THESIS

CLASSIFICATION OF P300 FROM NON-INVASIVE EEG SIGNAL USING CONVOLUTIONAL NEURAL NETWORK

Submitted by

Nazia Farhat

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2022

ABSTRACT

CLASSIFICATION OF P300 FROM NON-INVASIVE EEG SIGNAL USING CONVOLUTIONAL NEURAL NETWORK

Brain-Computer Interface system is a communication tool for the patients of neuromuscular diseases. The efficiency of such a system largely depends on the accurate and reliable detection of the brain signal employed in its operation. P300 Speller, a well-known BCI system, which helps the user select the desired alphabet in the communication process uses an Electroencephalography signal called P300 brain wave. The spatiotemporal nature and the low Signal-to-noise ratio along with the high dimensionality of P300 signal imposes difficulties in its accurate recognition. Moreover, its inter- and intra-subject variability necessitates case-specific experimental setup requiring considerable amount of time and resources before the system's deployment for use. In this thesis Convolutional Neural Network is applied to detect the P300 signal and observe the distinguishing features of P300 and non-P300 signals extracted by the neural network. Three different shapes of the filters, namely 1-D CNN, 2-D CNN, and 3-D CNN are examined separately to evaluate their detection ability of the target signals. Virtual channels created with three different weighting techniques are explored in 3-D CNN analysis. Both within-subject and cross-subject examinations are performed. Single trial accuracy with CNN implementation. Higher single trial accuracy is observed for all the subjects with CNN implementation compared to that achieved with Stepwise Linear Discriminant Analysis. Up to approximately 80% within-subject accuracy and 64% cross-subject accuracy are recorded in this research. 1-D CNN outperforms all the other models in terms of classification accuracy.

ACKNOWLEDGEMENTS

I am grateful to my advisor, Chuck Anderson, for his teaching, insights, ideas, and guidance in the progress and completion of this thesis as well as for his mentoring and constant support throughout the whole of my graduate studies. I would like to thank Theresa M. Vaughan for providing us with the data used in this analysis and for her valuable insights and advice relating to the practicality of this research. I would like to express my gratitude to my committee members for their time, suggestions, and support in this work. I would like to thank the professors and the Graduate Teaching Assistants for continuously helping me learn and build new skills. I am thankful to Colorado State University for giving me the opportunity to study Computer Science. Finally, I am indebted to my family, friends, and peers, without whose support it would not have been possible to complete this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Brain-Computer Interface (BCI) is a commonly used technology in neuroprosthetic applications to restore the damaged control functionalities by creating a direct communication pathway between brain and an external device [1, 2]. The communication is established by the analysis and the translation of the brain signals into commands required for the operation and control of the device. This computer-based system reads brain activity in the form of electric potential by either placing electrodes on the scalp (non-invasive) or implanting them on or in the relevant parts of the cortex (invasive), extracts and interprets features using the specified algorithm or model, and converts these interpreted features into instructions to operate an external device that replaces the human functionalities required for carrying out a desired action. This system aids people, who are disabled by neuromuscular disorders and diseases, such as amyotrophic lateral sclerosis (ALS), multiple sclerosis, spinal cord injury, stroke, dementia *etc.*, to perform tasks that they lost the ability to do.

The journey of BCI started with the discovery of electroencephalogram (EEG) signals, which are fluctuations of electrical current corresponding to the variations in the brain activity, by Hans Berger [3]. Detected by tiny electrodes EEG measures the brain signals in small amplitudes of few tens of microvolts and is contaminated by various physiological noises such as eye movements and blinks. This signal structured as a time-series of electrical potential extracted from the channels or electrodes has high temporal and low spatial resolution, which requires digitization, amplification, and filtering according to the analysis of interest. EEG is a non-invasive measure of brain signal as the electrodes are placed on the scalp instead of being implanted in the motor cortex. This type of non-invasive EEG-based system eliminates the risks of electrode implantation surgery.

One of the frequently used EEG-based BCI systems for the purpose of communication is P300 Speller. This tool displays the 26 alphabets along with few other symbols in matrix format on a computer or communication device screen. The user focuses his/her attention on the desired alphabet or symbol while the rows and columns of characters or standard stimuli are randomly flashed. When the row or column containing the desired element is flashed, a certain type of Event Related Potential (ERP) is elicited and detected by the system to determine which character in the screen the user wishes to communicate. This ERP component is known as P300 brain wave. After approximately 300 ms following the flashing of the desired character or target stimulus this P300 wave occurs as a positive deflection or spike in the evoked ERP waveform. This P300 signal happens to take place in an "oddball" paradigm among the standard stimuli when one focuses on detecting a target during the decision-making process [4, 5].

Although P300 has been proven as a reliable signal for BCI systems, its recording, processing, and detection is not without challenges. Wave distortion and amplitude reduction due to the high-pass filters during the recording of the signal, occurrence of larger potential due to ocular artifacts, variation of amplitude and latency from trial to trial, generation of multiple overlapping electric fields due to possible different processes simultaneously taking place over the scalp are few of the technical challenges when differentiating a P300 signal [6]. Because of the spatiotemporal characteristics and low signal-to-noise ratio (SNR) of non-invasive EEG recording, preprocessing of the raw data such as filtering, artifacts removal, re-referencing, downsampling, digitization, averaging over multiple trials etc. are necessary depending on the analysis.

Before deploying a system like P300 Speller for use it is required to setup an experiment, where multiple recordings over a number of sessions are performed for the user to build the P300 recognition system for accurate detection of the target character. This calibration process is time consuming and slows down the deployment of the system. Moreover, the lack of a reliable algorithm for accurately recognizing a P300 signal in a single trial, which refers to the data extracted over the flash time of a single character, limits the practicality of its application. The time of the P300 wave's appearance varies not only from person to person, but depends on the mental state, fatigue, attention of a single person as well. This inter- and intra-subject variability of this ERP signal is another barrier to creating a generalized model for its accurate, fast, and reliable recognition.

Various signal processing techniques and machine learning algorithms have been explored for detecting P300 signal. Almost all of these methods required averaging over multiple trials to produce reliable results. However, the image processing machine learning algorithm - Convolutional Neural Network (CNN) - showed promising performance in recognizing P300 signal in single trial. In this work binary classification approach of supervised machine learning is taken by implementing Convolutional Neural Network for automatic feature extraction and detection of P300 and non-P300 signals in both within-subject and cross-subject experimental design. This analysis was done offline on 6 healthy subjects' data. Three different shapes of filters along with several channel arrangements are explored to find the best performing approach. With 1D filters approximately 73% of the within-subject and 64% of the cross-subject single trial ERP samples are correctly classified averaging over 6 subjects. For some subjects individually about 80% within-subject and 70% cross-subject single trial accuracy are achieved. This analysis reconfirms the potential of CNN algorithm in faster detection of P300 signal among other random non-P300 brain waves paving the way for a more reliable BCI system.

Chapter 2 reviews various implementations of traditional machine learning techniques and especially Convolutional Neural Network in the domain of BCI. In Chapter 3 the data preparation and the methods of the experiment are explained in detail. This chapter also gives a brief overview of the CNN algorithm and the classification approach employed in this analysis. Chapter 4 discusses the results of the experiments in terms of accuracy and model fitting and tries to interpret the distinguishing features of the P300 signal extracted by the neural network. Chapter 5 gives a summary of this thesis by comparing different methods and discusses the future work that may be possible by extending this analysis.

Chapter 2

Background

BCI was developed as a communication pathway for disabled/locked-in (ALS) patients [7]. These days it is also used in mental state monitoring, neural prosthesis control, sleep analysis, transmission of image to blind person, epileptic seizure detection, IoT services, entertainment etc. The practicality of a BCI system depends on the accurate detection of the response signal used for its operation. The signals that BCIs usually rely on are Event Related Potential (ERP), Steady State Visual Evoked Potential (SSVEP), Event Related Desynchronization (ERD), Motor imagery, and Slow Cortical Potential [8, 9]. The response of interest in this analysis is P300 wave, which is an Event Related Potential measured by means of EEG.

2.1 Traditional ML in BCI

Traditional machine learning methods have been applied in P300 classification problem. Support Vector Machine (SVM), Bayesian Linear Discriminant Analysis (BLDA), and Fisher Linear Discriminant Analysis (FLDA) have been explored to classify and recognize EEG and P300 signal. Principal Component Analysis (PCA) and Independent Component Analysis (ICA) have been used for removing artifacts and signal's dimensionality reduction [10–13].

Krusienski et al. applied Stepwise Linear Discriminant Analysis (SWLDA) on the same dataset as the one used in this thesis. Their analysis with SWLDA showed consistent performance in P300 recognition. A binary classifier for P300 speller was constructed using Stepwise Linear Discriminant Analysis (SWLDA) on 7 healthy subjects' data, with which classification accuracy of more than 90% with 5 or more trials was achieved for some subjects but the single trial accuracy remained less than 60%. Four different sets of electrodes were analyzed to see the effect of channel selection, where the largest set employed to build the model consisted of 19 electrodes. Three different decimation factors were examined for the purpose of dimensionality reduction. Maximum number of features for each combination of channel-set and decimation factor were evaluated to determine the number of trials required for good generalization of the model on the unseen data. Furthermore, ear referenced data was compared with the Common Average Reference of all channels. The performance of the model varied notably with the varying channel-sets. The set with 6 central and posterior channels yielded the best results with no significant performance improvement observed with the larger 19 channels set. All these analyses were done on offline data and confirmed with an online analysis with the objective of building a baseline P300 classifier [14].

However, few problems exist with these traditional ML approaches. They require various preprocessing of the raw recorded data such as noise cancellation by averaging over multiple trials and calibration with multiple trials recording before deploying the system for use, which results in longer training time and deployment time. The performance is not consistent across multiple subjects either. These are not shift-invariant algorithms, and therefore, may not perform as good on some subjects as on the others. Moreover, depending on the condition of the recorded data these methods may require enhancing of the P300 potentials before feeding the data into the model. Selection of the most relevant electrodes or extraction of relevant features based on domain knowledge may be necessary as well [15].

Artificial Neural Network based machine learning techniques were employed to get around these drawbacks. Deep Learning has been used in the detection of P300 signal and EEG classification. Various algorithms have been implemented to design the building blocks of the neural network. Fully Connected layers of neural network (FCNN), Deep Belief Network (DBN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN) have most commonly been explored. In addition, Generative Adversarial Network (GAN) has been used for data augmentation [16]. Although neural networks can automatically extract distinguishing features, the downside of neural networks is that they require a large number of fairly clean or less noisy samples for training the models and achieving better generalization performance on the unseen data.

2.2 CNN in BCI

Convolutional Neural Network has shown significant target recognition performance in image processing field. It has been used in various computer vision fields such as face recognition, classification, object detection and recognition, gesture or movement recognition etc. It has the characteristics of automatic feature extraction, which makes it possible to extract features unknown or invisible to us. Its shift-invariant nature has the potential to achieve comparable performance across multiple input sources or subjects. In recent times CNN has been used in various brain signal analyses such as P300 detection, Mental task recognition, Emotion recognition, Seizure detection etc.

Lawhern et al. designed an EEG-based CNN architecture named EEGNet that is capable of detecting P300 signal across four different BCI paradigms, namely, Visual-Evoked Potentials, Error Related Negativity responses, Movement Related Cortical Potentials (MRCP), and Sensory Motor Rhythms (SMR). Their model was evaluated for both within-subject and cross-subject classification. 1D convolution was performed on 2D input samples, where the temporal features were extracted first with kernel shape of (1, half the sampling rate) and then the spatial features were extracted using a Depthwise Convolution layer with kernel shape of (number of channels, 1) followed by again Depthwise Convolution and Pointwise Convolution [17].

Shan et al. implemented a 2D CNN for character classification. Their network consisted of 1 hidden layer with 10 or more units followed by ReLU. They had one dropout layer too depending on the dataset. Kernel size was (N/15, C); N is 240 (1000 ms x 240 Hz) timesteps and C is number of channels. Each character had 15 epochs. It seemed like single trial accuracy measurement; but the character accuracy was determined from multiple trials [15].

In the analysis of Carabez et al. 3D Convolution was employed to detect P300 signal in auditory brain-computer interfaces. The analysis was done on 9 healthy subjects' data, where the stimuli were presented following the oddball paradigm. The input data were represented as an array of three-dimensional single trial samples keeping the arrangement of the electrodes similar to that of the experimental setup. For each timepoint all the 64 channels were reshaped into a 10x11 matrix keeping the relative spatial position of the electrodes, where the empty spaces in the matrix were set to 0. These 2D representations of data for all the timepoints employed in the analysis were stacked along a third axis to create the 3D representation of a single trial sample. Single trial accuracy above 80% was achieved with this novel representation of input data [18].

Wei et al. used CNN3D in epilepsy seizure detection by analyzing EEG signal. The input was arranged in the manner where each channel's time series was transformed into a 2D image and compressed to 256x256. The channels were than stacked along the third axis according to the adjacency degree similar to how CNN3D was applied in gesture recognition problem [19].

Cho et al. applied 3D CNN for emotion recognition using EEG data. CNN with 3D filters were used to extract features by taking the spatial relationship of the electrodes on the scalp into account, since the international 10-20 system of positioning the electrodes on the scalp entails variable distance between the electrodes. The input was rearranged in the form of a 3D image with timesteps represented as a 2D image and electrodes stacked along the z-axis according to the adjacency degree. Another input arrangement converted the data matrix with original electrode position at each time point into 64x64 representation with RBF interpolator and stacked the sequential time points along the z-axis to create the 3D array representation of each sample. The other kept the original positioning of the electrode and filled the empty spaces with null values before feeding the samples into the neural network [20].

A 5-layer 1D CNN was applied by Lun et al. to the Motor Imagery (MI-EEG) classification task, where raw EEG signal was the input and the feature extraction was performed using separate 4 temporal and 1 spatial filters. Dimensionality reduction was done using 4 max-pooling layers and overfitting was reduced using dropout and batch normalization. A final fully-connected layer was used for classification purpose. Each input sample contained a single pair of symmetrical electrodes data over the motor cortex of a single subject and the 4 target classes were imagination of opening and closing of left fist, right fist, both fists, and both feet. The accuracy achieved with this approach was 97.28% and FC3-FC4 electrodes pair was found to reach the highest accuracy among the 9 pairs that were employed in the construction of the model [21].

Li et al. combined Principal Component Analysis (PCA) and CNN to detect P300 signal. They reduced the dimensionality of the input data using PCA and fed the transformed features into the CNN network for classification. Their proposed model extracted spatial features from all the channels of an input sample in the first convolution layer. Three same-size convolution layers, each containing different kernels of different size, were placed in parallel to extract the temporal features from the output of the previous spatial convolution layer. The extracted features from these parallel layers were concatenated and passed to a dropout layer to prevent overfitting. This parallel layer was followed by another such layer, a pooling layer, and the final fully-connected layer with a SoftMax layer. Over 95% accuracy, which was calculated as the percentage of correct character recognition for multiple trials or repeats, was achieved with this proposed algorithm. However, the single trial accuracy remained at and below 30% [22].

Shukla et al. proposed a weighted ensemble of convolutional neural network weighted for single trial detection of target appliance such as, light fan, television, bulb, mobile phone, door, and electric heater in a P300 Speller-based home environment system. With this approach they obtained an average accuracy of 90.55% in P300 detection and an average target appliance accuracy of 93.22% over 9 subjects. They preprocessed the EEG data by passing it through a Chebyshev filter and removing the artifacts using ICA–recursive least square method [23].

Almost all of these models consisted of complex and deep networks, which make them difficult for mobile implementation for BCI systems. Here in this thesis we explore a number of simple network architectures for detecting the response signals and make a comparison among their performance and discuss ways to improve the signal recognition ability of the models.

Chapter 3

Methods

Machine Learning algorithms showed promising performance in analyzing brain signals. Deep Neural Network and Convolutional Neural Network were applied to classify P300 signals recorded from several healthy subjects using P300 speller.

3.1 Data Collection

The dataset used in this analysis was shared by the National Center for Adaptive Neurotechnologies, which is supported by the National Institutes of Health under Grant P41 EB018783. The collection of the data was supported in part by the National Institutes of Health under Grant NICHD HD30146 and Grant NIBIB/NINDS EB00856 and in part by the James S. McDonnell Foundation [14].

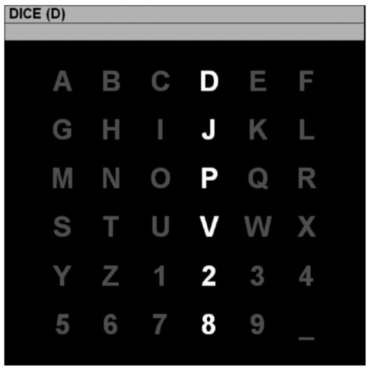


Figure 3.1: P300 Speller.

A P300 speller configured in a 6x6 matrix containing alphaneumaric characters was displayed to record the brain activity of 8 healthy subjects. The task was to focus on a certain character (target) and count the number of times the row or column containing that character was intensified (flashed) until a new one was specified. The data was collected from 64 channels at standard scalp positions. All of them were referenced to the right earlobe and grounded to the right mastoid. The data was bandpass filtered at 0.1-60 Hz, amplified, digitized, and stored. The sampling frequency for the digitization was 240 Hz.

3.2 Segmentation and Preprocessing

The data is collected over multiple sessions for each subject, where each session consists of 9 runs. Each run comprises the response for a series of characters called target stimuli. There are approximately 36 target stimuli appearing in each session, where the period stretching over each stimulus is called a Character Epoch. Each of these epochs consists of 15 cycles, each of which is called a Sequence. One such Sequence spans over the intensification of all the rows and columns in the matrix. Therefore, one target character is intensified twice per Sequence resulting in a total of 30 intensifications of the target-stimulus out of 180 intensifications in one Character Epoch. Each of these row or column intensifications spans over 100 ms followed by a 75 ms blank state. There is a 2.5 sec blank state period before starting each Character Epoch as well. Since the sampling rate is 240 Hz, the number of times the brain signal is sampled (responses) in one Run containing 3 target stimuli is approximated to be around 26,000.

When the intensified row/column contains the desired character, the stimulus type of the sampled data is 'target' and when the desired character is absent in the intensification, the stimulus type is 'non-target'. We segment the responses into P300 (target) and non-P300 (non-target) blocks based on the appearance of the row/column intensification and the type of stimulus (target or non-target). Each segment contains data of 500 ms time window starting from the appearance of the stimulus. A pre-stimulus mean of responses corresponding to 20 ms is subtracted from these sample segments.

With this approximation of the number of stimuli appearance and response segmentation procedure it is observed that some of the runs do not have the expected number of stimuli appearances in each Character Epoch. After sifting through all the subjects' data only the runs containing 30 target stimuli per Character Epoch are set apart. This reduces the data down to 6 subjects and that is finally used for the analysis.

The dataset is highly imbalanced with approximately 83% non-target segments in each run. In order to avoid classification bias towards the majority class of non-P300 signals in an imbalanced distribution, the dataset is balanced by randomly choosing equal number of non-target (non-P300 class) segments to match the number of target segments (P300 class) available in training, validation, and test set for each subject before employing them into the analysis. The training data is standardized over all the signal segments, channels, and time-instances using the sample mean and standard deviation. The P300 signal blocks are labeled as 1 and the non-P300 signal blocks are labeled as 0. 120 time-instances reflecting 500 ms after the appearance of the stimuli are taken to train the model, as larger time windows were causing the model to overfit in some cases.

3.3 Detection Algorithm

Neural network based supervised machine learning technique is applied to detect the response signals. Usually two types of classifications are analyzed in P300-based BCI problems. One of these approaches models the analysis as a binary classification problem, where one class is P300 response and the other class represents all the other random responses. The other approach attempts to recognize the characters displayed on the screen [8]. In this work we have taken the binary classification approach to distinguish P300 responses from the non-P300 responses.

Neural network based machine learning methods, known for automatic feature extraction algorithm, generally follow the feedforward network architecture. The input data are passed into the network, which learns to map each input sample to a fixed target output. If the target outputs are categorical, this is known as a classification model. If the classes of the trainable input samples are known and labeled, it is known as supervised machine learning. The hidden layers of the neural network contains weights or parameters that are optimized based on the error in the prediction found at each iteration during training of the model. The function used to compute and minimize the error or loss in the prediction is called objective function. The weights of the network are updated using backpropagation method. With this procedure the gradient of the objective function is computed and propagated backwards to calculate the gradients with respect to the weights of each of the hidden layers following the chain rule of derivatives. These gradients along with an optimal learning rate are then used to update the parameters following an optimizing method. Once the objective function is optimized the learning is complete and the model converges. Such a trained model is now capable of predicting a never-before-seen sample as its correct class. The drawback with this detection method is that it requires large number of fairly clean training samples. If the training dataset contains noises the model ends up making faulty prediction.

3.4 Fully-Connected Deep Neural Network

The first method deployed to classify the P300 and the non-P300 signals is a Fully-Connected Neural Network. This is a typical feedforward neural network characterised by hidden units employed for feature extraction, where the network parameters are updated following backpropagation procedure. The non-linear activation used in this architecture is hyperbolic tangent function. In this network structure each hidden unit in one hidden layer is connected to all the hidden units in the next hidden layer. The number of units in the output layer is equal to the number of classes available for prediction, which is two in our case of binary classification. The loss function and the weight optimizer are the same as the ones used in the Convolutional Neural Network methods and are described later in this chapter.

3.5 Convolutional Neural Network

Convolutional Neural Network (CNN), widely known as ConvNet or CNN, is a deep learning algorithm that has its root in the neocognitron [24] and explores the spatial relationship present in the data and performs automatic feature extraction. It is usually applied to analyze images in the computer vision field and is capable of tackling the problem of huge computation and memory cost in the processing of large input data or images. This shift-invariant algorithm has been successfully implemented in image and video analysis, classification and recognition, object detection, natural language processing, time-series analysis, and brain-computer interfaces.

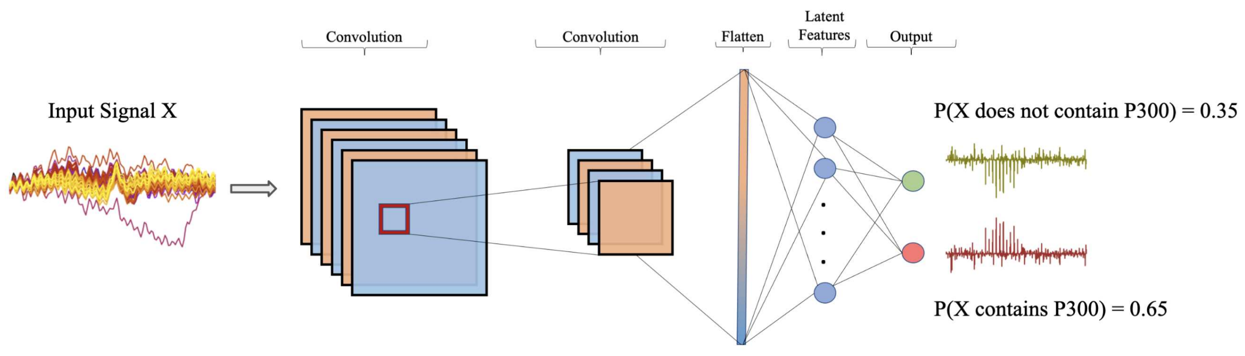


Figure 3.2: Convolutional Neural Network.

Convolutional Neural Network processes data that come in the form of multiple arrays or a matrix of rows and columns. The main ideas in processing such type of data are local connections, shared weights, and using hidden layer. Depending on the type of problem another processing called pooling is done as well. Figure 3.2 shows the architecture of a typical ConvNet. The first two layers are convolutional layers, each of which is generally followed by a non-linear activation layer. Each unit in the convolutional layer is a set of weights called filters or kernels. These filters slide over the feature maps in the previous layer and output a local weighted sum calculated from those previous features' local patches. This weighted sum is then passed through the non-linear activation function. Each unit or kernel in the convolution layer represents a distinct feature of the input data. The type of data that are usually processed using ConvNet, such as an image, has two important characteristics - the local group of values in the data has a correlation that forms distinctive feature and these distinctive features are invariant to location or shift-invariant in nature. The architecture of convolutional neural network works well for this type of data. Staying true to the nature of neural networks ConvNet reduces the dimensionality in addition to overcoming small shift and distortion of the latent features of the input data. The outputs from the activation layer following the last convolutional layer are then flattened and passed through the output layer of the network for the prediction based on the probability of the classes available for the data [25].

Depending on the type and shape of the input data the kernels can be 1, 2, or 3-dimensional. In this analysis all three types of kernels and the required shapes or arrangement of the input data are explored to determine the model and setup optimal for detecting P300 and non-P300 signals.

3.5.1 CNN1D

A convolutional neural network is referred to as 1D CNN, when its filters or kernels are one dimensional. 1D CNN is usually used in time-series data analysis such as, sensor, audio, or text data. In this convolution the kernels slide along one dimension known as the width of the data, where the shape of the input and the output of the convolution layer is 2D representing the features at each timestep.

The 1-dimensional kernels extract features from the input sample by convolving over a single dimension only. Here the kernels convolve individually over each of the channels and generate the feature maps by matrix multiplication of the extracted features by the respective kernels. That means with 1-D convolution only the temporal relationship of the EEG signal is explored for finding any distinguishing characteristics to classify the P300 and the non-P300 signals. The channel or spatial relationship is not explored using convolution filters. The electrodes are considered as the input channels of the network and used for dimensionality reduction by the neural network. Since the features are extracted by convolving the filters over the timesteps in each electrode's time-window only, the arrangement of the channels is not of any significance in this case. In this report this approach is referred to as CNN1D. With CNN1D model only the temporal features are extracted by the kernels and the channels information is incorporated into those features in reduced dimension.

3.5.2 CNN2D

Convolutional neural network is mostly known for its two dimensional filters or kernels and is referred to as 2D CNN. This type of network is used in computer vision problems for image analysis. The kernels slide along two dimensions, usually referred to as the height and the width of the image. The shape of the input and the output of the convolution layer is 3D with two of the dimensions being the height and the width and the other being the number of features at each image or data element. For example, they refer to the number of channels in an RGB image.

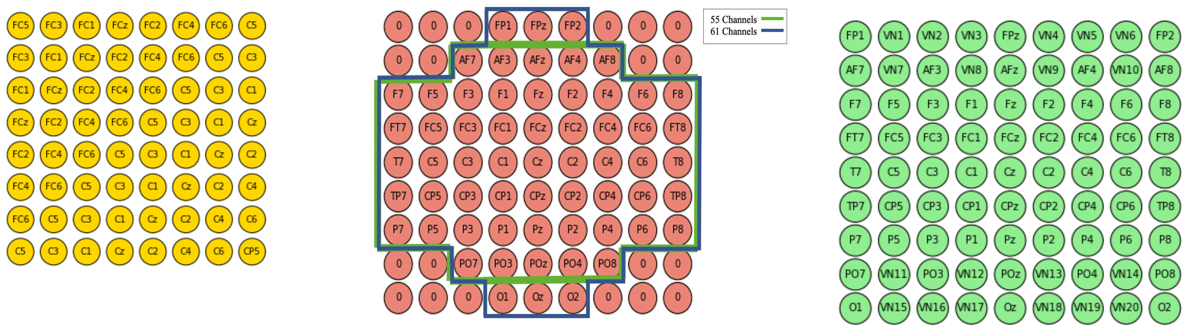
In the case of 2-dimensional kernels the feature extraction is performed along two dimensions of the input sample. The two dimensions that the kernels convolve over in the implementation of this approach in this analysis consist of the channels and the time-window in the input sample. The filters slide simultaneously along the electric potential of groups of electrodes at groups of consecutive timesteps. Since the shape of the filters is 2-D and one of the dimensions consists of the timesteps, the channels are stacked in a single dimension only. The exact spatial position of the channels on the scalp is not maintained. The channels are laid out in the ascending order of the electrode pin number in the input sample matrix regardless of their original position. In this arrangement the spatial dependencies of the neighboring channels cannot be captured perfectly as the electrodes' pin number in ascending order does not place all the channels according to their original position on the scalp. Therefore, this setup is expected to produce random results with little ability to recognize the P300 signal. This approach is referred to as CNN2D in this thesis.

3.5.3 CNN3D

Convolutional neural network with three dimensional filters or kernels are known as 3D CNN. 3D CNN has been used in medical image analysis, such as MRI or CT-scan, and in video prediction. The kernels slide along three dimensions including a depth axis along with the height and the width axes. The shape of the input and the output of the convolution layer is 4D with three of the dimensions being the height, the width, and the depth and the other being the number of features at each image element or timestep.

With 3-dimensional kernels the neural network can do convolution along a third dimension, usually referred to as the volume-convolution. In this setup there exists the scope of arranging the data channels in our analysis in their original scalp position and leverage the spatial relationship of the data in the recognition of P300 and non-P300 signals. Each of the input samples is shaped into a 3-D matrix, where the channels are arranged along the first two dimensions and the timesteps are placed along the third dimension to imitate the original setup of the electrodes and maintain the spatiotemporal characteristics of the EEG signals. The 3-D filters convolve over channels in two-dimensions and over time window in the third dimension to look for both spatial and temporal features of the data. Implementation of this approach is referred to as CNN3D in this report. The input sample matrix is designed in five different ways in the CNN3D implementation in this analysis to observe the effect of the relative positioning of the electrodes on the scalp.

The first design examined is 'Channels in Ascending Order' or CNN3D-asc. In this design all the 64 channels are arranged according to the ascending order of their pin numbers and shaped into an 8x8 matrix. The relative position of the electrodes in this setup can be visualized in Figure 3.3(a). This setup did not maintain the original scalp position of the channels and therefore, is not expected to show optimal performance.



(a) Channels in ascending order of pins.

(b) Channels in original position with empty virtual channels.

(c) Channels in original position with weighted virtual channels.

Figure 3.3: Channels arrangement for CNN3D networks. (a) All channels in ascending order. (b) Channels in original position with empty virtual nodes. (c) Channels in original position with weighted virtual nodes.

Since the actual relative position or proximity of the electrodes is not preserved with the CNN3D-asc. setup, the concept of virtual nodes are applied in this analysis. In order to feed the input data into the neural network according to the channels' scalp position it is required to organize them into a matrix of shape 10x11, if all the channels along the X and Y-axis are to be included. This ends up creating many empty spaces in the input matrix required to be padded with either 0 or some interpolated or extrapolated values. Considering that the padded values might interfere with the network's prediction, it is decided to exclude some of the channels data from the input matrix and create some virtual nodes to fill up the empty spaces in the input matrix. The virtual channels are created in three different ways - empty or null virtual channels, equally-weighted virtual channels, and variably-weighted virtual channels.

Empty virtual nodes are created in two different ways. The first approach excludes channel T9, T10, Iz, FP1, FPz, FP2, O1, Oz, and O2. Rest of the 55 channels are included and the 8 empty spaces required to create the 7x9 input matrix are filled with 0 value. The outermost electrodes have been left out in this arrangement to reduce the number of zero-weighted virtual nodes in the input matrix. It has also been expected to reduce the effect of eye blinks by leaving the FP electrodes data out of the analysis. This first approach is referred to as CNN3D-null55 and CNN3D-empty55 interchangeably in this report. The second approach referred to as CNN3D-null61 or CNN3D-empty61 excludes channel T9, T10, and Iz from the input data. The 20 empty spaces created due to the organization of the remaining 61 channels into a 9x9 matrix are then filled with null values translating to 0 weight similar to CNN3D-null55 arrangement before feeding the input matrix into the network. Channel T9 and T10 are used for referencing the EEG signals and are hypothesized to contribute little in recognizing the P300 signal. Channel Iz around the inion region could possibly hold important information [26]; but it is left out to avoid zero padding as well as to examine if its absence causes any significant performance drop in P300 recognition. Figure 3.3(b) visualizes these two types of channel arrangements.

The next interpolation method examined for the generation of the virtual nodes in the vacant spaces in the two frontal and the two parietal rows to construct the input array required for the 3D convolution is Equally Weighted Virtual Channels or CNN3D-eq.wt., where the empty spaces are filled with values other than 0. These weighted virtual channels are generated by taking the average of two adjacent channels and weighting all the newly created nodes between two channels equally. The weighted nodes are created following Equation (3.1).

$$VN_{ij} = \frac{C_i W_i + C_j W_j}{n} \tag{3.1}$$

Here VN_{ij} is the interpolated virtual node between two adjacent channels i and j, W_i and W_j are weights used to calculate the virtual node, and n is the number of virtual nodes required to be interpolated between two adjacent nodes. When interpolating nodes for CNN3D-eq.wt. model, virtual nodes in the first and the last rows of the input matrix are calculated using value 0.5 for both W_i and W_j. For the second first and the second last rows' virtual nodes W_i and W_j are taken as 1.

The final interpolation method examined is Variably Weighted Virtual Channels, referred to as CNN3D-var.wt. in this report. With this approach the virtual nodes are not equally weighted. Rather the interpolated values are calculated based on the proximity of the virtual nodes from their adjacent channels. The ratios of the distances of the three virtual nodes between two adjacent nodes in the first and the last rows of the input matrix are assumed to be $\frac{1}{2}$, $\frac{1}{1}$ and $\frac{2}{1}$ from left to right. In equation (3.1) W_i and W_j are taken based on the value of these ratios, where the nearer channel has higher weight. For the second first and the second last rows' virtual nodes W_i and W_j are taken as 1 similar to CNN3D-eq.wt. arrangement. The node arrangement in CNN3D-eq.wt. and CNN3D-var.wt. is visualized in Figure 3.3(c).

3.6 Model Training, Validation, and Evaluation

All the models are trained, validated, and evaluated following the same procedure. Weight initialization method, activation function, regularization technique, loss function, and parameters or weights optimizer are kept the same across all the CNN methods when training the models. The validation and the evaluation process are discussed in this section as well.

3.6.1 Training

The weights and the biases of the neural network are generally initialized with random floating point numbers close to zero. There have been different heuristics employed and analyzed in initializing the parameters for performance improvement and faster convergence of the models based on the activation function or the number of inputs to the hidden units. This analysis uses PyTorch for modeling the neural networks and employs its default weight initialization technique. This initializes the parameters uniformly and scales them by the square root of the number of inputs to each unit in order to ensure similar output distribution of all the hidden units. The number of inputs can be found by multiplying the kernel size with the number of incoming channels/inputs.

Activation functions are applied on the weighted sum outputs of the hidden units for non-linear transformation of the neural network. Rectified Linear Unit or ReLU is used in this analysis. This is a piecewise linear function that outputs the positive values as they are and the negative values as zero.

Overfitting is a significant problem in machine learning models. When the model fits too closely to the training data, it ends up learning about the trivial details and the noises in the training set and fails to capture the actual pattern of the data and therefore, does not generalize well to the unseen data samples. There is a trade-off between the complexity of the model and its generalization accuracy on the validation or test data and this is achieved by a technique called regularization that simplifies the model by penalizing its weights. Various regularization techniques are employed while training the model to reduce the overfitting. In this analysis 'dropout' is used to regularize the models' weights. It reduces the complexity by randomly dropping a specified number of neurons from the network in each iteration of the training process. A dropout layer with a probability of 0.2 following each activation layer is employed in the network [27].

Since it is a classification task, the loss function used for calculating the error was Cross Entropy Loss Function utilizing the concept of negative log-likelihood.

$$\mathbf{L}(y,p) = -\sum_{i} y_i \log(p_i) \tag{3.2}$$

where $\mathbf{L}(y,p)$ is the loss or error in the prediction results during the training, y_i is the actual class label and p_i is the Softmax probability of the i^{th} class. When it is reduced to a binary classification problem, it takes the following form -

$$\mathbf{L}(y,p) = -y\log(p) - (1-y)\log(1-p) \tag{3.3}$$

Adam optimizer with a learning rate of 0.01 is used for updating the network parameters in each training iteration based on error minimization.

3.6.2 Validation and Evaluation

One of the methods to measure the performance of neural network and finetune its hyper-parameters is k-fold cross-validation. It is usually used in predictive analysis for selecting the optimal-performing model.

K-fold cross validation is a resampling technique that estimates the model's prediction ability on a set of unseen data called validation set. In this method the available sample set is randomly shuffled and split into approximately equal size of K groups or folds. (K-1) folds of the data are employed as training set and the remaining fold, which is not seen by the network during training, is used as the validation set. The model is trained on the training set and evaluated on the unseen validation set. This process is repeated k times, where each time a different fold is held out as the validation set and the other folds are employed in the model's training. The mean of all these evaluations is considered as the resulting performance of the model. The parameter k is chosen in such a way that the training and the validation sets are large enough to be statistically representative of the population.

In our analysis we have used 5x2-fold cross validation to finetune our networks and select the best-performing model [28]. Here the trainable sample set, which consists of the 90% of the available samples for each subject, is randomly shuffled and split into two folds. The model is trained on one fold and evaluated on the other fold. This process is repeated five times and the final result is measured as the mean of all these runs. The sample set is shuffled randomly each time.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalseNegative + FalsePositive} \times 100\% \tag{3.4}$$

The models are evaluated in terms of accuracy and model fitting using a test set, which contains data samples unseen by the model during the training and validation process. This test set consists of the 10% of the samples available for each subject. The percentage accuracy of detecting the

P300 and non-P300 signals are recorded for test set. The training and the validation accuracy results are recorded as well in order to observe the model fitting and its ability to generalize to the unseen data.

Chapter 4

Results and Discussion

The results of the experiments detailed in chapter 3 are presented and discussed in this chapter. The discussion begins by presenting the optimal hyperparameters of each type of neural network found in the analysis and elaborating the process that has been followed for their selection. These hyperparameters include the hidden units, kernel size, stride, learning rate, and number of epochs employed in the training. Next, the within-subject classification results are discussed by detailing the correct signal recognition ability and the generalization ability of the models. Models are compared to determine the best performing model found for this dataset. Next, the cross-subject classification results are presented. After that the network weights and the feature maps of the best performing model are examined to interpret what the neural network is learning and considering as the distinguishing features for classifying the P300 signal. Finally, a brief discussion is offered to shed light on how the number of steps in the time-window of the input samples employed in this anylysis has been determined. Although it has not been utilized in this work, it is attempted to find the size of the training dataset required for developing the best performing model.

4.1 Hyperparameters Selection

Hyperparameters are used to control the training process of machine learning models. Model hyperparameters, which are typically used for model selection, in our analysis are the hidden layers, hidden units, kernel size and stride. The algorithm hyperparameters, which are responsible for the optimal speed and convergence of the model, are the learning rate and the training epochs. All of these are determined by manual tuning while observing the models' performance, since they cannot be inferred during the training process. The selection of these hyperparameters is made based on the trade-off between the best accuracy on the validation set and the least overfitting with the training set. For example, Figure 4.1 shows that the network with 2 hidden layers with 1 and 2 units respectively in the CNN1D model is the optimal network structure for subject 5.

Similar performance is observed for all the other subjects with this network architecture. Although the structure with 1 hidden unit in a single hidden layer shows similar results, in some cases this has caused the model to underfit. Besides, in order to get a better insight into what features the model is learning as the distinguishing characteristics of the P300 signal the single-layer-single-unit architecture has been avoided, unless the model's performance is compromised.

Table 4.1: Hyperparameters of the optimized networks.

Hyperparameters	CNN1D	CNN2D	CNN3D
Within-subject Hidden Units	[1, 2]	[1]	[2, 2]
Cross-subject Hidden Units	[3, 2]	[2]	[5, 10]
Kernel size	[10, 10]	[5x5]	[5x5x5, 2x2x2]
Stride	[3, 1]	[3]	[3, 1]
Training Epochs	2000	5000	2000

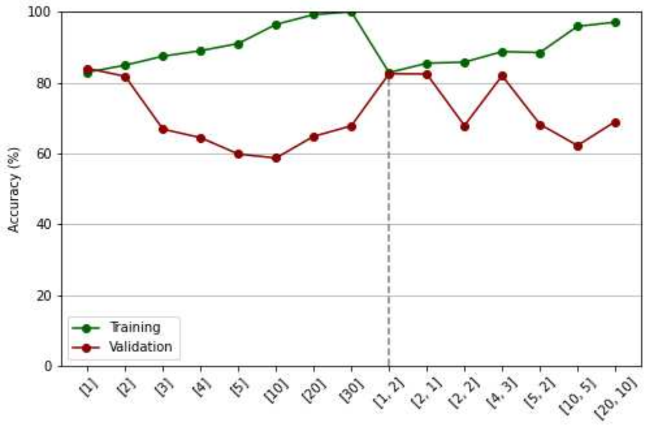


Figure 4.1: CNN1D model's performance over varying hidden units.

In similar way the kernel size and the stride for each model are determined. In case of CNN2D the simplest model with only 1 hidden unit in a single hidden layer is observed to be overfitting the training data. Deeper networks with more hidden layers and units continues to increase the overfitting, which is why, the simplest architecture is employed in the CNN2D analysis of the data. More training epochs are required for the models to converge as well as to reach the similar performance level as the other approaches in case of CNN2D. Different kernels other than square and cube shape have been examined for CNN2D and CNN3D models but those do not show any significant improvement in the model's performance. Table 4.1 records the optimal hyperparameters found in both the within-subject and the cross-subject analyses of signal classification, where the difference is observed only in the number of hidden layers and units of the neural network. A learning rate of 0.01 with Adam optimizer results in the fastest convergence of the models' parameters or weights. The hidden units' values are nonlinearized through ReLU (Rectified Linear Unit) activation function in conjunction with 20% dropout regularization.

4.2 Within-Subject Classification

4.2.1 Accuracy

The intrasubject or within-subject accuracy measures the percentage of the samples correctly classified by the trained model for a single subject. The model is trained, validated, and tested using the data collected from the same subject. All the models have been trained and tested on the 6 subjects separately. This metric evaluates the P300 and non-P300 signal recognition ability of the model accounting for the intrasubject variability that is induced by the data collection over multiple sessions and days.

Table 4.2: Test accuracy % of all the models.

Model	Sub_2	Sub_3	Sub_4	Sub_5	Sub_6	Sub_7	Mean
FC-DNN	62.88	79.07	78.7	80.46	63.96	61.39	71.08
CNN1D	62.97	80.93	80.56	78.15	71.8	68.89	73.88
CNN2D	57.84	80.09	75.12	75.46	64.5	59.54	68.76
CNN3D-asc.	61.53	79.17	80.32	76.76	68.83	66.57	72.2
CNN3D-null55	61.98	77.96	75.93	72.69	61.53	67.69	69.63
CNN3D-null61	58.74	80.74	77.66	77.59	65.77	67.41	71.32
CNN3D-eq.wt.	63.69	77.13	79.98	76.57	66.76	70.83	72.5
CNN3D-var.wt.	62.79	79.17	78.82	76.57	68.11	61.3	71.13

Table 4.2 contains each method's single trial accuracy of recognizing a P300 and non-P300 signal in the balanced Test set for each of the subjects individually. Fully-connected deep neural network, referred to as FCDNN, achieves high accuracy closer to 80% for subject 3, 4, and 5 and comparatively lower accuracy of approximately 62% for subject 2, 6, and 7. With CNN1D the performance of the models on subject 6 and 7 increases about 7% while maintaining similar results as that of FCDNN for the other subjects. This improvement in the classification performance increases the mean accuracy of CNN1D models. The performance of the neural networks goes down with the application of CNN2D models with a decrease of about 4% in the mean accuracy compared to the mean result of CNN1D models. CNN3D with electrodes arranged in the ascending order, expressed as CNN3D-asc. models, raises the accuracy to comparable level of CNN1D results but still shows lower performance for subject 7. With CNN3D-null55 and CNN3D-null61 the performance of the models on subject 4 and 5 decreases to the similar level as that of CNN2D, although incorporating 61 channels instead of only 55 does show an improvement in the mean accuracy. CNN3D-eq.wt. and CNN3D-var.wt. models show similar performance as that of CNN3D-asc. models. Therefore, it is observed that CNN1D models are able to achieve the highest mean accuracy, whereas CNN2D has the lowest recognition accuracy. All the models perform their best with subject 3 data and had the most difficulty with subject 2 data. CNN1D outperforms all the other models in terms of test accuracy.

4.2.2 Model Fitting

Model fitting is a measure of the generalization capability of a machine learning model. If the model generates higher classification accuracy on the training set than on the validation set or the unseen test set, it is deemed as an overfitting model. If the accuracy of the training set is less than that of the unseen test set, the model is called underfitting. Overfitting model learns more about the noises and outliers in the training set rather than the underlying distinguishing features of the target classes. A well fit model learns the features of the training set as well as shows good generalization on the unseen dataset.

It has been possible to achieve above-chance single-trial accuracy for all the subjects with Fully-connected deep neural network (FC-DNN), where the average test accuracy is about 71%. However, this model is highly overfitting with the training data as shown in Figure 4.2. This plot depicts the training and the validation accuracy with 100 units in a single hidden layer. This behavior persists with fewer hidden units but results in a reduced accuracy in the classification indicating the necessity of more exploration for better generalization of the unseen data and improvement in the recognition task. This behavior is characteristic of a classic FC-DNN network as it contains a large number of parameters, each of which is connected to all the others. For high dimensional data like EEG responses overfitting is not an unexpected phenomenon.

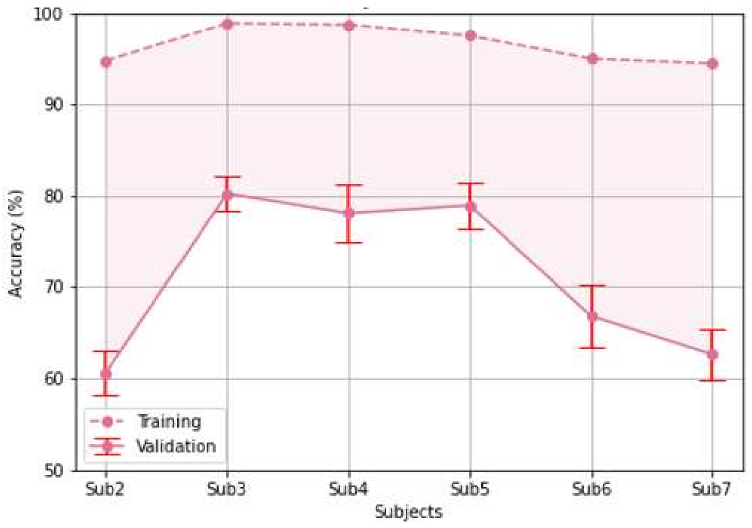
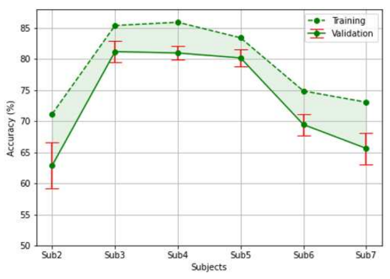
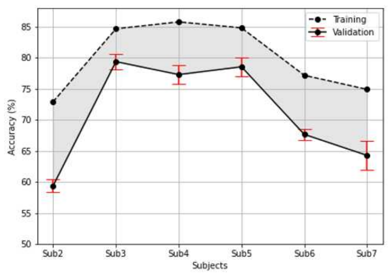


Figure 4.2: Training and Validation accuracy of Fully-Connected Deep Neural Network.

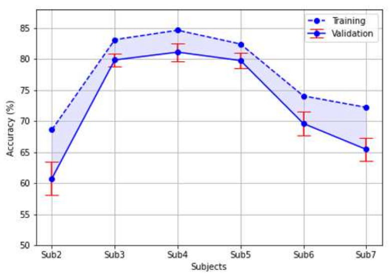
With the target of reducing the overfitting problem and in search of better generalization performance automatic feature extraction is examined by implementing convolutional neural network. The first approach with 1-dimensional filters, expressed as CNN1D in this analysis, not only reduces the overfitting, it also results in higher accuracy in the classification task. The overfitting is reduced from 25% to approximately 5% on average with CNN1D. With the second approach, CNN2D, containing 2-dimensional filters the models are observed to overfit more in comparison with CNN1D but the generalization is better than that of FC-DNN models. The overfitting



(a) CNN1D.



(b) CNN2D.



(c) CNN3D with ascending order of channels.

Figure 4.3: Training and validation accuracy with all channels. (a) CNN1D results. (b) CNN2D results. (c) CNN3D with ascending ordered channels results.

again is increased to approximately 8% with CNN2D. CNN3D, which is the third approach with 3-dimensional filters with the input data channels arranged in the ascending order, reduces the overfitting further while maintaining comparable performance in classification accuracy as the CNN1D models. It is to be noted that all 64 channels are fed into the network with all these three approaches. Figure 4.3 shows each of their performance with the training and the validation sets.

More examinations with different arrangements of the input data channels before feeding them into the neural network are performed with the objective of observing CNN3D models' potential for further improvement in the recognition ability of the target signal classes. To keep the original scalp position of the electrodes the concept of virtual nodes have been considered for the analysis. The prefrontal channels FP1, FPz, and FP2 and the occipital channels O1, O2, and Oz including the reference electrodes T9 and T10 and channel Iz are eliminated. The remaining 55 channels are fed into the network with 8 empty virtual nodes filling the empty spaces of the input matrix of shape 7x9. Figure 4.4 shows the results of this approach. It is to note that the classification accuracy drops about 2-5% depending on the subject both in the training and the validation sets with this selection and arrangement of input electrodes. The overfitting of this model remains at similar level as that of the CNN3D-asc. model.

Since the removal of the above mentioned 9 channels and including 8 empty or null values results in reduced classification accuracy of the models, in the next arrangement of the channels examined in this analysis 61 channels are fed into the network excluding only channel T9, T10, and

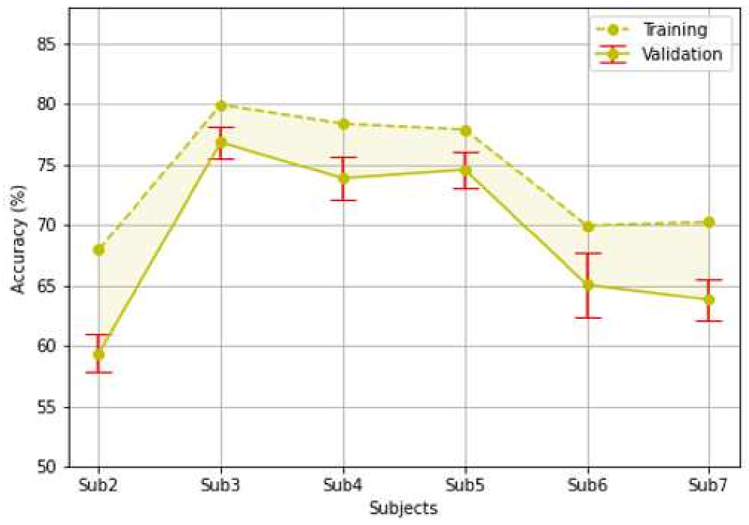


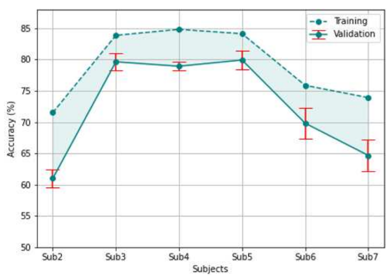
Figure 4.4: Training and validation accuracy of CNN3D with 55 channels and 8 empty virtual nodes.

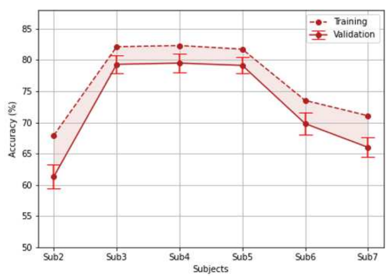
Iz. In this case the virtual nodes required to generate the symmetric input data structure, which is a square matrix of order 9, are created by interpolation using three different weighting techniques - Empty or null weights, Equal weights regardless of the proximity of adjacent nodes, Variable weights based on the relative proximity of the adjacent nodes.

Figure 4.5 presents the results of these arrangements. CNN3D-empty61 again raises the training accuracy to approximately similar level as that of CNN3D-asc. model; but the validation accuracy remains a little lower resulting in a slightly higher overfitting. With CNN3D-eq.wt. arrangement this drawback is overcome as the validation accuracy goes up and the training accuracy goes down. Almost identical results are achieved with CNN3D-var.wt. arrangement. The error bar for all the subjects in the equally-weighted virtual nodes arrangement shows a more stable behavior compared to the other approaches explored in this analysis. The overfitting reduces to approximately 3% on average with CNN3D-eq.wt. and CNN3D-var.wt., which is the lowest among all the models.

4.2.3 Model Comparison

The models are compared using the within-subject classification accuracy on the unseen test set. It is interesting to notice that all the CNN3D models perform in similar manner. Comparison


(a) CNN3D-empty61.


(b) CNN3D-eq.wt.

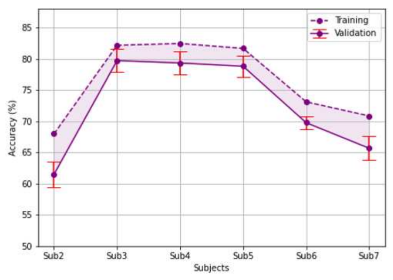

(c) CNN3D-var.wt.

Figure 4.5: Training and validation accuracy with 61 channels and virtual nodes. (a) CNN3D-empty virtual nodes. (b) CNN3D-equally weighted virtual nodes. (c) CNN3D-variable weighted virtual nodes.

of single trial test accuracy of CNN3D models is shown in Figure 4.6. The models with CNN3D-empty55 arrangement shows lower classification accuracy in comparison to the other CNN3D models. All the other models show similar performance. CNN3D-asc. model generates highest recognition accuracy for subject 4 and subject 6. But CNN3D-eq.wt. catches up with it and additionally performs better than any other approach for subject 2 and subject 7. CNN3D-empty61 produces better results for subject 3 and subject 5. Leaving out 9 channels from the input sample in CNN3D-empty55 model seems to have affected the recognition ability of 3D convolutional neural network for subject 4, 5, and 6.

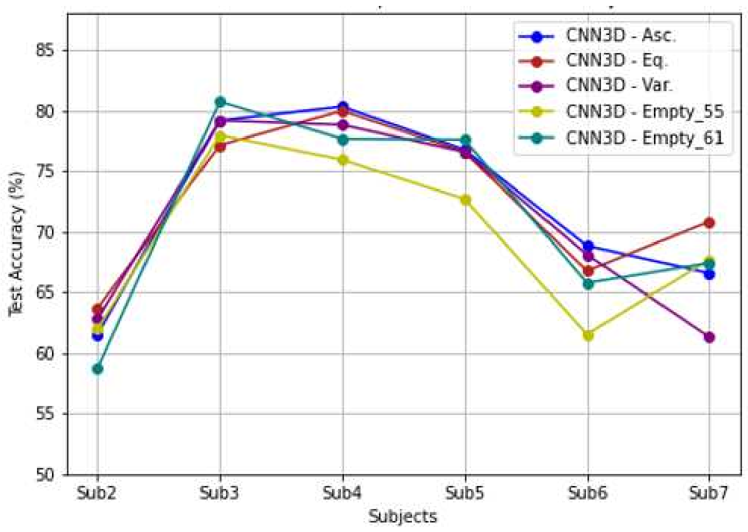


Figure 4.6: Accuracy comparison of CNN3D models.

The models trained by feeding all 64 channels' data into the neural network are CNN1D, CNN2D, and CNN3D-asc. CNN1D outperforms the other two, whereas CNN2D has the lowest accuracy on the test set. Models (with all channels) accuracy comparison is shown in Figure 4.7. The lowest signal recognition ability of CNN2D is understandable if the channel arrangement in the input sample is considered. Since the channels are stacked one after another and this arrangement does not preserve the actual placement of the electrodes on the scalp, the kernels do not have access to a group of electrodes at once that may be representative of a localized portion of the brain. Rather the features extracted with this model are compilations of data characteristics from both adjacent and nonadjacent electrodes, which mainly capture the temporal features along with the features of a group of channels appearing at the same timepoint simultaneously. That is why, it is difficult to interpret the extracted features of this model in terms of spatial relationship among the electrodes or localized pattern in the brain. The combination of adjacent and nonadjacent spatial features may be the reason behind lower classification ability and higher overfit as some of these features may be representative of some other random processes going on in the brain. Therefore, although this model may be more suitable for capturing more variability in the brain signal, it may not be ideal for distinguishing between a P300 and a non-P300 signal specifically as this model may be learning information not needed for this purpose. For example, FC5, FC3, FC1, FCz, FC2 electrodes are adjacent but C4, C6, CP5, CP3, CP1 electrodes are nonadjacent and do not represent any locality in the brain. Still most of the groups of electrodes are adjacent in this arrangement and that may be why the performance of CNN2D model is not bad. The few groups of nonadjacent electrodes may be causing the model to overfit with only a single hidden unit.

CNN1D kernels have extracted temporal features while reducing the spatial dimension. As 64 channels are reduced to only 2 dimensional or spatial representations, CNN1D model may be losing many important spatial information. The noises in the spatial dimension are also getting included in the features extracted by the network, which may be causing the model to overfit. This noise inclusion is reduced a little with original electrodes placement in CNN3D-eq.wt. and CNN3D-var.wt. models.

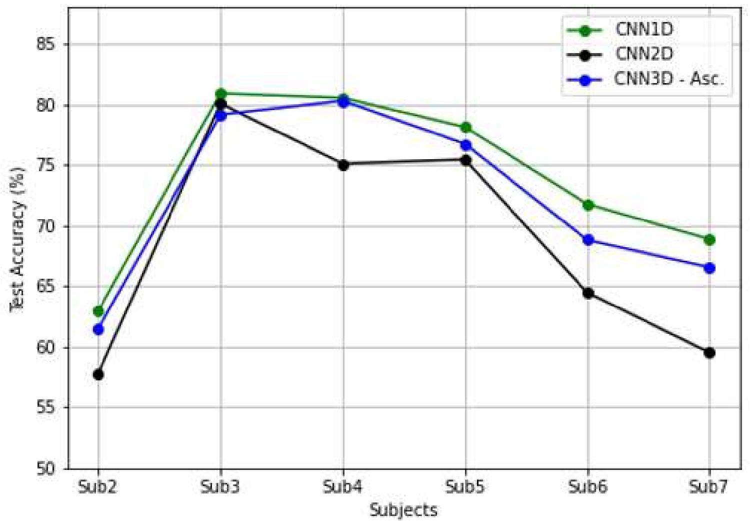


Figure 4.7: Accuracy comparison of CNN1D, CNN2D, CNN3D-asc. models with all channels fed into the network.

CNN3D-asc. model, where the input sample's channels are arranged in the ascending order, possesses characteristics from both the CNN1D and CNN2D model. This model extracts features along the temporal direction without merging it with the spatial dimension. This way the temporal feature extraction of CNN1D is maintained. It also convolvs over a group of adjacent and nonadjacent channels similar to CNN2D. Moreover, this model incorporates all the channels' data as is the case for both CNN1D and CNN2D. Due to maintaining the temporal features along a separate direction as well as incorporating the spatial information in the extracted features, this model reduces the overfit of the CNN2D model while producing classification results closer to that of CNN1D. The interesting point to note is that original electrode positioning in CNN3D-eq.wt. and CNN3D-var.wt. has not shown any significant performance improvement in recognizing the P300 signal. This may be due to performing an interpolation to generate the virtual nodes. This arrangement although maintains the relative spatial positioning of the electrodes, it does not take the actual distance between the electrodes into account. Therefore, an extrapolation while incorporating the between-channels distance information to generate the virtual nodes may show an improved performance in recognizing the target signals.

4.3 Cross-Subject Classification

Intersubject or cross-subject classification accuracy is measured by training a model on the data collected from any 5 subjects and testing on the unseen data of the 6th subject. For evaluating a model's generalization ability on each subject's data, the training is performed on all the other subjects keeping the test subject's data out of training dataset. This shows a model's ability to transfer the learning from other subjects' features to recognize the P300 wave of a new subject accounting for the intersubject variability.

Table 4.3: Cross-Subject Test accuracy % of all the models.

Model	Sub_2	Sub_3	Sub_4	Sub_5	Sub_6	Sub_7	Mean
FC-DNN	52.59	67.94	64.44	65.43	58.95	54.32	60.6
CNN1D	52.29	69.72	69.96	71.41	63.33	57.8	64.1
CNN2D	53.32	69.58	70.57	68.94	64.6	56	63.8
CNN3D-asc.	52.01	69.52	70.82	69.37	63.49	53.94	63.2
CNN3D-null55	50.43	63.36	64.63	63.1	56.36	52.88	58.46
CNN3D-null61	52.16	66.92	67.29	68.94	60.53	54.74	61.8
CNN3D-eq.wt.	52.31	67.61	69.62	67.16	63.22	55.16	62.5
CNN3D-var.wt.	52.34	68.21	68.06	67.81	64.32	55.82	62.8

From Table 4.3 it is observed that CNN1D again outperforms all the other models in the cross-subject P300 recognition task. Averaging over all 6 subjects the single trial test accuracy of CNN1D is 64.1%, which is the highest among all the models. The lowest performance of 58% is observed for CNN3D-null55 models. However, it is interesting to note here that CNN2D performs better than CNN3D models and produces classification accuracy closer to that of CNN1D. CNN3D-asc. model does slightly better than CNN3D-eq.wt. and CNN3D-var.wt. models. The reduced performance with CNN3D virtual nodes approach may be an indicator of the importance of channel Iz in detecting a P300 signal, which has been left out when building these models.

The hyperparameters of the cross-subject networks are shown in Table 4.1. It is to note here that the number of hidden units in the hidden layer are required to be increased in comparison with that of the within-subject networks to overcome the underfitting problem of the cross-subject models. Since the models are trained on 5 subjects' data, which have been randomized before being fed into the neural network, it is possible that the network needs more units to capture features of more variations due to the intersubject variabililty. Again for CNN2D model one additional unit is sufficient to fit the model. Only the hidden units are shown in the table as the kernel size, stride, and number of training epochs are the same for the cross-subjects models as that for the within-subject models.

4.4 Network Weights and Feature Maps

The interpretation of the features extracted by the neural networks is a difficult task, since they do not always give meaningful insights into the structure of the function approximated by the networks. The interpretability method is usually employed based on the problem at hand and the type of network being studied. In this work only the best performing model CNN1D's weights and feature maps are reported. Due to the large drop in the spatial dimensionality in the CNN1D models, the interpretation of their hidden layers' feature maps and weights is not very intuitive. As the feature maps generated by the hidden layers do not show a consistent pattern across all the samples and the subjects, only the output layer's weights and corresponding features are reported here.

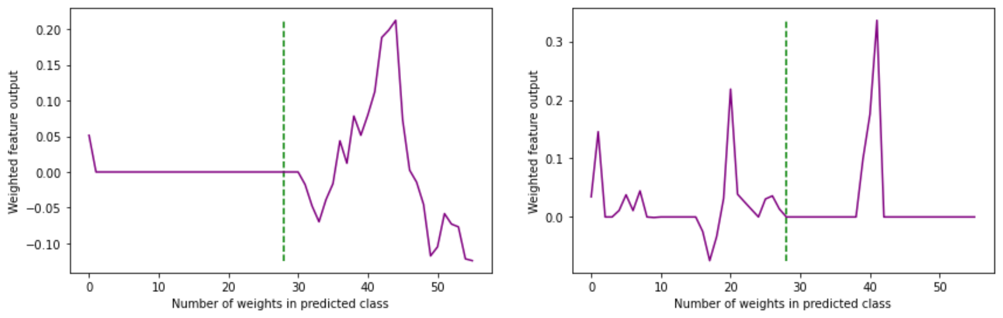


Figure 4.8: CNN1D Weighted Outputs of Within-Subject True Positive Samples.

In the final hidden convolutional layer there are 2 hidden units representing all the 64 input channels. Each unit generates the temporal features extracted from the 120 timesteps of the input sample by the 1-dimensional kernels of the network. This way the 120 timesteps are reduced to 28 points representing the 500 ms of the input time-window. The ReLU outputs of these 2 units are concatenated and multiplied by their respective weights in the predicted class unit in the output layer to generate the weighted outputs of the samples. These weighted outputs of a couple of true positive and true negative samples are shown in Figure 4.8 and Figure 4.9. These are the outputs of the unit in the output layer of the neural network corresponding to the class predicted by the model for the input sample. The green dashed line in the figure at point 28 in the X-axis (Number of weights in predicted class) separates the features generated by the 2 hidden units in the penultimate convolutional layer. Therefore, the 28 points in both the halves represent the 500 ms time-window. For the first True Positive sample of subject 5 (left) in Figure 4.8 there seems to be a peak at around 215 - 270 ms (40-45 step on X-axis) in the features of the second hidden unit. Another True Positive sample (right) of the same subject shows a peak at 320 - 357 ms (18-20 step in the X-axis in the same figure) in the first hidden unit's features as well. However, this pattern is not always seen in the other samples and subjects and the timing of the peak seems to vary depending on the sample and the subject.

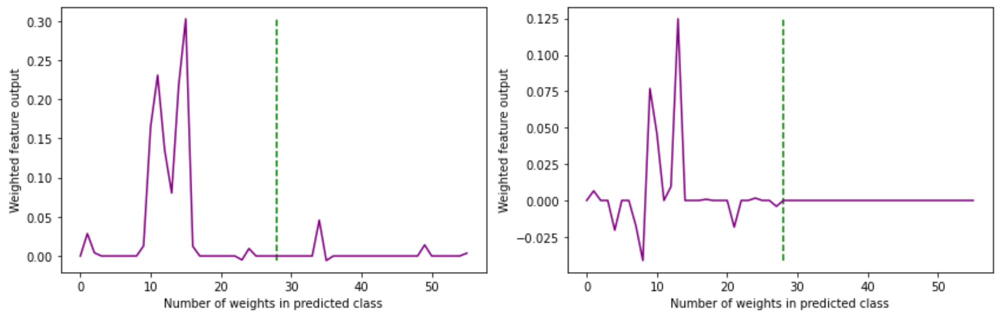


Figure 4.9: CNN1D Weighted Outputs of Within-Subject True Negative Samples.

Figure 4.9 shows the plots of 2 True Negative samples of the same subject 5. It is important to note here that the weighted output features in these samples show an opposing pattern from the true positive samples at each timestep. This opposing pattern is observed to be consistent in the extracted features by the CNN1D models irrespective of the subjects and the samples.

The weights of the output layer's units representing the P300 and the non-P300 classes also show an opposing pattern as shown in Figure 4.10. For P300 class unit the highest value weights are observed for the second hidden unit of the penultimate convolutional layer at steps 10-16 which correspond to 178-286 ms of the time-window. The weights corresponding to the same position in the non-P300 class unit shows the smallest value. The appearance of the P300 signal peak might be occurring at around this time for Subject 5. Due to the intrasubject variability and the variation in the latency of the P300 occurrence this pattern in the output layer's weights should vary depending on the subject and that is what we have observed in our analysis as well.

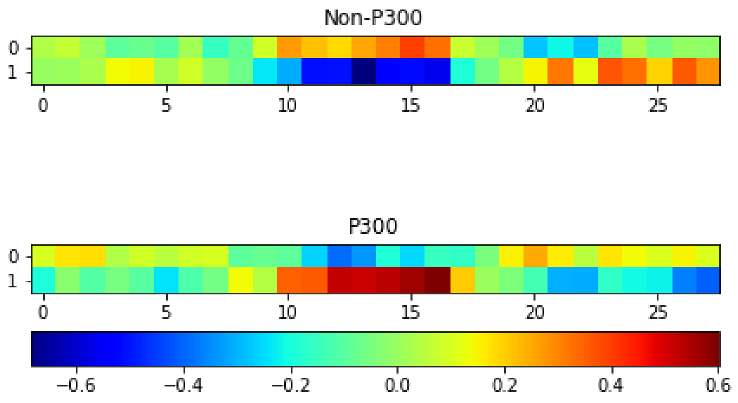


Figure 4.10: CNN1D Output Layer Weights of Within-Subject Model for Subject 5.

4.5 Size of Time-Window and Training Dataset

Based on how the experiment for the data collection is designed the shortest and the longest gap between two target stimuli in one intensification sequence could be 175 ms and 1925 ms respectively. As the P300 signal is assumed to peak a large positive potential at around 300 ms following the appearance of the target stimulus, there is a possibility of overlapping targets and therefore, the size of the time window representing a target and non-target signal plays a vital role in the classification task. Multiple window sizes are experimented starting from approximately

400 ms window in order to have a better understanding of the hidden features extracted by the network from the input signals. Figure 4.11 shows CNN1D model's performance over varying window sizes for subject 5 and 4. It is observed that with increasing window size until around 700 ms the model's performance either dwindles or remains similar. However, for some subjects larger time windows of around 800 ms causes the model to overfit and results in highly spread-out and uncertain prediction as shown for subject 5. For other subjects this results in a small improvement in the performance as shown for subject 4. Therefore, same time window seemed to behave differently depending on the subjects and the initial weights of the models. Since the larger window of 800 ms tends to produce less reliable results and smaller windows show comparable and sometimes better performance in terms of classification accuracy, generalization and reliability of the model, a window size of 500 ms corresponding to 120 data point or steps is used in the analysis considering the time gap of P300 observation as well as to keep the time window reasonably shorter in the input signal.

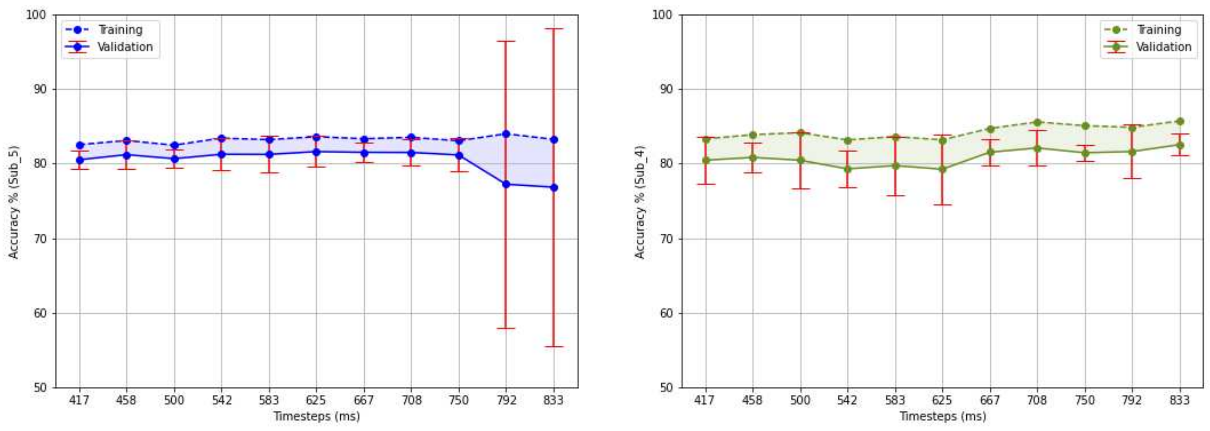


Figure 4.11: CNN1D model's performance over varying timesteps.

Although all the samples available for each subject except the test set have been employed in training the networks and validating them with 5x2-fold cross validation, it is examined to find whether the sample size required for training and generalizing the models capable of producing similar performance is less than what has been used in the analysis. It is interesting to observe that the sample size required to achieve comparable performance for subject 5 is about 40 - 50% of the data employed in the training and reaches the optimal performance at the 70% amounting to around 6,800 samples, as shown in Figure 4.12. The performance of the model drops at 80% and 100% and shows high error range indicating unstable behavior. This might be due to the confusing and noisy labels or samples in the training set. This tells that if the noisy samples can be got rid of, the models have the potential to produce better prediction of the target signals.

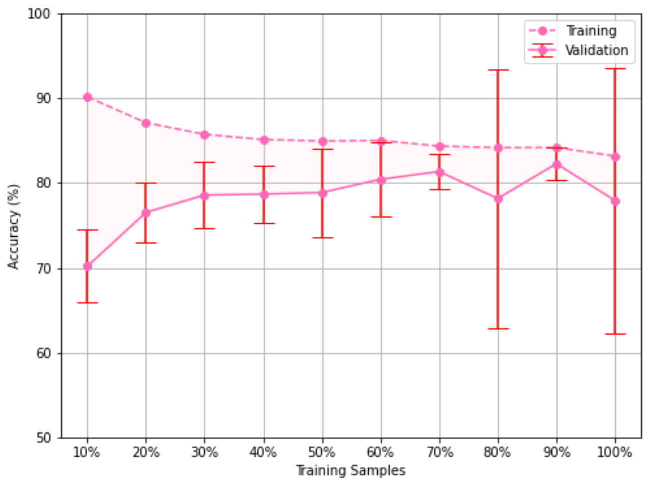


Figure 4.12: CNN1D model's performance over varying training sample size for subject 5.

Chapter 5

Conclusion

The results and findings of this thesis are summarized in this chapter. Some of the analyses that can be done in future based on the findings of this work are discussed as well.

5.1 Summary

In this analysis five different methods using convolutional neural network architecture for detecting the P300 signal are examined and compared. Three types of filters are analyzed on minimally preprocessed EEG data. The 1-dimensional filters of CNN1D models are observed to outperform all the other models in terms of signal recognition accuracy. Approximately 73.9% within-subject and 64.1% cross-subject single trial accuracy averaging over 6 subjects are achieved with CNN1D. The highest within-subject single trial accuracy of 80.9% is achieved for subject 3 using 1-D filters. The lowest single trial accuracy achieved with this method is above 62.9%, which is more than what has been achieved by Krusienski et al. [14] in their implementation of SWLDA on the same dataset. CNN2D's 2-D filters' performance in detecting P300 and non-P300 signals is similar to that 1-D filters of CNN1D in case of cross-subject analysis and results in an average of 63.8% single trial accuracy.

The concept of nonzero virtual nodes is introduced with 3-D filters in CNN3D and analyzed to determine its effect on the signal recognition ability. No significant performance improvement is observed but the overfitting of the model is considerably reduced for virtual nodes arrangement. CNN1D and CNN3D-asc. implementations that include all the channels in developing the response signal detection model show higher accuracy compared to all the other methods in the within-subject paradigm. CNN2D shows comparable performance to these methods in case of cross-subject signal detection as well. The network architecture in the cross-subject models requires more hidden units in the convolutional layers to learn the latent features of the training data of 5 subjects and to fit the models. The size of the time-window used in this analysis is 120 steps representing 500 ms after the appearance of the stimulus. Incorporating more steps in building the models does not make much difference in their signal recognition ability. The required number of samples for training the models for extracting similar performance seems to be much less than what is used in this analysis in the preliminary studies.

5.2 Future Work

There are a number of questions that are yet to be answered following the findings of this thesis such as, the counterintuitive results of the virtual nodes implementation, the possibility of noisy labels in the training data, the inconsistent pattern in the feature maps etc. Some of these points and the possible methods for their analysis are discussed in this section.

Although the single-trial accuracy achieved with these models is higher than that of most other state-of-the-arts detection algorithms reviewed in chapter 2, it needs to be improved for practical use and deployment in the BCI applications. Taking average over multiple trials gets rid of noises in the training samples and results in better signal detection ability. Feeding multiple-trials input samples into the network may result in improved recognition ability of the models.

In our implementation of virtual nodes interpolation has been used without taking the absolute distance between the adjacent nodes into account. Extrapolation of the virtual nodes by incorporating the distance between the neighboring channels may perform more intuitively in the weighted CNN3D models.

Leaving out 9 channels results in decreased performance in this analysis. Therefore, it is important to understand which channels contribute the most and the least in detecting a response signal. These channels may vary depending on the subject but there is also a possibility that some of the channels are not necessary for the classification irrespective of the subject. The method of occlusion can shed light on the least performing and the most contributing channels in distinguishing a P300 signal.

One of the objectives of implementing the convolutional network architecture in detecting the P300 signal is to observe the latent features or pattern learned by the models. However, the feature

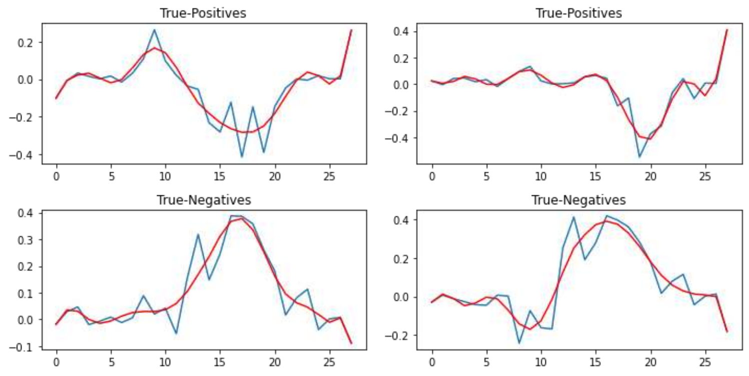


Figure 5.1: Weighted output features of CNN1D followed by a GRU layer.

maps and the weights of the best performing model CNN1D do not show any consistent pattern for the response signal and that may be due to the dimensionality reduction of the channels. Adding a GRU layer after the last CNN layer to map all the reduced spatial dimensions to a single dimension containing only the temporal feature steps may capture the sequential properties of the response signal and give more insight into what pattern or latent characteristics in the input data the neural network is using as the distinguishing feature in predicting a P300 signal. The results of a simple initial test of the use of a GRU layer are shown in Figure 5.1.

The models' performance on some subjects' data is lower than that on the other subjects' data. That may be due to the noises or faulty labeling of the response signals. Abstention loss function [29] may be utilized to get rid of the noisy and confusing labels and improve the models' performance.

Bibliography

[1] J J Vidal. Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2(1):157–180, 1973. PMID: 4583653.

[2] Andrea Kübler, Boris Kotchoubey, Jochen Kaiser, Niels Birbaumer, and Jonathan R. Wolpaw. Brain-computer communication: Unlocking the locked in. *Psychological Bulletin*, 127(3):358–375, May 2001.

[3] Hans Berger. Über das elektrenkephalogramm des menschen. *Archiv für Psychiatrie und Nervenkrankheiten*, 87(1):527–570, Dec 1929.

[4] L.A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.

[5] F. Nijboer, E. W. Sellers, J. Mellinger, M. A. Jordan, T. Matuz, A. Furdea, S. Halder, U. Mochty, D. J. Krusienski, T. M. Vaughan, J. R. Wolpaw, N. Birbaumer, and A. Kübler. A p300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 119(8):1909–1916, Aug 2008.

[6] Terence Picton. The p300 wave of the human event-related potential. *Journal of clinical neurophysiology : official publication of the American Electroencephalographic Society*, 9:456–79, 11 1992.

[7] Eric Sellers and Emanuel Donchin. A p300-based brain-computer interface: Initial tests by als patients. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 117:538–48, 04 2006.

[8] Hubert Cecotti and Axel Graeser. Convolutional neural networks for p300 detection with application to brain-computer interfaces. *IEEE transactions on pattern analysis and machine intelligence*, 33:433–45, 03 2011.

[9] Reza Fazel-Rezai, Brendan Z. Allison, Christoph Guger, Eric W. Sellers, Sonja C. Kleih, and Andrea Kübler. P300 brain computer interface: current challenges and emerging trends. *Frontiers in neuroengineering*, 5:14–14, Jul 2012.

[10] Ulrich Hoffmann, Jean-Marc Vesin, Touradj Ebrahimi, and Karin Diserens. An effcient p300-based brain computer interface for disabled subjects. *Journal of neuroscience methods*, 167:115–25, 02 2008.

[11] Alain Rakotomamonjy and Vincent Guigue. Bci competition iii: Dataset ii- ensemble of svms for bci p300 speller. *IEEE Transactions on Biomedical Engineering*, 55:1147–1154, 2008.

[12] Abdulhamit Subasi and Mehmet Gürsoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert Systems with Applications*, 37:8659–8666, 12 2010.

[13] Priya Velu and Virginia de Sa. Single-trial classification of gait and point movement preparation from human eeg. *Frontiers in neuroscience*, 7:84, 06 2013.

[14] Dean Krusienski, Eric Sellers, Dennis Mcfarland, Theresa Vaughan, and Jonathan Wolpaw. Toward enhanced p300 speller performance. *Journal of neuroscience methods*, 167:15–21, 2008.

[15] Hongchang Shan, Yu Liu, and Todor Stefanov. A simple convolutional neural network for accurate p300 detection and character spelling in brain computer interface. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1604–1610. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[16] Alexander Craik, Yongtian He, and José Contreras-Vidal. Deep learning for electroencephalogram (eeg) classification tasks: A review. *Journal of Neural Engineering*, 16, 02 2019.

[17] Vernon Lawhern, Amelia Solon, Nicholas Waytowich, Stephen Gordon, Chou Hung, and Brent Lance. Eegnet: A compact convolutional network for eeg-based brain-computer interfaces. *Journal of Neural Engineering*, 15, 11 2016.

[18] Eduardo Carabez, Miho Sugi, Isao Nambu, and Yasuhiro Wada. Convolutional neural networks with 3d input for p300 identification in auditory brain-computer interfaces. *Computational Intelligence and Neuroscience*, 2017:1–9, 11 2017.

[19] Xiaoyan Wei, Lin Zhou, Ziyi Chen, Liangjun Zhang, and Yi Zhou. Automatic seizure detection using three-dimensional cnn based on multi-channel eeg. *BMC medical informatics and decision making*, 18(Suppl 5):111–111, Dec 2018.

[20] Jungchan Cho and Hyoseok Hwang. Spatio-temporal representation of an electoencephalogram for emotion recognition using a three-dimensional convolutional neural network. *Sensors*, 20:3491, 06 2020.

[21] Xiangmin Lun, Zhenglin Yu, Tao Chen, Fang Wang, and Yimin Hou. A simplified cnn classification method for mi-eeg via the electrode pairs signals. *Frontiers in Human Neuroscience*, 14:338, 2020.

[22] Feng Li, Xiaoyu Li, Fei Wang, Dengyong Zhang, Yi Xia, and Fan He. A novel p300 classification algorithm based on a principal component analysis-convolutional neural network. *Applied Sciences*, 10:1546, 02 2020.

[23] P. Shukla, R. Chaurasiya, and S. Verma. Brain–computer interface-based single trial p300 detection for home environment application. *Electronics Letters*, 56:1392–1395(3), December 2020.

[24] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[25] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[26] Luis Moctezuma and Marta Molinas. Towards a minimal eeg channel array for a biometric system using resting-state and a genetic algorithm for channel selection. *Scientific Reports*, 10, 09 2020.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[28] Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 10 1998.

[29] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. Combating label noise in deep learning using abstention. *Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97*, 2019.