximum frequency(64), the time period of the wavelet which is equal to the sampling rate of the signal (256), the result matrix.
2. Find the FFT of the Raw data.
3. while *Frequency* $\leq$ *64* **do**
  4. Create a complex morlet wavelet from frequency by convolution of sine wave and gaussian.
  5. Find the FFT of the wavelet.
  6. Find the convolution of FFT of signal and FFT of wavelet by pointwise multiplication.
  7. Find inverse fourier transform of convoluted signal to convert back to time domain.
  8. Extract the magnitude of the complex signal and square it to get the absolute power component of the signal and add it to the result matrix.
end
9. Initialize a new matrix (final result) of size N x 64 x 64 x number of channels, where N is the total number of datapoints.
10. while *i* $\leq$ *len(result matrix)* **do**
  11. Divide the data in result matrix into overlapping windows of 64 and store in the matrix initialized in step 9 i.e. final result matrix
end
12. Return the final result matrix.

get the desired prediction.

5.3.2 Architecture used for Att-MC-BiLSTM

Three layers of Bidirectional LSTM in which the signal propagates backward as well as forward in time. The input of (Nx64x64x4), where N is the data size, shape was fed to first BiLSTM layer of 20 LSTM units, the output of this was fed to second BiLSTM layer of 20 LSTM units and finally the output of second layer was fed to final BiLSTM layer of similar configuration. This output form the stacked unit was passed to self-attention, after this a softmax was applied to get the classification result.

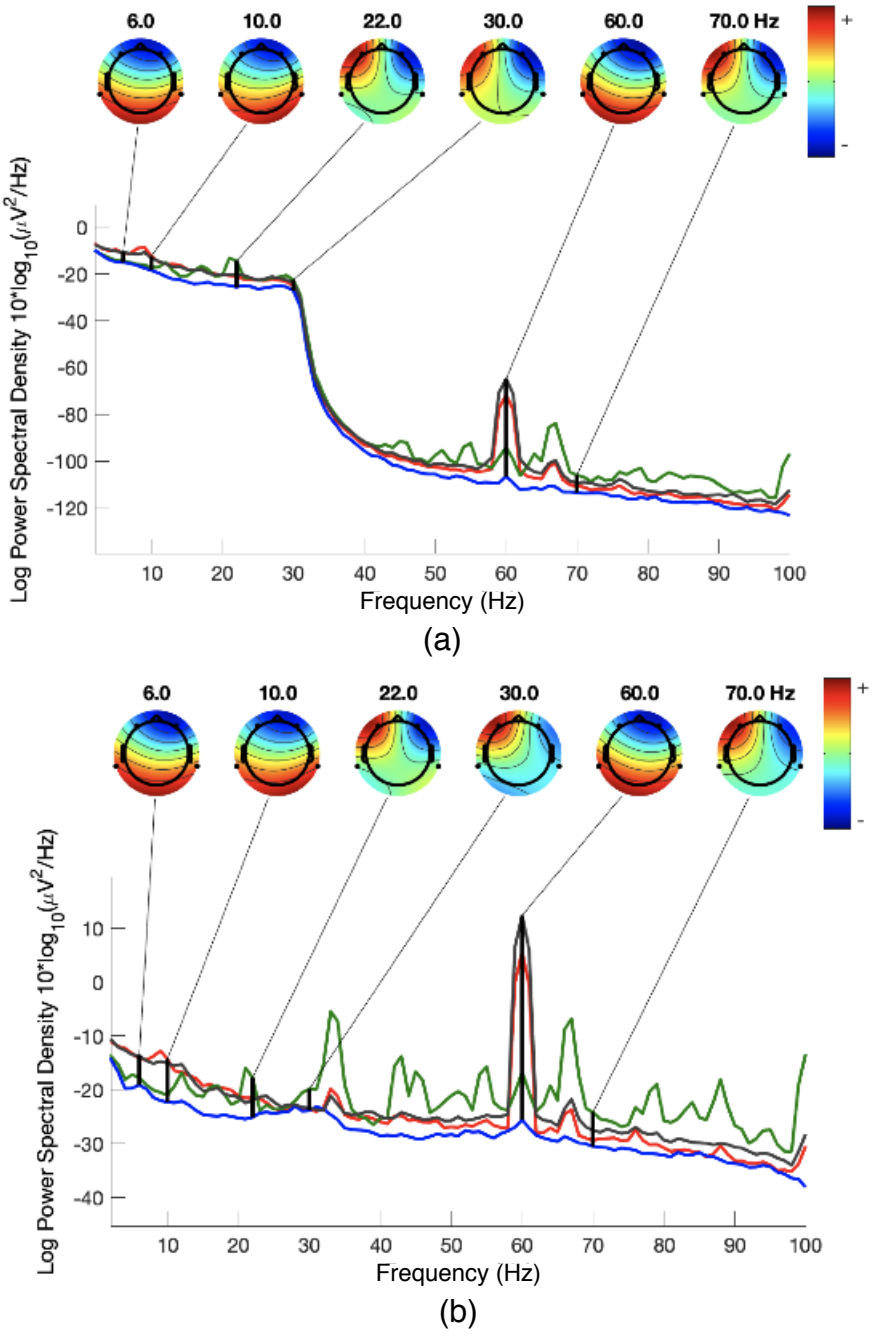


Figure 23: (a) Spectral maps for different frequencies for low cognitive tasks; (b) Spectral maps for different frequencies for high cognitive tasks

5.3.3 Architecture used for Att-MC-CNN-LSTM

In this model, we fed the input in form of 64x64 spectrogram-like matrix from all the four channels so the shape of our input is 64x64x4, where the last dimension denotes the number of channels. This input was fed to a Conv2D layer with 32(5x5) filters and the output of this layer was fed to another Conv2D layer with 32(3x3) filters. A dropout layer was then applied to avoid overfitting. The output of the dropout layer was then maxpooled with pooling filter of size 2x2 and then flattened. The flattened data was then sent to two LSTM layers, after which self-attention was applied to the encoded data from LSTM. The last layer was a Dense/fully connected (FC) layer. The output of FC was passed through softmax to get the prediction i.e. low-cognitive and high-cognitive. Batch Normalization was applied in training to normalize the outputs of the layer.

It has been seen that adding convolutional layers to capture local, temporal pattern on top of LSTM layers can be immensely helpful [35, 85]. The CNN LSTM architecture involves using CNN layers for feature extraction on input data combined with LSTMs to support sequence prediction.

5.3.4 Architecture used for Att-MC-CLSTM

In this model we fed the input (Nx64x64x4), where N is the data size, shape to a ConvLSTM2D layer that comprised of 32(2x2) filter and a relu activation. The output from ConvLSTM2D was then batch normalized and fed to another similar ConvLSTM2D layer and was also batch normalized. Self-attention layer was then applied to the data. A dropout layer was then applied and the data was then flattened and fed to a dense layer of 100 neurons. This data from dense layer was then passed through a softmax to get the output.

5.4 Transfer Learning

We used the technique of transfer learning to perform inter-subject classification. Transfer Learning is a machine learning method in which the model trained on one task can be reused for another similar task. In our approach, we used the best performing model (Att-MC-CLSTM) and added two dense layers of 32 and 16 neurons respectively and a softmax in the end and trained the last layers using the train data of a single subject (from a total of S subjects) that was not used in training, leaving the original model frozen (which was originally trained with data of S-1 subjects). To make a clear intuition of our methodology consider that we have data from S participants and each participant's data is x_i s.t. $i \in [1, S]$, then the training set A can be denoted as $A = \{x_i : \forall x_i \exists i = j : x_j \notin A\}$. A is used to train the initial model leaving data from one subject, x_j, for transfer learning and the weights are frozen. We then use the data of left out subject, x_j to train the newly added layers by using the frozen weights. Thus, we reused the weights from the training of other subjects' data and used them to retrain the modified model with extra layers using a single subject data that was not initially used for training. In Figure 24, we show the methodology we used for transfer learning. The use of transfer learning enabled us to reuse the DL model by only training the newly added layers to adjust the old weights according to the new data. Thus a model that has already been trained on the data of other participants can be utilised on new data of some new participant by only adding a few layers in the original model. This saves time for training the complete model, promotes reusability of the trained model on unseen data, and it can be helpful to classify data of a new participant in case when there is no or less data for that participant. We initially used old model without adding the extra layers and evaluated it using leave one subject out cross validation (cv) however that model did not perform well and therefore we added some layers in the trained model to retrain on new subject's data so as to fine tune the weights according to the new subject using Transfer Learning. We explain the procedure in Algorithm 3.

Algorithm 3: Transfer Learning of Att-MC-CLSTM

1. Save the weights of the best performing model (Att-MC-CLSTM) for S-1 participants.
2. Add two dense layers of 32 and 16 neurons respectively and a softmax in the end of the model.
3. Freeze the initial layers of the saved model.
4. Retrain the modified model with K fold cross validation with extra layers on the data of left out participant by initializing the weights with the saved weights.

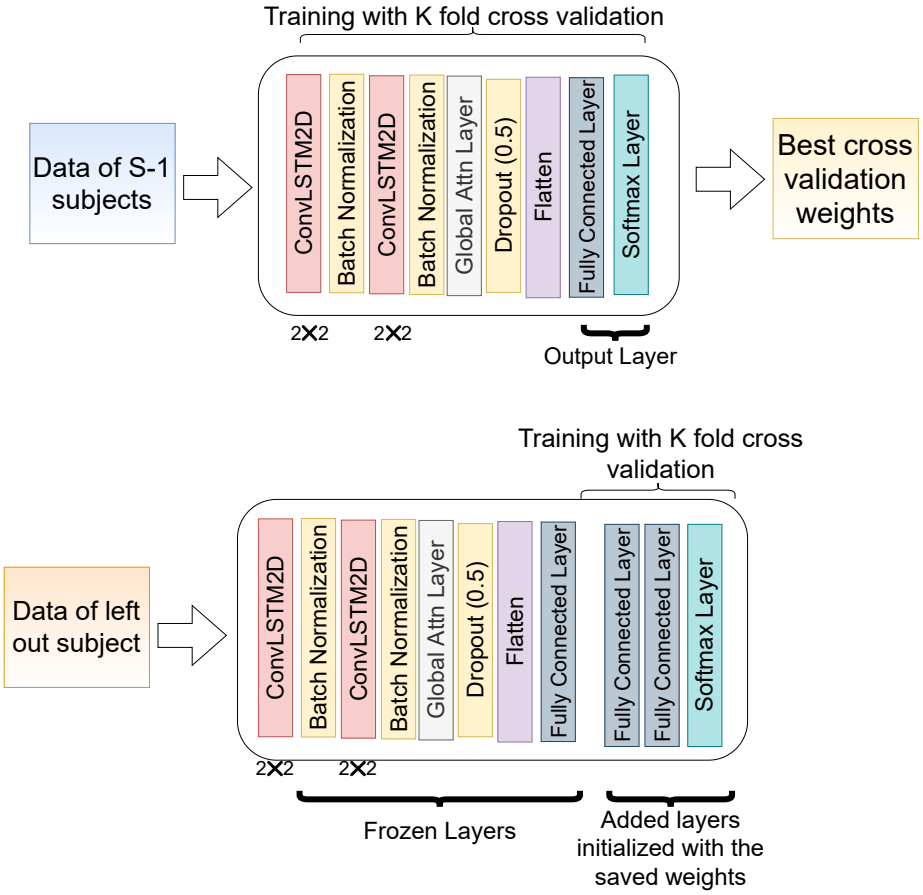


Figure 24: The training methodology used for transfer learning for inter-subject classification. The original model is trained with the data from all subjects except one and the best cross validation weights are saved. Two additional dense layers are added to the model and the original layers are frozen; the newly added layers are initialized with the best cross validation weights and are trained on the data of one remaining subject.

5.5 Evaluation Metrics for binary classification

The evaluation metrics used to measure the performance of our model were mainly four, namely, Accuracy, Precision, Recall and F1 score.

The accuracy indicates the samples that were correctly classified from all the samples.

$$\text{Accuracy}=\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{5.1}$$

where TP = True Positives (low cognitive classified as low cognitive) , TN = True Negatives (high cognitive classified as high cognitive), FP = False Positives (low cognitive classified as high cognitive), and FN = False Negatives (high cognitive classified as low cognitive).A TP is an outcome where the model correctly predicts the positive class. Similarly, a TN is an outcome where the model correctly predicts the negative class. A FP is an outcome where the model incorrectly predicts the positive class. And a FN is an outcome where the model incorrectly predicts the negative class.

Precision expresses the proportion of the data points our model says was relevant actually were relevant.

$$\text{Precision}=\frac{\text{TP}}{\text{TP+FP}} \tag{5.2}$$

Recall is the model's ability to find all the data points of interest in a dataset. It is also called sensitivity or true positive rate (TPR).

$$\text{Recall}=\frac{\text{TP}}{\text{TP+FN}} \tag{5.3}$$

The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation.

$$\text{F1 Score}=2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}} \tag{5.4}$$

5.6 Experimental Results

We have applied four time-series classification models to perform binary classification task on EEG data. We classified data into low-cognitive and high-cognitive. The raw EEG data was preprocessed and spectrograms were obtained by performing continuous wavelet transform. The data after time-frequency analysis was divided into overlapping windows of size 64 with an overlap of 50%. Thus the data shape we got after wavelet transform was (Nx64x4), where N represents the size of data, and after applying sliding window the final input shape obtained was (Mx64x64x4), where M represents the data size after dividing the data into overlapping windows of 64, and this was fed to the models for classification. The timewindow of 64 was chosen after experiment with different window sizes (32,64,128,256) and an overlap of 50% gave the best results when compared to other overlap of 0%, 25% and 75%. In our results we report four metrics accuracy (acc), precision (pre), recall (TPR) and F1-score (F1).

Table 5.1: Performance of DL models on train data for intra-subject classification.

Data Set	**Att-MC-CNN**				**Att-MC-BiLSTM**			
	Acc	Pre	TPR	F1	Acc	Pre	TPR	F1
Best Subject	94.2%	0.940	0.939	0.939	92.5%	0.921	0.923	0.922
Avg Subject	89.3%	0.892	0.894	0.893	86.5%	0.863	0.864	0.863
Data Set	**Att-MC-CNN-LSTM**				**Att-MC-CLSTM**			
	Acc	Pre	TPR	F1	Acc	Pre	TPR	F1
Best Subject	96.3%	0.963	0.961	0.962	**98.1**%	0.979	0.982	0.980
Avg Subject	94.4%	0.941	0.946	0.943	**95.8**%	0.955	0.954	0.954

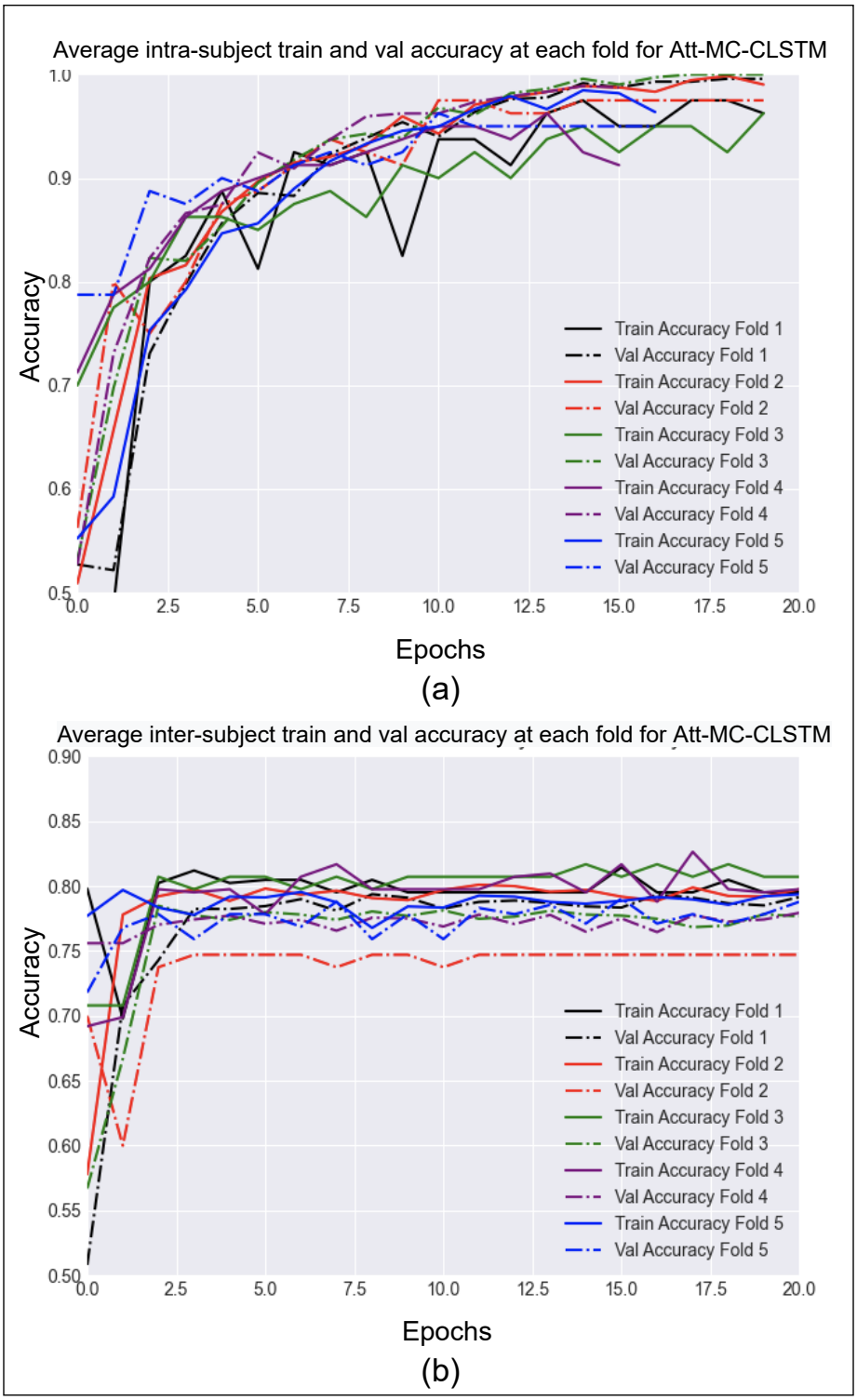


Figure 25: (a)The average accuracy of Att-MC-CLSTM during training phase on training and validation data for intra-subject classification;(b)The average accuracy of transfer learning on Att-MC-CLSTM during training phase on training and validation data for inter-subject classification

Table 5.2: Performance of DL models on test data for intra-subject classification.

Data Set	**Att-MC-CNN**				**Att-MC-BiLSTM**			
	Acc	Pre	TPR	F1	Acc	Pre	TPR	F1
Best Subject	93.2%	0.931	0.932	0.931	90.8%	0.903	0.907	0.905
Avg Subject	87.8%	0.863	0.860	0.861	83.6%	0.843	0.826	0.820
Data Set	**Att-MC-CNN-LSTM**				**Att-MC-CLSTM**			
	Acc	Pre	TPR	F1	Acc	Pre	TPR	F1
Best Subject	94.3%	0.955	0.943	0.949	**96.1**%	0.961	0.959	0.960
Avg Subject	92.4%	0.927	0.918	0.922	**93.8**%	0.936	0.939	0.937

5.6.1 Training Procedure for Intra-subject Classification

The data of each subject was divided into train and test where 30% of data was kept for testing. The training was done using 5 fold cross validation on each subject's data seperately. We trained the the data using the training data and validated it with the validation data with 20 epochs for each fold. During the training we saved the weights for the best epoch for each fold. The final train accuracy was calculated by evaluating the best performing weights over the complete training set. We also used Early Stopping to end the training for each fold if the accuracy did not improve upto 5 epochs. This made the training procedure efficient. In Table 5.2 , we report the best subject train metrics achieved for intra-subject classification,where we provide the highest performance metric from among all the 12 subjects and we also provide the average subject performance metric where we give a mean for all the subjects, this helps to understand how the models perform for all subjects and in Table 5.1 we provide the test metrics.

5.6.2 Training Procedure for Inter-subject Classification using Transfer Learning

In order to perform inter-subject classification, we used Transfer Learning technique. We performed inter-subject classification by using the best performing model i.e. Att-MC-CLSTM. Firstly we trained the model using the same procedure as we described in intra-subject classification, i.e. we combined the data of all the subjects except one (which was kept aside for retraining and testing) to form the training data, with which we performed 5 fold cross validation training, each fold having 20 epochs. We saved the best weights while the training. The one subject's data we left we used it for transfer learning. We froze the all layers of the model and assigned them the best weights that we saved by training procedure from the combined data set. We added two dense layers to the model and retrained the model with one subject's data we left initially using the best performing weights. The training was done by dividing the one subject's data into train and test with 30% test data. We trained the new combined model using the the train data of one subject using 5 fold cross validation. After the training was done we calculated the train accuracy using the whole train data and the test accuracy using the test data. We repeated this for all the subjects.

Att-MC-CNN-LSTM and Att-MC-CLSTM performed relatively well on the data. The possible reason could be that had CNN and LSTM models integrated into the architecture that accounted for the spatial and temporal dependencies of the data.The highest test accuracy of 96.1% was achieved by Att-MC-CLSTM for intra-subject and the highest average test accuracy for intra-subject was also achieved by Att-MC-CLSTM which was 93.8%, this shows that the model performed relatively well for all the subjects.

The average inter-subject test accuracy of 86.3% was obtained using Transfer Learning on Att-MC-CLSTM. This shows that with little training on some data the model could be used by other people. Our proposed methodology has shown improvement over the past approaches [31, 99] that have worked on similar problems using wearable EEG by 10% from the highest achieved accuracy and we have achieved comparable performance to the works that have used complex

Table 5.3: Average performance metics of Att-MC-CLSTM with Transfer Learning for inter-subject classification.

Data Set	**Att-MC-CLSTM**			
	Acc	Pre	TPR	F1
Train Data	87.5%	0.877	0.874	0.875
Test Data	86.3%	0.863	0.860	0.861

medical grade EEG. The results achieved show that our model is promising for this type of classification on EEG data. Our results are comparable to the results obtained by medical grade EEG devices. The results for inter-subject can be seen in Table 5.3.

In Table 5.4 we compare our results with previous approaches.

5.7 Summary

We used the data from a CMI task to classify cognitive load using DL. The proposed architecture used a novel technique to classify data based on multi-channel approach where the input was in form of spectrogram stack from all the channels of Muse 2. We compared state-of-the-art time-series classification DL models on the basis of their performance on our data and achieved the highest test accuracy of 96.1% for intra-subject classification using attention multi-channel ConvLSTM. Our model surpassed the performance of previously proposed models and hence showed promise in reliable detection of cognitive load.

Our proposed model obtained the highest inter-subject test accuracy of 86.3% for inter-subject classification using the concept of transfer learning on attention multi-channel ConvLSTM. Our study proposed the use of transfer learning in EEG cognitive classification problem. Current work has shown improvement over the past approaches that have worked on similar problems using wearable EEG by 9% from the highest achieved accuracy and we achieved comparable performance to the works that have used complex medical grade EEG .

Table 5.4: Comparison with past approaches

Study	Device	Electrodes	Model	Accuracy	Validation Approach
Bird et al. [45]	Muse	4	Random Forest	87.16%	Intra-subject (10-fold cv)
Appriou et al. [100]	Medical-grade EEG	28	CNN	72.7%	Intra-subject
Pouya et al. [35]	Muse	4	SVM	75%	Inter-subject (leave-one-subject-out)
Xiong et al. [36]	Medical-grade EEG	128	SVM	96.3%	Inter-subject (leave-one-subject-out)
Liang et al. [37]	Not Known	6	ELM	86.7%	-
Kuanar et al. [38]	Neurofax EEG-1200	64	ELM	92.5%	Inter-subject (leave-one-subject-out))
Liu et al. [10]	Not known	64	CNN	93%	-
Saha et al. [40]	Medical-grade EEG	64	SDAE+MLP	89.51%	Inter-subject (MonteCarlo cv with four splits)
Wang et al. [101]	Emotiv EEG	14	PCA,SVM	97.14%	Intra-subject (4-fold cv)
Chakladar et al. [102]	-	14	BLSTM-LSTM	86.33%	-
Dimitrakopoulos et al. [86]	ANT waveguard system	64	SVM	86%	Inter-subject (leave-one-subject-out)
Plechawska et al. [103]	Mitsar EEG 201	21	SVM	91.50%	Inter-subject (5 fold cv)
Ours	Muse	4	Att-MC-CLSTM	**98.1%**	Intra-subject train
				96.1%	Intra-subject test
				87.5%	Inter-subject train
				86.3%	Inter-subject test

Identifying cognitive load effectively can help one to design smart, adaptive, personalized and intelligent information systems and thus in this work we have devised a DL approach to classify cognitive load using EEG data from a wearable headband from a CMI task. We have applied four models to perform binary classification task to classify data into low cognitive and high cognitive. We converted the raw data into spectrograms and then trained the models on this data. We have also proposed using multi-channel models that accept the data from all the channels of the EEG at the same time. The addition of self-attention in our models made the classification more efficient. Out of the four models two models outperformed the other two Att-MC-CNN-LSTM and Att-MC-CLSTM gave the best results with the highest test accuarcy of 96.1%. This result is very promising considering we are using a 4-channel EEG headband.

In future work, an integrated IoT application can be designed using our proposed models. There are several other wearable devices available in the market like Muse S, Emotiv, Neurosky, Starlab etc, one can apply our methodology using data from these headbands also. The performance of the models can be checked for improvement by applying more preprocessing on the data and using representations other than spectrograms.

Chapter 6

Sabotage detection using Deep Learning models on EEG data from cognitive-motor integration task

6.1 Introduction

Sabotage detection has many clinical and industrial applications especially where one needs to detect willful deceit. In this work, we have proposed a novel method to detect performance sabotage using a deep learning approach on the electroencephalogram (EEG) signals from a wearable device. The EEG signals from a four-channel headband were acquired from a cognitive-motor integration (CMI) task. Each participant completed sabotage and no-sabotage conditions in random order. A multi-channel CNN-LSTM (convolutional neural network with long short term

memory) model with self-attention has been used to perform the time-series classification into sabotage and no-sabotage, by transforming the time series into 2D image-based spectrogram representations. This approach allows the inspection of frequency-based, as well as temporal features of EEG and the use multi-channel model facilitates in capturing correlation and causality between different EEG channels. By treating the 2D spectrogram as an image, we show that the trained CNN-LSTM classifier based on automated visual analysis can achieve high levels of discrimination and an overall accuracy of 98.71%, as well as low false-positive rates. We also compare the spectrogram-based results with the results that we obtained by using raw timeseries. Our proposed methodology has outperformed previous studies that have been done on deceit detection. Our method can be applied in clinical applications such as baseline testing, assessing current state of injury and recovery tracking as well as industrial applications like monitoring performance deterioration in workplaces.

6.2 Participants

Electroencephalography (EEG) and behavioural data were collected from 12 healthy volunteers (8 female, 4 male) aged 22-50 years of age. Participants had no history of substance abuse, neurological illness or impairment, brain injury, psychoactive drug treatment, or concussion. All procedures were approved by York University's Human Participants Research Committee, and all participants provided informed consent to participate.

6.3 Experimental Task

The experimental task for similar to the one described in Section 5.2.1 with the difference being that for each condition, they were instructed to either complete the task as quickly and as accurately as possible (true effort condition), or to willfully perform poorly while still completing the trials (sabotage condition). In summary, participants completed two sets (true effort condition, sabotage condition) of the two tasks (standard task, CMI task), for a total of 80 trials. The entire

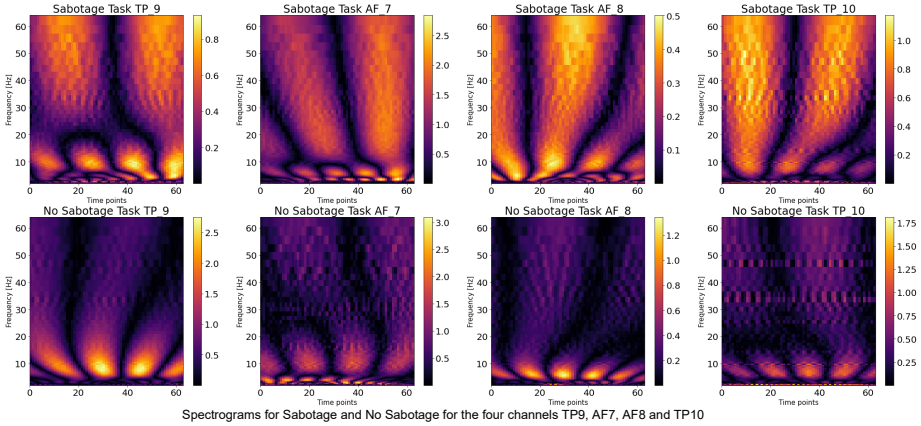


Figure 26: Spectrograms for Sabotage and No Sabotage activities for Subject 1 for the four channels TP9, AF7, AF8 and TP10

behavioural task took approximately 10 minutes (2-3 minutes for each 20-trial individual task). Both the conditions and the standard vs. CMI task within a condition were randomized for each participant.

6.4 Data Preprocessing and Feature Engineering

The steps similar to the ones described in Section 5.2.3 were followed and spectrograms of $64 \times 64 \times 4$ were obtained. The process is shown in Figure 27.

6.5 Deep Learning Approach

In our methodology, we used four time-series classification models namely, Attention Multi-Channel Convolutional Neural Network(CNN), Attention Multi-Channel Convolutional Neural Network with Long Short Term Memory (CNN-LSTM), Attention Multi-Channel ConvLSTM and Stacked Bidirectional LSTM on our data, however, we only discuss the results of Self-Attention based Multi-Channel Convolutional Neural Network- Long Short Term Memory as we got the best results using this model. We call this model Multi-Channel because we sent the input in form of spectrograms from all the four channels at the same time to the model.

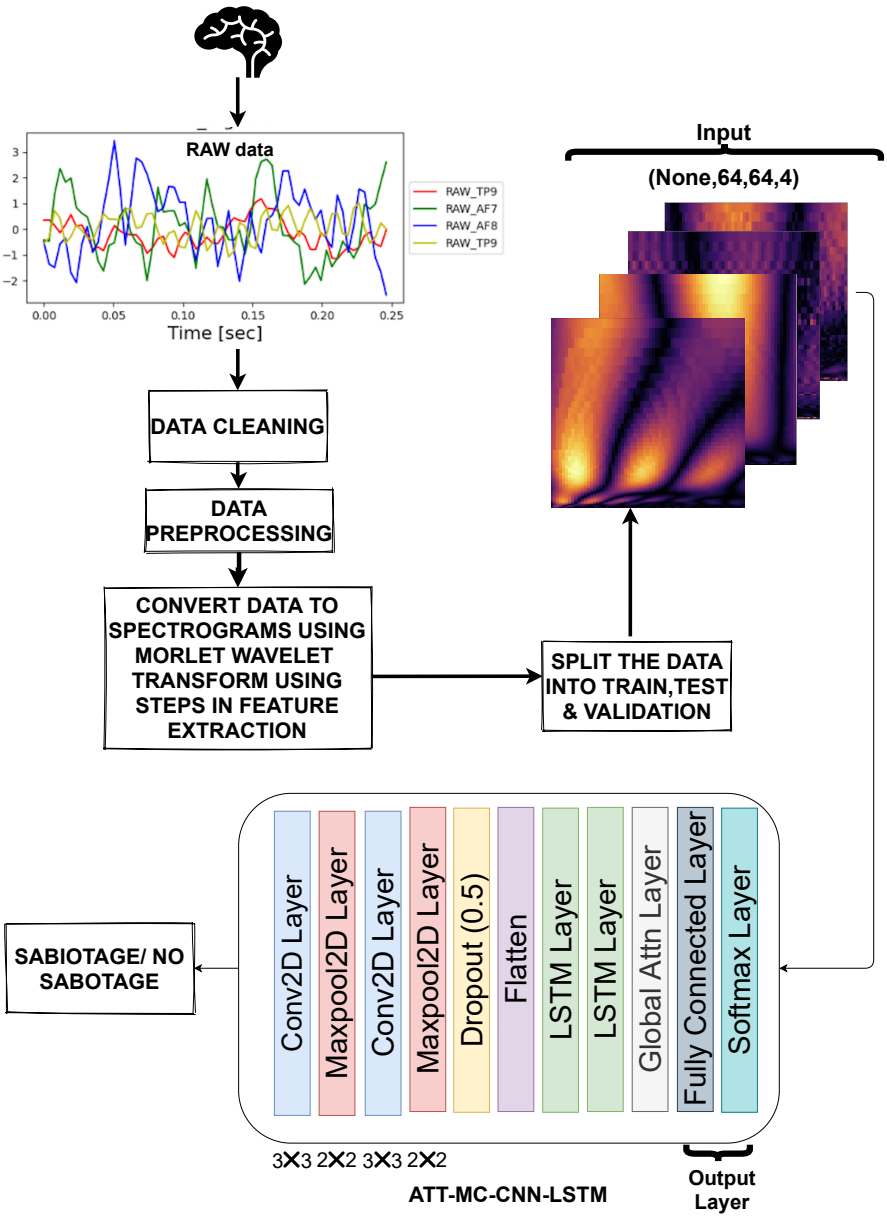


Figure 27: The methodology used in our approach for Sabotage detection.

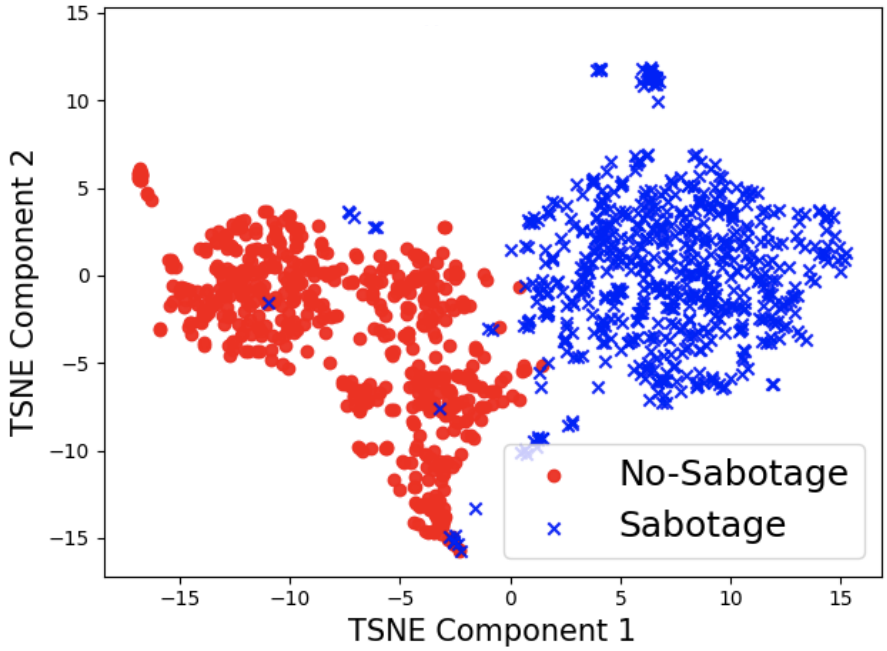


Figure 28: TSNE visualization of data for Sabotage detection.

6.5.1 Architecture used for Att-MC-CNN-LSTM

In our proposed model, the input took the form of 64×64 spectrogram-like matrix from each of the four channels. The overall shape of input was 64×64×4, where the last dimension denotes the number of channels. This input was fed to a Conv2D layer with 32(5×5) filters followed by a maxpool layer of 2×2 and the output of this layer was fed to another Conv2D layer with 32(3x3) filters followed by a maxpool again. A dropout layer was then applied to avoid overfitting. The output of the dropout layer and then flattened. The flattened data was then sent to two LSTM layers, after which self-attention was applied to the encoded data from LSTM. The last layer was a dense/fully connected (FC) layer. The output of FC was passed through softmax layer to get the prediction i.e. sabotage or no-sabotage. Batch Normalization was applied in training to normalize the outputs of the layer.

Although, CNN and LSTM effectively capture spatio-temporal information, there is a need to target specific information from the embeddings generated by the combination of CNN and LSTM

and bring them together since multiple components can together form relevant semantics for decoding the activity being performed. This has been done using the self-attention mechanism that forms a 2-dimenstional matrix to represent the embedding of the input, such that each row of the matrix caters to a different part of the time-series. Along with CNN and LSTM, we show that self-attention leads to a statistically significant results.

6.5.2 Transfer Learning for Sabotage Detection

We applied Transfer learning in case of Sabotage detection too, we used the same methodology used in Section 5.4, the only difference being that we added only two additional layers to the transfer learning model one fully connected layer and one softmax layer. The use of two FC layers did not show much improvement when compared to one FC layer and therefore we went ahead with single FC layer as this saved the retraining time of the new layers.

6.6 Experimental Results

In order to analyse the benefit of time-frequency analysis on classification result on our data we applied our proposed model both on raw data with time-frequency analysis i.e. on raw data spectrograms as well as on raw data without time-frequency analysis. We discuss the result on both of type of data in this section. As we earlier mentioned, that we divided the data into overlapping windows of 64 data points with different values of overlap. We tried different window sizes 32,64,128 and 256 and different overlap values (no-overlap, 25%, 50% and 75%) for the both the types of data i.e. with and without time-frequency analysis. We got the best average accuracy for a window of 64 with 50% overlap so we went ahead with this time-window for further experimentation.

We divided the data into training data and test data. The train and test splits were in the ratio 70% and 30% respectively. The data was trained using 5-fold cross validation on the training data with 20 epochs for each fold and after each epoch the model was cross-validated on the validation

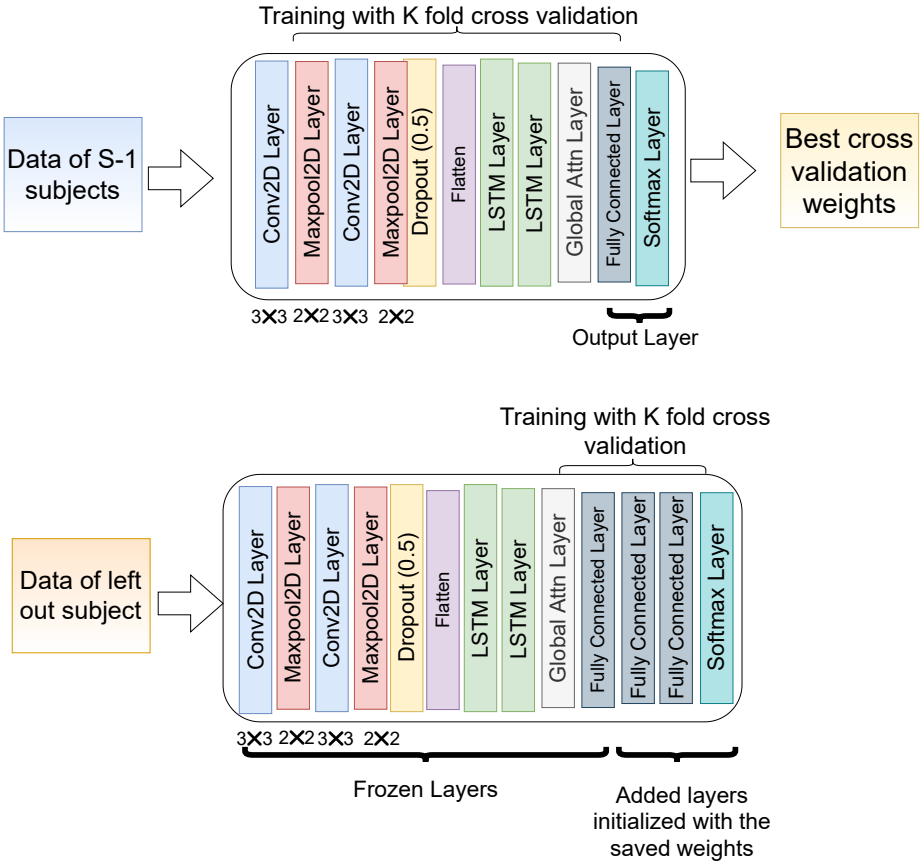


Figure 29: The methodology used for transfer learning for inter-subject classification for sabotage detection.

Table 6.1: Performance of the model on raw data

Subject	Accuracy	Precision	Recall	F1-Score
Subject 1	92.05	0.934	0.909	0.919
Subject 2	94.05	0.921	0.941	0.931
Subject 3	88.12	0.914	0.791	0.848
Subject 4	85.56	0.923	0.779	0.853
Subject 5	94.43	0.961	0.923	0.939
Subject 6	91.36	0.924	0.908	0.915
Subject 7	91.66	0.952	0.871	0.909
Subject 8	87.83	0.879	0.866	0.872
Subject 9	88.78	0.897	0.882	0.893
Subject 10	80.76	0.729	1.0	0.843
Subject 11	86.67	0.872	0.859	0.865
Subject 12	85.85	0.808	0.989	0.889
`Average Intra-subject`	89.19	0.899	0.876	0.877

data. The metrics in the results are reported on the test data. We performed classification on individual participant data that we called intra-subject classification, as well as we combined the data from all the participants and performed classification and we called it inter-subject classification. In case of inter-subject classification we performed 5-fold leave one subject out cross-validation i.e. we trained the model by combining data from all participants except one and used this data for testing. We can see the performance with different timewindows and overlap in Figure 30.

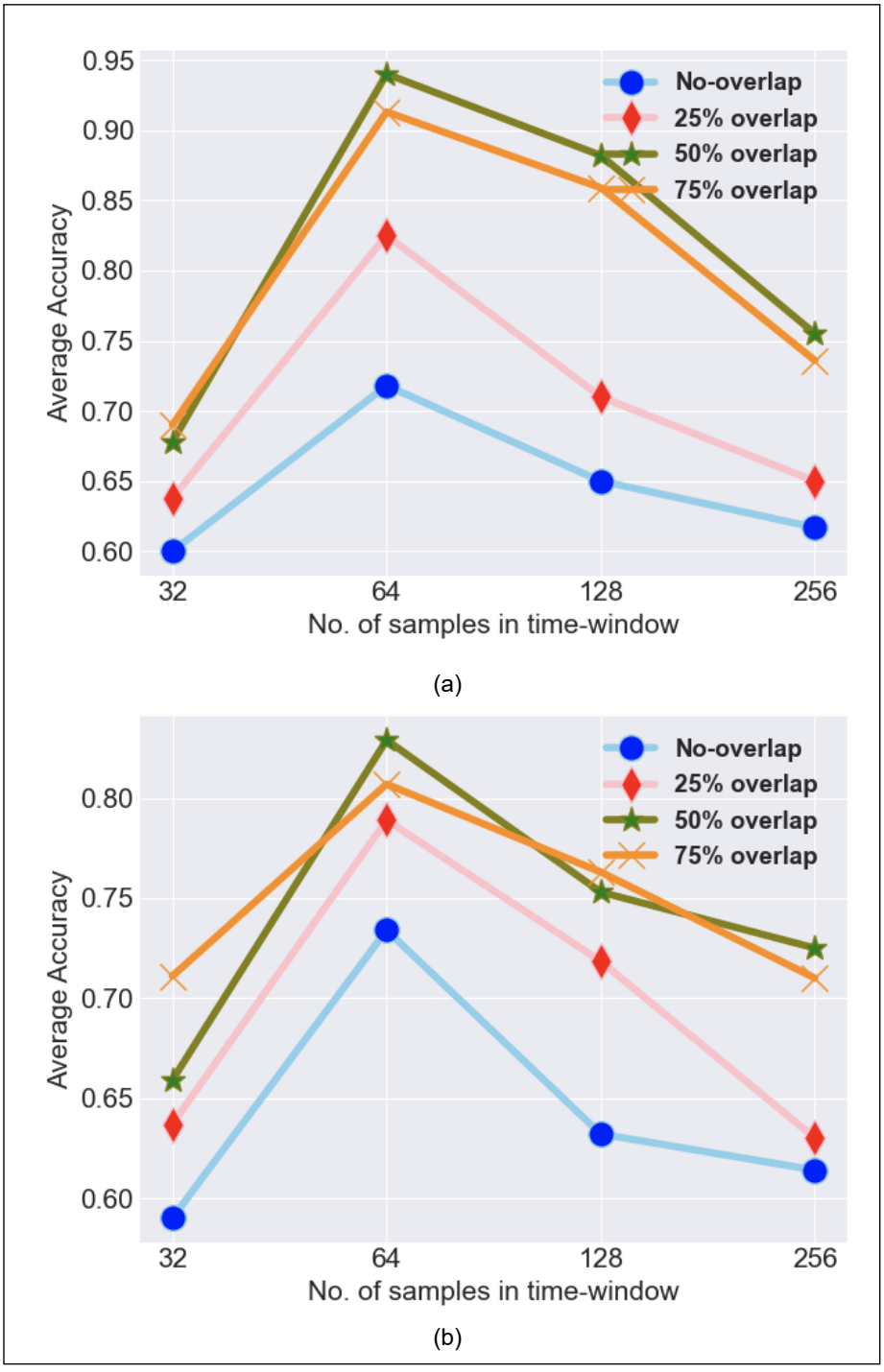


Figure 30: (a) The average accuracy for different timewindows and overlap values for spectrogram data (b) The average accuracy for different timewindows and overlap values for raw data

Table 6.2: Performance of the model on spectrograms of raw data.

Subject	Accuracy	Precision	Recall	F1-Score
Subject 1	96.80	0.975	0.961	0.968
Subject 2	98.71	1.0	0.975	0.987
Subject 3	93.23	0.959	0.870	0.912
Subject 4	89.10	0.967	0.848	0.903
Subject 5	96.01	0.977	0.942	0.959
Subject 6	93.57	0.949	0.925	0.936
Subject 7	93.15	0.971	0.884	0.925
Subject 8	92.94	0.915	0.935	0.925
Subject 9	83.89	0.849	0.833	0.841
Subject 10	95.27	0.915	0.980	0.946
Subject 11	90.23	0.899	0.902	0.901
Subject 12	93.10	0.895	0.992	0.9414
`Average Intra-subject`	92.98	0.943	0.914	0.927

Table 6.3: Performance of Att-MC-CNN-LSTM on Inter-subject with Transfer Learning.

Data Set	**Att-MC-CNN-LSTM**			
	Accuracy	Precision	Recall	F1-Score
Train	82.6%	0.826	0.813	0.819
Test	80.3%	0.803	0.800	0.801

Table 6.1 shows the performance of our model on raw EEG data i.e. data without spectrogram representation. The highest intra-subject accuracy we got with this data set was 94.43% and the minimum intra-subject accuracy was 85.56%. The average intra-subject accuracy was 89.19%, this shows that the model performed relatively well for all the subjects. The inter-subject accuracy obtained was 72.50%. The inter-subject classification was done to see the generalized performance of the model.

We then applied our model on the spectrogram representation of the raw data from the four channels. The results improved on this data set and are shown in Table 6.2. We got the highest

intra-subject accuracy of 98.71% and the lowest intra-subject accuracy of 89.10%. The average accuracy of all subjects was 92.98% which shows that the performance of the model improved the average accuracy for all subjects. The inter-subject accuracy also improved and became 76.15%. The results were relatively very good using spectrograms and the performance of the model improved on the spectrograms. Our proposed model is spectrograms with Self-Attention based Multi-Channel Convolutional Neural Network- Long Short Term Memory. We also performed Transfer Learning using the methodology and we report the results on test and train data in Table 6.3.

6.7 Summary

The primary aim of this study was to determine if performance sabotage could be detected by analyzing EEG data collected during a visuomotor skill assessment. Using a multi-channel deep learning approach, we found that analysis of EEG spectral data enabled us to differentiate intentionally poor task performance from maximal effort performance with 98.71% accuracy. The present results revealed that the time-frequency representation as the input to the model led to better results than the use of raw data. For future work we recommend applying different feature extraction techniques on the data or using topographical maps as input to the model to further improve classification accuracy.

The visuomotor skill assessment used in the present study has shown promise in as an objective tool to determine if a concussion has been sustained, as well as whether full recovery has occurred. Used as a baseline test, individuals at known risk of concussion would complete this assessment when they are known to be concussion-free and then again once an injury is suspected. However, this protocol is only useful if both tests reflect an individual's best effort, and the threat of litigation or a desire to resume or avoid certain activities leads some individuals to sabotage their test results, under-performing on a pre-injury test to enable earlier return to activity, or under-performing on a post-injury test to delay a diagnosis of recovery. The addition of EEG spectral analysis to the visuomotor task used in this study ensures that an individual's best

effort is captured during the assessment, and improves the clinical utility of the results.

A strength of this analysis was that it used technology consumer-grade EEG technology (MUSE 2™ EEG headband), along with a tablet-based task to assess cognitive-motor integration. The technology is portable and affordable; as a result, this is a protocol that can be easily translated to clinical patient care settings. With the advancement in wearable computing, more portable EEG devices are available and future studies replicating these findings with such devices would be useful to further expand the utility of this form of performance monitoring.

This proof-of-concept study showed that a multi-channel CNN-LSTM model can be used on EEG data to detect sabotage at an average approximately 93% of the time during the performance of a cognitive-motor integration task. Expanding our data collection to a larger sample size will allow for further refinement of the model, as well as the development of a user interface and standard operating procedure to translate this work into clinical practice. Future work will also focus on applying this sabotage detection process to a range of brain-computer interfaces for real-time, in-situ deceit detection. Such an application would increase the utility of both injury diagnosis and recover assessment using behavioural performance measures.

Chapter 7

Conclusion and Future work

In this chapter we summarise our work and present the concluding remarks. We also discuss the future work.

7.1 Conclusion

The work in the thesis can be summarised as follows:

- We proposed a novel ML-IoT pipeline software architecture that encompasses the essential components from data ingestion to runtime use of ML models. The proposed pipeline is highly maintainable, testable and independently deployable as opposed to the monolithic machine learning archietcures where the same script is used to extract the data, clean and prepare it, model it, and deploy model trained on it. Our architecture divides ML pipeline into components that work independently across edge-core for training and inference.
- The architecture has been validated using three BCI applications. We have used EEG signals from a wearable consumer-grade EEG headband to acquire the raw EEG data and then use this data to design IoT applications. These applications can be used in a

variety of scenarios such as recognizing color stimuli from brain waves which can be used in developing techniques for cognitive tasks using color cues such as controlling IoT devices by looking at primary colors for individuals with restricted motor abilities, detecting performance sabotage to cater various clinical maneuvers like baseline testing and classifying cognitive load for monitoring behavioural performance.

- We have achieved state-of-the-art results using our methodology for each application and our results have significant improvement over the results achieved by previous approaches. Our applications are novel in the terms of technique we used to preprocess and engineer the data and the multi-channel based approach we have used in the cognition and sabotage application. Also, the use of transfer learning in cognitive study has not been done in any previous study to classify inter-subject data. We propose to use transfer learning to analyse the generalizability of our models.

- The work on color classification using EEG signals has mainly focussed on on Alpha and Beta frequencies. In order to classify EEG data to RGB we extracted various spectral, correlation and statistical features from the data and applied ML models to it. Our proposed model of Random Forest with forward feature selection showed significant improvement when compared to previous approaches. Our methodology achieved an improvement of almost 20% in the average accuracy of classification. The intra-subject classification accuracy of 80.6% shows that wearable devices can be used in integrated IoT frameworks where they can be used in various control applications.

- We also proposed application to classify cognitive load using DL models. This application can help one to design smart, adaptive, personalized and intelligent information systems. We converted the raw data into spectrograms and then trained the models on this data. We have also proposed using multi-channel models that accept the data from all the channels of the EEG at the same time. The addition of self-attention in our models made the classification more efficient. Out of the four models two models outperformed the other two Att-MC-CNN-LSTM and Att-MC-CLSTM gave the best results with the highest test

accuarcy of 96.1%. This result is very promising considering we are using a 4-channel EEG headband.

- A third application that is proposed in this work analyses rich data sets for the purpose of detecting behavioural performance sabotage. Our results supported our hypothesis that such deep learning networks were capable of detecting significant neural activity differences as a function 'true effort' versus 'intentionally poor effort'. Using a multi-channel deep learning approach, we found that analysis of EEG spectral data enabled us to differentiate intentionally poor task performance from maximal effort performance with 98.71% accuracy. The visuomotor skill assessment used in the present study has shown promise in as an objective tool to determine if a concussion has been sustained, as well as whether full recovery has occurred. Used as a baseline test, individuals at known risk of concussion would complete this assessment when they are known to be concussion-free and then again once an injury is suspected. A strength of this analysis was that it used technology consumer-grade EEG technology (MUSE 2 EEG headband), along with a tablet-based task to assess cognitive-motor integration. The technology is portable and affordable; as a result, this is a protocol that can be easily translated to clinical patient care settings.
- In this work, we propose an architecture that can accept sensor data from IoT infrastructure, and, automatically address tasks like data ingestion, data cleaning, preprocessing, modeling and deployment in form of computation units that are easy to replace such that it is possible to rework them independently without changing the rest of the system to ensure better implementation.

7.2 Future work

In future work, we plan to design an architecture to dynamically distribute the ML computing and storage across an IoT infrastructure. We plan to propose the placement of pipeline components

and the dynamic adaptation of the edge core pipeline based on various optimization options like CPU utilization, response time, memory usage, etc, and set up the parameters like response time threshold, memory utilization threshold and using these the module would dynamically distribute the architecture at run time to ensure that the optimization standards are met.

We would try to emulate a real situation, similar to the static one and report the performance of adaptation and the improvement it achieves based on the various adaptation settings. A dedicated module called 'Adaptive Manager' could be designed to take care of the performance of the architecture.

In case of BCI applications, with the advancement in wearable computing, more portable EEG devices are available and future studies replicating these findings with such devices would be useful to further expand the utility of this form of performance monitoring. There are several other wearable devices available in the market like Muse S, Emotiv, Neurosky, Starlab etc, one can apply our methodology using data from these headbands also. The performance of the models can be checked for improvement by applying more preprocessing on the data and using representations other than spectrograms. Future work will also focus on applying the cognitive process to a range of brain-computer interfaces for real-time, in-situ deceit detection. Such an application would increase the utility of both injury diagnosis and recover assessment using behavioural performance measures.

Bibliography

[1] S. K. Sharma and X. Wang, "Live data analytics with collaborative edge and cloud processing in wireless iot networks," *IEEE Access*, vol. 5, pp. 4621–4635, 2017.

[2] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A knowledge-based approach for real-time iot data stream annotation and processing," in *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, 2014, pp. 215–222.

[3] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive internet of things: A new paradigm beyond connection," *Internet of Things Journal, IEEE*, vol. 1, 03 2014.

[4] G. Ian, B. Yoshua, and C. Aaron, "Deep learning," *The MIT Press*.

[5] J. Wolpaw, N. Birbaumer, W. Heetderks, D. Mcfarland, P. Peckham, G. Schalk, E. Donchin, L. Quatrano, C. Robinson, and T. Vaughan, "Brain-computer interface technology: A review of the first international meeting," *IEEE transactions on rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 8, pp. 164–73, 07 2000.

[6] S. Åsly, L. Moctezuma, M. Gilde, and M. Molinas, "Towards eeg based classification of rgb color-based stimuli," 09 2019.

[7] S. Rasheed and D. Marini, "Classification of eeg signals produced by rgb colour stimuli," *Journal of Biomedical Engineering and Medical Imaging*, 10 2015.

[8] J. Dawes, M. Msc, and T. Spiteri, "Sports and exercise medicine relationship between pre-season testing performance and playing time among ncaa dii basketball players corresponding author article history citation," vol. 2, pp. 47–54, 10 2016.

[9] K. Dossa, G. Cashman, S. Howitt, B. West, and M. N., "Can injury in major junior hockey players be predicted by a pre-season functional movement screen - a prospective cohort study." vol. 2, 2014.

[10] Y. Liu and Q. Liu, "Convolutional neural networks with large-margin softmax loss function for cognitive load recognition," 07 2017.

[11] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala, "Overview and importance of data quality for machine learning tasks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & amp; Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3561–3562. [Online]. Available: https://doi.org/10.1145/3394486.3406477

[12] D. C. Klonoff, "Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things," *Journal of Diabetes Science and Technology*, vol. 11, no. 4, pp. 647–652, 2017, pMID: 28745086. [Online]. Available: https://doi.org/10.1177/1932296817717007

[13] J. Kaur and A. Kaur, "A review on analysis of eeg signals," in *2015 International Conference on Advances in Computer Engineering and Applications*, 2015, pp. 957–960.

[14] Z. K. andMartin Strmiska., "Introduction to the identification of brain waves based on their frequency," *MATEC Web of Conferences*, 2018.

[15] Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networksy," *Neural Information Processing Systems*, 2012.

[16] B. J., "Notes on convolutional neural networks," 2006.

[17] H. S. and S. J., "Long short-term memory," *Neural computation*, 1997.

[18] Greff, R. Srivastava, and J. Koutnik, "Lstm: A search space odyssey[j]," 2016.

[19] V. Ashish, S. Noam, P. Niki, U. Jakob, J. Llion, N. Aidan, K. L., and P. Illia, "Attention is all you need," *NIPS*, 2017.

[20] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 06 2015.

[21] J. Johannesen, J. Bi, R. Jiang, J. Kenney, and C.-M. Chen, "Machine learning identification of eeg features predicting working memory performance in schizophrenia and healthy adults," *Neuropsychiatric Electrophysiology*, vol. 2, 12 2016.

[22] M. Chaouachi, I. Jraidi, and C. Frasson, "Modeling mental workload using eeg features for intelligent systems," in *User Modeling, Adaption and Personalization*, 07 2011, pp. 50–61.

[23] M. Almogbel, A. Dang, and W. Kameyama, "Eeg-signals based cognitive workload detection of vehicle driver using deep learning," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 02 2018, pp. 256–259.

[24] C.-T. Lin et al., "Eeg-based assessment of driver cognitive responses in a dynamic virtual-reality driving environment," *IEEE transactions on bio-medical engineering*, vol. 54, pp. 1349–52, 08 2007.

[25] D. Aclo et al., "Eeg-based color classification system using artificial neural networks," 10 2015.

[26] E. Alharbi, S. Rasheed, and S. Buhari, "Feature selection algorithm for evoked eeg signal due to rgb colors," in *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 10 2016, pp. 1503–1520.

[27] A. Rakshit and R. Lahiri, "Discriminating different color from eeg signals using interval-type 2 fuzzy space classifier," in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 07 2016, pp. 1–6.

[28] K. Phillips, O. Fosu, and I. Jouny, "Separation and classification of eeg responses to color stimuli," in *2015 41st Annual Northeast Biomedical Engineering Conference (NEBEC)*, 04 2015, pp. 1–2.

[29] Z. Huiran and T. Zheng, "To judge what color the subject watched by color effect on brain activity," in *IJCSNS International Journal of Computer Science and Network Security*, 02 2011, p. 80.

[30] S. Åsly, L. Moctezuma, M. Gilde, and M. Molinas, "Towards eeg based classification of rgb color-based stimuli," 09 2019.

[31] S. Åsly, "Supervised learning for classification of eeg signals evoked by visual exposure to rgb colors," Ph.D. dissertation, 2019.

[32] M. Angelovski et al., "Application of bci technology for color prediction using brainwaves," 09 2012.

[33] J. Teo and J. Chia, "Eeg-based excitement detection in immersive environments: An improved deep learning approach," 09 2018, p. 020145.

[34] O. Krigolson, C. Williams, and F. Colino, "Using portable eeg to assess human visual attention," 05 2017, pp. 56–65.

[35] P. Bashivan, I. Rish, and S. Heisig, "Mental state recognition via wearable EEG," *CoRR*, 2016.

[36] R. Xiong, F. Kong, X. Yang, G. Liu, and W. Wen, "Pattern recognition of cognitive load using eeg and ecg signals," *Sensors (Basel, Switzerland)*, vol. 20, 09 2020.

[37] N. Liang, P. Saratchandran, G. Huang, and N. Sundararajan, "Classification of mental tasks from eeg signals using extreme learning machines," *International journal of neural systems*, vol. 16, 03 2006.

[38] S. Kuanar, V. Athitsos, N. Pradhan, A. Mishra, and K. R. Rao, "Cognitive analysis of working memory load from eeg, by a deep recurrent neural network," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[39] Y. Liu and Q. Liu, "Convolutional neural networks with large-margin softmax loss function for cognitive load recognition," 07 2017.

[40] A. Saha, M. Vikash, B. Sanjith, Sreeja, C. Ritwika, and S. Debasis, "Classification of EEG signals for cognitive load estimation using deep learning architectures," in *Intelligent Human Computer Interaction - 10th International Conference, IHCI 2018, Allahabad, India, December 7-9, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11278. Springer, 2018.

[41] D. Das, D. Chatterjee, and A. Sinha, *Unsupervised approach for measurement of cognitive load using EEG signals.*, 12 2013.

[42] A. A. et al., "Classification of perceived mental stress using a commercially available eeg headband," *IEEE Journal of Biomedical and Health Informatics*, 2019.

[43] B. Pinar, "A comparative study of mental states in 2d and 3d virtual environments using eeg," *IEEE International Conference on Systems, Man and Cybernetics*, 2019.

[44] Richer, "Real-time mental state recognition using a wearable eeg," *IEEE*.

[45] Bird, "A study on mental state classification using eeg based brain-machine interface," *IS*, 2018.

[46] A. Subhani, "Machine learning framework for the detection of mental stress at multiple levels," *IEEE Access*, 07 2017.

[47] N. Lu, Y. Wu, L. Feng, and J. Song, "Deep learning for fall detection: 3d-cnn combined with lstm on video kinematic data," *IEEE Journal of Biomedical and Health Informatics*, vol. PP, 02 2018.

[48] Ravi, Wong, Deligianni, Berthelot, Andreu, Lo, and Yang, "Deep learning for health informatics," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, 2017.

[49] R. Gardner and K. Yaffe, "Epidemiology of mild traumatic brain injury and neurodegenerative disease," *Molecular and Cellular Neuroscience*, vol. 66, 03 2015.

[50] M. Ellis, J. Leddy, and B. Willer, "Physiological, vestibulo-ocular and cervicogenic post-concussion disorders: An evidence-based classification system with directions for treatment," *Brain Injury*, vol. 29, 10 2014.

[51] P. McCrory, W. Meeuwisse, J. Dvorak, M. Aubry, J. Bailes, S. Broglio, R. C. Cantu, D. Cassidy, R. J. Echemendia, R. J. Castellani, G. A. Davis, R. Ellenbogen, C. Emery, L. Engebretsen, N. Feddermann-Demont, C. C. Giza, K. M. Guskiewicz, S. Herring, G. L. Iverson, K. M. Johnston, J. Kissick, J. Kutcher, J. J. Leddy, D. Maddocks, M. Makdissi, G. T. Manley, M. McCrea, W. P. Meehan, S. Nagahiro, J. Patricios, M. Putukian, K. J. Schneider, A. Sills, C. H. Tator, M. Turner, and P. E. Vos, "Consensus statement on concussion in sport—the 5th international conference on concussion in sport held in berlin, october 2016," *British Journal of Sports Medicine*, vol. 51, no. 11, pp. 838–847, 2017. [Online]. Available: https://bjsm.bmj.com/content/51/11/838

[52] J. Hurtubise, D. Gorbet, Y. Hamandi, A. Macpherson, and L. Sergio, "The effect of concussion history on cognitive-motor integration in elite hockey players," *Concussion*, vol. 1, p. CNC17, 09 2016.

[53] L. Sergio, D. Gorbet, M. Adams, and D. Dobney, "The effects of mild traumatic brain injury on cognitive-motor integration for skilled performance," *Frontiers in Neurology*, vol. 11, p. 541630, 09 2020.

[54] M. Adams, E. Niechwiej-Szwedo, W. McILROY, and W. Staines, "A history of concussion affects relevancy-based modulation of cortical responses to tactile stimuli," *Frontiers in Integrative Neuroscience*, vol. 14, 07 2020.

[55] C. Baker and M. Cinelli, "Visuomotor deficits during locomotion in previously concussed athletes 30 or more days following return to play," *Physiological reports*, vol. 2, 12 2014.

[56] K. Manning, J. Brooks, J. Dickey, A. Harriss, L. Fischer, T. Jevremovic, K. Blackney, C. Barreira, A. Brown, R. Bartha, T. Doherty, D. Fraser, J. Holmes, G. Dekaban, and R. Menon, "Longitudinal changes of brain microstructure and function in nonconcussed female rugby players," *Neurology*, vol. 95, 06 2020.

[57] S. Slobounov, M. Gay, B. Johnson, and K. Zhang, "Concussion in athletics: Ongoing clinical and brain imaging research controversies," *Brain imaging and behavior*, vol. 6, pp. 224–43, 06 2012.

[58] A. Tapper, D. Gonzalez, E. Roy, and E. Niechwiej-Szwedo, "Executive function deficits in team sport athletes with a history of concussion revealed by a visual-auditory dual task paradigm," *Journal of sports sciences*, vol. 35, pp. 1–10, 03 2016.

[59] M. Thériault, L. De Beaumont, N. Gosselin, M. Filipinni, and M. Lassonde, "Electrophysiological abnormalities in well functioning multiple concussed athletes," *Brain injury : [BI]*, vol. 23, pp. 899–906, 10 2009.

[60] Higgins, T. Caze, and A. Maerlender, "Validity and reliability of baseline testing in a standardized environment," *Arch Clin Neuropsychol.*, vol. 23, pp. 437–443, June 2018.

[61] A. Rebchuk, H. Brown, M. Koehle, and G. Siegmund, "Using variance to explore the diagnostic utility of baseline concussion testing," *Journal of Neurotrauma*, vol. 37, 01 2020.

[62] J. Immanuel, A. Joshua, and s. George, "A study on using blink parameters from eeg data for lie detection," 01 2018, pp. 1–5.

[63] V. Abootalebi, M. Moradi, and M. Khalilzadeh, "A new approach for eeg feature extraction in p300-based lie detection," *Computer methods and programs in biomedicine*, vol. 94, pp. 48–57, 12 2008.

[64] A. Bablani, R. E. Damodar, and K. Venkatanareshbabu, "Deceit identification test on eeg data using deep belief network," 10 2018.

[65] J. Gao, Z. Wang, Y. Yang, Z. Wenjia, T. Chunyi, G. Jinan, and R. Nini, "A novel approach for lie detection based on f-score and extreme learning machine." 6 2013.

[66] A. Simbolon, A. Turnip, J. Hutahaean, and Y. Siagian, "An experiment of lie detection based eeg-p300 classified by svm algorithm," 10 2015.

[67] R. Cakmak and A. Zeki, "Neuro signal based lie detection," 10 2015, pp. 170–174.

[68] G. Ian, B. Yoshua, and C. Aaron, "Deep learning," *The MIT Press*, 2016.

[69] D. Michie, ""memo" functions and machine learning." *Nature 218*, 09 1968.

[70] N. Nilsson, "Principles of artificial intelligence," *The MIT Press*, 2014.

[71] Ordenez and Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, 2016.

[72] K. Kim, S. Choi, M. Chae, H. Park, J. Lee, and J. Park, "A deep learning based approach to recognizing accompanying status of smartphone users using multimodal data," *Journal of Intelligence and Information Systems*, vol. 25, 2019.

[73] S. Zhao, M. Talasila, G. Jacobson, C. Borcea, S. Aftab, and J. Murray, "Packaging and sharing machine learning models via the acumos ai open platform," 10 2018.

[74] W. Wang, S. Wang, J. Gao, M. Zhang, G. Chen, T. Ng, and B. Ooi, "Rafiki: Machine learning as an analytics service system," *Proceedings of the VLDB Endowment*, vol. 12, 04 2018.

[75] D. Crankshaw, X. Wang, G. Zhou, M. Franklin, J. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," 12 2016.

[76] J. Alves, L. Honorio, and M. Capretz, "Ml4iot: A framework to orchestrate machine learning workflows on internet of things data," *IEEE Access*, vol. 7, pp. 1–1, 10 2019.

[77] S. Ranganathan, "Edge to core to cloud architecture for ai," *NetApp*, 2018.

[78] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," 01 2017.

[79] V. Iyer and K. Roy Chowdhury, "Spectral analysis: Time series analysis in frequency domain," *The IUP Journal of Applied Economics*, vol. VIII, pp. 83–101, 01 2009.

[80] Y. Lei and Z. Wu, "Time series classification based on statistical features," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, 02 2020.

[81] J. Peirce et al., "Psychopy2: Experiments in behavior made easy." 51 2019.

[82] S. M. Arnaud Delorme, "Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis," *J Neurosci Methods*, 2004.

[83] T. da Silveira, A. Kozakevicius, and C. Rodrigues, "Automated drowsiness detection through wavelet packet analysis of a single eeg channel," *Expert Systems with Applications*, vol. 55, 03 2016.

[84] D. Das, D. Chatterjee, and A. Sinha, *Unsupervised approach for measurement of cognitive load using EEG signals.*, 12 2013, pp. 1–6.

[85] O. Krigolson, C. Williams, and F. Colino, "Using portable eeg to assess human visual attention," 05 2017, pp. 56–65.

[86] W. Hao, C. Zhihua, Z. Safeng, and Z. Li, "The continuous analysis of eeg's alpha wave by morlet wavelet transform." *National Center for Biotechnology Information*, 2010.

[87] M. Chaouachi, I. Jraidi, and C. Frasson, "Modeling mental workload using eeg features for intelligent systems," in *User Modeling, Adaption and Personalization*, 07 2011, pp. 50–61.

[88] J. Johannesen, J. Bi, R. Jiang, J. Kenney, and C.-M. Chen, "Machine learning identification of eeg features predicting working memory performance in schizophrenia and healthy adults," *Neuropsychiatric Electrophysiology*, vol. 2, 12 2016.

[89] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cogn. Sci.*, vol. 12, pp. 257–285, 1988.

[90] S. Kalyuga, "Cognitive load theory: How many types of load does it really need?" *Educational Psychology Review*, vol. 23, pp. 1–19, 03 2011.

[91] P. Blumenfeld and A. Paris, "School engagement: Potential of the concept, state of the evidence," *Review of Educational Research - REV EDUC RES*, vol. 74, pp. 59–109, 03 2004.

[92] C. Wickens, "Multiple resources and mental workload," *Human factors*, vol. 50, pp. 449–55, 07 2008.

[93] S. Miyake, "Multivariate workload evaluation combining physiological and subjective measures," *International journal of psychophysiology : official journal of the International Organization of Psychophysiology*, vol. 40, pp. 233–8, 05 2001.

[94] F. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, p. 115, 01 2016.

[95] S. Mukhopadhyay and S. Banerjee, "Learning dynamical systems in noise using convolutional neural networks," *Chaos 30*, vol. 30, 2020.

[96] S. Mukhopadhyay and M. Litoiu, "Fault detection in sensors using single and multichannel weighted convolutional neural networks," *IOT*, 2020.

[97] G. Ruffini, I. David, M. Castellano, L. Dubreuil Vall, A. Soria-Frisch, R. Postuma, J. Gagnon, and J. Montplaisir, "Deep learning with eeg spectrograms in rapid eye movement behavior disorder," *Frontiers in Neurology*, vol. 10, p. 806, 07 2019.

[98] A. W. Gaillard, "Comparing the concepts of mental load and stress." *Ergonomics*, 1993.

[99] B. et al., "A study on mental state classification using eeg based brain-machine interface," *IS*, 2018.

[100] A. Appriou, A. Cichocki, and F. Lotte, "Towards robust neuroadaptive hci: Exploring modern machine learning methods to estimate mental workload from eeg signals," 04 2018.

[101] Q. Wang and O. Sourina, "Real-time mental arithmetic task recognition from eeg signals," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 21, 01 2013.

[102] D. Chakladar, S. Dey, P. Roy, and D. Dogra, "Eeg-based mental workload estimation using deep blstm-lstm network and evolutionary algorithm," *Biomedical Signal Processing and Control*, vol. 60, 07 2020.

[103] M. Plechawska-Wojcik, M. Tokovarov, M. Kaczorowska, and D. Zapała, "A three-class classification of cognitive workload based on eeg spectral data," *Applied Sciences*, vol. 9, 12 2019.

Appendix A: Implementation

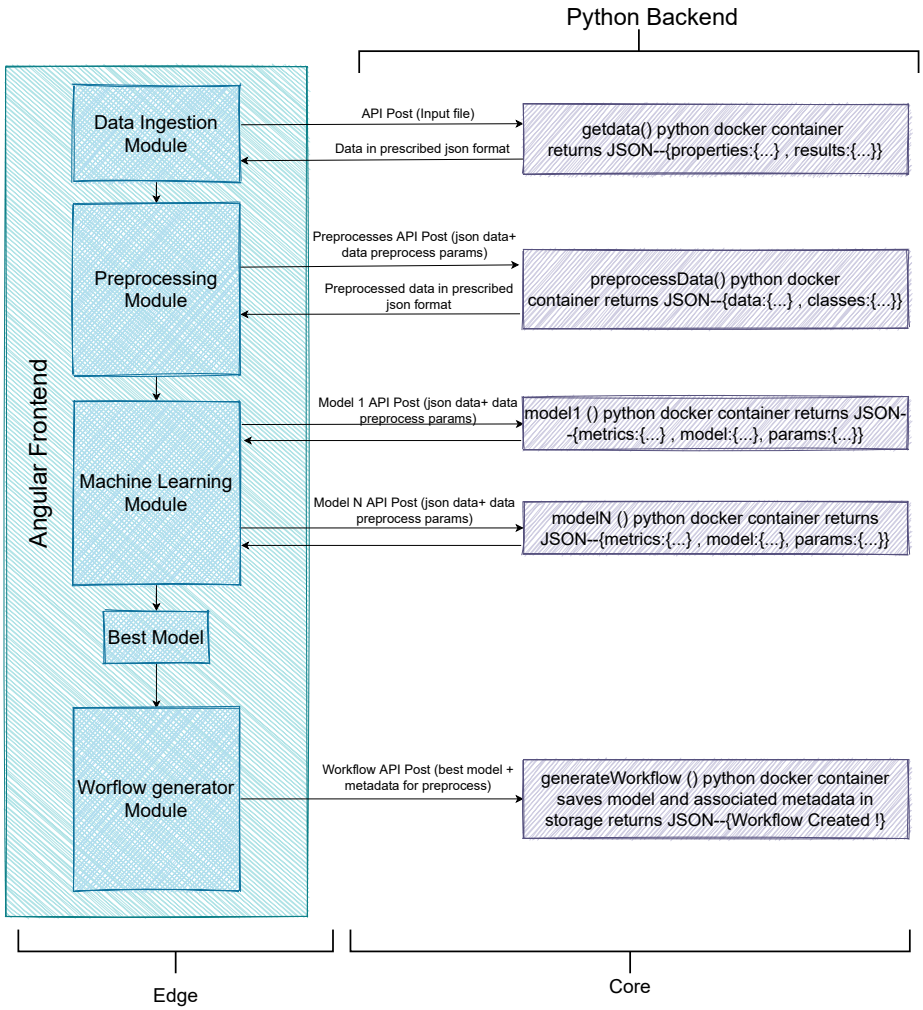


Figure 31: Block diagram for the proposed implementation.

We have implemented our proposed application using Angular frontend and Python backend. Each component in frontend communicates to the backend by sending GET/POST HTTP requests using respective API endpoints. The endpoints have been written using python's flask framework. The main components that we propose to use are Data Ingestion, Data Preprocessing, Machine Learning and Workflow generator module. Each module works independently and can interact with one another by sharing data. The flask services have been containerized using docker and hosted on AWS and can be accessed by simple API calls. We also provide a easy to use interface that makes the interaction of the user with our architecture easy. Writing dedicated and customized software code to execute these steps often leads to duplicate and hard-to-maintain glue code. Defining ML workflows employing a Graphical User Interface (GUI) helps to avoid unnecessary duplication of code and produces standardized representations of those workflows. Our application splits the ML workflow into granular and independent tasks. Also, splitting the ML workflows into small and specialized services facilitates the reuse of those software components. Lastly, it contributes to the extensibility of our application, because when ML workflows are divided into well-defined components, it creates natural points of extension for brand new functionalities. We shall now dicuss each component in detail:

User Interface

The user interface has been designed using Angular in form of a web application. Various angular functionalities have been used to like components, directives, routing, services etc. The interface is easy to use and interactive and helps the user to easily use various functionalities that are provided by the app.

Data Ingestion Module

The data ingestion module accepts the data in CSV format (with a header) and converts it to JSON for further processing by sending request to getData API. The JSON format helps to

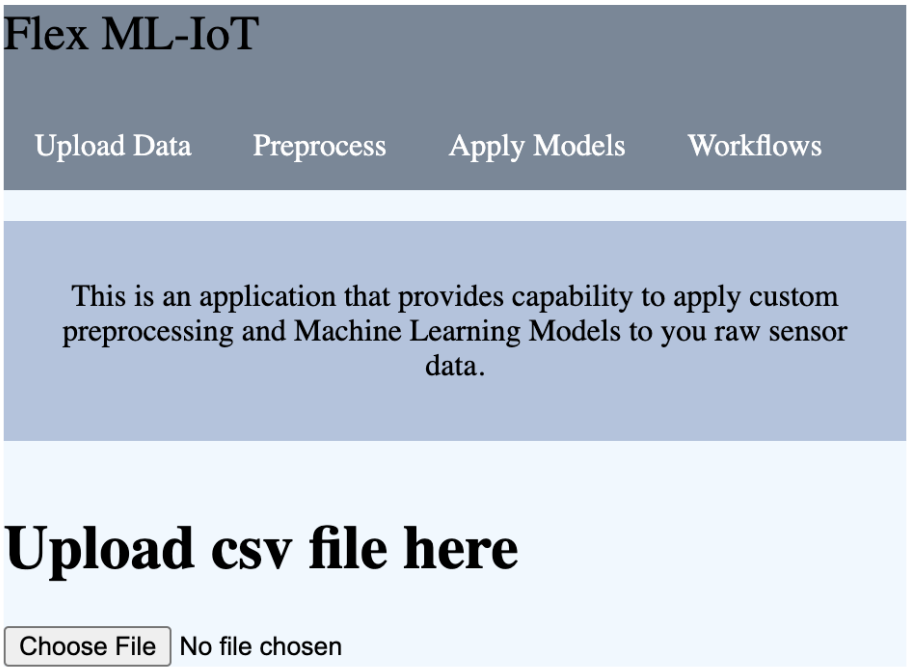


Figure 32: The UI for proposed application.

further analyse data and sharing the data to other API endpoints.

Data Preprocessing Module

The data preprocessing module enables the user to apply various data preprocessing, feature engineering and feature reduction techniques. Each process has a different API end point so that each component can operate independently. The user can select the preprocessing they want to apply and the module will perform the preprocessing in a sequential manner and give the final output of preprocessed data as a JSON object. The main python libraries used for various preprocessing steps are pandas, scipy, numpy, pywavelet and sklearn.preprocessing.

Machine Learning Module

This module provides various ML models that the user can apply on raw or preprocessed data. The models are provided by different docker images and can be accessed by endpoints. The training of models is done using the cross validation approach and the hyperparameter optimization is done using GridSearchCV. The ouput of this module is the best performing model with the best hyperparameters. The main libraries used in the backend for this module are scikit-learn, keras and tensorflow. The advantage of using containerized approach for providing ML models is that models that are provided by different languages can also be incorporated in the architecture, however for our case the models provided by python were suffice so we only used them.

Workflow generator Module

This module allows the user to generate and save ML workflows which give the best result. When a user saves the workflow what is saved is the best performing model along with a JSON file storing the preprocesses applied before the training. This can be used by the user at the time of online or batch testing for predicting the results on the test data. One advantage of this approach is there is no dedicated container allotted to each workflow, when the user runs the workflow the model and metadata is retrieved and a single container always addresses the prediction request. This container can also be horizontally scaled for serving multiple requests. The workflow generated is stored in MongoDB and can be retrieved on demand when a request is sent to use it for prediction. The model is saved in a binary format using Python's pickle library along with the other metadata in JSON format. Each model is stored in separate collection in the same database. We used MongoDB's M0 Sandbox (General) tier to store this data, it is freely available tier.