Application of Deep Learning Techniques for EEG Signal Classification

by

Omkar Muglikar

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2021 by the
Graduate Supervisory Committee:

Yalin Wang, Chair
Jianming Liang
Hemanth Kumar Demakethepalli Venkateswara

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Communicating with computers through thought has been a remarkable achievement in recent years. This was made possible by the use of Electroencephalography (EEG). Brain-computer interface (BCI) relies heavily on Electroencephalography (EEG) signals for communication between humans and computers. With the advent of deep learning, many studies recently applied these techniques to EEG data to perform various tasks like emotion recognition, motor imagery classification, sleep analysis, and many more.

Despite the rise of interest in EEG signal classification, very few studies have explored the MindBigData dataset, which collects EEG signals recorded at the stimulus of seeing a digit and thinking about it. This dataset takes us closer to realizing the idea of mind-reading or communication via thought. Thus classifying these signals into the respective digit that the user thinks about is a challenging task. This serves as a motivation to study this dataset and apply existing deep learning techniques to study it.

Given the recent success of transformer architecture in different domains like Computer Vision and Natural language processing, this thesis studies transformer architecture for EEG signal classification. Also, it explores other deep learning techniques for the same. As a result, the proposed classification pipeline achieves comparable performance with the existing methods.

To Mom and Dad and all people who helped me throughout the journey

ACKNOWLEDGMENTS

I would first like to express my gratitude to my chair, Dr. Yalin Wang, for his support and encouragement over the years. Thanks also to all members of the Geometry Systems Laboratory group for providing a great research environment. I am also grateful to my committee members Professor Hemanth Kumar Demakethepalli Venkateswara and Professor Jiangming Liang for their input in my work. Finally, I am most grateful to my family for their support and encouragement throughout my studies. Many thanks to you all.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

Communication with computers through thought/brain signals has been a remarkable achievement in recent years. A brain-computer interface (BCI) is an interface where we do not need peripheral devices to communicate with computers. Essentially, we communicate with computers directly from our brain without any intermediate devices. BCI is defined as a direct communication pathway between the human brain and an external device(Wolpaw and Wolpaw 2012). BCI relies heavily on Electroencephalography (EEG) signals for communication between humans and computers.

EEG signals quantify the voltage fluctuations occurring at the brain's surface resulting from the ionic currents due to neuronal activity of the brain. These signals are captured via electrodes placed over different locations on the brain surface. There are many other techniques like fMRI, PET, fNIRS, etc., to visualize the undergoing brain activity. Among all these, EEG is preferred for BCI applications as it is usually non-invasive, portable, and gives better time resolution signals. Also, commercial-grade EEG headsets are available in the market at reasonable costs, which naturally implies a more significant impact on users. Furthermore, with the advent of deep learning, many studies conducted recently use EEG data to perform various tasks like emotion recognition, motor imagery classification, sleep analysis, Epileptic seizure prediction, etc.

According to (Roy et al. 2019) and (Gong et al. 2020), the number of publications

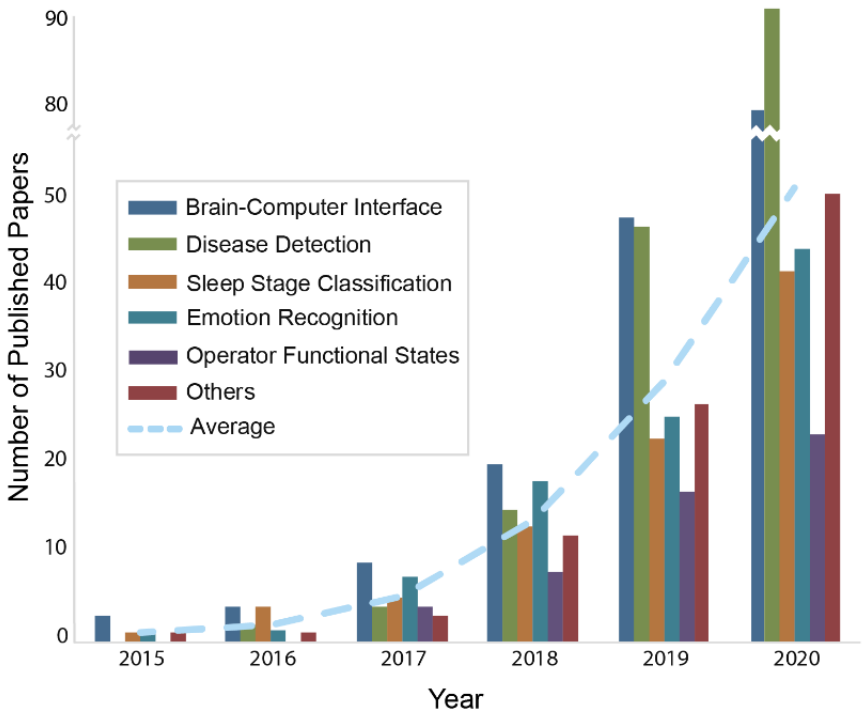

Figure 1: Number of publications per year(2015-2020)

Source: (Gong et al. 2020)

has increased drastically from 2015 to 2020 with the introduction of deep learning in BCI. This implies that the application of deep learning to EEG-based BCI is a topic of growing interest among the community. Figure 2 shows the distribution of publications from 2010 to 2018 related to EEG and BCI research and Figure 1 shows a recent version from 2015 - 2020. Especially after the breakthroughs in Deep learning, the research in this area grew exponentially. Similarly, the study presented in (Gu et al. 2021) focuses on recent advancements(2015 - 2019) at the intersection of EEG-based BCI and computational intelligence techniques, which can be used for an effective BCI system. Both these studies are essential to understand the current research direction in this field and provide a road map for future research directions.

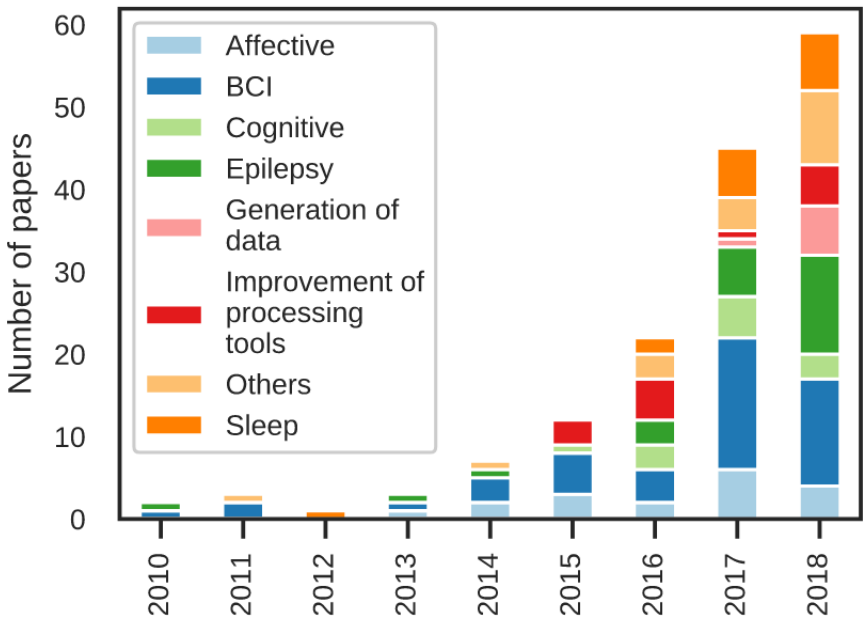

Figure 2: Number of publications per year(2010 - 2018)

Source: (Roy et al. 2019)

Given this sudden increase in the interest in classifying EEG signals for various applications, many studies have applied deep learning and machine learning techniques to different domains like Motor Imagery recognition, Sleep stage classification, Epilepsy Detection, etc. However, among all these recent studies, not many have explored the application of EEG signal classification for the MindBigData(MBD) [1] dataset. Thus, this thesis tries to answer the question - Given a set of EEG signals from a subject captured with the stimulus of seeing a digit from 0 to 9 and thinking about it, can we classify which number the user was thinking/seeing? This question can be further rephrased from a machine learning point of view: Given multivariate EEG signals, can we classify them into respective digits? This question can give rise to 2 scenarios that we are interested in -

[1] https://http://www.mindbigdata.com/opendb/

1. Can we pinpoint which digit the user was thinking/seeing about? (multiclass classification -1 to 10 (including not thinking about any digit))
2. If not, can we at least tell if the user was thinking about any digit at all? (binary classification - yes or no)

Answering these research questions can significantly benefit the advancement of EEG-based BCI applications. Solving the given problem can empower people with disabilities to communicate with devices without the need for physical movement; it can help build intelligent systems where users can control devices with just thinking. This research is a step towards realizing this goal of Brain-computer interfacing, where we start from recognizing numbers thought by a user. This can further lead to recognizing words and sentences from human brain signals. This study may also benefit in understanding the patterns involved for such subtle tasks in EEG recordings, if any. Overall, this study will explore the new MindBigData dataset and try to classify the EEG signals based on which number the user is thinking about. Furthermore, this study will take us a step closer to realizing the dream of controlling computers through "thoughts."

Given that EEG signals are essentially multivariate time series data where features are different channels, we can expect deep learning models harnessing recurrent nature to suit the task of EEG signal classification. Studies like - (Song et al. 2017) have shown that Transformer architectures can improve classification performance for Time-series data. However, there are only three studies to the best of our knowledge (Bird et al. 2019),(Pratama, Kesiman, and Gunadi 2021), and (Jolly et al. 2019) that try to solve the above questions. (Jolly et al. 2019) tries to solve the above problems for the four-channel Muse dataset using Gated Recurrent Units, Autoencoder, and Convolutional Neural Networks. On the other hand, (Bird et al. 2019) tries to

approach the problem using LSTM and MLP and improve their performance using an Evolutionary Algorithm. (Pratama, Kesiman, and Gunadi 2021) tries 2 different data preprocessing techniques to obtain features from EEG signals namely - frequency band features and Principal Component Analysis. The features obtained from them are further passed to KNN classifier to obtain results on MBD Epoc as well as MBD Muse dataset.

1.2 Thesis outline

This thesis is divided logically into following sections -

- Chapter 2 discusses the Background about the topic which includes History of EEG based BCIs followed by Relevant works and problem description.
- Chapter 3 starts with the description of Data used and then dives deep into discussing about methods used for this study. It discusses the entire classification pipeline used for classifying EEG Signals.
- Chapter 4 discusses the experiments performed and the results obtained.
- Chapter 5 discusses the findings and limitations of this study.
- Chapter 6 discusses the conclusions of the study followed by the future work.

1.3 Contributions

This study develops a classification framework using deep learning techniques for raw EEG signal classification. Also, we compare the performance of Traditional machine learning techniques and Deep learning techniques to observe that both of them give similar performance. The contributions of this study are as follows -

- This study applies deep learning techniques on MindBigData (MBD) EPOC and Muse datasets which are not explored much.
- This is the first study to apply Transformer architecture on the above mentioned datasets to the best of our knowledge.
- Developing an end to end classification pipeline for raw EEG classification
- Acheived comparable results compared to existing literature with both - hand crafted features as well as features extracted from deep learning architecture. Thus proving the efficacy of deep learning techniques for automated feature extraction.

Chapter 2

BACKGROUND

2.1 History of Brain Computer Interfacing and Electroencephalography

EEG Signals were the foundation on which Brain-Computer Interfaces were built. But well before acquiring EEG signals from the human brain, there were some studies done on animals to record their brain electrical responses. According to (Kawala-Sterniuk et al. 2021), An English physicist - Richard Caton, recorded the first electrical signals from animals and published his results in 1875 in the British Medical Journal. Around the same time, two polish scientists - Napoleon Nikodem Cybulski and Adolf Beck - were working on brain signal recording. In 1913, Vladimir Vladimirovich Prawdicz-Nieminski was the first person to show seven different types of changes in the animal brain's electrical activity, which he called "electrocerberogram." According to (Kawala-Sterniuk et al. 2021), Hans Berger recorded the first EEG signals from the human brain in 1924. He published his findings in 1929 in a paper titled "Über das Elektrenkephalogramm des Menschen" (Berger 1929). This was the starting point for all the applications of EEG in different fields. His findings were confirmed in early 1934 by English scientist Lord Adrian and an American researcher from Harvard University—Hallowell Davis. After the era of EEG discovery, real experiments for BCI began on animals around the 1970s. According to (Kawala-Sterniuk et al. 2021), The first experiment was carried out on monkeys in 1969 and 1970. Cats were also experimented on around the same time. The tests on Humans began around the 1990s. One of the breakthroughs for BCI can be considered when Wolpaw and his colleagues

showed that a cursor on a monitor could be controlled by brain waves, particularly the SMR (sensorimotor rhythm) in 1991(Kawala-Sterniuk et al. 2021).

Given this success, the next phase in BCI was invasive BCI systems. According to(Kawala-Sterniuk et al. 2021), the first invasive BCI chip was implanted in 1998 on a human subject. The 2000s saw an increased interest in BCI, and two major breakthroughs happened in 2012. According to (Kawala-Sterniuk et al. 2021), both the studies demonstrated that BCI systems were able to reinstate neural arm control in paralyzed subjects. The first study showed the same in monkeys, and the second one showed it in human subjects. Both these studies involved invasive BCI systems.

The study of Non -invasive EEG-BCI's has become more widespread recently with the advent of machine learning and deep learning techniques. It is evident from Fig2 that the number of publications related to EEG-based applications has increased exponentially after 2012. It is particularly important because that year, AlexNet was introduced as a deep learning model for solving Image classification for the first time. After that, there have been numerous attempts to apply Deep learning techniques in EEG signal processing.

Overall, the EEG-based BCI has come a long way and seems that the day won't be far where we can control devices with our thoughts given the rapid developments in this field.

2.2 Relevant works

Recently, many studies have been conducted to classify EEG signals using Convolutional Neural Networks, Gated Recurrent Units, and many other deep learning architectures. However, very few studies try to solve the specific problem of classifying

given EEG signals into respective digits that the user was thinking about, i.e., not many studies have explored the MBD Epoc and MBD Muse dataset. The following studies have attempted to solve the above-mentioned problem -

- (Bird et al. 2019) proposes an evolutionary approach for selecting hyperparameters for classifying the MBD Muse dataset. This study experiments with Multilayer Perceptron(MLP) and Long Short Term Memory (LSTM) network architectures for classifying 3 different datasets - one of them being the MBD Muse dataset. It reports a maximum accuracy of 10.77% for LSTM architecture and 31.35% for boosted MLP with an evolutionary approach. This study experiments with a very small size of data selected from the original dataset. It selects 15 samples for each class totaling 150 samples. This study attempts to perform a 10 class classification (digits 0-9) and does not consider the samples for random thinking(-1).
- (Jolly et al. 2019) develops a universal encoder for extracting features from EEG signals which can generalize to different datasets. This study experiments with 5 different datasets, among which one is the MBD Muse dataset. This study proposes a Gated recurrent unit for feature extraction and achieves an accuracy of 0.338, an F1 score of 0.16 for multiclass classification(11 classes), an accuracy of 0.993, and an F1 score of 0.991 for binary classification(thinking about the digit or not) on the MBD Muse datasets with 40983 samples in total. This study also compares their approach to well-known approaches like EEGnet (Lawhern et al. 2018) which reports an F1 score of 0.138 and 0.926 for multiclass and binary classification, respectively.
- According to (Pratama, Kesiman, and Gunadi 2021), MBD Epoc and MBD Muse datasets were used to experiment with the effects of extracting features

using frequency band and Principal component analysis. This study reports max accuracy of 12.3% using PCA feature extraction technique for MBD Epoc data for multiclass classification of digits 0-9. This paper does not mention f1 scores for their experiments. It uses 40 samples for each digit from the MBD Epoc data for experimentation. It applies the K-Nearest Neighbors algorithm to the extracted features from the data. Overall, this study reports no clear winner among both the techniques as every dataset is different.

2.3 Problem Description

Given a multichannel EEG signal recording from a person thinking/seeing a digit from 0 to 9, can we recognize if the user was thinking about a particular number? This is the fundamental question that this study tries to address. However, the problem is non-trivial and is expected to be much more challenging as very little work has been done on such datasets. Therefore, this study also tries to solve a simpler version of the above problem - Can we recognize if the person was thinking about a digit (yes/no)?

This problem can be formulated as a classification problem (Binary and Multiclass) from the machine learning point of view. Also, EEG signals can be considered multivariate time series data where different channels are equivalent to various time-series variables. So the problem boils down to multivariate time-series classification.

This study investigates the problem by building a classification pipeline for two datasets with 14 channel EEG data and 4 channel EEG data, which is described in Chapter 3. We experiment with deep learning architectures like - Transformer Encoder and CNNs. Also, we experiment with traditional machine learning techniques like

random forest and consequentially compare the results obtained. Finally, this problem is perfectly suited for deep learning applications because of the following reasons -

- The size of data is large enough to train a deep learning model
- Architectures like Transformers and Convolutional Neural Networks have proved to improve the classification performance for time series data as mentioned in (Zerveas et al. 2021) and (Wang, Yan, and Oates 2016).
- Feature extraction can be automated using deep learning without going through the tedious process of traditional feature extraction resulting in similar performance.

Chapter 3

METHODS

3.1 Data Description

This study employs two datasets for experiments - MindBgData(MBD) EPOC and MindBigData(MBD) Muse. These two datasets differ in the devices used to capture the EEG signals - 4 channels or 14 channels. The MBD dataset consists of data gathered from two more devices - Neurosky mindwave and Emotiv Insight with single channel and 4 channels, respectively. Being a single channel device, mindwave data is not considered in this study as we believe a single channel could not capture intricate patterns of EEG signals. Furthermore, the Insight data did not contain labels for "not thinking about the number" and hence could not be used for binary classification.

All the data is collected from a single subject. The data consists of 2 seconds EEG recordings from the person, which are captured after the subject receives a stimulus of seeing a digit from 0-9 and thinking about it. A single subject has collected the data - David Vivancos - over the course of 2 years between 2014 and 2015. The EEG devices used for capturing these signals are commercial devices and not medical-grade equipment, and hence the quality is sub-optimal compared to medical-grade EEG devices. These devices follow the 10/20 system of electrode positioning, which is shown in Fig 3. The data is presented in a flat text file format with all the fields separated by a single space, as shown in Table 1.

MBD EPOC contains recordings captures from Emotiv Epoc headband which has 14 channels to collect EEG signals. The EEG signals are collected from the following

[id]	[event]	[device]	[channel]	[code]	[size]	[data]
27	27	MW	FP1	5	952	18,12,13,12,..
67650	67636	EP	F7	7	260	4482.564102,..
978210	132693	MU	TP10	1	476	506,508,509,..
1142043	173652	IN	AF3	0	256	4259.487179,..

Table 1: Data Format

Source: (**http://www.mindbigdata.com/opendb/**)

locations - “AF3”, “F7”, “F3”, “FC5”, “T7”, “P7”, “O1”, “O2”, “P8”, “T8”, “FC6”, “F4”, “F8”, “AF4” which are shown in dark blue color in Fig 3. The sampling frequency for this headset is 128Hz.

MBD Muse contains recordings captured from Interaxon Muse headband, which consists of 4 channels. The EEG Signals are collected from the following channels - “TP9”, “FP1”, “FP2”, “TP10,” which are shown in red color in Fig 3. The sampling frequency for this headset is 220Hz.

The data is processed using pandas library python where the input plain text file is broken down into individual EEG samples of shape

$$no.of channels \times length of EEG signal$$

Each such individual sample is stored as a CSV file in the following format - Data_no_<Id>_for_digit_<label>.csv. This helps in the further processing of data in the classification pipeline. Finally, not all the samples are used for training deep neural networks and traditional machine learning models. We do some data preprocessing described in the next section to select specific samples from the entire dataset.

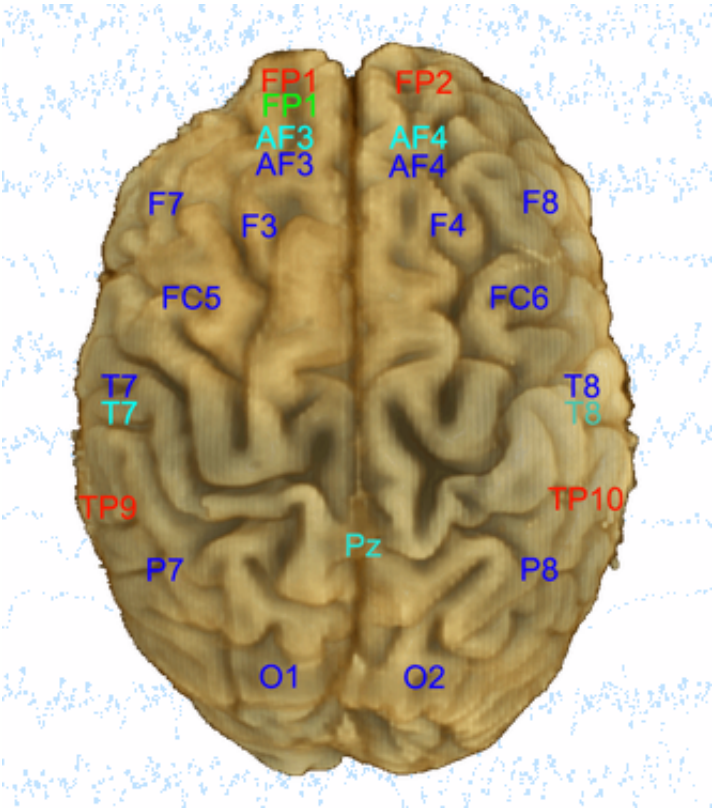

Figure 3: Electrode locations

Source: (**http://www.mindbigdata.com/opendb/**)

3.2 Classification Pipeline

The overall classification pipeline is shown in Fig4The input to the pipeline is a raw EEG signal from the previously generated CSV files of shape (no of channels, length of EEG signal). E.g. - (1,14,250) where 1 is the number of samples, 14 is the EEG channels, and 250 is the length of signal/data points in 2 seconds. Then it is passed through the data preprocessing step, where different operations are applied on the raw EEG signals to get them in a format required to train the deep learning model. The details of data preprocessing are discussed in section 3.3. Next, the prepared data is passed to a classifier (Deep learning or Traditional machine learning). The

experiments are carried out on various classification techniques. In Deep learning based classification, we employ architectures like Transformers and CNNs, which are trained on input data to get the final classification label. On the other hand, Traditional machine learning methods are also experimented with where an additional step of hand-crafted feature extraction is needed, followed by choice of classifier like Random Forest to get the final classification results. The details of each of the steps mentioned above are discussed in detail in the following section.

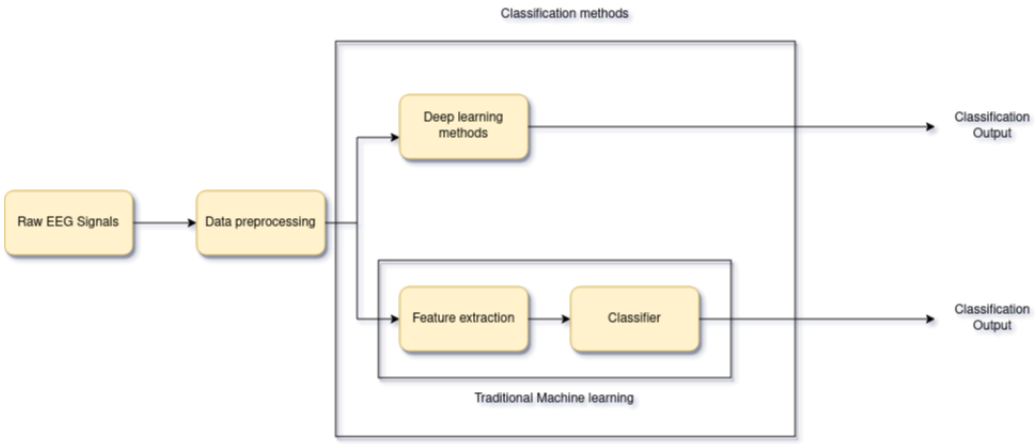

Figure 4: Classification Pipeline

3.3 Data Preprocessing

Firstly, the raw EEG signals are standardized channel-wise using a min-max scaler to ensure that the values are between 0 and 1. This is important, especially for deep learning models, as they tend to have the problems like vanishing and exploding gradients. Next, a rolling mean of signal in each channel is calculated with a window size of 5 for MDB EPOC and a window size of 10 for MBD Muse datasets. This helps in smoothening out the signals, which helps to remove some noise from the signal.

Now that we have somewhat cleaner signals within [0,1] range, we further try to drop the samples (here each sample is considered as a matrix of (no of channels, signal length) shape) who have bad channels in them. Bad channels have values abnormally higher or lower than the rest of the channels, as shown in Fig 6. On the other hand we have good signals/channels as shown in Fig 5

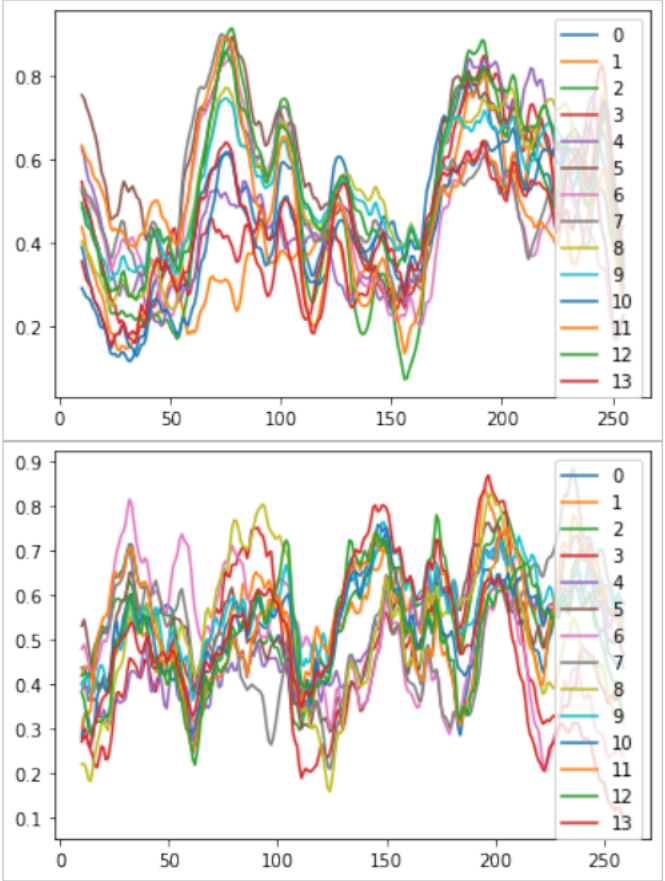

Figure 5: Examples of good channels

Such bad channels may adversely affect our classification performance. These samples can be considered as outliers. We drop such outlier samples as we do not have a dearth of data. We calculate the average maximum and minimum values for

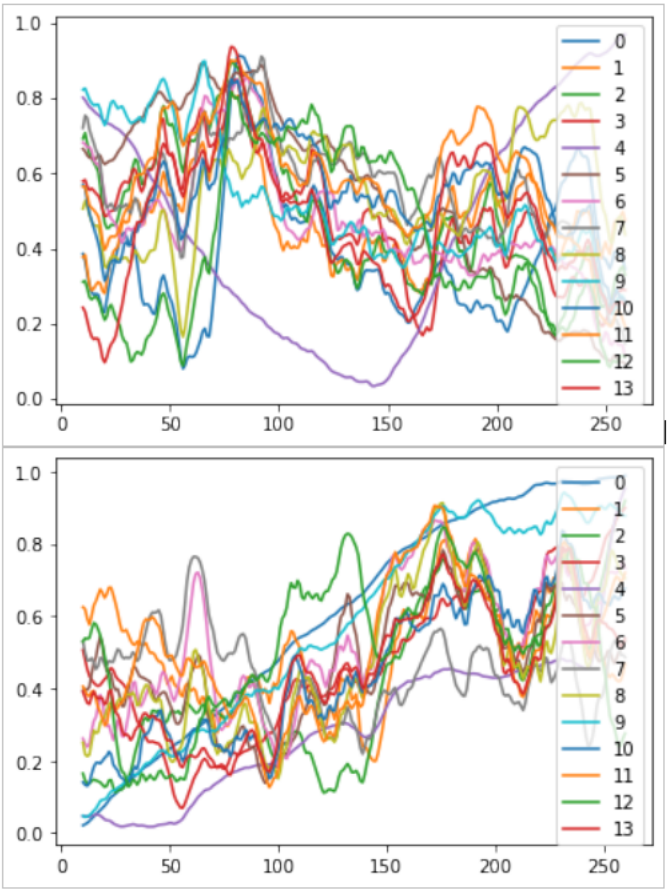

Figure 6: Example of bad channels

all samples across all channels and filter all the samples whose channel values do not cross that threshold. This is important as we want to focus on good samples rather than providing a model with bad samples as the task is already difficult to classify into digits. We do not want to make the model learn something that does not help it find meaningful patterns in data. This step is only done for classes 0-9. This processing is done for both the datasets, and finally, we select the minimum number of samples of the available good samples to maintain even class distribution. Hence, we randomly selected 159 samples for each digit (including -1) for the MBD EPOC dataset and 900 random samples for each digit (including -1) for the MBD

Digit	Epoc	Muse
0	159	900
1	159	900
2	159	900
3	159	900
4	159	900
5	159	900
6	159	900
7	159	900
8	159	900
9	159	900
-1	159	900
Total	1749	9900

Table 2: Data distribution for each digit

Muse dataset. The classwise distribution is shown in Table 2. Data preprocessing is purposefully kept at the minimum to make the deep learning model learn as much as possible from real-world data without any exclusive data preprocessing as it is done in conventional EEG preprocessing pipelines. This makes sure that the model is robust to the somewhat noisy data captured from commercial electrodes.

3.4 Feature Extraction Using Deep Learning

Now that we have the data in the required format, we can apply various classification techniques to this data. These deep learning techniques do not require hand-crafted feature extraction techniques as traditional machine learning algorithms require. Also, deep learning automates this feature extraction process completely. Thus we can directly pass the EEG signals to the model. We will discuss each classification technique in detail below -

3.4.1 Transformer Architecture

Transformer architecture was introduced by(Vaswani et al. 2017) in 2017. Before Transformers, LSTM(Long Short Term Memory Network) and GRU(Gated Recurrent Units) were the gold standards for any sequence modeling tasks. However, Transformer architecture has proved to surpass LSTMs and GRUs in many fields and is the current state of the art for sequence modeling. It has also outperformed many existing deep learning architectures for time series representation learning as shown by (Zerveas et al. 2021).

The architecture for Transformer is shown in Fig7. Transformer architecture can broadly be divided into two parts - Encoder and Decoder. The left part of the figure is the Encoder, and the right is the decoder. Since we deal with sequence to vector tasks, we only need the Encoder part of Transformer architecture. First, the input embeddings are added with the positional embeddings, and the resultant vector is passed to the Encoder layer. We discuss each component of the Transformer in detail below -

- **Positional Encoding**: Positional Encoding is used to account for the ordering of input vectors. These embeddings help to store information regarding the relative positions of vectors passed to the Encoder layer. This is very important in time series transformers. After adding the positional embedding vector, the resultant is passed on to the Encoder layer. The Encoder layer consists of a multi-head attention layer followed by a feed forward layer.
- **Attention**: To understand multi-head attention, we need to understand what attention is in Transformer. Fig8 describes the working of Attention in Trans-

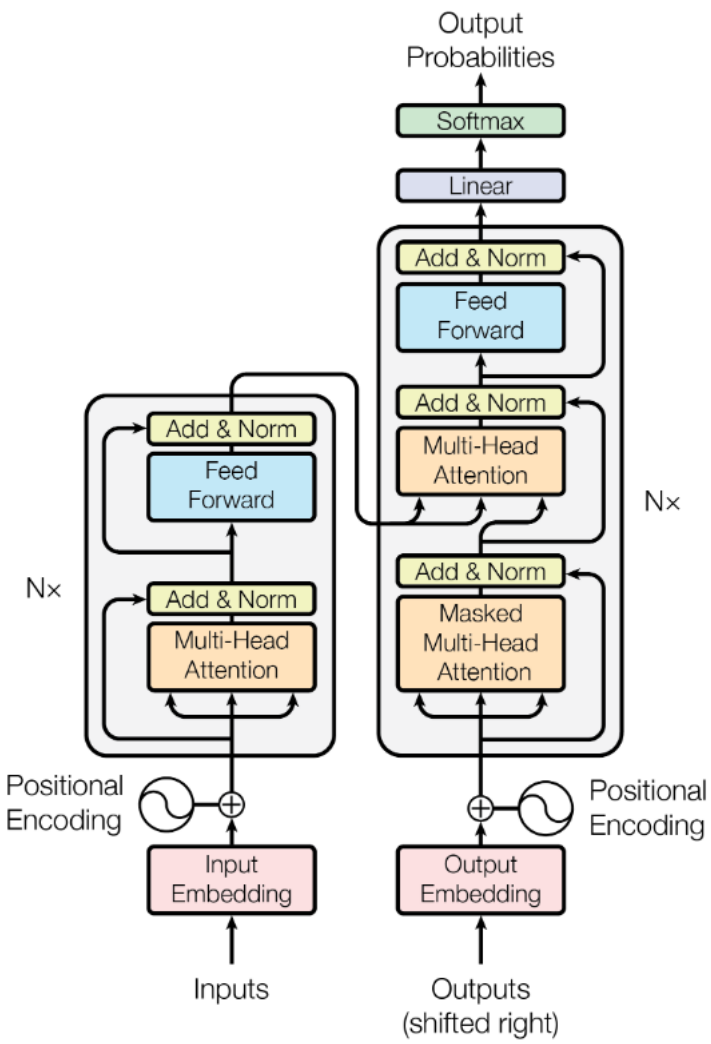

Figure 7: Transformer architecture

Source: (Vaswani et al. 2017)

former. In simple words, the input to the attention function are three vectors - K, Q, and V. The Attention is calculated from these three vectors using the following equation -

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

These three vectors are created from the input vector by multiplying it with

three different matrices, namely Wq, Wk, and Wv. These matrices are updated during the training process of the Transformer.

- **Multi-head Attention**: Now that we know how the scaled dot product attention mechanism works, we can see from Fig 8 that multi-head attention simply repeats the scaled dot product attention n times parallelly and concatenates the output vectors. This concatenated vector is further passed to the feed-forward network.
- **Feed Forward layer**: The feed-forward layer is a simple network with a ReLU activation applied to it. It can be described as follows -

$$FFN(x) = max(0, xW1 + b1)W2 + b2$$

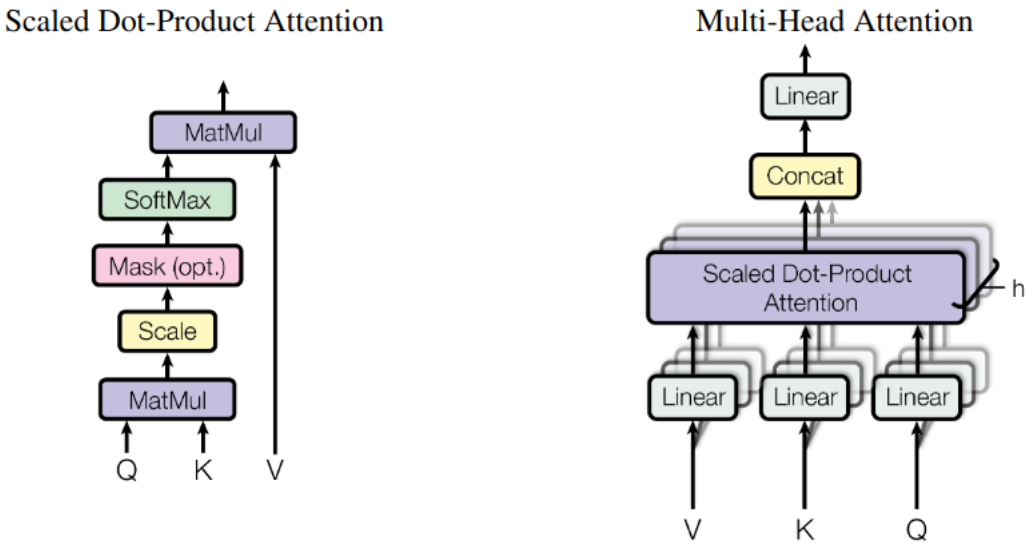

Figure 8: Attention mechanism

Source: (Vaswani et al. 2017)

We do not pass the resultant vector to the decoder part of the Transformer as we are only interested in the features extracted from the encoder layer. The output

obtained from the encoder layer is concatenated with outputs from other heads and is flattened before passing it to the classifier. This last layer holds the features extracted from data that is used for classification.

3.4.2 Long Short Term Memory Neural Networks

Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is how LSTM is formed.

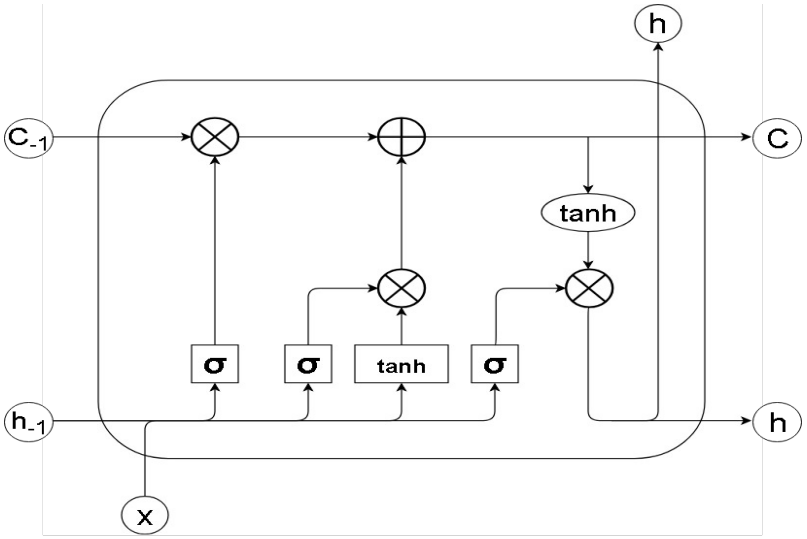

Figure 9: LSTM Architecture

Each layer of LSTM has:

- Trained neural network layers
- Point wise operations
- Vector concatenations

LSTM has 4 main gate functions [2]:

- Forget Gate

  This gate is used to help the current cell forget information from the previous cell. This is done with the help of a sigmoid layer:

  $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

  Given the inputs h_{t-1} and x_t, it produces a number between 0 and 1 for each input vector in the cell state C_{t1}

  0 Represents completely forget

  1 Represents completely remember

- Input Gate

  This gate is used to decide which values will be updated in the cell. This is achieved using a sigmod layer. We also use a tanh layer to find candidates values, $\tilde{C}_t$, which can be added to the state.

  $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

  $$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- Cell State

  We find the current cell state by using the values of the previous gates and performing point wise addition and multiplication

  $$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

[2] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Output Gate

  First, a sigmod layer is used on h_{t-1} and x_t. Then we pass tanh on the current cell state to cell state in the range of -1 and. Finally, we multiply these two results to get the output h_t

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times tanh(C_t)$$

3.4.3 Convolutional Neural Network

- CNN mainly comprises of 3 main operations namely -
  1. Convolution Operation
  2. Activation function
  3. Pooling Operation
- Convolution is mathematically defined as below.

$$(f * g)(c) = \sum_a f(a) \cdot g(c - a)$$

  In mathematics, convolution is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other [3] . This operation is beneficial in CNN as the operation defines the sliding nature of the kernel over the input image to produce the desired output.
- **Convolution** operation is the basis of this algorithm. Convolution operation produces a high value output for a given position if the convolution feature is

[3]https://en.wikipedia.org/wiki/Convolution

present in that location; else, it outputs a low value. Thus convolution operation transforms the time series according to the kernel weights. Different transformations have different kernels and are usually hand-crafted values. Training a CNN aims to learn these weights for specific tasks like classification instead of using hand-crafted values. Fig10shows a typical flow of CNN for time series data -

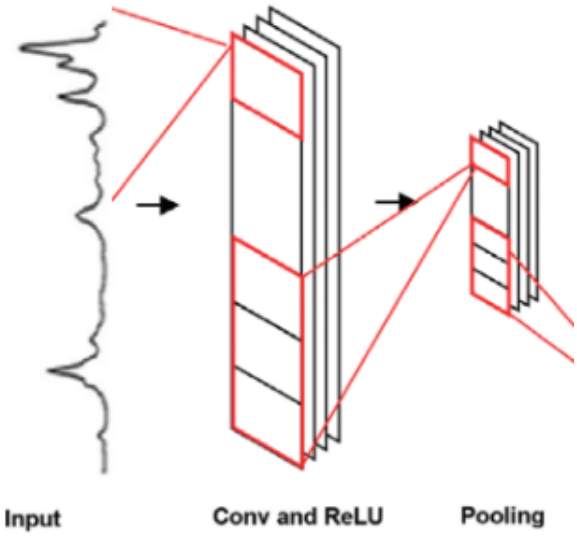

Figure 10: 1D Convolution

Source: (Mozaffari and Tay 2020)

- The next operation is **activation function**. This function decides the output value. If a certain threshold is crossed, the neuron fires the output. Various activation functions are used like ReLu, GELU, ELU, softmax, softplus, tanh, etc. In this study, we use the RELU (Rectified Linear Unit) and GELU (Gaussian Error Linear Unit) function. The functions are mathematically defined as below -

$$ReLU(x) = max(0, x)$$

$$GELU(x) = 0.5x(1 + tanh(\sqrt{2/\pi}(x + 0.044715x^3)))$$

Various activation functions are shown in Fig11.

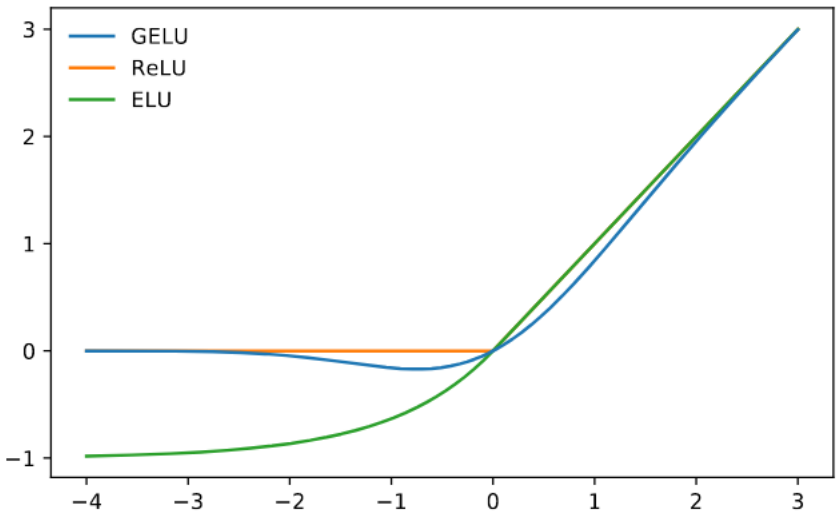

Figure 11: Activation functions

Source: (Hendrycks and Gimpel 2020)

- The next operation following the convolution is **pooling**. Pooling is used for reducing the dimension of output while preserving spatial information. Various types of pooling are used like max pooling, average pooling, etc. However, we restrict our discussion to max-pooling in this section. Max pooling selects the maximum value from the subset of vector values.

3.5 Feature Extraction using Traditional Machine Learning

Unlike Deep learning, traditional machine learning requires hand-crafted features for learning from data. Hence extracting correct features from given EEG signals is important. We start with extracting some statistical features like -

- Aggregated Fast Fourier Transform(FFT) kurtosis - Calculates the spectral kurtosis of absolute Fourier transform spectrum
- FFT coefficients - Calculates the FFT coefficients for real-valued input series. It is calculated as follows -

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\}, \qquad k = 0, \ldots, n-1.$$

  where a is the input time series, k is the kth coefficient. We consider the first 10 coefficients as features.
- Fourier Entropy - Calculates the binned entropy of the power spectral density of the time series (using the welch method)[4]. we choose bin size=100.
- c3 - Calculates c3 statistics(Schreiber and Schmitz 1997) for measuring nonlinearity of signal. Following is the equation for calculating it -

$$\frac{1}{n-2lag} \sum_{i=1}^{n-2lag} x_{i+2\cdot lag} \cdot x_{i+lag} \cdot x_i$$

  We use lag values of 1,2 and 3.
- (Complexity invariant distance)cid_ce (Batista et al. 2013) - This quantifies the complexity of EEG signals and is calculated as follows -

$$\sqrt{\sum_{i=1}^{n-1} (x_i - x_{i-1})^2}$$

  . We calculate this feature for both normalized as well as non-normalized signals.
- partial autocorrelation(Wilson 2016) - Calculates the partial correlation at specified lag value. We use lag values from 0-9.

[4]https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.welch.html

- cwt coefficients - Calculates the continuous wavelet transform for Mexican hat wavelet. The following equation defines it -

$$\frac{2}{\sqrt{3a}\pi^{\frac{1}{4}}}(1-\frac{x^2}{a^2})exp(-\frac{x^2}{2a^2})$$

  where a is the width parameter
- cwt peaks - Calculates the number of peaks after smoothening the signal using a mexican hat wavelet for different widths. We use widths=1 and 5.
- permutation entropy(Bandt and Pompe 2002) - It is calculated by following a three-step process. First, we divide the signal into windows of size D and strides of tau. Then we replace each window with a permutation that captures its ordinal ranking. Lastly, we calculate the frequency of these permutations and calculate their entropy.
- variation coefficient of signal
- variance of the signal
- skewness of signal
- kurtosis of signal

As we can clearly see, the feature calculation is a complex and tedious process. Despite calculating such advanced features, we get classification results comparable to those of deep learning models. All these features are extracted for every channel of the EEG signal. Thus the total number of features is multiplied by the number of channels present in the dataset. The MBD Muse dataset generates 216 features in total, whereas the MBD Epoc dataset generates 756 features. After extracting the features, we apply Principal Component Analysis(PCA) to reduce the number of dimensions. In the case of MBD Muse, first 30 components capture 99.959% of variance, and for MBD Epoc, 98.829% of variance is captured. Now, these new components are used for classification purposes.

3.5.1 Random Forest

Random Forest is an ensemble technique for combining various decision trees. This algorithm belongs to the class of bagging ensembles. In this category of ensembles, weak classifiers come together to produce a strong classifier. Random forest classification is a technique where multiple decision trees are built from the same dataset with different root nodes. These trees are used for majority voting to get the final class label of the given sample. This way, weak learners are combined via a voting mechanism to improve the accuracy of the machine learning model.

Decision trees are the most explainable machine learning models available. It gives us an intuition about which features are important for classification and is not just a black box as many other deep learning/machine learning models are. The root of a decision tree is chosen in such a way that the cost of splitting (Gini score) is lowest at each step. Gini score quantifies the similarity of samples in the same group. A score of 0 implies that all the samples in the group belong to the same class, whereas the worst score can be 0.5, which implies that 50% of the data belongs to one group. Thus, the decision tree tries to reduce this score at each step while splitting the data based on any node/feature. The decision tree stops splitting the data further when one of the below-mentioned conditions is achieved -

- The samples at leaf nodes are less than the specified minimum number.
- Maximum depth of the tree is reached.

Thus, Random Forest is a strong classifier created from a set of weak classifiers, and it improves the overall performance of the machine learning model.

3.5.2 K-Nearest Neighbors

K-Nearest Neighbors(KNN) is a simple yet powerful supervised learning technique. It is a non-parametric technique developed by Evelyn Fix and Joseph Hodges in 1951. The working of this method is quite easy to follow -

- Given a sample/data point, we first calculate the distance of this point from all other points. The distance is calculated using the euclidean distance.
- Then we select k nearest points/neighbors of that sample point and count the number of points/samples belonging to a particular class.
- We then take a majority voting, and the sample point is assigned the label of majority neighboring points belonging to that class.

Throughout this process, the choice of k is crucial. Usually, k is chosen as an odd number to avoid situations with an even number of samples/data points for both classes. Thus, we employ KNN for our classification task as one of the least complex models available. Furthermore, this will act as a baseline model for comparing with other methods discussed in this study.

3.6 Evaluation metrics

Once the model is ready, we need some metrics for evaluating the model. In this study, since our dataset becomes imbalanced in the case of binary classification, we rely heavily on the F1 score rather than accuracy. The deep learning models and traditional machine learning models are evaluated differently. In deep learning models, the data is split in 90/10 proportion, and we take the average of accuracy and weighted F1 score for the last five epochs whereas, for traditional machine learning, we do 10

fold cross-validation and report the mean of accuracies and F1 scores obtained on each fold. The following section describes all the evaluation methods in detail -

3.6.1 k-fold Cross Validation

Cross-validation is a process of dividing the dataset for training and testing the model. In this process, the dataset is divided into k equal sizes of partitions. In each iteration, one partition is used as testing or validation data, and others are used for training the model. This is shown in Fig 12 for 3 fold cross validation. This process ensures that the model is trained and validated on all the data at least once. This helps in reducing bias and reducing the overfitting of the model.

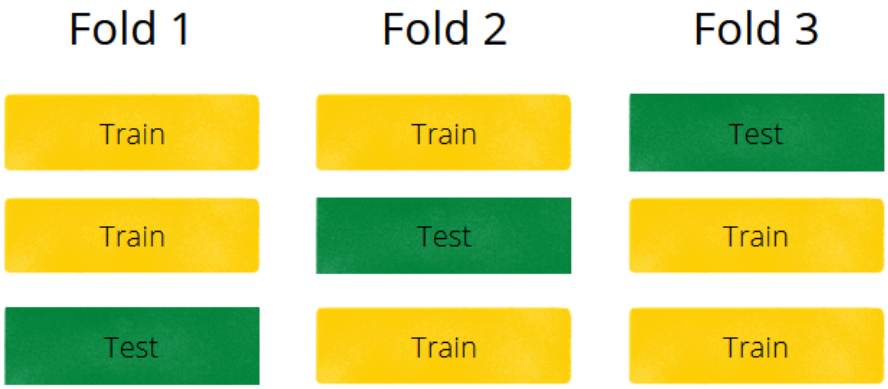

Figure 12: K-fold cross validation

3.6.2 Confusion Matrix

All the evaluation metrics for binary classification used in this study can be traced back to the confusion matrix shown in Table3.

	Predicted YES	**Predicted NO**
Actual YES	TP	FN
Actual NO	FP	TN

Table 3: Confusion Matrix

The same metrics can be used for multiclass classification evaluation with some minor changes. These metrics are discussed below -

- **TP**: True positives are the number of samples belonging to the positive class and are predicted as positive by the model.
- **TN**: True negatives are the number of samples belonging to the negative class and are predicted as negative by the model.
- **FN**: False negatives are the number of samples belonging to the positive class and are predicted as negative.
- **FP**: False positives are the number of samples belonging to the negative class and are predicted as positive.
- **Precision**: Precision quantifies: out of all the predictions, how many predictions were actually correct. It is defined as below -

$$\frac{TP}{TP + FP}$$

- **Recall**: Recall quantifies : out of all the actual positive classes, how many were predicted correctly. It is defined as below -

$$\frac{TP}{TP + FN}$$

- **F1 score**: This is the harmonic mean of precision and recall and is defined as -

$$\frac{2 \times Precision \times Recall}{Precision + Recall}$$

This is slightly modified for multiclass classification, where the f1 score is calculated for each class and is weighted by the number of samples actually belonging to that class(support).

Chapter 4

EXPERIMENTS AND RESULTS

4.1 System configuration

The experiments throughout this study have been carried out on a machine with the following specifications -

1. CPU - 9th Gen Intel Core i7-9750H
2. GPU - NVIDIA Geforce RTX 2060
3. CPU Memory - 32GB RAM, 512GB SSD
4. GPU Memory - 6GB GDDR6
5. OS - Ubuntu 20.04.3 LTS
6. No of cores - 12

4.2 Languages and Software used

1. Python Pandas for data preprocessing
2. Python tsai (PyTorch at backend) (Oguiza 2020) library for implementing Transformer, LSTM, and CNN
3. Python tsfresh(Christ et al. 2018) library for extracting time series features
4. Python sklearn for traditional machine learning models - Random forest and KNN

4.3 Experimental setup

The classification pipeline is explained in Fig 4. We experiment with different deep learning and traditional machine learning classifiers, as explained in the previous section. The goal of experimentation is to evaluate these methods on two datasets - MBD Epoc and MBD Muse. Further, we also experiment with some of the individual models to identify the best performing hyperparameters for that specific model. Finally, all these models are evaluated using the evaluation metrics explained in Section 3.6.

4.4 Transformer architecture for classification

Transformer architecture from (Zerveas et al. 2021) is applied to this dataset. According to (Zerveas et al. 2021), the positional embeddings are not deterministic, i.e., they are learned along with the entire architecture. A batch size of 32 is used for this experiment. The data is split into 90% train and 10% validation. As mentioned before, we do not use the decoder part of the transformer and use five encoder layers for this experiment. The number of multi-attention heads is set to 16, and the input embedding dimension is 128. Batch normalization with gelu activation is applied. Adam optimizer is used with a learning rate of 1.4e-05. The model is trained for 50 epochs. Results are included in Table 6 and Table 7.

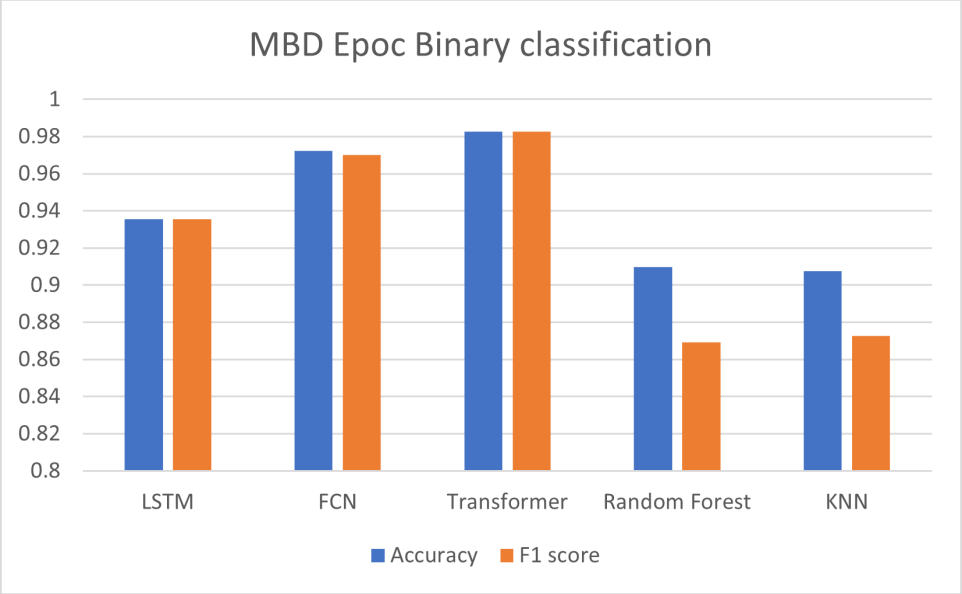

Figure 13: MBD Epoc Binary Classification results

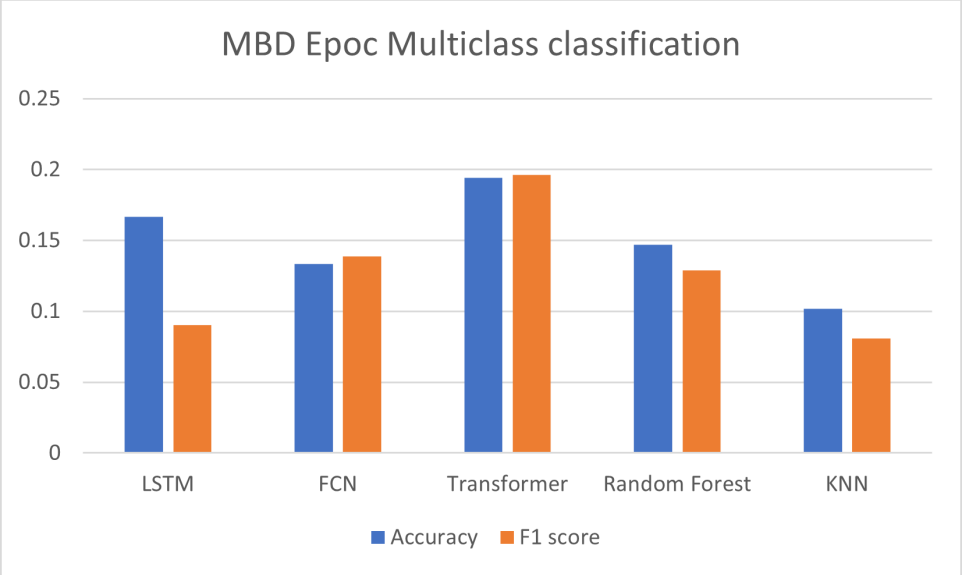

Figure 14: MBD Epoc Multiclass Classification results

4.4.1 Effect of number of Encoder layers

In this experiment, we try to analyze the behavior of the transformer model while varying the number of encoder layers of the architecture. We use the Epoc dataset

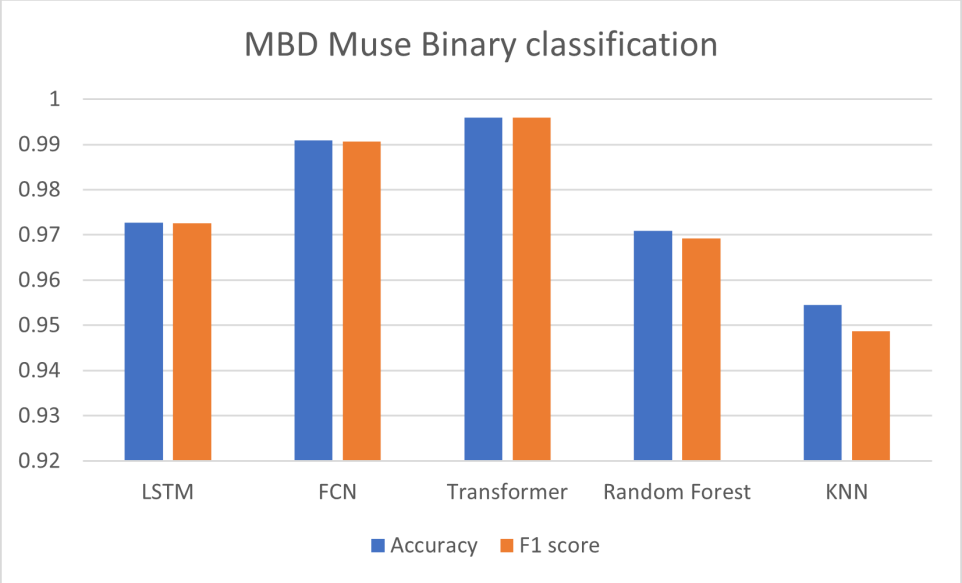

Figure 15: MBD Muse Binary Classification results

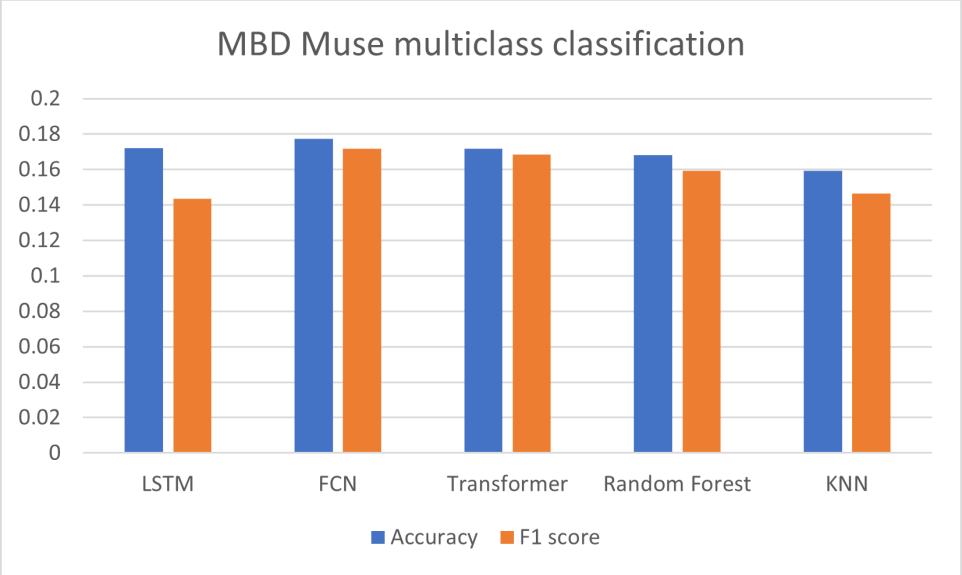

Figure 16: MBD Muse Multiclass Classification results

for experimentation. This dataset is divided into 80%train and 20%validation data. The model is trained for 20 epochs for 1,2,3,5, and 7 Encoder layers. The results are shown in Table 4.

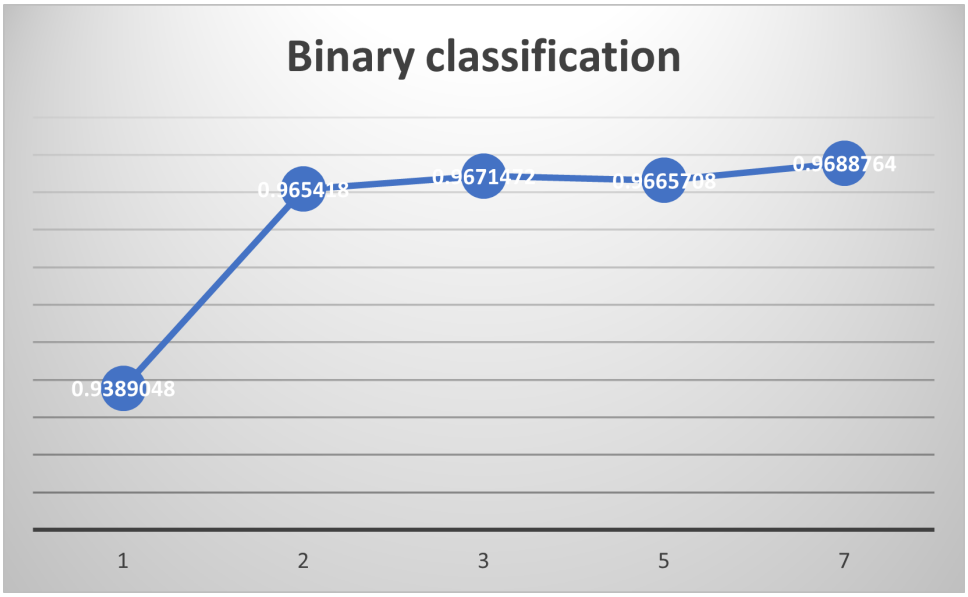

Figure 17: Effect of choosing No. of Encoder layers for Binary classification

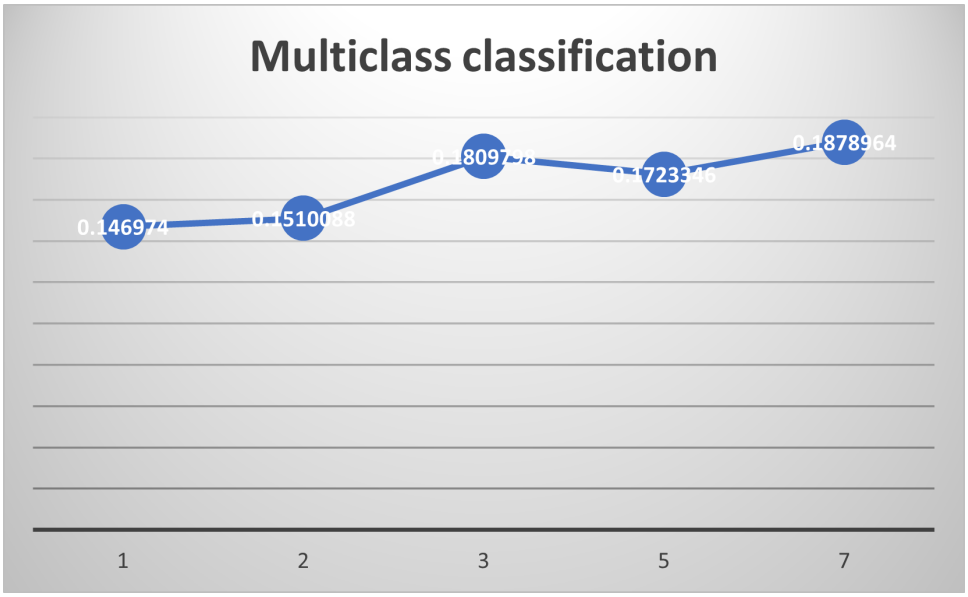

Figure 18: Effect of choosing No. of Encoder layers for Multiclass classification

From the table, we can see that overall, the model's performance increases with an increase in the number of Encoder layers. This is expected as more layers will

Number of layers	Binary classification	Multiclass classification
1	0.9389048	0.146974
2	0.965418	0.1510088
3	0.9671472	0.1809798
5	0.9665708	0.1723346
7	**0.9688764**	**0.1878964**

Table 4: Effect of Encoder layers on accuracy

result in more parameters to train, which can capture more complex patterns from data. However, after 3 Encoder layers, we see that the performance increase is not very significant. Thus we stick to 3 Encoder layers for the next experiments. The final results are show in Table 6 and Table 7. Transformers achieve an accuracy of 0.982759 and F1 score of 0.982508 for MBD epoc dataset which is the for binary classification. It achieves an accuracy of 0.194252 and F1 score of 0.1962678 which is also the highest for multiclass classification for Epoc dataset. As far as Muse dataset is concerned, it has the highest accuracy of 0.995947 and F1 score of 0.995947 for binary classification which is the highest. For multiclass, it achieves an accuracy of 0.171631 and a F1 score of 0.168474.

4.5 Long Short term memory neural networks for classification

We use LSTM(Hochreiter and Schmidhuber 1997) architecture with five layers. These layers are not bidirectional, just simple 5 LSTM layers stacked onto each other. Batch size of 32 is used for this experiment. The data is split into 90% train and 10% validation. Adam optimizer is used with a learning rate of 1.4e-05. Categorical cross entropy is the loss function used for multiclass classification and binary cross entropy is used for binary classification. The final layer of the model outputs the probability distribution over all the classes using the Softmax function. The model is trained

for 50 epochs. Results are included in Table 6 and Table 7. We can see that LSTM achieves an accuracy and F1 score of 0.9356326 and 0.9269712 respectively for binary classification and 0.166667 and 0.0901832 for multiclass classification which is very low compared to other methods. As far as Muse dataset is concerned, it achieves accuracy and f1 score of 0.972727 and 0.9725895 respectively for binary classification and 0.1722221, 0.1434158 for multiclass classification.

4.6 Convolutional Neural Networks for classification

FCN(Wang, Yan, and Oates 2016) is a 1d convolutional neural network with 3 layers. The first layer consists of 128 activation maps and a kernel size of 3 followed by batch normalization and relu, as shown in Fig19This network uses only 1d convolution for extracting features from time-series data. Adam optimizer is used with a learning rate of 0.000575. Categorical cross entropy is the loss function used for multiclass classification and binary cross entropy is used for binary classification. The final layer of the model outputs the probability distribution over all the classes using the Softmax function. Results are included in Table 6 and Table 7. We can see from the tables that FCN achieves an accuracy of 0.9724136 and an F1 score of 0.9701642 for Binary classification on the MBD Epoc dataset. For multiclass classification, it achieves an accuracy of 0.1333334 and an F1 score of 0.1385574 on the same dataset. As far as MBD Muse dataset is concerned, it achieves an accuracy of 0.990909 and an F1 score of 0.990692 which is second best after the transformer architecture. It achieves the best multiclass performance on this dataset with an accuracy of 0.1772726 and an F1 score of 0.1716725. This shows that FCN has a comparable performance to transformer architecture.

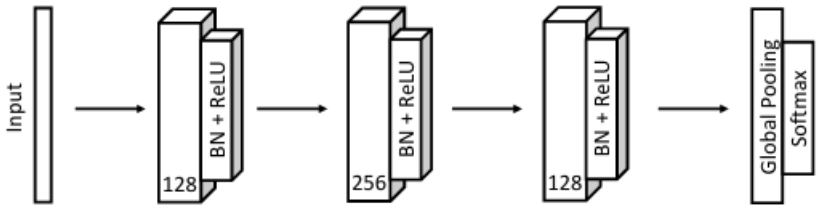

Figure 19: FCN architecture

Source: (Wang, Yan, and Oates 2016)

4.7 K-Nearest Neighbors for classification

K nearest Neighbors is a well-known and relatively not very complex algorithm for classification tasks. We report the average of accuracies and F1 scores over 10 fold cross-validation. We use this classifier and experiment with the only parameter - K. The results are shown in Table 5. We experiment with both the datasets while varying the value of K. We can see from the results that lower values of K give poor results compared to the higher values of K. This is expected as the more the neighbors in the KNN algorithm, the larger is the sample of data points for majority voting giving better results. However, after k=3, we do not see any significant improvement in the performance of the classifier. Thus we choose k=3 as we do not want to increase the model parameters unnecessarily. As shown in Table 6 and Table 7, the accuracy and F1 scores for KNN algorithm are not very high compared to other deep learning models. This is expected as deep learning models extract the features relevant to the task and may have better features compared to traditional features as a result.

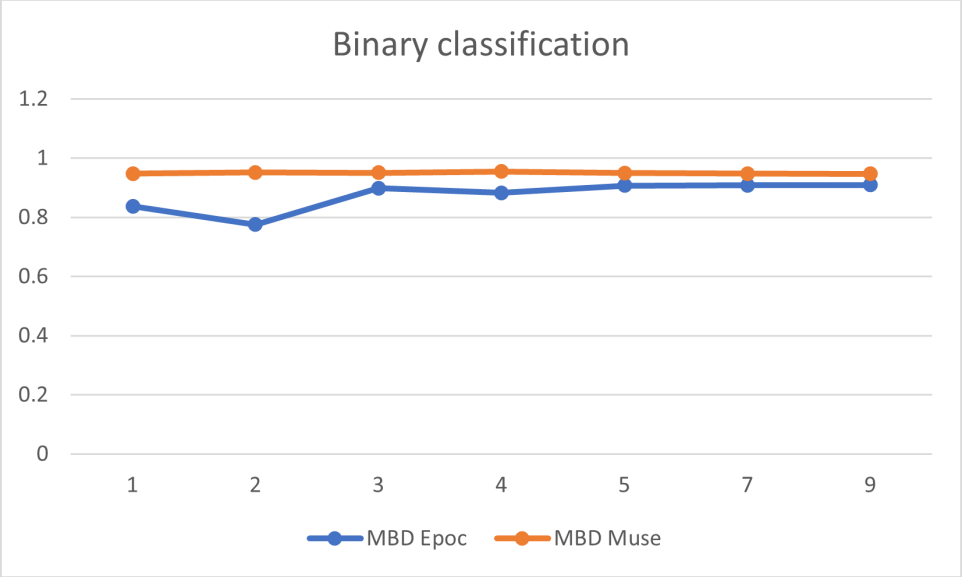

Figure 20: Effect of choosing K on Binary classification

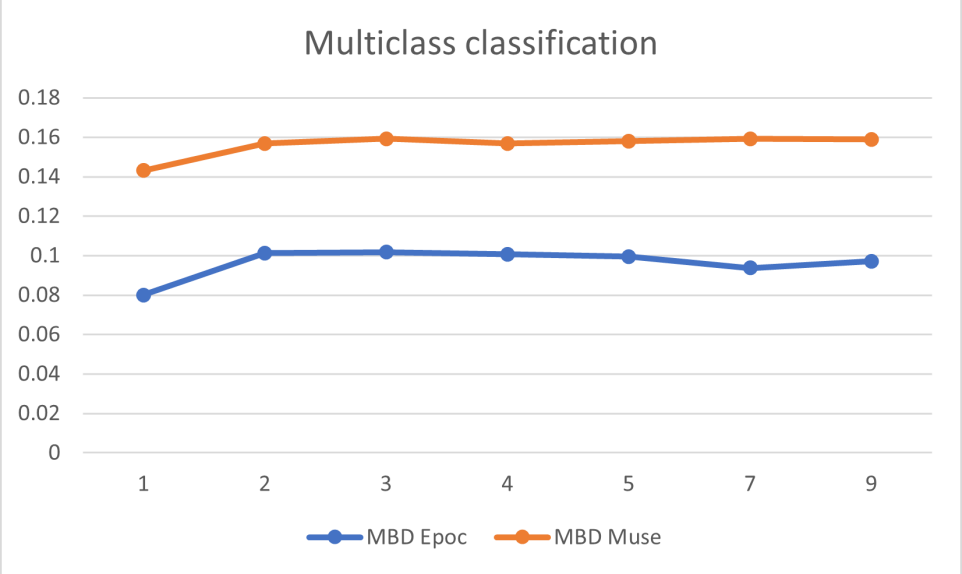

Figure 21: Effect of choosing K for Multiclass classification

4.8 Random Forest

Random forest is an ensemble technique based on bagging. It essentially comprises of multiple decision trees that produce an output label for each input sample. All

Value of K	**Binary**		**Multiclass**	
	MBD Epoc	MBD Muse	MBD Epoc	MBD Muse
1	0.8370706	0.9478787	0.0800426	0.1431313
2	0.7758981	0.9511111	0.1011986	0.1568686
3	0.8982561	0.9507070	**0.1017668**	**0.1592929**
4	0.8822463	**0.9544444**	0.1006174	0.1569696
5	0.9074022	0.9501010	0.0994778	0.1580808
7	0.9079737	0.9479797	0.0937635	0.1591919
9	**0.9091133**	0.9465656	0.0972085	0.1588888

Table 5: Effect of choosing value of K on accuracy

these outputs from different decision trees are combined via voting mechanism where the label corresponding to highest votes is assigned to the new sample. For this experiment, we choose 100 decision trees and maximum depth of 30 for each tree. The input to this classifier are the first 30 components of PCA output. The model is evaluated using 10 fold cross validation method. As shown in Table 6 and Table 7, for MBD Epoc dataset, random forest report an accuracy of 0.909681 and 0.1469228 for binary and multi-class classification respectively. It reports an F1 score of .869268 and 0.1288593 for binary and multi-class classification respectively. Similarly for MBD Muse dataset, it reports an accuracy of 0.9709090 and 0.1679797 for binary and multi-class classification respectively. It reports an F1 score of 0.9692250 and 0.1591471 for binary and multi-class classification respectively. Overall, we can observe that it does not perform well compared to its deep learning counterparts.

Method	**Binary (thinking about digit or not)**		**Multiclass (digits 0-10 and others Total – 11 classes)**	
	Accuracy	F1 score	Accuracy	F1 score
LSTM	0.9356326	0.9269712	0.166667	0.0901832
FCN	0.9724136	0.9701642	0.1333334	0.1385574
Transformer	**0.982759**	**0.982508**	**0.1942526**	**0.1962678**
Random Forest	0.909681	0.869268	0.1469228	0.1288593
KNN	0.9074022	0.8727334	0.1017668	0.0806819

Table 6: Final Results for MBD Epoc dataset

Method	**Binary**		**Multiclass**	
	Accuracy	F1 score	Accuracy	F1 score
LSTM	0.972727	0.9725895	0.1722221	0.1434158
FCN	0.990909	0.990692	**0.1772726**	**0.1716725**
Transformers	**0.995947**	**0.995947**	0.1716312	0.1684744
Random Forest	0.9709090	0.9692250	0.1679797	0.1591471
KNN	0.9544444	0.9486992	0.1592929	0.1463025

Table 7: Final Results for MBD Muse dataset

Chapter 5

DISCUSSION

Chapter 3 and 4 discussed about the methods used and the experiments along with their results. This chapter focuses on findings from these experiments. We also state some limitations of the study and discuss some methods which can improve our work further.

5.1 Findings

As discussed in previous sections, EEG signal classification is perfectly suited for deep learning architectures. Out of all the available deep learning architecture, we implement CNN, LSTM and Transformer architectures which are widely used for modelling time series data in general. We find that among these architectures, Transformers are the best performing ones which is in accordance with the latest research done on these architectures. Thus proving the efficacy of these architectures.

On the other hand, we find that traditional machine learning classifiers do not perform well compared to their deep learning counterparts which is expected. The reason may be because of the features extracted from deep learning models may be more comprehensive compared to hand crafted features for traditional machine learning algorithms.

Transformer being a promising architecture for modelling our data, we do some experiments with the number of encoder layers. We find that generally, the performance of this model increases with increase in number of encoder layers but after 3 layers,

the increase in performance is not significant. Also, as the number of layers increase, the complexity of model increases as well. So having 3 or 5 layers is the optimal solution for this trade off.

Among all the deep learning architectures experimented with, we observe that Transformer architecture outperforms others. This may be attributed to the fact that Transformers utilize the multi-head attention mechanism, which is pivotal in extracting features from time series. Multi-head attention helps to encode the information regarding the context of each time step. Also, learnable positional embeddings may have further improved the performance. We use only the encoder blocks of transformers, which help in extracting features from raw time series. Thus, each time step is considered an embedding vector passed on to the transformer as input. The output is a group of vectors corresponding to each time step. This output is concatenated to form a single matrix which is passed to an MLP layer for classification. Thus, transformer architecture parallelizes the operations providing a speedup over existing deep learning techniques like RNN and LSTM, where the output of each time step depends on previous time steps. Hence, Transformers completely bypass the recursion step in RNNs and LSTMs, thus providing a speedup. However, transformers can be computationally expensive if the length of the time series increases. This can be taken care of by introducing 1-D CNNs as described in (Zerveas et al. 2021), where we do a 1D convolution pass over the time steps to downsample the number of input vectors. Thus given our problem, we observe that transformers perform better than other deep learning architectures for the reasons mentioned above.

In case of traditional machine learning, we extract many features (216 to 756) and only 30 of them capture around 99% of variance. This helps us to reduce the number

of dimensions drastically which further helps in reducing space and time constraints without affecting the performance of classifiers.

The reason for low accuracies on the mentioned datasets can be attributed to the fact that recognising which specific digit is the user thinking is a very difficult task and using only EEG signals can not be sufficient to find patterns. On other hand, detecting if a person is thinking about the digit or not is relatively easy which is observed from the accuracies obtained. Thus multiclass classification will require much more sophisticated methods and corresponding multimodal data for better results.

Table 8 compares our classification pipeline with the existing methods. We can see that every existing approach is different with respect to data preprocessing techniques used, number of samples used and the associated task. Hence a direct comparison is not possible. However, we can see that F1 score obtained with our approach is comparable to existing methods and we believe that F1 score is a better evaluation metric compared to simple accuracy as we deal with imbalanced dataset for binary classification task.

We try to follow (Jolly et al. 2019) with regard to experimental setup and is evident from Table 8 that we achieve better F1 score compared to (Jolly et al. 2019). Thus we can say that Transformers perform better compared to existing deep learning models like CNN, GRU, LSTM, Autoencoder etc.

5.2 Limitations

The sample size for MBD Epoc data for this study is small compared to entire dataset. This is because of the memory and time resource constraints. Applying these deep learning models on entire dataset could give better results.

Classification approach	**Accuracy**	**F1 score**	**No. of samples used**	**Dataset used**	**Classification task**
(Jolly et al. 2019)	0.338	0.16	40983	Muse	Multiclass (11 claass)
(Jolly et al. 2019)	0.993	0.991	40983	Muse	Binary (-1 vs rest)
(Bird et al. 2019)	0.3135	-	150	Muse	Multiclass (10 class)
(Pratama, Kesiman, and Gunadi 2021)	0.31	-	-	Muse	Multiclass (11 class)
(Pratama, Kesiman, and Gunadi 2021)	0.123	-	400	Epoc	Multiclass (10 class)
Our Approach	0.1942526	0.1962678	1749	Epoc	Multiclass (11 class)
	0.1772726	0.1716725	9900	Muse	Multiclass (11 class)
	0.982759	0.982508	1749	Epoc	Binary (-1 vs rest)
	0.995947	0.995947	9900	Muse	Binary (-1 vs rest)

Table 8: Comparison with existing approaches

This study focuses on application of deep learning models for EEG classification. But exploring some ensemble methods in traditional machine learning could provide us some more insights about their comparative performance with deep learning counterparts.

Being an extremely difficult dataset for EEG classification, experimenting with some more popular datasets could help evaluate the true potential of the deep learning models discussed in this study.

Very few hyper-parameters were tuned for the deep learning models in this study.

Exploring some more hyper-parameter tuning could lead to models with better performance.

This study focuses on EEG signals collected from commercial headbands available in the market. EEG signals collected from such devices are not known to be very clean and contains noise. Experimenting with medical grade EEG signals might lead to a better performance of models.

Data preprocessing involves a process for rejecting samples. First, we calculate the maximum value a given EEG signal for a sample can have. Similarly, we calculate the minimum value for each sample. We then take the average of these obtained values over all samples to get the average max and minimum values. These two values act as a threshold for rejecting our samples. If the sample contains any value above/below the given threshold, we reject the sample. The intuition behind this process was to reject samples with EEG channels having abnormally high or low values compared to other channels. However, this process may not be the best possible process for selecting good samples. We may lose some samples with peak values above the threshold and not necessarily the channel having abnormally high or low values compared to other channels.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This study develops a classification pipeline for applying deep learning methods - particularly transformer architecture - for classifying the EEG signals in MBD datasets. Very few studies have explored these datasets, and this study helps to explore these datasets by applying various deep learning techniques and reporting their performance. Our comprehensive experiments showed that Transformer architectures not only perform better in Natural Language Processing and Computer Vision tasks but also for EEG signal classification tasks. Also, the features extracted by deep learning models are much better for classification compared to their traditional machine learning counterparts. The experiments show that Transformer architecture performs better than other related deep learning techniques like LSTM and CNN, thus emphasizing the efficacy of Transformer architectures for EEG signal classification. The F1 scores obtained on both datasets for the transformer model are in accordance with the existing literature for this data.

Despite the task for MBD data being extremely difficult, We hope that this study sparks an interest in exploring the MindBigData dataset and applying deep learning techniques for Brain-Computer Interfacing.

6.2 Future work

This study focuses on specific kinds of EEG signals captured from commercial EEG headbands and can be extended to other EEG signals as well. Below listed are some future works for this study -

- This study relies completely on EEG signals for classifying what the user is thinking about. Instead, we can apply deep learning techniques to multi-modal data, which has EEG and other signals related to brain activity. Recently (Oct 2021), such a dataset has been published as a subset of MindBigData. Seeing the performance of transformer architecture on this data could be interesting. Since fMRI is known for its good spatial resolution data, we can combine BOLD signals with EEG signals to get multi-modal data with good temporal and spatial resolution.
- It would be interesting to see the applications of transformer models to medical-grade EEG signals.
- EEG signals are applied for epileptic seizure detection, and hence this study can be extended to classify EEG for seizure prediction.

REFERENCES

Bandt, Christoph, and Bernd Pompe. 2002. "Permutation Entropy: A Natural Complexity Measure for Time Series." *Phys. Rev. Lett.* 88 (17): 174102. https://doi.org/10.1103/PhysRevLett.88.174102.

Batista, Gustavo E. A. P. A., Eamonn J. Keogh, Oben M. Tataw, and Vinicius M. A. Souza. 2013. "CID: an efficient complexity-invariant distance for time series." *Data Mining and Knowledge Discovery* 28:634–669.

Berger, Hans. 1929. "Über das Elektrenkephalogramm des Menschen." *Archiv für Psychiatrie und Nervenkrankheiten* 87, no. 1 (December): 527–570. https://doi.org/10.1007/BF01797193.

Bird, Jordan J., Diego R. Faria, Luis J. Manso, Anikó Ekárt, and Christopher D. Buckingham. 2019. "A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction." *Complexity* 2019 (March): 4316548. https://doi.org/10.1155/2019/4316548.

Christ, Maximilian, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)." *Neurocomputing* 307:72–77. https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067.

Gong, Shuxiao, Kaibo Xing, Andrzej Cichocki, and Junhua Li. 2020. "Deep Learning in EEG: Advance of the Last Ten-Year Critical Period." *ArXiv* abs/2011.11128.

Gu, Xiaotong, Zehong Cao, Alireza Jolfaei, Peng Xu, Dongrui Wu, Tzyy-Ping Jung, and Chin-Teng Lin. 2021. "EEG-based Brain-Computer Interfaces (BCIs): A Survey of Recent Studies on Signal Sensing Technologies and Computational Intelligence Approaches and Their Applications." *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* 1–1. https://doi.org/10.1109/TCBB.2021.3052811.

Hendrycks, Dan, and Kevin Gimpel. 2020. *Gaussian Error Linear Units (GELUs).* arXiv: 1606.08415 [cs.LG].

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Comput.* (Cambridge, MA, USA) 9, no. 8 (November): 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Jolly, Baani Leen Kaur, Palash Aggrawal, Surabhi S Nath, Viresh Gupta, Manraj Singh Grover, and Rajiv Ratn Shah. 2019. "Universal EEG Encoder for Learning Diverse

Intelligent Tasks." In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM),* 213–218. https://doi.org/10.1109/BigMM.2019.00-23.

Kawala-Sterniuk, Aleksandra, Natalia Browarska, Amir Al-Bakri, Mariusz Pelc, Jaroslaw Zygarlicki, Michaela Sidikova, Radek Martinek, and Edward Jacek Gorzelanczyk. 2021. "Summary of over Fifty Years with Brain-Computer Interfaces-A Review" [in eng]. Brainsci11010043[PII], *Brain sciences* 11, no. 1 (January): 43. https://doi.org/10.3390/brainsci11010043.

Lawhern, Vernon J, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. 2018. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces." 15, no. 5 (July): 056013. https://doi.org/10.1088/1741-2552/aace8c.

Mozaffari, M. Hamed, and Li-Lin Tay. 2020. *A Review of 1D Convolutional Neural Networks toward Unknown Substance Identification in Portable Raman Spectrometer.* arXiv: 2006.10575 [eess.SP].

Oguiza, Ignacio. 2020. *tsai - A state-of-the-art deep learning library for time series and sequential data.* Github. https://github.com/timeseriesAI/tsai.

Pratama, I Wayan, Made Windu Antara Kesiman, and I Gede Aris Gunadi. 2021. "Frequency Band and PCA Feature Comparison for EEG Signal Classification." *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi* 12 (1): 1–12. https://doi.org/10.24843/LKJITI.2021.v12.i01.p01.

Roy, Y., H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert. 2019. "Deep learning-based electroencephalography analysis: a systematic review." *J Neural Eng* 16, no. 5 (August): 051001.

Schreiber, Thomas, and Andreas Schmitz. 1997. "Discrimination power of measures for nonlinearity in a time series." *Phys. Rev. E* 55 (5): 5443–5447. https://doi.org/10.1103/PhysRevE.55.5443.

Song, Huan, Deepta Rajan, Jayaraman J. Thiagarajan, and Andreas Spanias. 2017. *Attend and Diagnose: Clinical Time Series Analysis using Attention Models.* arXiv: 1711.03905 [stat.ML].

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention is All you Need." In *Advances in Neural Information Processing Systems,* edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett,

vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, Zhiguang, Weizhong Yan, and Tim Oates. 2016. "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline." *CoRR* abs/1611.06455. arXiv: 1611.06455. http://arxiv.org/abs/1611.06455.

Wilson, Granville Tunnicliffe. 2016. "Time Series Analysis: Forecasting and Control, 5th Edition , by George E. P. Box , Gwilym M. Jenkins , Gregory C. Reinsel and Greta M. Ljung , 2015 . Published by John Wiley and Sons Inc. , Hoboken, N." *Journal of Time Series Analysis* 37, no. 5 (September): 709–711. https://ideas.repec.org/a/bla/jtsera/v37y2016i5p709-711.html.

Wolpaw, J., and E.W. Wolpaw, eds. 2012. *Brain–Computer Interfaces: Principles and Practice* [in en]. Accessed October 20, 2021. https://oxford.universitypressscholarship.com/view/10.1093/acprof:oso/9780195388855.001.0001/acprof-9780195388855..

Zerveas, George, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. "A Transformer-Based Framework for Multivariate Time Series Representation Learning." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining,* 2114–2124. KDD '21. Virtual Event, Singapore: Association for Computing Machinery. https://doi.org/10.1145/3447548.3467401.