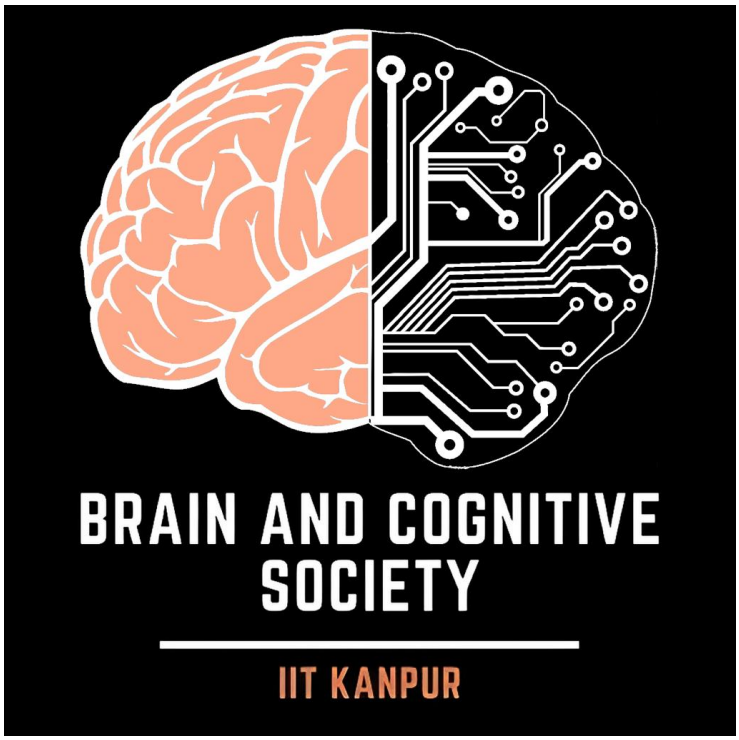


The Mind's Mirror

Endterm Report

Mentors:

Poojal Katiyar
Debarpita Dash
Shrimi Agrawal
Eeshwari Sunkersett

Mentees:

Akshay Reddy , Anika Gupta , Hitarth Makawana , Chaitanya Nitawe , Kshitiz Tyagi , Lavina Goyal , Om Chaudhari , Shivang Sonker , Shruti Sekhar , Sparsh Gupta , Umesh Varun

Abstract

This report summarizes the learnings of mentees during the Summer Project The Mind's Mirror, organized by the Brain and Cognitive Society at IITK. The project's primary objective is to create a visual object classifier using human brain signals.

Contents

1. Introduction

This project aims to utilize EEG signals for the classification of responses corresponding to visual stimuli using 1D CNNs which are effective in capturing temporal dependencies within sequential data, making them suitable for analyzing the dynamic responses of the brain to visual stimuli. Various feature extraction methods are utilized to enhance the representation of EEG data. These methods extract relevant information from raw EEG signals, such as spectral features (like power spectral density), time-domain features (such as mean and variance), and other statistical measures that characterize the signal's complexity and pattern. Furthermore, the project employs a range of machine learning techniques including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes, and XGBoost for EEG classification.

The classification of EEG data holds significant implications across several domains. In neuroscience, it provides insights into brain activity patterns in response to visual stimuli, facilitating the study of brain function, disorders, and treatment protocols.In healthcare, the ability to classify EEGs based on specific image stimuli has transformative potential for diagnostics. By identifying distinct EEG patterns associated with certain images, this approach enhances the detection and early intervention of neurological conditions such as epilepsy and Alzheimer's disease, thereby enabling more personalized and effective treatment strategies.

2. EEG Experiment

2.A. The Device

- Our EEG device consists of 4 electrodes that we have to place on the head and 2 others that are supposed to remain behind the ears in contact with the skin at all times.
- This is necessary to get accurate readings as when the electrodes touch the skin, you get negative values.
- The device lights up green when it's turned on and lights up blue when it's connected to another device. Yellow is displayed when the device is on standby.
- While taking readings, be sure to check whether your device is still connected to the EEG device (it must be lighted up blue during the entire collection process).

2.B. Setting up environment

Install pygame and websocket-client and install pylsl according to the method for your respective Operating System.

2.C. The Experiment

The experiment will display 10 numbers (0-9) and one non-numerical image in a random sequence, each shown for 5 seconds. To assess attention, a cross will randomly appear on the screen. Clicking the cross will return it to the center, and clicking it again will reveal the image, ensuring the participant's attention is engaged.

2.D. Performing the experiment

- Turn on the machine
- Connect your PC to the machine and connect both the machine and the PC on the same network
- Run main.py in the terminal and input the name of the user
- The program will save the data in a CSV of name username with CSV extension

3. Pre-Processing of EEG Data

Each dataset corresponds to images labeled 0 to 9, along with an additional non-numerical image represented by -1. To ensure consistency and comparability between the datasets pre-processing is applied on the data.

3.A. Normalization

Normalization in pre-processing scales data to a standard range, usually [0, 1] or [-1, 1]. This step is crucial for:

1. **Uniform Scale**: Ensures data points are comparable, improving the performance of machine learning models.
2. **Numerical Stability**: Reduces numerical errors during computation.
3. **Improved Convergence**: Enhances the performance of optimization algorithms.

Techniques used:

- **Min-Max Normalization**: This technique scales the data to a fixed range, where the minimum value of the feature becomes 0 and the maximum value becomes 1. It is calculated by subtracting the minimum value of the feature from each value and then dividing by the range (difference between maximum and minimum values).
- **Z-Score Normalization**: Also known as standardization, this technique scales the data to have a mean of 0 and a standard deviation of 1. It involves subtracting the mean of the feature from each value and dividing by the standard deviation of the feature.

3.B. Interpolation

To ensure consistency and comparability between the datasets from different devices, the data was resampled using linear interpolation. Interpolation is a method of constructing new data points within the range of a discrete set of known data points. In this case, linear interpolation was used to resample the data so that all data arrays were of the same size, facilitating subsequent analysis and model training.

3.C. Bandpass Filtering

Bandpass filtering in preprocessing refers to the technique of selectively allowing a range of frequencies to pass through while attenuating frequencies outside this range. This step is essential for:

1. **Frequency Selection**: By isolating specific frequency bands, bandpass filtering helps focus analysis on relevant signal components, such as eliminating noise or extracting desired features.
2. **Signal Enhancement**: It improves the signal-to-noise ratio by attenuating unwanted frequencies, thereby enhancing the quality of the signal for further analysis or processing.
3. **Preprocessing for Analysis**: Bandpass filtering prepares signals for subsequent analysis techniques, such as feature extraction or classification, by highlighting relevant frequency components.

3.C.I. Techniques used:

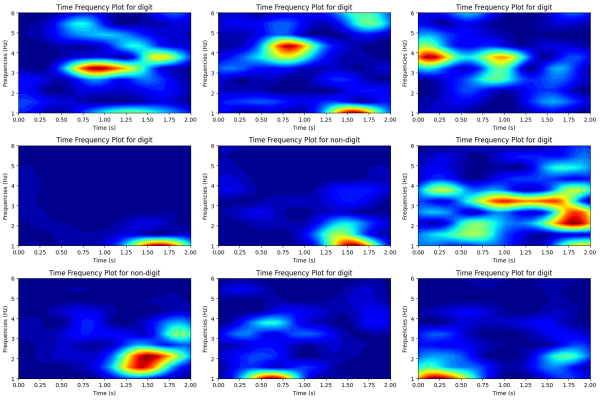
Finite Impulse Response (FIR) Filter: This filter is characterized by a finite duration impulse response, typically used for linear phase response and precise control over frequency characteristics.

Infinite Impulse Response (IIR) Filter: This filter uses feedback in its implementation, allowing for a more efficient design with sharper frequency response characteristics.

Bandpass filtering involves specifying parameters such as the center frequency and bandwidth to define the range of frequencies to pass through. It is applied as a preprocessing step to signal data before conducting detailed analysis or feeding into machine learning models. Bandpass filtering ensures that signals are appropriately processed to focus on specific frequency components relevant to the intended analysis or application, enhancing the overall effectiveness of signal processing techniques.

3.D. Time-frequency plots

For a more comprehensive understanding of the data characteristics, we generated time-frequency plots for visualization.

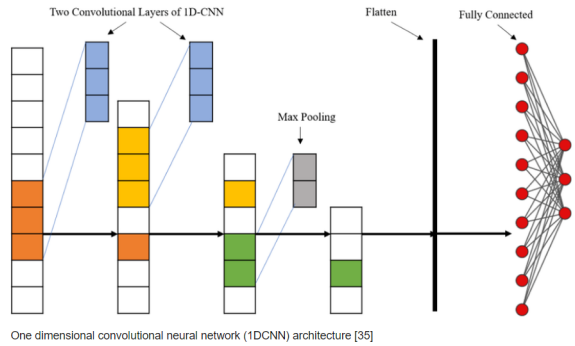


4. Classification of Signals into Numeric and Non-Numeric using 1D CNN

A 1D Convolutional Neural Network (CNN) is designed for analyzing sequences of data , such as time series or signals. Unlike 2D CNNs used for image processing, 1D CNNs apply convolutional operations along one dimension.

4.A. Convolution

Convolutions in a 1D Convolutional Neural Network (1D CNN) involve applying filters to a sequence of data to extract features. These filters slide over the input data, performing element-wise multiplications and summing the results to produce feature maps. This process helps the network learn important patterns in the data.



One dimensional convolutional neural network (1DCNN) architecture [35]

4.A.I. Filters

A filter (also called a kernel) is a small matrix of weights. They are small arrays of numbers used to detect specific patterns in the input data. The size of the filter is smaller than the input data. For a 1D CNN, the filter might be a 1D array

4.A.II. Sliding the filter

The filter slides over the input data (signal or image) from the beginning to the end. At each position, an element-wise multiplication is performed between the filter and a subset of the input data (the same size as the filter).

4.A.III. Dot Product and Summation

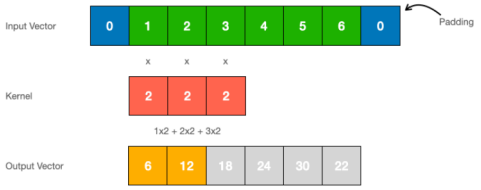
The element-wise multiplication results are summed up to produce a single value. This single value becomes an entry in the output feature map.

4.A.IV. Strides

The stride is the step size by which the filter moves across the input data. A stride of 1 means the filter moves one position at a time, while a stride of 2 means it moves two positions at a time.

4.A.V. Padding

Padding involves adding extra values (typically zeros) around the input data to ensure the filter can fit properly at the borders. 'Valid' padding means no padding, and the output size is smaller than the input size. 'Same' padding means padding is added to keep the output size the same as the input size.



4.B. Pooling

Pooling in a 1D Convolutional Neural Network (1D CNN) is a technique used to reduce the dimensionality of feature maps, making the network more manageable and efficient while retaining the most critical information. Pooling reduces the size of feature maps by summarizing the values in a local neighborhood of the feature map.

Types of Pooling:

1. **Max Pooling** : Selects the maximum value within the pooling window
2. **Average Pooling** : Computes the average value within the pooling window.

Pooling reduces the size of the feature maps, decreasing the number of parameters and computations in the network.By reducing the dimensionality, pooling helps in controlling overfitting by forcing the network to generalize better.

4.C. Activation

Activation functions play a critical role in introducing non-linearity into the model, enabling it to learn complex patterns and

representations from the input data. Activation functions are applied after each convolutional layer to the feature maps produced by the convolutions.

Types of Activation function:

1. **ReLU (Rectified Linear Unit)** : ReLU is one of the most commonly used activation functions in CNNs. It replaces all negative values in the feature map with zero and keeps positive values unchanged.Introduces non-linearity, helps mitigate the vanishing gradient problem, and is computationally efficient.It can lead to dead neurons where some neurons never activate during training.

   f(x)=max(0,x)

2. **Leaky ReLU** : A variant of ReLU that allows a small, non-zero gradient when the unit is not active. This helps mitigate the dying ReLU problem.

   f(x)=max(0.01x,x)

3. **Sigmoid** : The sigmoid function squashes input values to the range [0, 1]. It is often used in binary classification tasks for the final output layer.Ii is useful for binary classification problems.

4. **Hyperbolic Tangent(Tanh** : Tanh squashes input values to the range [-1, 1]. It is zero-centered, making it preferable to sigmoid in some cases. Its zero-centered, which can make optimization easier.

5. **Softmax** : It used in the output layer for multi-class classification problems. It converts logits into probabilities that sum to 1. It provides a probabilistic interpretation of the output

4.D. Fully Connected Layers

Fully Connected Layers (also known as dense layers) are typically used at the end of the network to perform classification or regression tasks based on the features learned by the convolutional layers.

After convolutional and pooling layers, the feature maps are typically flattened into a 1D vector.This flattening process converts the spatial hierarchy of the feature maps into a single vector representation.

Fully connected layers take this flattened vector as input and perform classification or regression based on these extracted features. Each neuron in a fully connected layer is connected to every neuron in the previous layer, allowing the network to learn complex relationships.

4.E. Regularization Techniques: Dropout,Early Stopping and Learning Rate Scheduler

1. **Dropout** : A powerful regularization technique used to prevent overfitting and improve the generalization ability of the model. By dropping neurons, the network is forced to learn redundant representations of the data, reducing the reliance on specific neurons.

   Dropout is implemented as a separate layer where rate specifies the fraction of neurons to drop during training. During evaluation (prediction), Dropout is turned off, and all neurons are used to ensure accurate predictions.

2. **Early Stopping** : A regularization technique commonly used to prevent overfitting and optimize model performance during training. It helps prevent unnecessary computational resources from being spent on training when further training epochs are unlikely to improve the model's performance.

   Components of Early Stopping

   a) Monitor: Specifies the metric to monitor during training .

   b) Patience: Number of epochs with no improvement after which training will be stopped .

   c) Restore Best Weights: Restores the weights of the model to those that yielded the best monitored quantity.

3. **Learning rate scheduler** : A technique used to dynamically adjust the learning rate during training. This helps in improving convergence, optimizing training speed, and potentially achieving better model performance.

   It adjusts the learning rate based on the training progress, allowing for faster

convergence and potentially more stable training.

Components of Learning rate scheduler

a) Scheduler Function: Defines how the learning rate changes based on epoch and current learning rate .

b) LearningRateScheduler: Keras callback that applies the defined scheduler function during training .

c) Learning Rate Adjustment: In the example, the learning rate is reduced exponentially after the 10th epoch .

4.F. Insights

Initially when same number of kernels were applied in all the convolution layers,accuracy of the model was somewhat less. Accuracy of the model got increased when we increased the number of kernels in the inner convolution layers as it was able to extract more features.

Initially when data augmentation was applied to the EEG data and then passed into 1D CNN accuracy was somewhat low when we didn't use dropout. Dropout actually helped the model by preventing it from learning the noise from data and hence accuracy got increased.

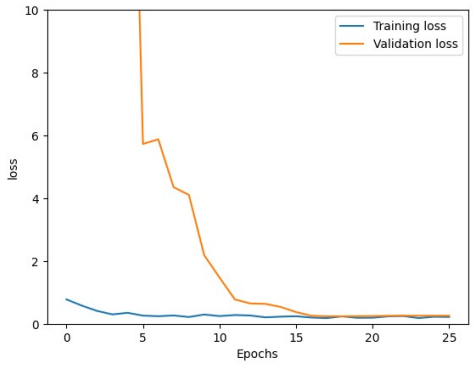
5. Data Augmentation Methods

5.A. SMOTE

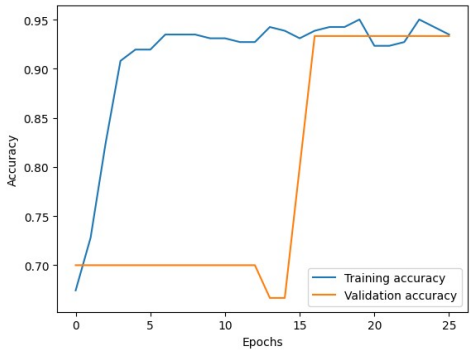
SMOTE is an oversampling technique that differs from traditional oversampling. The minority data population is duplicated in traditional oversampling like random oversampling.While it increases the amount of data, it does not provide the machine learning model with any additional knowledge or variation.

SMOTE generates synthetic data using knn. SMOTE begins by selecting random data from the minority class after which the data's k-nearest neighbors are determined. The random data and the randomly chosen k-nearest neighbour would then be combined to create synthetic data.

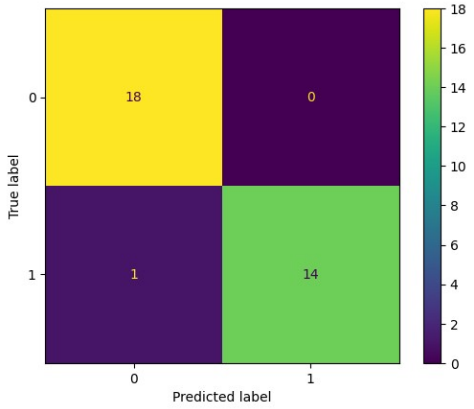
These are the results obtained after applying SMOTE oversampling



Loss vs Epochs graph



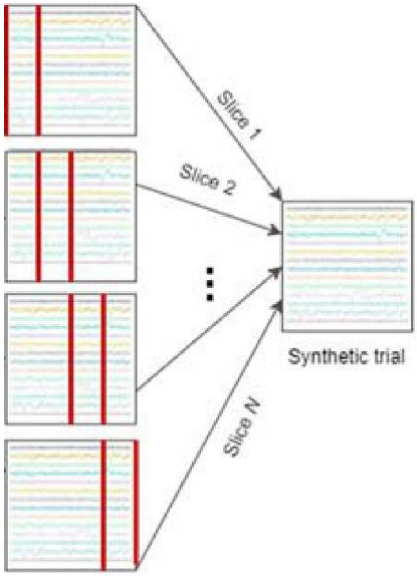
Accuracy vs Epochs graph



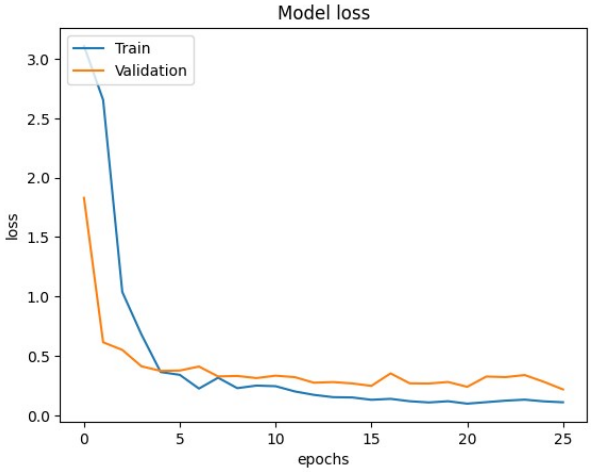
Confusion Matrix

5.B. Time Slicing

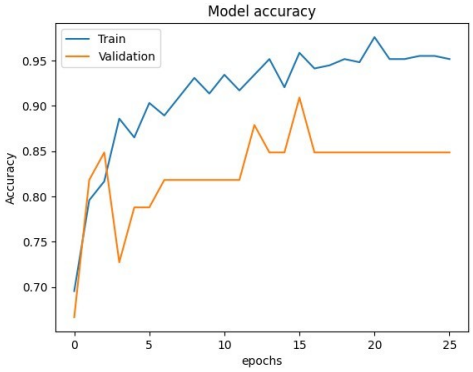
This involves selecting N trials and make every trail into N equal time slices. Then, will generate a new trial by recombining using one time slice from each of the selected trials in an order



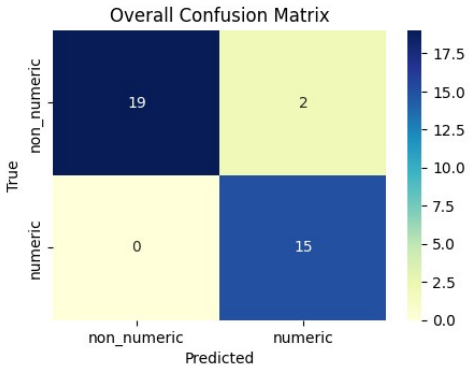
Time Slicing



Loss vs Epochs graph



Accuracy vs Epochs graph

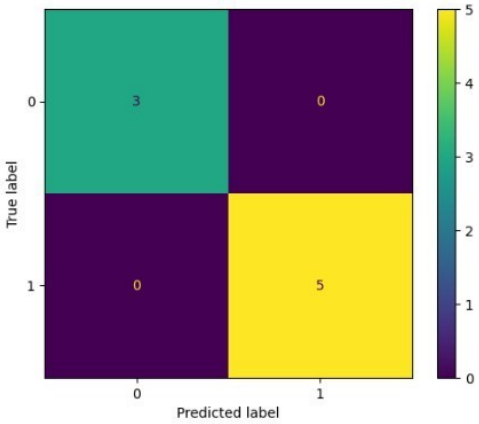


Confusion Matrix

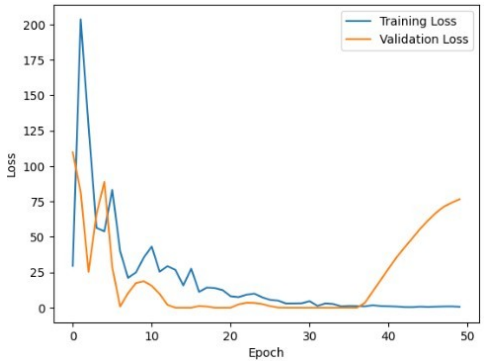
5.C. Downsampling

Downsampling is a technique used to handle class imbalance in datasets, especially in binary classification problems. The goal is to reduce the number of instances of the majority class to match the number of instances of the minority class.

These are the results obtained after applying downsampling to the EEG data.

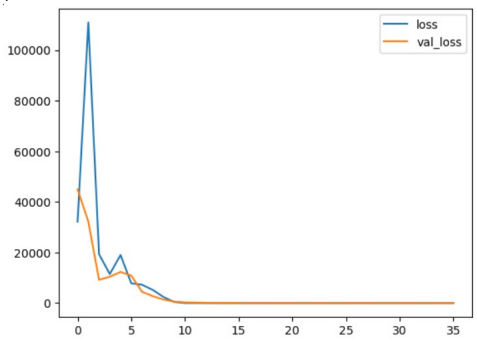


Confusion Matrix of test data

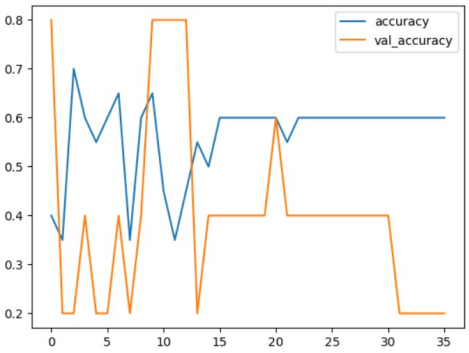


Loss vs Epochs graph

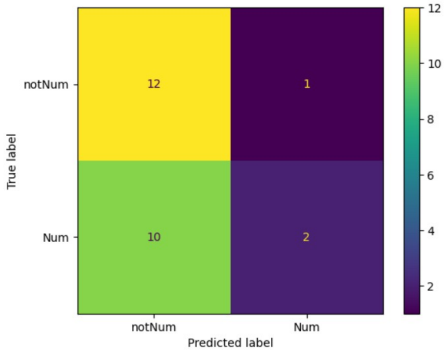
In order to neglect any kind of bias towards a particular kind of data, the data to be included in the downsampled dataset is chosen on a random basis. And the final training data is created by merging the minority and downsampled datasets after shuffling them using numpy.random.shuffle() (optional for datasets which are dependent on previous data).



Loss vs Epochs graph



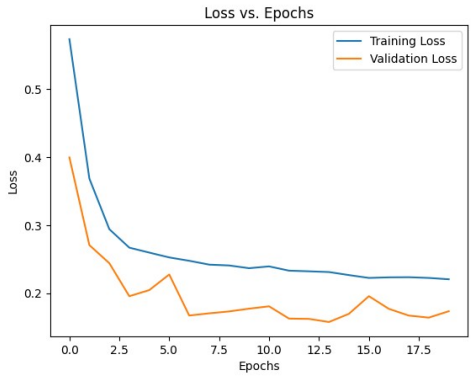
Accuracy vs Epochs graph



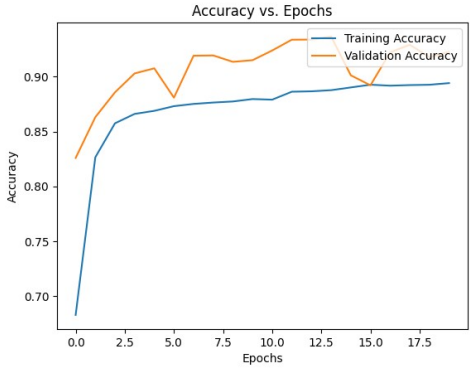
Confusion Matrix

5.D. Bootstrap Sampling

Bootstrap sampling is a statistical method used to estimate the distribution of a sample statistic by resampling with replacement from the original data. We do this to make sure the data in the samples from the non-numerical EEG (1/10th) data can be resampled to the length of the data from the numerical EEG (9/10th) data to ensure there is no bias toward the numerical EEG samples. Then we use the 'sample' method to randomly sample rows in the dataframe so the model can't "guess" the data. Here are some of the results we achieved using this type of resampling:



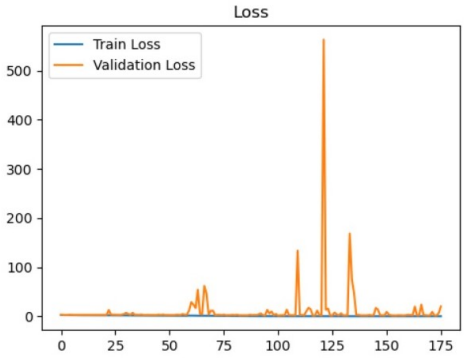
Loss vs Epochs graph



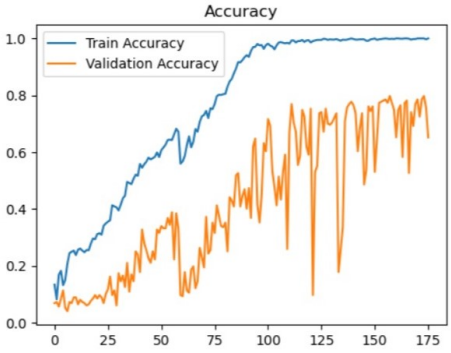
Accuracy vs Epochs graph

5.E. Frequency Shift

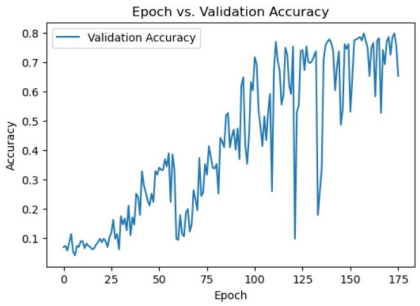
Frequency shift is a data augmentation technique used for time series data. It involves shifting the frequency spectrum of a signal up or down by a certain amount, effectively changing the pitch of the audio without altering its duration. The key idea behind frequency shifting is that shifting the frequency content of an audio signal by a small amount can introduce variations that help improve the robustness and generalization of machine learning models trained on the augmented data.



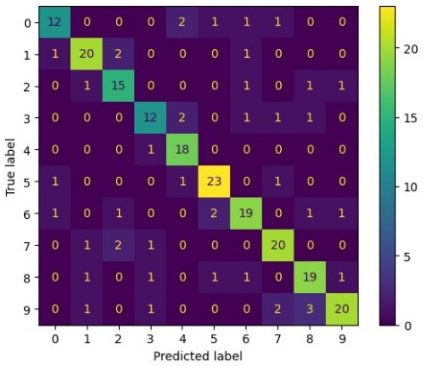
Loss vs Epochs graph



Accuracy vs Epochs graph



Epochs vs val-accuracy graph



Confusion Matrix

5.F. Additional methods

5.F.I. Simple Shift

Shifting signal during training exposes the model to more diverse data, helping it learn features that are invariant to the position of the signal. This improves the model's ability to generalize to real-world scenarios.

5.F.II. Smooth Time Mask

The smooth time mask is a data augmentation technique for time series data, such as electroencephalogram (EEG) signals. It is designed to improve the robustness and generalization of predictive models trained on EEG data.

5.F.III. Adding Noise

Adding noise is an effective data augmentation technique that can improve the robustness and generalization of machine learning models, especially for deep learning on image data.

5.G. Analysis of data augmentation methods

Since the EEG dataset contains the data for 0-9 numerical images and one non-numerical image, the primary calssification into numerical/non-numerical classes suffered a high bias towards the numerical data over non-numerical dataset, due to which different data augmentation methods, stated above were applied. As observed from the accuracy vs epochs and loss vs epochs graphs, SMOTE, time slicing, bootstrap sampling give accurate results for classification of data, whereas Downsampling gives lower accuracy because of the limitation of the availability of data for training. The dataset contains 198 dataframes in total, whereas there are about 20 dataframes corresponding to non-numerical image, as a result of which, when downsampling is applied, the size of the dataset reduces drastically, approximately equal to double the amount of available dataset for non-numerical image, thereby affecting the accuracy of classification.

6. Multiclass Classification of EEG signals

After classifying the EEG signals into numeric and non-numeric classes we tried multiclass classification of EEG signals. In this we extracted features from the EEG signals using

Data augmentation method	Accuracy	Precision	Recall	F1Score
SMOTE Oversampling	96	97	97	97
Time Slicing	94	95	94	94
Downsampling	87.5	100	100	100
Frequency Shift	40	54	60	57
Bootstrap Sampling	95	99	90	94

Table 1: Results after applying various data augmentation methods

various feature extraction methods and then used various machine learning models like knn,svm,naive bayes and xgboost to classify the signals.

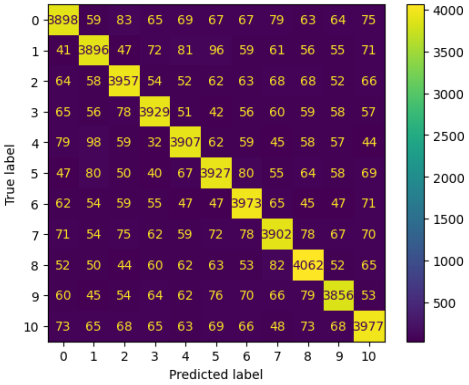
Another approach involved generating spectrograms from the EEG data and leveraging various pretrained CNNs for classification.

6.A. Auto Correlation Method

In this method we obtain features by extracting the information of power.

Autocorrelation gives the information about how similar a signal is to itself over time. In this method first we compute the autocorrelation function of the signal and then apply fast fourier transform to the autocorrelation function. This gives us the power spectrum which tells us how the power of signal is distributed over various frequencies.

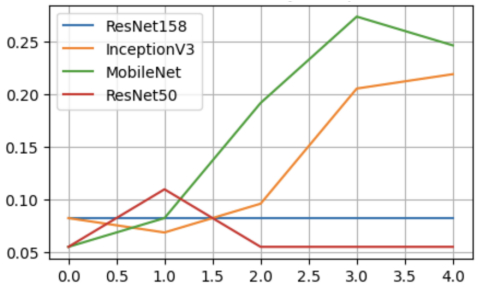
Then we use this data of powers into the machine learning models for classification.These are the results obtained after applying this method and then passing it on to knn:



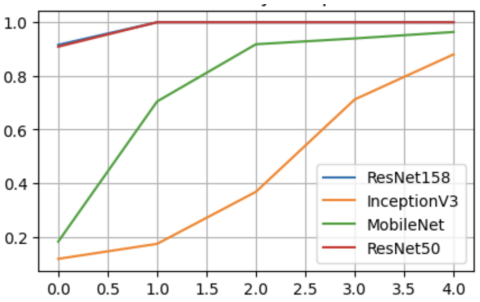
Confusion Matrix

6.B. Spectrogram-Based Classification with Pretrained CNNs

Classification of EEG data is done using spectrogram images generated from the EEG signals of four channels. The spectrograms are combined into a grid-like image, which is then used as input to various pre-trained convolutional neural network (CNN) models like ResNet158, InceptionV3, MobileNet, and ResNet50.A time-slicing augmentation method was applied to enhance the dataset and improve the model's generalisation capabilities. The augmentation process involved creating multiple shifted versions of each EEG signal. MobileNet outperformed the other models in terms of validation accuracy, indicating better generalization to unseen data. InceptionV3 also showed promise, while ResNet158 and ResNet50 exhibited signs of overfitting, with high training accuracy but low validation accuracy.



Validation Accuracy vs Epochs graph

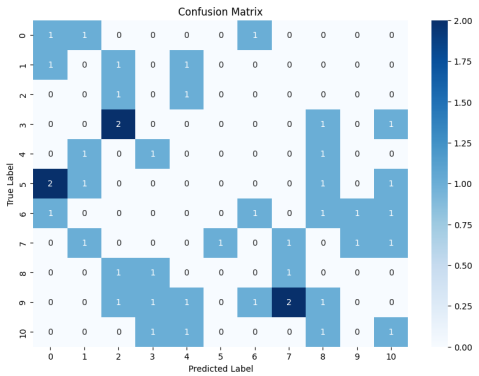


Train Accuracy vs Epochs graph

The models' low validation accuracy may stem from their pre-training on ImageNet, which features natural images. Spectrograms derived from EEG signals have distinct characteristics not well-captured by these pretrained weights. Fine-tuning models specifically for spectrogram data or using models pretrained on more relevant datasets could enhance classification accuracy.

6.C. STFT-Based Feature Extraction

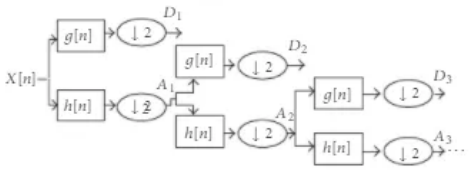
Using STFT for feature extraction from EEG data is a powerful technique that leverages the rich information contained in both the time and frequency domains, facilitating more accurate and insightful analysis and classification of EEG signals. These features are input to machine learning algorithms such as SVM and KNN for classification. However, STFT has limitations as the data has to be very large in order for there to be proper overlapping windows for spatial and temporal resolution. This is primarily one of the reasons for the accuracy to be around 0.36 with SVM and 0.32 with KNN. As you can see from the confusion, this model does not seem to provide any meaningful results:



Confusion Matrix

6.D. Feature extraction using Discrete Wavelet Transformation(DWT)

Wavelet transform feature extraction is a method used to analyze and decompose signals or data into different frequency components with both time and frequency localization. This method is particularly useful for analyzing non-stationary signals like EEG signals, where frequency components change over time.Wavelet transform feature extraction provides a powerful tool for analyzing complex signals by capturing both time and frequency information, making it suitable for a wide range of applications in machine learning and signal processing. Wavelet Transformation is again classified into Discrete Wavelet Transorm (DWT) and Continuous Wavelet Tranform (CWT) DWT has evolved to address the weakness of CWT that is the scaling and translation parameter changes continuously. DWT is defined in the base of multiscale representation. Each scale represents the unique thickness of EEG signal. The multiresolution decomposition of EEG data is as follows,



Decomposition of DWT

Each step has digital filters they are,g(n) and h(n).g(n) is discrete mother wavelet, it is high pass in nature and h(n) is low pass in nature.The number of steps depends on EEG data component with dominant frequency. Each step gives two, one is detail about the signal (D) and the other is approximation of the signal (A). A becomes the output of the next step.

There are a lot of advantages in this method because of precisely describing the features of the signal segment within specified frequency domain and localized time domain.

6.D.I. Insights

We used features obtained from DWT decomposition over EEG signal from 4 channels individually for classification.Using 1st channel, we got 92.7 percentage accuracy with SVM based classifier.Using 2nd channel,we got 86.7percentage accuracy with SVM.Using 3rd channel,we got 83 percentage accuracy with SVM based classifier.Using 4th channel,we got 75 percentage accuracy with SVM based classifier.From this we can observe that EEG signals from channels 1 and 2 are more effective to image stimuli.

Feature extraction method	Classification model	Accuracy	Precision	F1Score
Autocorrelation	knn	86	86	86
Fast Fourier Transform	CNN	79	79	79
Auto regressive (Burg's method)	knn	98	98	98
Discrete Wavelet Transform	SVM	92.5	93	93

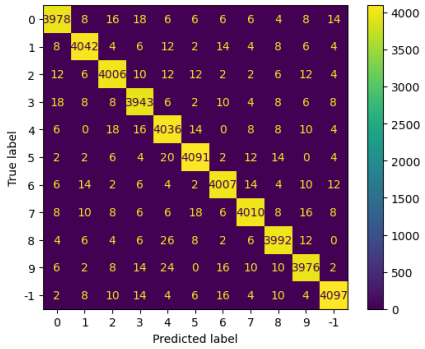
Table 2: Feature Extraction Methods

6.E. Auto Regressive Methods - Burg's Method

The Burg's method involves calculating coefficients for the Power Spectral Density distribution i.e. PSD coefficients for the given EEG potential values for different frequencies. The Burg's method is applied using burg method from linear regression model from statsmodel library, it outputs rho and sigma. Rho typically represents the reflection coefficients (also known as partial correlation coefficients) obtained during the estimation process. Sigma usually represents the prediction error variance. Using them the PSD coefficients are calculated which represents the power values for different frequencies. after standardizing these powers, they are fed into classification models such as KNN for predicting the classes. The input in KNN is somewhat different from general dataframes. The input are given as one row at a time just like how the data is collected by EEG device. For hyperparameter tuning, GridSearchCV is used.

6.E.I. Insights

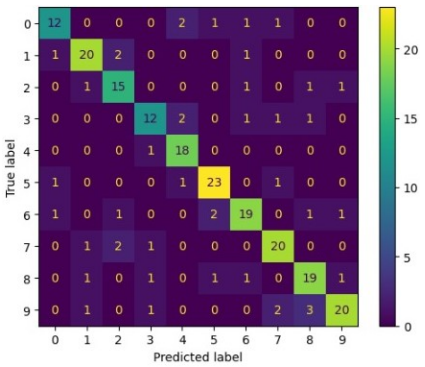
The Burg's method of feature extraction followed by classification using knn gave a very high accuracy of 98 percentage. The reason of very high accuracy might be the process of feeding the data to the model. The data can be thought of a continuous signal, meaning a particular element of data is just a row and not an enitre dataframe because it is also the same way in which an EEG device records a reading i.e. one row at a time which is then clustered together by knn by analysing similar features from the data. Taking into account one row at a time, the dataset length becomes large enough unlike the no. of classes available for classification, which then have to be repeated for each rows corresponding to an image.



Confusion Matrix after applying Burg's Method

6.F. Feature Extraction Using Fast Fourier Transform

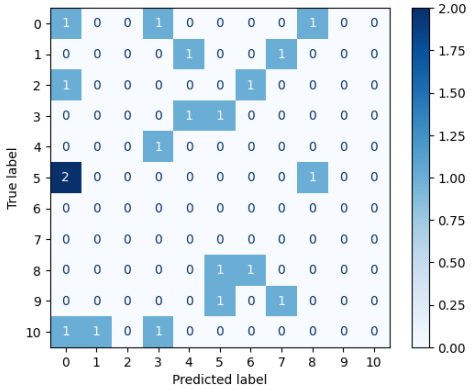
FFT (Fast Fourier Transform) is a commonly used method for feature extraction in EEG signal classification. Here are the key points: FFT transforms EEG signals from the time domain into the frequency domain, allowing features to be extracted based on the frequency spectrum. FFT is easy to implement and computationally efficient compared to other frequency domain methods. However, it lacks time localization and is sensitive to noise. In summary, FFT is a simple yet effective technique for extracting frequency-based features from EEG signals for classification tasks like sleep staging and brainwave identification.



Confusion Matrix

6.G. Feature extraction using Principal Component Analysis and Independent Component Analysis

Principal Component Analysis (PCA) and Independent Component Analysis (ICA) for feature extraction. PCA transforms the original correlated features into uncorrelated principal components, maximizing variance and reducing dimensionality while preserving significant patterns. ICA, on the other hand, finds statistically independent components, useful for separating mixed signals in EEG data.



Confusion Matrix after applying PCA and ICA

We employed Support Vector Machine (SVM), Random Forest, and Gradient Boosting models our models yielded validation accuracies around 5.26% and test accuracies around 10%. These results suggest that while PCA and ICA effectively reduced dimensionality, the extracted features might not have captured the necessary patterns for accurate image prediction from EEG data. The low accuracies indicate potential areas for improvement, such as further tuning the feature extraction process, exploring alternative models, or incorporating additional preprocessing steps to enhance signal quality. The combination of PCA and ICA for feature extraction, followed by SVM, Random Forest, and Gradient Boosting models, provided valuable insights into the complexities of predicting images from EEG data, highlighting the challenges and underscoring the need for continued research and refinement to achieve better predictive performance.

7. Future Work

7.A. Improving Accuracy

In the future, we can look into further methods to improve our current accuracy. Some methods we found that increased our accuracy drastically are below: Data Augmentation: The data we collected for our project was not enough due to limitations of the device and the number of people. Therefore, to increase number of samples we implemented various data augmentation techniques to ensure better training. Extracting more features: Extracting various features apart from mean and variance can ensure better data classification. Hyperparametric Tuning: This method sets the best parameters for SVM and KNN classification which helped increase our accuracy by a lot.

7.B. RCNN and LSTM

This is another method we can look into for data classification. This can be an especially powerful method when dealing with sequential data like EEG signals. Implementing RCNN for EEG classification involves balancing between capturing temporal dependencies and spatial features within EEG signals. We can use recurrent layers like Long Short-Term Memory (LSTM) to capture temporal dependencies within each segment of EEG data. To learn spatial features from the EEG signals within each segment, we can use convolutional layers. Then we can add pooling layers (like MaxPooling) to downsample the features learned by the convolutional layers.

8. Resources:

Learning EEG - https://www.learningeeg.com/

8..I. References for Approach:

Brain Digi CNN - https://link.springer.com/article/10.1007/s13369-022-07313-3
CNN GAN - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10708586/pdf/sensors-23-09351.pdf
CNN,LSTM,DNN - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10708586/pdf/sensors-23-09351.pdf
CNN - https://eurasip.org/Proceedings/Eusipco/Eusipco2021/pdfs/0001226.pdf

Tensorflow - https://eurasip.org/Proceedings/Eusipco/ Eusipco2021/pdfs/0001226.pdf

Time Slicing - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9433610/
Smote oversampling - https://wellsr.com/python/upsampling-and-downsampling-imbalanced-data-in-python/

8..II. Data Pre-processing

Data Preprocessing - https://www.javatpoint.com/ data-preprocessing-machine-learning
Cross-Validation - https://ai.plainenglish.io/leave-one-subject-out-cross-validation-for-machine-learning-model-557a09c7891d
Batch Normalization - https://deepchecks.com/glossary/batch-normalization/
Chirp Signal - https://www.gaussianwaves.com/2014/07/chirp-signal-frequency-sweeping-fft-and-power-spectral-density-matlab-python/
Gaussian Window - https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.windows.gaussian.html
Triangular Window - https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.windows.triang.html
Rectangular Window - https://docs.scipy.org/doc/scipy/reference/signal.windows.html
Hamming Window - https://docs.scipy.org/doc/scipy/reference/signal.windows.html
Short-Time Fourier Transform - https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.stft.html
Low Pass Filter - https://medium.com/analytics-vidhya/how-to-filter-noise-with-a-low-pass-filter-python-885223e5e9b7
Discrete Fourier And Fast Fourier Transform - https://www.youtube.com/watch?v=TFEf8yOlXyYab$_channel = BarryVanVeen$
Downsampling - https://www.youtube.com/watch?v=TFEf8yOlXyYab$_channel = BarryVanVeen$
PSD Method - https://www.scicoding.com/calculating-power-spectral-density-in-python/
Welch Method - https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html
Periodogram Method - https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.periodogram.html

8..III. Feature Extraction :

Feature extraction in eeg - https://medium.com/@kalaivaaninatarajan/feature-extraction-in-eeg-signals-f8e5dfb94cdf
Methods used:
Spectrum Analysis - https://www.geeksforgeeks.org/spectrum-analysis-in-python/
Calculating PSD - https://www.scicoding.com/calculating-power-spectral-density-in-python/
Stft Feature extraction - https://github.com/AryaKoureshi/Signal-Processing-and-Analysis-of-EEG-Data-Using-Python/blob/main/Analysis$_of_EEG_Data.py$
PCA for feature extraction - https://www.geeksforgeeks.org/principal-component-analysis-pca/amp/
ICA for feature extraction - https://www.geeksforgeeks.org/ml-independent-component-analysis/amp/

8..IV. Data Augmentation: