UNIVERSITY OF PIRAEUS

School of Information Technologies and Communication Technologies

Department of Informatics

Thesis Title	Machine learning methods for EEG signal classification
Τίτλος Πτυχιακής Εργασίας	Μέθοδοι μηχανικής μάθησης για την ταξινόμηση σημάτων ηλεκτροεγκεφαλογραφίας
Student full name	Vasileios Chartomatzidis
Father's name	Odisseas
Student ID	P18171
Supervisor	Aggelos Pikrakis, Assistant Professor

Submission date September 2024

Copyright ©

Table of Contents

Abstract

Electroencephalography (EEG) together with a Brain-computer interface (BCI), is a common approach in brain monitoring and control of prosthetic devices. One common challenge in research applications is the cross-subject classification of the motor movement and motor imagery signals, as the EEG recordings are highly individualized. In this project, a Shallow 2D Convolutional Neural Network was developed, which was trained on lightly preprocessed EEG data. The data was provided by the PhysioNet EEG dataset. The dataset contained 109 subjects, with each subject having recordings of motor imagery and motor movements. The developed model in combination with the segmented data achieved high statistical results (2 classes: 80%, 4 classes: 58%), comparable to other results from research papers in this domain.

The code for this assignment is provided at the link bellow
https://github.com/VasilhsXart/EEGClassification.git

Η ηλεκτροεγκεφαλογραφία (EEG) σε συνδυασμό με ένα σύστημα διεπαφής εγκεφάλου-υπολογιστή (Brain-computer interface BCI), αποτελεί μια συνήθης προσέγγιση για την παρακολούθηση των ηλεκτρικών σημάτων του εγκεφάλου και τον χειρισμό προσθετικών συσκευών. Μια κοινή πρόκληση στον ερευνητικό τομέα είναι η ταξινόμηση των σημάτων που απεικονίζουν σωματική και πλασματική κίνηση ανάμεσα σε διαφορετικά πρόσωπα, καθώς τα εγκεφαλοεγραφήματα διαφέρουν από άτομο σε άτομο. Στην παρούσα πτυχιακή, αναπτύχθηκε ένα ρηχό συνελεκτικό νευρωνικό δίκτυο (CNN), το οποίο εκπαιδεύτηκε σε ελαφρώς προεπεξεργασμένα σήματα EEG. Τα δεδομένα προήλθαν από την βάση δεδομένων της PhysioNet. Το σύνολο των δεδομένων περιλάμβανε 109 εθελοντές, με κάθε εθελοντή να έχει καταγραφές σημάτων κίνησης και πλασματικής κίνησης. Το ανεπτυγμένο μοντέλο, σε συνδυασμό με τα προεπεξεργασμένα δεδομένα, απέδωσε υψηλά στατιστικά αποτελέσματα(2 κλάσεις 80%, 4 κλάσεις: 58%), συγκρίσιμα με άλλα αποτελέσματα από ερευνητικές εργασίες σε αυτόν τον τομέα.

Ο κώδικας της εργασίας βρίσκεται στον παρακάτω σύνδεσμο
https://github.com/VasilhsXart/EEGClassification.git

1. Introduction

Electroencephalogram (EEG) is a low cost, typically non-invasive method that is used to record the spontaneous electrical activity of the brain [1]. EEG is used in the medical sector to monitor and identify various neurological conditions, as well as in neuroscience where it's used to monitor consciousness states and brain activity during various tasks and behaviors.

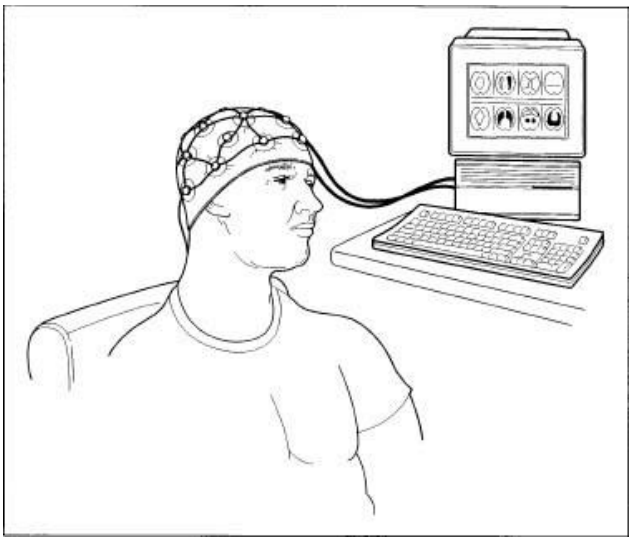

Figure 1. EEG in combination with a BCI device. [3]

Brain-computer interfaces (BCIs) facilitate a communication link between an external device and a brain's electrical signals. Frequently used in research and medicine, BCIs can have many applications, such as controlling prosthetics in rehabilitation cases or recordings of brain activity for diagnostic purposes [2].

By combining these two technologies as shown in **Figure 1**, we can have real-time brain monitoring in combination with the control of an external device such as the forementioned prosthetics to record Motor Movements (MM) or Motor Imagery (MI). One such example is given to us by Schalk et al. [4], where it showcases the capabilities of the BCI2000 system, which 'can incorporate alone or in combination any brain signals, signal processing methods, output devices, and operating protocols' (p.1034). The EEG signals that were recorded are publicly available at PhysioNet [5].

Due to the highly individualized nature of the EEG recordings, it is difficult to analyze signals, which come from cross-subject sessions, and be able to figure out correlations between them [6]. This in turn becomes a classification problem. One common technique that is used in classification problems, is the use of machine learning techniques, more specifically Neural Networks.

Neural Networks are inspired by the structure and function of biological neural networks in animal brains. It consists of several connected artificial neurons that constantly update based on custom parameters. A Neural Network is used for various tasks where the purpose is to derive conclusions from complex information.

With the help of a Neural Network, the software can identify similar and relevant features between the signals produced by the EEG recordings and captures by the BCI device. That way the Neural Network can classify, process or decode these signals.

There have been many different research studies in EEG MM/MI classification using different models that categorize the signals with high accuracy. In existing research, the review by Altaheri et al. [7] shows that the most common approach is by using a Convolutional Neural Network (CNN) alone or in combination with other NNs to work as the classifying model as presented in **Figure 2**. Another notable point in the review, is the high preference for raw 2D matrices when training a CNN.

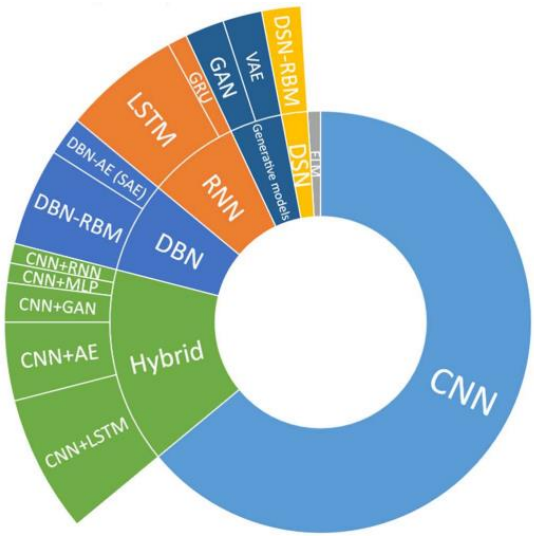

Figure 2. Deep learning methods across all studies. [7]

Amongst the many studies, the work done by Dose et al. [8] is noteworthy for also using PhysioNet's EEG data and achieving a very high accuracy of 80.4% in 2 class classification by using raw 2D matrices on a 1D CNN model. Despite the model being shallow, with only 2 convolutional layers, the researchers achieved high results showcasing what can be achieved with raw data and a simple model.

While using an LSTM is a less popular method, some studies were able achieve results comparable to the CNN approach. For example, the study by Wang et al. [9] achieved an accuracy of 79.6% while using an LSTM on the BCI-C IV-2a dataset. The study used feature extraction, specifically the statistical measures analysis technique instead of raw signals.

Most of the research studies identified in [7], use 2D matrices or extract features out the signal for the purpose of classification. There is a lack of 2D CNN research done to show the effect of using 2D convolutional layers and treating the input signal as an image. When treating the signal as an Image, the 2D convolutional layers can capture and learn patterns across all channels simultaneously providing better results for sequential data [10].

The purpose of this thesis is to provide a 2D CNN solution, with results comparable to previously mentioned research studies, by using 2D convolutional layers with the intent of treating the EEG signals as images. There will also be experiments made to customize the data to test the model on different inputs. This attempt will be made using the Python Tensorflow Keras library for the development of the model, on the PhysioNet using the MNE package [11] on the EEG database, by cutting down the signals into smaller sections which contain only a specific motor movement. Afterwards the signals will be processed by the 2D CNN and will be classified into their corresponding categories.

2. Methods

2.1. Dataset

The data provided by the EEGMMIDB [5] dataset is of 109 participants, with each subject going through 14 experimental runs. The data was recorded is using the BCI2000 System [2], which is an open-source general-purpose software used for BCI research, as it handles the acquisition and preprocessing of the EEG data. The dataset is provided in the European Data Format (EDF) with each file containing the 64 signals, which are sampled at 160 samples per second. The 64 electrodes were recorded as per the international 10-10 system as shown in **Figure 3**.

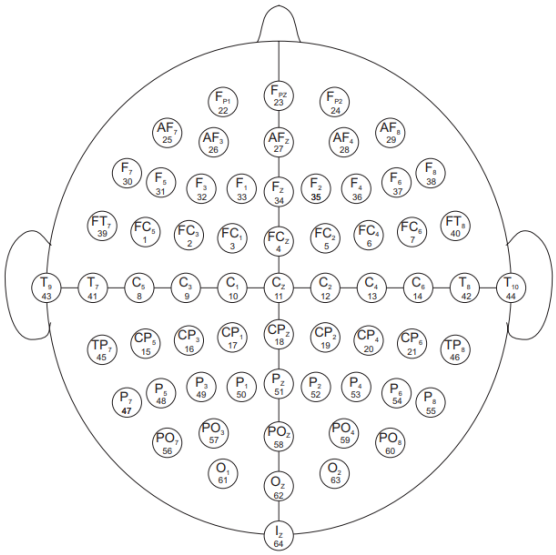

Figure 3. The 64 electrodes as per the international 10-10 system. [12]

The first 2 experiments are two-minute experimental baseline runs with the participants having his eyes open and closed respectively. The rest of the 12 experimental runs are 4 tasks which are repeated 3 times. The 4 tasks are:

1. A target appears on either the left or the right side of the screen. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes.
2. A target appears on either the left or the right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes.
3. A target appears on either the top or the bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.
4. A target appears on either the top or the bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

In summary, the experimental runs were:

	Run 1	Run 2	Run 3
Baseline, eyes open	open and close left or right fist	Task 1	Task 1
Baseline, eyes closed	imagine opening and closing left or right fist	Task 2	Task 2
	open and close both fists or both feet	Task 3	Task 3
	imagine opening and closing both fists or both feet	Task 4	Task 4

Table 1. Summary of the experimental runs. 2 Baseline runs, followed by 4 experimental runs repeated 3 times.

Each EDF file is annotated with each annotation corresponding to the movement the subject is performing at the specified timepoint.

Each annotation includes one of three codes (T0, T1, or T2):

- **T0** corresponds to rest
- **T1** corresponds to onset of motion (real or imagined) of
    - the left fist (in runs 3, 4, 7, 8, 11, and 12)
    - both fists (in runs 5, 6, 9, 10, 13, and 14)
- **T2** corresponds to onset of motion (real or imagined) of
    - the right fist (in runs 3, 4, 7, 8, 11, and 12)
    - both feet (in runs 5, 6, 9, 10, 13, and 14)

The MNE Package is used for the purpose of reading the EDF files from the EEGMMIDB dataset, specifically the SxxxRxx.edf files, as they contain the signal we want to use. The MNE package was used as it contains easy-to-use functions for the purpose of reading, processing, and handling the EEG signals.

2.2. Preprocessing

The processing of the EDF files starts with simple techniques used for preprocessing signals:

A second-order IIR zero-phase notch filter is applied on the signal per frequency, with the targeted frequency being 50 Hz. The FIR filter is provided by the MNE package, with its window type being Hamming.

A bandpass filter is applied on the signal with the low cutoff frequency being at 0 Hz and the high cutoff frequency being at 40 Hz. The FIR filter is provided by the MNE package, with its window type being Hamming.

The sampling frequency of the signal was adjusted to 160 samples.

After applying the above techniques on the signal, the annotated sections of the signal are extracted to be cut down into chunks, where each chunk contains a specific activity. The chunks are then categorized for training, based on the activity that the subject is doing. The classes that were chosen are shown in **Table 2**.

Action	**Imagination**
LeftHand	LeftHand
RightHand	RightHand
BothHands	BothHands
BothFeet	BothFeet

Table 2. The labelled Action Movements and Imagery Movements from the experimental runs.

The **T0** data, which corresponds to rest periods, was not used in this study therefore it was removed.

Afterwards the data is organized into two lists, one containing the Chunks which contain the segmented activities, and the other containing their Classes which are presented in their categorical form (example left hand = 2 → class = 2 → class → [1,0]) for more efficient classification. At the same time, the maximum length of a Chunk sequence is being stored in a value, while also ignoring Chunks with a length sequence of above 1000 as they are problematic Signals from the original dataset.

After labeling each chunk with the forementioned classes, each chunk is then padded to reach the highest sample size using the stored maximum length.

With the now padded chunks, we go through each channel and apply Z-Score normalization shown in **Equation 1** to standardize the data, so that it can fit a normal distribution and improve the machine learning performance.

$$Z = \frac{\chi - \mu}{\sigma}$$

Equation 1. Z-Score normalization. Z equals to the standard score, χ equals to the observed value, μ equals to the mean of the sample and σ equals to the standard deviation of the sample.

2.3. Baseline Model Architecture

For this thesis, a feed-forward Neural Network is used. Specifically, a shallow Convolutional Neural Network (CNN) is used to classify each MM/MI signal to its respective class. **Table 3** is the structure obtained from Python's TensorFlow Keras package. The developed model is portrayed in **Figure 4**.

Layer (Type)	Output Shape	Param #
reshape (Reshape)	(None, 616, 64, 1)	0
conv2d (Conv2D)	(None, 607, 1, 32)	20,512
activation (Activation)	(None, 607, 1, 32)	0
max_pooling2d (MaxPooling2D)	(None, 151, 1, 32)	0
batch_normalization (BatchNormalization)	(None, 151, 1, 32)	128
spatial_dropout2d (SpatialDropout2D)	(None, 151, 1, 32)	0
conv2d_1 (Conv2D)	(None, 147, 1, 64)	10,304
activation_1 (Activation)	(None, 147, 1, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 36, 1, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 36, 1, 64)	256
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295,040
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 4)	516

Table 3. The developed neural network in array form. Each row of the table corresponds to a layer in the Neural Network.

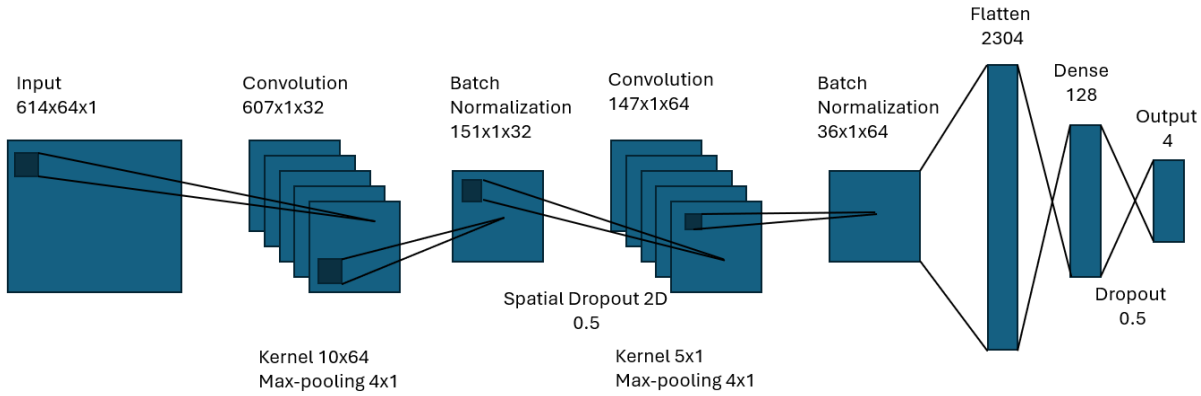

Figure 4. The developed model.

The developed Neural Network model starts by reshaping the input data using the **Reshape Layer**. This transforms the data from [data, channels] to [data, channels, layers] as that allows the application of 2D Convolutional Layers on 3D matrixes.

At the core of the CNN model, the **2D Convolutional Layer** functions as a small window (which is the kernel) that goes through the input serially (sliding), adjusting its variables as it moves along the array. By configuring a [x,64] kernel in the first layer, the model can learn patterns across all channels simultaneously.

Following the convolutional layers, the **Activation layer** (or Activation function), determines if a neuron should be activated or not based on the input value. By applying a non-linear activation function, the model can capture complex patterns and relationships in the data.

Next, the model has a **Max Pooling 2D layer**. This layer reduces the spatial dimensions of the data by down-sampling, which means it only retains the most important features by selecting the maximum value from each small region within the convolutional output. By reducing the dimensionality, the Max Pooling layer preserves important information while decreasing computational load, which helps make the model more efficient.

To further improve performance and training stability, a **Batch Normalization layer** is then used. This layer normalizes the input data, which helps to reduce the sensitivity of the model to initial weight parameters. In turn, batch normalization contributes to faster convergence and better generalization, making the model both more stable and less likely to overfit.

With this architecture, the segmented data is processed through the Neural Network, where the Convolutional layer extracts the most important features from all channels, allowing the model to recognize the most important features of the segmented signals.

A **Spatial Dropout layer** is used which unlike traditional dropout, which randomly drops individual neurons, Spatial Dropout randomly drops entire feature maps during training. By removing entire feature maps, Spatial Dropout reduces the model's reliance on specific sets of features and prevents overfitting.

Finally, after further layers of convolution, activation, max pooling, and normalization, the data is **flattened** and fed into a **Dense** (fully connected) **Layer** with 128 units, followed by a dropout layer to further stop overfitting. The network concludes with another **Dense layer** that outputs the class predictions using the loss function.

2.4. Training and Testing Strategy

Table 4 shows the values of the hyperparameters used for the training of the model.

Parameter	Value
Optimizer	Adam
Activation	ELU
Padding	Valid
Pool Size	[4, 1]
Loss function	Categorical cross-entropy
Batch size	256
Epochs	10

Table 4. The hyperparameters of the developed Neural Network.

For the hyperparameters of the model, the Adam optimizer shown in **Equation 2** is used as its optimization algorithm. Its widely used in Neural Network development as it is effective and efficient in training [13]. The loss function is set as categorical cross-entropy as it is commonly used for classification problems.

$$H(y, \hat{y}) = -\sum_{i} y_i \, log(\hat{y}_i)$$

Equation 2. The categorical cross-entropy function. y is the true class distribution while ŷ is the predicted class distribution.

Categorical cross entropy compares the output given from the previous soft max function, which the output contains the probabilities of the classes the signal belongs to. The highest probability is the one that the signal belongs to.

For the Convolutional Layer, the Exponential Linear Unit (ELU) shown in **Equation 3** is chosen as the activation function as it can handle negative inputs. The activation function is used to determine the output of each node in the neural network. They are crucial as they determine whether a neuron is activated or not. If a neuron is not activated, the value is not passing through the rest of the Neural Network.

$$f(x) = \begin{cases} x & x > 0 \\ a(e^x - 1) & x \leq 0 \end{cases}$$

Equation 3. The Exponential Linear Unit function. y is the true class distribution while ŷ is the predicted class distribution. x is the input of the function, and a is the parameter that controls the negative values. The default value of a is 0.

It is computationally more expensive than ReLU, but it allows for faster learning, while also preventing dead neurons (if their output is zero, a common problem in ReLU) as it allows the negative values.

The Convolutional Layer also supports padding, and the parameter for it is set as Valid. This is done as to not extend the window size by adding zeroes which may introduce artifacts near the borders of the signal.

The Max Pooling 2D layer allows for the parameterization of its pool size. This value is set at [4, 1], as the objective of the layer is to focus on the most important features of the timesteps but keep the number of channels intact.

Finally, for the training of the model, the batch size has been set to 256 with 10 epochs of training. The batch size can be changed as it does not affect the training results. The epochs have been set to 10, as after that the model starts overfitting.

Below are the shapes and sizes of the inputs in the 3 executions made for the dataset. The final arrays are stated like [all signals, channels, signal size]

Afterwards the data is split into Train, Test and Validate sets with them being shown in **Table 5.**

Data	Percentage
Train Data	70%
Test Data	15%
Validation Data	15%

Table 5. Represents the percentages chosen to split the data into Train, Test and Validate sets.

2.5. Experiments

2.5.1. Comparative Models

For this thesis, different models were introduced to compare their results against the baseline model using the same data.

At first, a shallow Long Short-Term Memory (LSTM) Model was introduced.

Layer (Type)	Output Shape
lstm (LSTM)	(None, 821, 16)
batch_normalization (BatchNormalization)	(None, 821, 16)
lstm_1 (LSTM)	(None, 32)
batch_normalization_1 (BatchNormalization)	(None, 32)
dense (Dense)	(None, 32)
dense_1 (Dense)	(None, classes)

Table 6. The model architecture of the LSTM model.

The LSTM model is a recurrent neural network, which is particularly effective for working with sequential data, such as an EEG signal. LSTM models are not often used in EEG classification problems on their own (**Figure 2**), so an attempt had been made to benchmark it against the developed model. The architecture of the model is shown in **Table 6**.

The second model is a CNN with 1D convolutional layers.

Layer (Type)	Output Shape
conv1d (Conv1D)	(None, 814, 32)
activation (Activation)	(None, 814, 32)
max_pooling1d (MaxPooling1D)	(None, 203, 32)
batch_normalization (BatchNormalization)	(None, 203, 32)
spatial_dropout1d (SpatialDropout1D)	(None, 203, 32)
Conv1d_1 (Conv1D)	(None, 196, 64)
activation_1 (Activation)	(None, 196, 64)
max_pooling1d_1 (MaxPooling1D)	(None, 49, 64)
batch_normalization_1 (BatchNormalization)	(None, 49, 64)
spatial_dropout1d (SpatialDropout1D)	(None, 49, 64)
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)
dense (Dense)	(None, 64)
dense_1 (Dense)	(None, 4)

Table 7. The model architecture of the 1D CNN model.

A 1D CNN model was introduced as to compare it with its 2D counterpart. The 1D convolutional layer uses a kernel, which goes through each channel, unlike its counterpart which can go through the whole signal at the same time. The architecture of the model is shown in **Table 7**.

2.5.2. Data Experiments

In the data preparation phase, experiments have been made on the EEG Signals to further customize the data, as to provide better results in the baseline model's training.

The **first approach** was to select specific channels, which have been shown to provide information related to the MM/MI signals. Numerous studies [14] have been made to identify which channels have been shown to affect motor imagery classifications the most. For this thesis 17 prevalent channels have been chosen out of 64. The 17 channels are depicted in **Figure 5**.

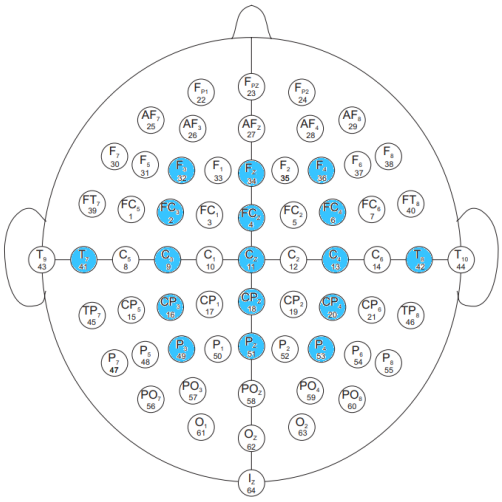

Figure 5. The selected channels on the 10-10 international system [C3, C4, Cz, Cp3, Cp4, Cpz, Fc3, Fc3, Fcz, F3, F4, Fz, P3, P4, Pz, T7, T8]. [15]

By using less channels that focus on precise information from the brain, an attempt to train the model with more specialized data can be made, as to verify the significance of the channels. This can be done by choosing the channels before processing the signal.

The **second approach** was to use a method to remove artifacts from the EEG signal. ICA is a popular method used for artifact removal in EEG data. Artifacts can be eye blinks, muscle movements, and heartbeats that are often mixed with the brain signals. ICA can separate the artifacts from the brain signal as shown in **Figure 6** allowing for a clearer overview of the neural activity.

By using ICA, an attempt to train the model with less noise caused during the recording session can be made, as to verify the significance of the noise. The function used in the program was provided by MNE. For proper application of the ICA method, in preprocessing, the band pass filter was adjusted to start from 1.0 Hz as it improves the results.

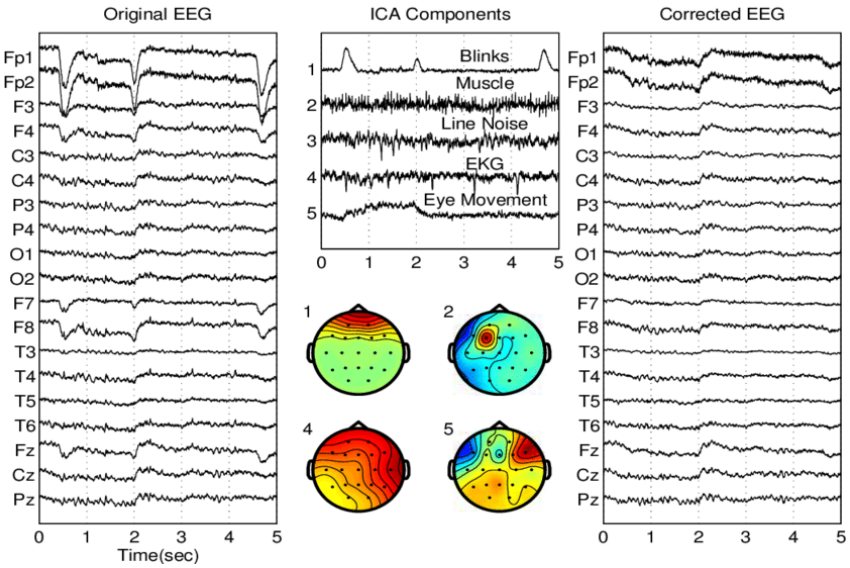

Figure 6. (left) A 5-sec portion of an EEG time series. (center) ICA components accounting for eye movements, cardiac signals, and line noise sources. (right) The same EEG signals corrected for artifacts by removing the ve selected components. [16]

The algorithm used in the parameters was Infomax, as it is more robust than FastICA, and FastICA was not able to converge on the dataset that was provided. The number of components chosen was 20.

The **third approach** is to combine the specialized channels and the ICA method to train the model with more specialized data inputs rather than the general data given from the start.

3. Results

3.1. Baseline Model

Using the developed baseline model in combination with lightly processed data, 3 different executions were made with 2, 4 and 8 classes.

The results of the executions are shown in **Table 8**.

	2 Classes		4 Classes		8 Classes	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Fit	0.8617	0.3105	0.6495	0.8982	0.4217	1.5438
Evaluation	0.8024	0.4382	0.5859	1.0887	0.3460	1.7515
Prediction	0.8005	0.4461	0.5801	1.0995	0.3486	1.7361

Table 8. The results of the developed model's executions on 2, 4 and 8 classes.

The model achieves good results, 30% above the random base chance of 50% and it is comparable to the research paper [8] with an accuracy of 80.4%, which also works with two classes. It shows signs of overfitting after the preselected 10 epochs.

The input data is [4927, 64, 821] and the distribution of the two classes is shown in **Table 9**.

Classes	**Size**
LeftHandAction	2471
RightHandAction	2456

Table 9. Data distribution on 2 Classes.

For the 2 classes, the confusion matrix on **Table 10** shows that the data is preferring the right hand over the left hand in training.

		Predicted Classes	
	Classes	**LH**	**RH**
Actual Classes	**LH**	**0.79**	0.21
	RH	0.18	**0.82**

Table 10. The 2 Class Confusion Matrix of the baseline model. LH stands for Left hand and RH stands for right hand.

The classification graph on **Figure 7** shows that after the 9th epoch the model slowly starts overfitting.

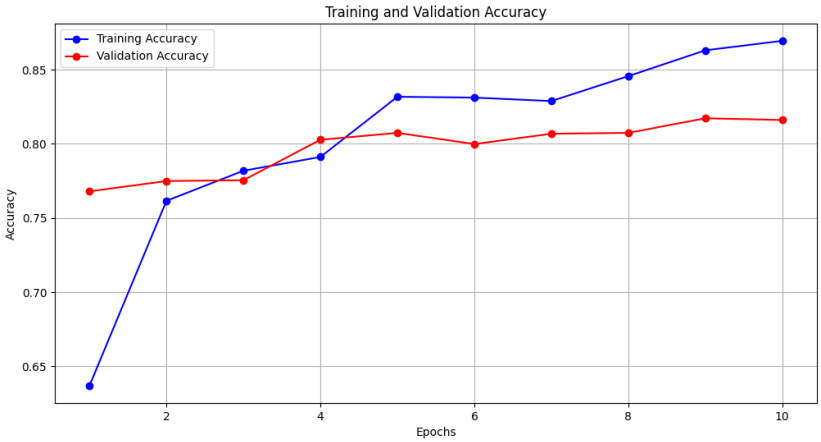

Figure 7. 2 Class classification graph.

The second execution features 4 classes. The classes selected were Left-hand, Right-hand, both feet and Both hands action movement. The input data size is [9836, 64, 821] and the distribution of the classes is shown in **Table 11**.

Class	Size
LeftHandAction	2471
RightHandAction	2456
BothHandsAction	2469
BothFeetAction	2440

Table 11. Data distribution on 4 Classes.

As per **Table 8**, The model seems to work relatively well on 4 classes, achieving an accuracy higher than the random baseline of 25%.

For the 4 classes, the confusion matrix on **Table 12** shows that the model predicts Left-Hand and Right-Hand movements more accurately than Both Feet or Both Hands.

		Predicted Classes			
	Classes	**LH**	**RH**	**BF**	**BH**
Actual Classes	**LH**	**0.64**	0.08	0.11	0.17
	RH	0.12	**0.65**	0.12	0.12
	BF	0.14	0.13	**0.57**	0.16
	BH	0.16	0.20	0.17	**0.47**

Table 12. The 4 Class Confusion Matrix of the baseline model. LH stands for Left hand, RH stands for right hand, BF stands for Both feet and BH stands for Both hands.

Figure 8. 4 Class classification graph.

For the third execution features 8 classes. The classes selected were all the action and imagery movements. the data size is [19674, 64, 821] and the distribution of the classes is shown in **Table 13**.

Label	Size
LeftHandAction	2471
RightHandAction	2456
BothHandsAction	2469
BothFeetAction	2440
LeftHandImagination	2480
RightHandImagination	2438
BothHandsImagination	2465
BothFeetImagination	2455

Table 13. Data distribution on 8 Classes.

As per **Table 8**, the model seems to struggle with 8 classes, but still achieving results higher than the random baseline of 12.5%

Table 14 Showcases that the model struggles in half of the classes but performs better in some others.

		Predicted Classes							
	Classes	**LHa**	**LHi**	**BHa**	**BHi**	**RHa**	**RHi**	**BFa**	**BFi**
Actual Classes	**LHa**	**0.41**	0.25	0.07	0.07	0.06	0.04	0.06	0.03
	LHi	0.31	**0.39**	0.03	0.08	0.04	0.03	0.15	0.05
	BHa	0.12	0.09	**0.25**	0.21	0.12	0.07	0.75	0.05
	BHi	0.08	0.11	0.15	**0.29**	0.09	0.11	0.10	0.07
	RHa	0.04	0.04	0.06	0.07	**0.40**	0.30	0.07	0.03
	RHi	0.05	0.05	0.04	0.08	0.26	**0.42**	0.05	0.06
	BFa	0.09	0.05	0.08	0.06	0.08	0.06	**0.37**	0.21
	BFi	0.06	0.08	0.06	0.12	0.06	0.11	0.26	**0.26**

Table 14. The 8 Class Confusion Matrix of the baseline model. LH stands for Left hand, RH stands for right hand, BF stands for Both feet and BH stands for Both hands. The a stands for action and the i stands for imagery.

The classification graph shows that after the 9th epoch the model slowly starts overfitting.

Figure 9. 8 Class classification graph.

3.2. Experiment Results

3.2.1. Baseline Model compared to the Comparative Models

The execution for the comparison of the 3 models, is done on the same lightly preprocessed data. The input data size is the same as the baseline model, shown in **Table 8**.

The classes selected were Left-hand and Right-hand action movement. The results of the executions are shown on **Table 15** and **Table 16**.

	Baseline 2D CNN		LSTM		1D CNN	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Fit	0.8617	0.3105	0.7231	0.5642	0.6388	0.6386
Evaluation	0.8024	0.4382	0.5131	0.6958	0.6388	0.6081
Prediction	**0.8005**	**0.4461**	0.5321	0.6932	0.6727	0.6065

Table 15. Prediction results on all the models used. The 2D CNN solution outperforms the shallow LSTM and the 1D CNN.

Model	**Accuracy**	**Precision**		**Recall**		**F-Score**		**Training Time**
		Left	Right	Left	Right	Left	Right	
Baseline	80%	81%	80%	79%	82%	80%	81%	37s
LSTM	53%	53%	54%	62%	45%	57%	49%	70s
1D CNN	67%	69%	63%	55%	76%	61%	68%	20s

Table 16. The precision, recall and F-score of the models.

The 2D CNN outperforms the other two models, with its higher accuracy and lower loss values. The LSTM Model underperforms severely on the data. In **Table 17**, the differences in training can be seen, as the two other models are classifying the two classes properly.

		Baseline Model		LSTM		1D CNN	
	Classes	**LH**	**RH**	**LH**	**RH**	**LH**	**RH**
Actual Classes	LH	**0.79**	0.21	**0.62**	0.38	**0.55**	0.45
	RH	0.18	**0.82**	0.55	**0.45**	0.24	**0.76**

Predicted Classes

Table 17. The confusion matrixes of the models.

The LSTM model quickly overfits and does not perform above the random baseline of 50% as per **Figure 10**.

Figure 10. LSTM classification graph.

The 1D CNN model trains well on the data, but 10 epochs are not enough for the model to train completely on the data.

Figure 11. 1D CNN classification graph.

3.2.2. Baseline Model on the Data Experiments

For the execution of the baseline model with different techniques, the data is of variable length.

	Baseline Data		**17 Channels**		**ICA**		**17 Ch + ICA**	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Fit	0.8617	0.3105	0.8595	0.3277	0.8639	0.3107	0.8527	0.3341
Evaluation	0.8024	0.4382	0.7932	0.4595	0.8050	0.4259	0.7835	0.4953
Prediction	**0.8005**	**0.4461**	0.7775	0.4727	**0.8059**	**0.4368**	0.7707	0.5080

Table 18. Prediction results on all the different data techniques. The data with ICA artifact removal and the data with no techniques applied to it bring similar results.

Table 18 depicts that the data with ICA and the data with no methods applied to it, bring similar results.

Model	Accuracy	Precision		Recall		F-Score		Training Time
		Left	Right	Left	Right	Left	Right	
Baseline	80%	81%	80%	79%	82%	80%	81%	37s
17 Channels	77%	76%	79%	81%	75%	78%	77%	23s
ICA	80%	82%	79%	79%	82%	80%	81%	51s
17 Ch + ICA	77%	82%	74%	70%	84%	75%	79%	23s

Table 19. The precision, recall and F-score of the different techniques.

Table 19 in combination with **Table 20**, depicts that the data with ICA is much more consistent, because the data with no processing is more biased towards left hand movements.

		Baseline Data		17 Channels		ICA		17 Ch + ICA	
	Classes	**LH**	**RH**	**LH**	**RH**	**LH**	**RH**	**LH**	**RH**
Actual Classes	LH	**0.75**	0.25	**0.81**	0.19	**0.79**	0.21	**0.70**	0.30
	RH	0.15	**0.85**	0.25	**0.75**	0.18	**0.82**	0.16	**0.84**
		Predicted Classes							

Table 20. The confusion matrix of the different methods.

Figure 12. 2 Class 17 Channels classification graph.

The model trained on 17 channels seem to overfit early, and the final prediction percentage is lower as per **Table 19**. **Figure 12** shows that the model starts to struggle after the 5th epoch. A shallower model may provide better results.

Figure 13. 2 Class ICA classification graph.

The results on **Table 19** and **Table 20** showcase that using ICA does give some improvements in the categorization of the classes. A deeper Neural Network or changes to the parameters of the model's layers may provide even better results in the future.

Figure 14. 2 Class 17 Channels ICA classification graph.

The final accuracy with the 2 techniques is lower than the implementation without them. Just as the implementation with only the selected channels, it may require a shallower Neural Network, as the developed model overfits quickly.

4. Discussion

This project attempted to provide alternative and efficient model for classifying EEG data, by treating the small segments of MM/MI signals as images using 3D arrays. The CNN model developed using 2D Convolutional Layers, captures spatial dependencies from all the channels which helps with classification. The data contains only the motor or imagery movements without the sections where the participants rest, which were cut according to the annotations provided within PhysioNet's dataset. The model developed in combination with limited preprocessing, is an efficient approach, as the accuracy is comparable to similar studies handling EEG cross-subject classification.

The model was tested as a baseline against other simple models with similar structure as the baseline using the same preprocessed data, specifically an LSTM and a 1D CNN. The results provide evidence that using a 2D CNN with this setup can produce better results.

Experiments were also done to see how the model performs when using different preprocessing techniques on the original data. Unlike using less channels, **Table 20** shows that using ICA improves the results by being more consistent, because the data with no processing is more biased towards left hand movements.

One of the limitations is that this model was only tested on the PhysioNet dataset. Further research is needed to test the model on other datasets, as they may have recorded the data using different techniques.

More research could be done by introducing different models or customizing the experimented ones to further support the preprocessed data. Further research can also be done on the techniques involving less channels, by adjusting and improving the Model to the new requirements.

In summary, the 2D Convolutional Neural Network that was developed, combined with slightly processed segmented data, provided a quick and efficient classifier for EEG motor-based data, with results comparable to other studies.

5. List of Acronyms

EEG	Electroencephalogram
BCI	Brain-Computer Interface
MI	Motor Imagery
EEGMMIDB	Electroencephalogram Motor Movement/Imagery Dataset Bank
EDF	European Data Format
MNE	Magnetoencephalography
CNN	Convolutional Neural Network
ICA	Independent Component Analysis
ELU	Exponential Linear Unit
LSTM	Long Short-Term Memory

6. References

[1]. Teplan M. Fundamentals of EEG measurement. Measurement science review. 2002 Jan;2(2):1-1.

[2]. McFarland DJ, Wolpaw JR. Brain-computer interface operation of robotic and prosthetic devices. Computer. 2008 Oct 10;41(10):52-6.

[3]. Moyes C, Jiang M. Recording a User's Brain Waves Using EEG. 2012. Available from: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/cwm55/cwm55_mj294/index.html .

[4]. Schalk G, McFarland DJ, Hinterberger T, Birbaumer N, Wolpaw JR. BCI2000: a general-purpose brain-computer interface (BCI) system. IEEE Transactions on biomedical engineering. 2004 May 24;51(6):1034-43.

[5]. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. circulation. 2000 Jun 13;101(23):e215-20.

[6]. Bidgoly AJ, Bidgoly HJ, Arezoumand Z. A survey on methods and challenges in EEG based authentication. Computers & Security. 2020 Jun 1;93:101788.

[7]. Altaheri H, Muhammad G, Alsulaiman M, Amin SU, Altuwaijri GA, Abdul W, Bencherif MA, Faisal M. Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review. Neural Computing and Applications. 2023 Jul;35(20):14681-722.

[8]. Dose H, Møller JS, Iversen HK, Puthusserypady S. An end-to-end deep learning approach to MI-EEG signal classification for BCIs. Expert Systems with Applications. 2018 Dec 30;114:532-42.

[9]. Wang P, Jiang A, Liu X, Shang J, Zhang L. LSTM-based EEG classification in motor imagery tasks. IEEE transactions on neural systems and rehabilitation engineering. 2018 Oct 18;26(11):2086-95.

[10]. Wu Y, Yang F, Liu Y, Zha X, Yuan S. A comparison of 1-D and 2-D deep convolutional neural networks in ECG classification. arXiv preprint arXiv:1810.07088. 2018 Oct 16.

[11]. Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L, Hämäläinen M. MEG and EEG data analysis with MNE-Python. Frontiers in Neuroinformatics. 2013 Dec 26;7:267.

[12]. PhysioNet. 64 channel sharbrough. Available from: https://physionet.org/content/eegmmidb/1.0.0/64_channel_sharbrough.pdf

[13]. Kingma DP. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.

[14]. Alotaiby T, El-Samie FE, Alshebeili SA, Ahmad I. A review of channel selection algorithms for EEG signal processing. EURASIP Journal on Advances in Signal Processing. 2015 Dec;2015:1-21.

[15]. PhysioNet. 64 channel sharbrough. Available from: https://physionet.org/content/eegmmidb/1.0.0/64_channel_sharbrough.pdf [adapted]

[16]. Jung TP, Humphries C, Lee TW, Makeig S, McKeown M, Iragui V, Sejnowski TJ. Extended ICA removes artifacts from electroencephalographic recordings. Advances in neural information processing systems. 1997;10.