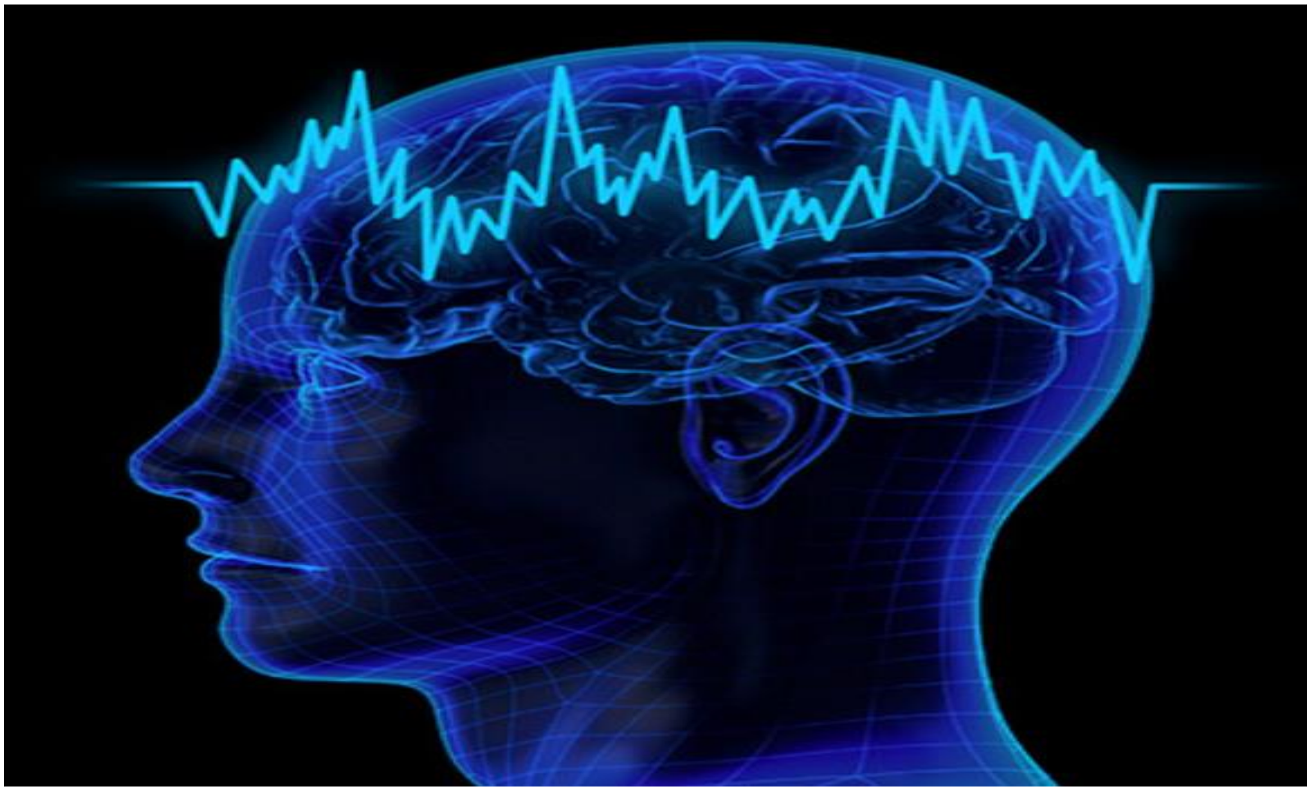


Seizure Classification: Unraveling EEG Signals

IE6400 Foundations Data Analytics Engineering

Group Number 3

Jahnavi Mishra (002724552)

Sahita Bonthu (002823336)

Namrit Sheth (002244393)

Part 1: Introduction and Background Information

Epilepsy, a neurological disorder marked by recurrent seizures and temporary aberrations in cerebral electrical activity, presents formidable challenges in management, especially in cases of intractable seizures. Addressing these challenges requires a comprehensive understanding of the disorder, and the Children's Hospital Boston EEG database emerges as a pivotal resource in this context. This project encapsulates EEG recordings obtained from pediatric subjects grappling with intractable seizures, specifically collected post the cessation of anti-seizure medication. The primary objective is to delineate the characteristics of seizures and evaluate the viability of surgical intervention as a therapeutic avenue.

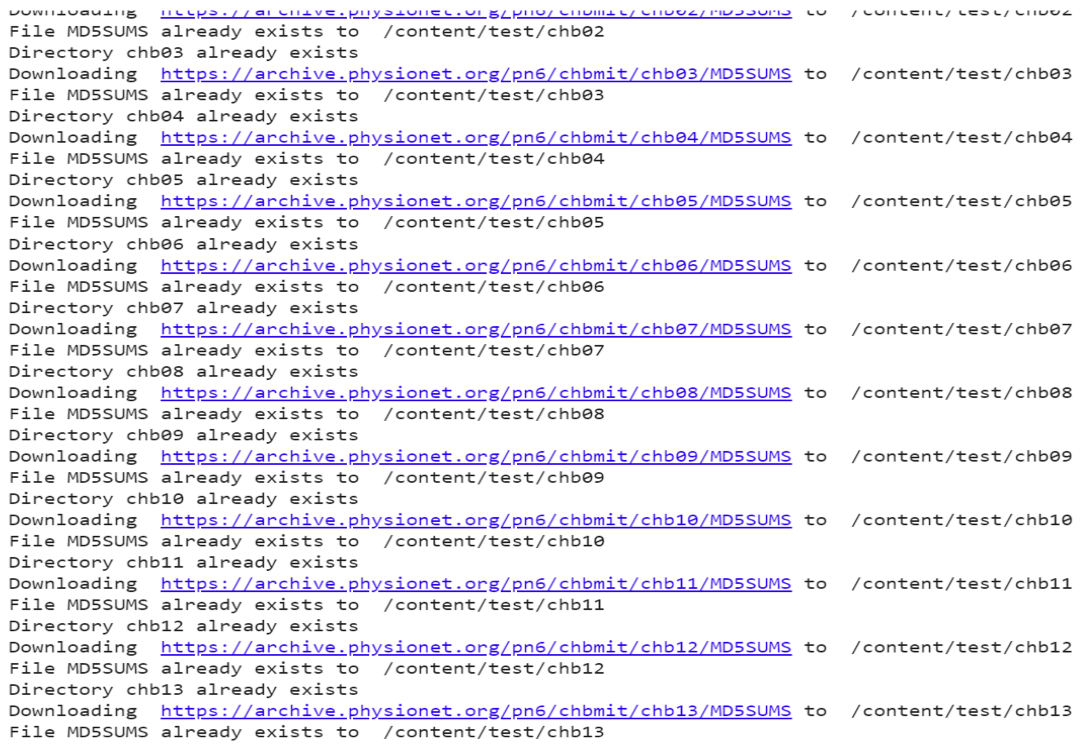
The inherent unpredictability of seizures in epilepsy patients underscores the critical need for efficient detection and response methodologies. This project endeavors to harness the power of machine learning methodologies to construct a sophisticated classification model. The model's purpose is to analyze EEG data and effectively categorize it into distinct seizure types. For this ambitious endeavor, we will leverage a prominent dataset : the CHB-MIT EEG Database. This datasets will serve as the bedrock for training and evaluating the proposed classification model.

In essence, this undertaking not only aims to contribute to the scientific understanding of epilepsy but also seeks to provide a practical and technologically advanced solution for seizure classification through the fusion of machine learning and EEG data analysis. The dataset selected offers a diverse array of EEG recordings, ensuring the model's robustness and adaptability across varying seizure scenarios. As we delve into the technical intricacies of this project, the subsequent sections will elaborate on data preprocessing, feature extraction, model selection, training strategies, evaluation metrics, and future work, collectively charting the course for a comprehensive exploration of epilepsy management through computational methodologies.

Part 2: Data Preprocessing and Feature Extraction

We begin our group project by initializing key variables "folders," "dataset_path," and "base_url"— which form the backbone of our data organization strategy for the CHB-MIT Scalp EEG Database. Through this foundational step, we create the "list_of_folders" and introduce the "find_nth" function, essential components for streamlined dataset management and efficient substring search functionality in our subsequent analyses.

Continuing our project, we systematically address directory management, file downloading, and organization. Starting with the capture and manipulation of the working directory, we dynamically create folders and download crucial files, exemplified by obtaining the 'MD5SUMS' file. This meticulous process ensures a well-structured dataset, a critical requirement for the success of our project.



```
Downloading  https://archive.physionet.org/pn6/chbmit/chb02/MD5SUMS to  /content/test/chb02
File MD5SUMS already exists to  /content/test/chb02
Directory chb03 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb03/MD5SUMS to  /content/test/chb03
File MD5SUMS already exists to  /content/test/chb03
Directory chb04 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb04/MD5SUMS to  /content/test/chb04
File MD5SUMS already exists to  /content/test/chb04
Directory chb05 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb05/MD5SUMS to  /content/test/chb05
File MD5SUMS already exists to  /content/test/chb05
Directory chb06 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb06/MD5SUMS to  /content/test/chb06
File MD5SUMS already exists to  /content/test/chb06
Directory chb07 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb07/MD5SUMS to  /content/test/chb07
File MD5SUMS already exists to  /content/test/chb07
Directory chb08 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb08/MD5SUMS to  /content/test/chb08
File MD5SUMS already exists to  /content/test/chb08
Directory chb09 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb09/MD5SUMS to  /content/test/chb09
File MD5SUMS already exists to  /content/test/chb09
Directory chb10 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb10/MD5SUMS to  /content/test/chb10
File MD5SUMS already exists to  /content/test/chb10
Directory chb11 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb11/MD5SUMS to  /content/test/chb11
File MD5SUMS already exists to  /content/test/chb11
Directory chb12 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb12/MD5SUMS to  /content/test/chb12
File MD5SUMS already exists to  /content/test/chb12
Directory chb13 already exists
Downloading  https://archive.physionet.org/pn6/chbmit/chb13/MD5SUMS to  /content/test/chb13
File MD5SUMS already exists to  /content/test/chb13
```

We further advance our project by introducing crucial parameters such as "time_window" and "time_step," pivotal for temporal considerations in EEG data analysis. It meticulously checks for the existence of preprocessed data files and, if absent, engages in the comprehensive reading and processing of EEG data files designated for training. This phase intricately examines channel labels, extracts pertinent information, and applies specific criteria to ensure the creation of a balanced dataset. Each processed file is meticulously logged, capturing essential details such as the presence of seizure events, the appropriateness of channel labels, and the count of signals integrated into the dataset.

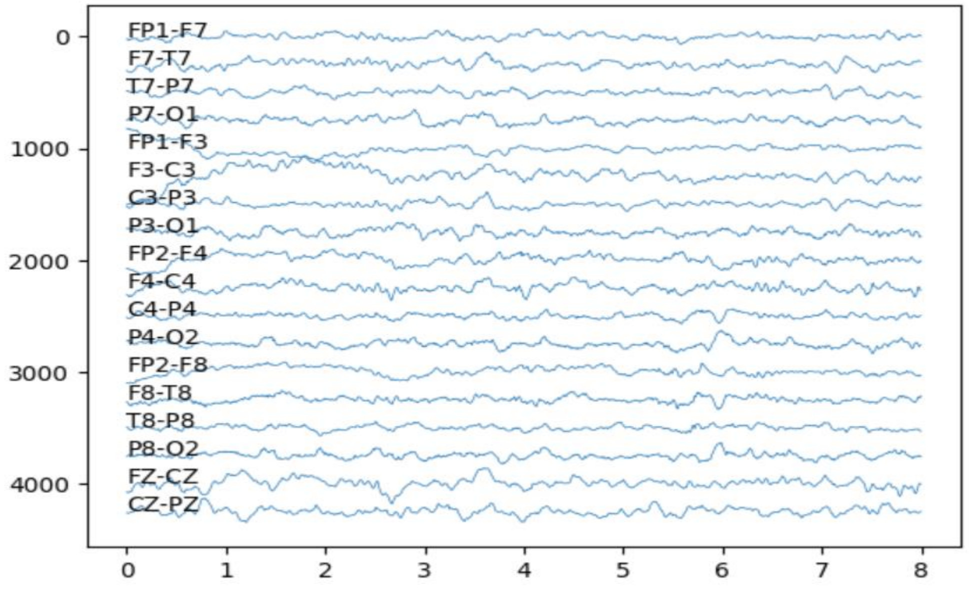
As we progress through the training files, we dynamically construct numpy arrays to store the resulting EEG signals and their corresponding labels, indicating the occurrence of seizure events. This organized representation is crucial for the subsequent stages of machine learning analysis. To provide insights into the processed data, the script concludes by printing the final shape of the resulting array. Additionally, it incorporates a downsampling step, selecting every second sample of the EEG signals, optimizing data efficiency while preserving essential information. This meticulous preprocessing phase, as detailed by the script, enhances the quality and utility of the EEG dataset for our project's machine learning endeavors.

```
Train PT:  02 03 04 05 06 09 11 12 13 14 15 16 17 18 19 20 21 23 24
Test PT:  01 07 08 10 22
```

```
100%|██████████| 116/116 [02:19<00:00,  1.20s/it]
```

```
[ ]  array_signals.shape

     (3913, 18, 2048)
```

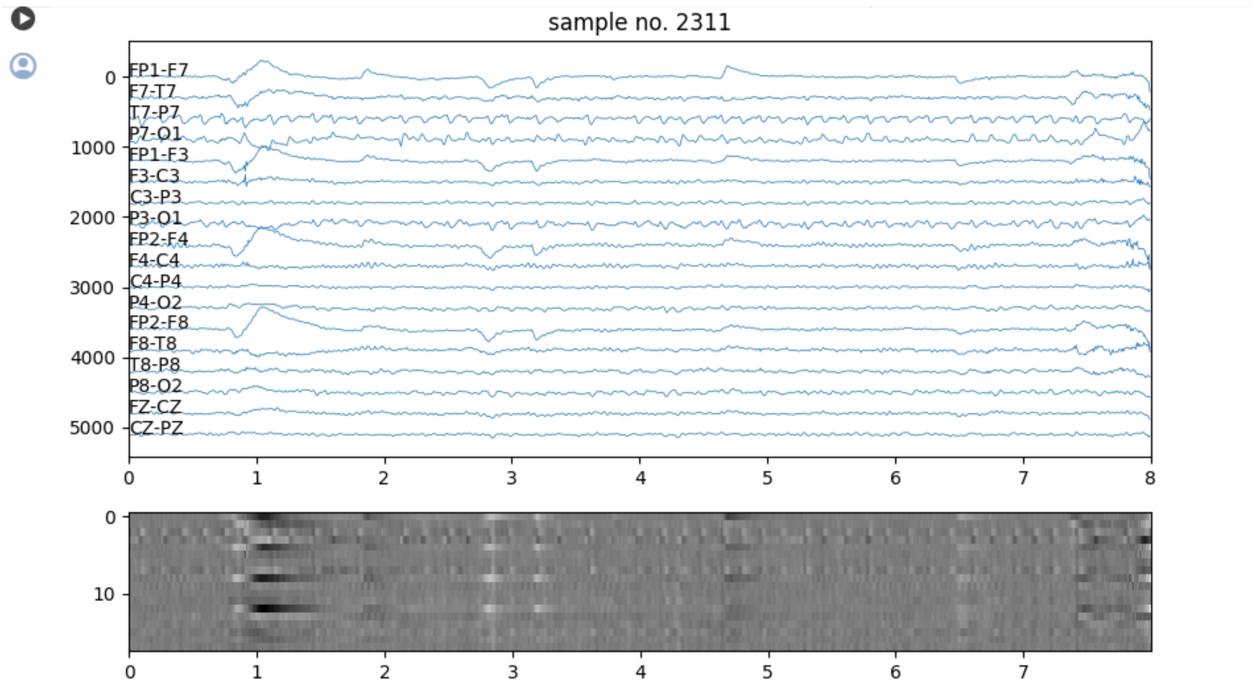
visualizing a sample of the extracted EEG signals, focusing specifically on the last set. This visualization aids in gaining insights into the nature of the recorded data. The script utilizes matplotlib to create a plot, with each channel's signal depicted in a distinct color and vertically separated for clarity. Channel labels are annotated accordingly. The inverted y-axis enhances readability, providing an intuitive representation of the EEG signals.

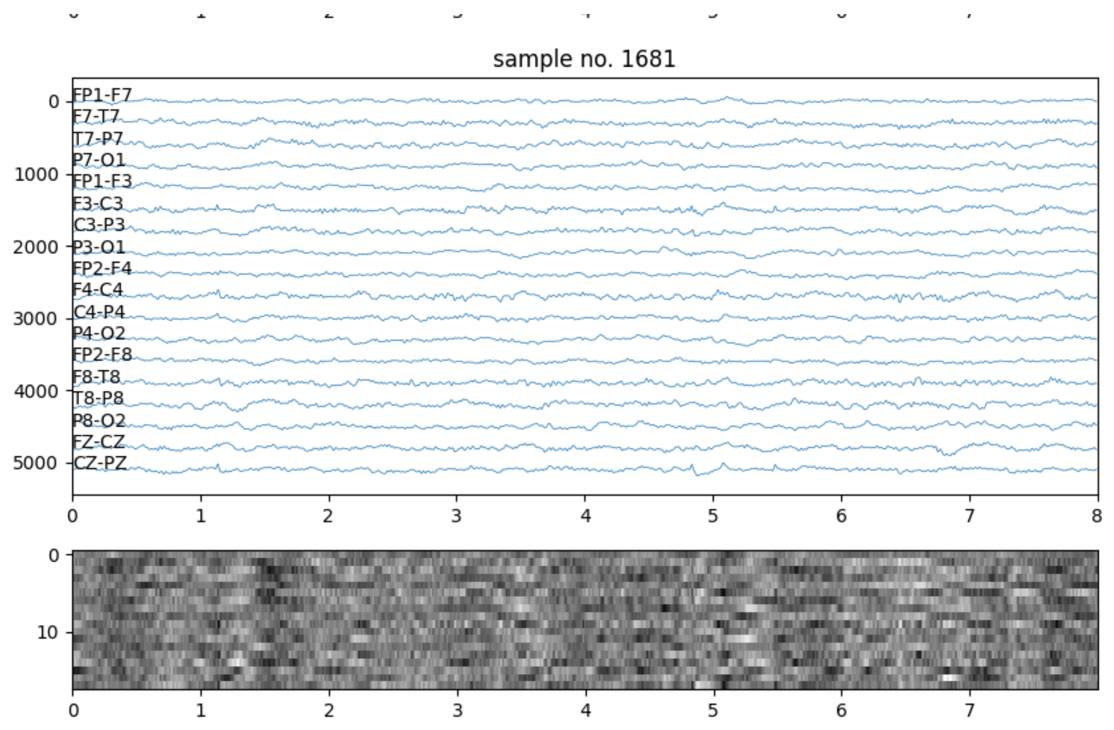


Following the visualization, the assessment of the presence of seizure events within the extracted signals has taken. By querying the array of labels indicating seizure events, the number of signals with seizures is determined, offering valuable insights into the distribution of seizure occurrences in the dataset. This quantitative analysis reveals important metrics, including the total number of signals and the ratio of signals containing seizures.

```
Number of all the extracted signals: 391
Number of signals with seizures: 2581
Ratio of signals with seizures: 0.660
```

We delve into samples featuring seizure events within the extracted EEG signals. Then visualized 10 randomly selected instances from the array of signals that indicate the presence of seizures. Each subplot displays the EEG signals from different channels, emphasizing the temporal evolution of the signals over a specified time window. The upper subplot employs individual lines for each channel, with channel labels annotated for clarity. Simultaneously, the lower subplot employs a colormap to showcase the same set of signals, offering an alternative representation that accentuates signal patterns.





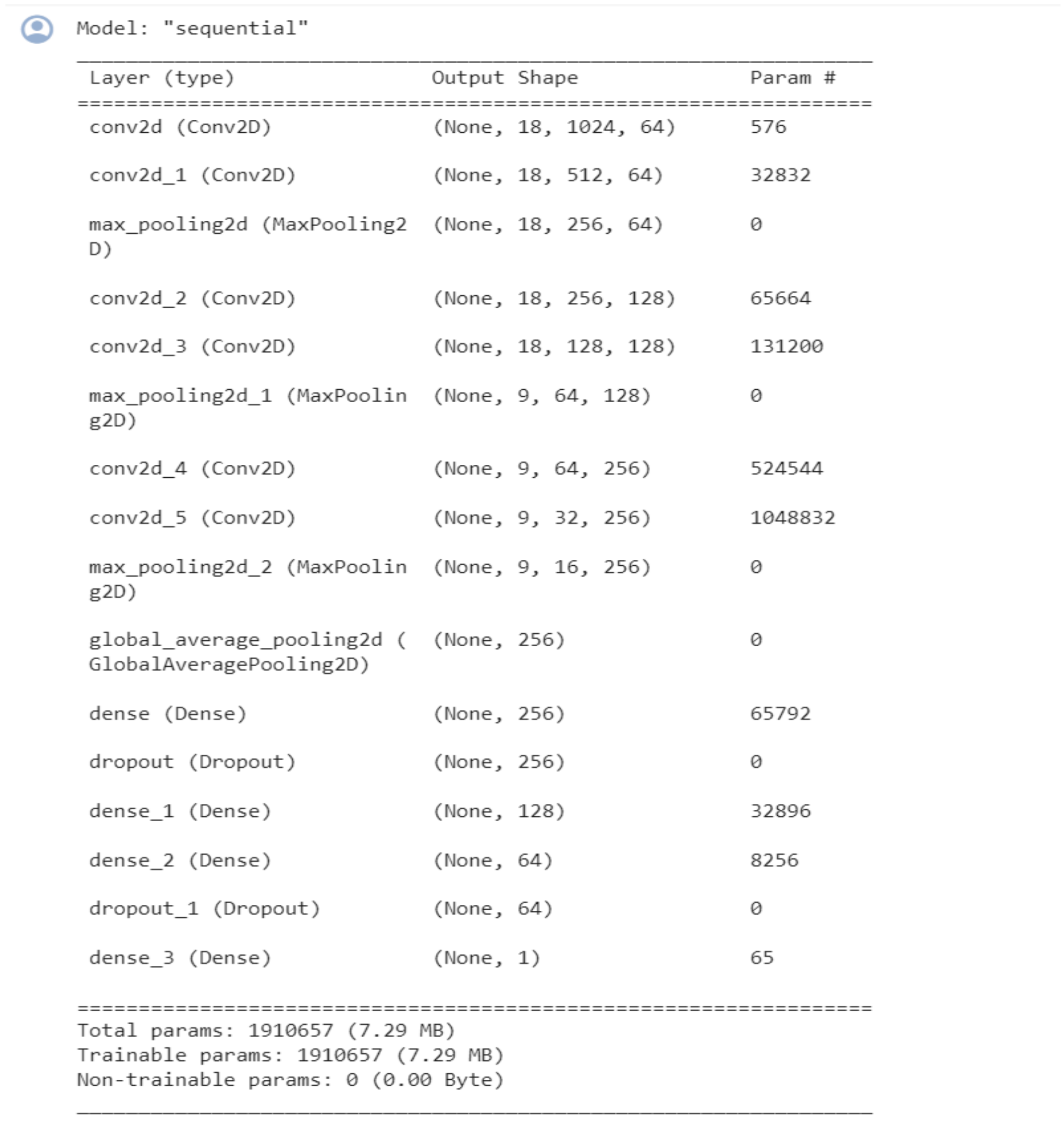
```
array_signals = array_signals[:, :, :, np.newaxis]

array_signals.shape
```

```
(3913, 18, 1024, 1)
```

Part 3: Model Architecture and Training

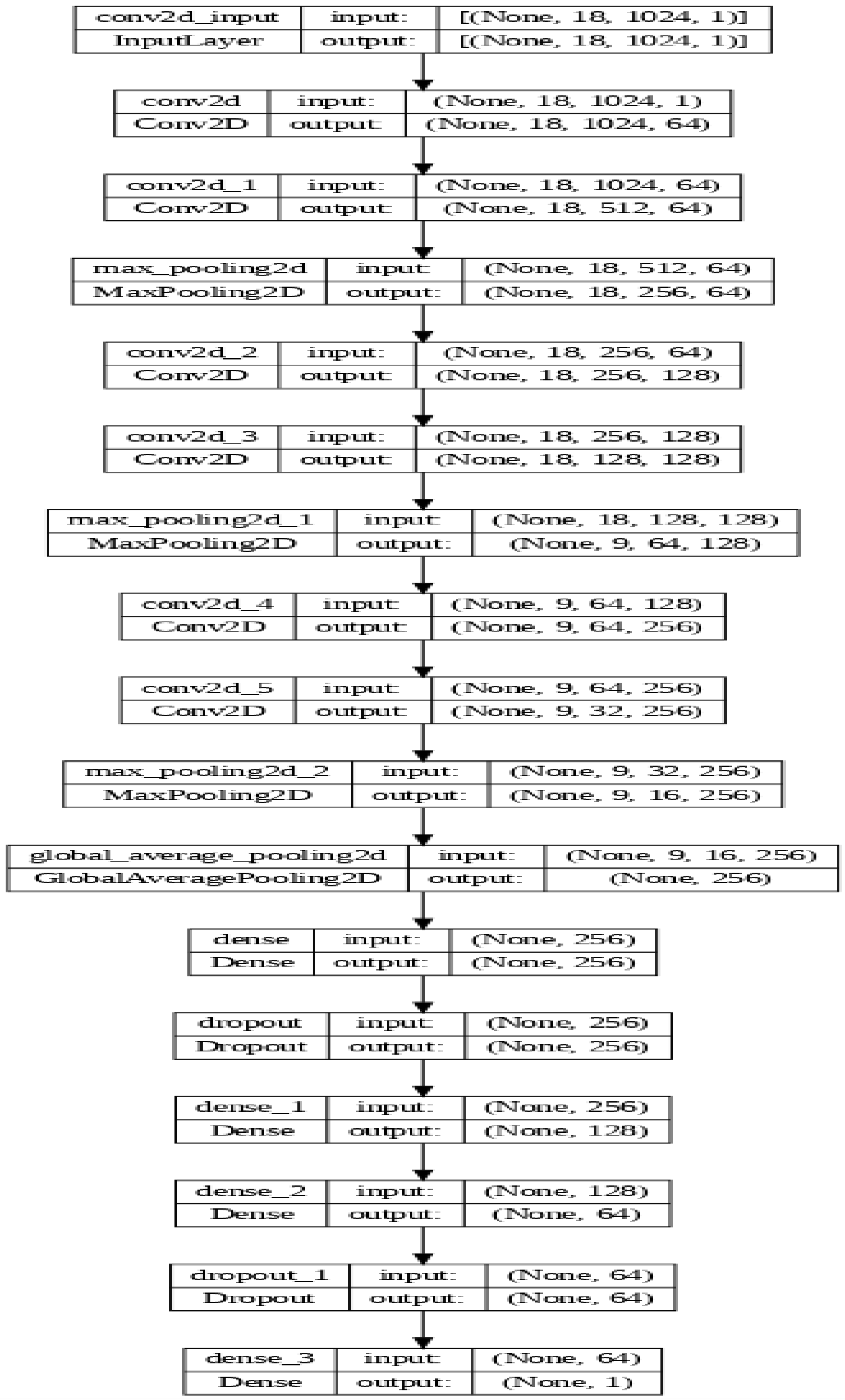
we have undertaken the design and implementation of a convolutional neural network (CNN) using the Keras library to address a binary classification task. The model is meticulously structured within a sequential framework, reflecting our deliberate approach to architectural choices. Commencing with an input layer tailored to the dimensions of our training data (X_train), the model incorporates two sets of convolutional layers, each followed by a MaxPooling layer for spatial dimension reduction. A Global Average Pooling layer is strategically introduced to capture essential features effectively.

The subsequent three fully connected (dense) layers showcase a progressive reduction in the number of neurons (256, 128, 64) to facilitate the learning of hierarchical representations. Incorporating rectified linear unit (ReLU) activation functions, these layers are designed to capture intricate patterns within the data. To combat overfitting, dropout regularization is strategically applied.

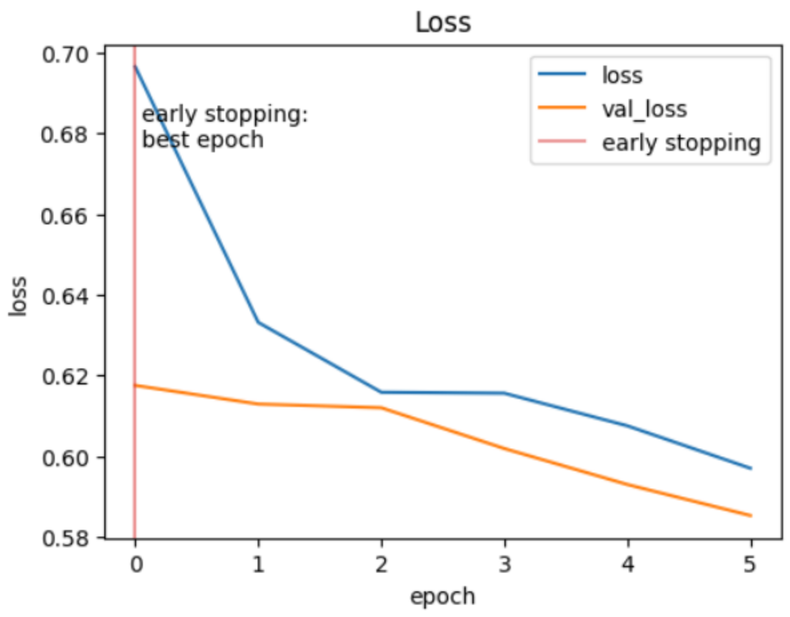


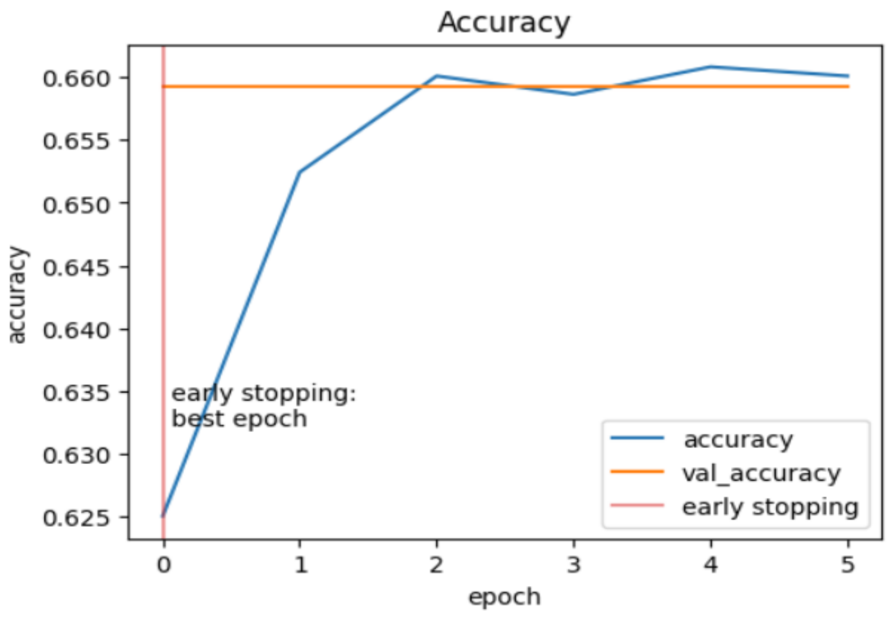
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 18, 1024, 64)      576

 conv2d_1 (Conv2D)           (None, 18, 512, 64)       32832

 max_pooling2d (MaxPooling2  (None, 18, 256, 64)       0
 D)

 conv2d_2 (Conv2D)           (None, 18, 256, 128)      65664

 conv2d_3 (Conv2D)           (None, 18, 128, 128)      131200

 max_pooling2d_1 (MaxPoolin  (None, 9, 64, 128)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 9, 64, 256)        524544

 conv2d_5 (Conv2D)           (None, 9, 32, 256)        1048832

 max_pooling2d_2 (MaxPoolin  (None, 9, 16, 256)        0
 g2D)

 global_average_pooling2d (  (None, 256)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               65792

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 1910657 (7.29 MB)
Trainable params: 1910657 (7.29 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

This layer, crucial for binary classification, consists of a single neuron utilizing a sigmoid activation function. This layer outputs probabilities, indicating the likelihood of the input belonging to the positive class. The architecture is meticulously visualized using the plot model function from Keras, offering a comprehensive overview of the network's layer-wise configuration. Our collective efforts in crafting this CNN architecture align with our project's goals of effectively capturing and learning discriminative features from EEG data for subsequent binary classification tasks.


conv2d_input | input: | [(None, 18, 1024, 1)]
InputLayer | output: | [(None, 18, 1024, 1)]
conv2d | input: | (None, 18, 1024, 1)
Conv2D | output: | (None, 18, 1024, 64)
conv2d_1 | input: | (None, 18, 1024, 64)
Conv2D | output: | (None, 18, 512, 64)
max_pooling2d | input: | (None, 18, 512, 64)
MaxPooling2D | output: | (None, 18, 256, 64)
conv2d_2 | input: | (None, 18, 256, 64)
Conv2D | output: | (None, 18, 256, 128)
conv2d_3 | input: | (None, 18, 256, 128)
Conv2D | output: | (None, 18, 128, 128)
max_pooling2d_1 | input: | (None, 18, 128, 128)
MaxPooling2D | output: | (None, 9, 64, 128)
conv2d_4 | input: | (None, 9, 64, 128)
Conv2D | output: | (None, 9, 64, 256)
conv2d_5 | input: | (None, 9, 64, 256)
Conv2D | output: | (None, 9, 32, 256)
max_pooling2d_2 | input: | (None, 9, 32, 256)
MaxPooling2D | output: | (None, 9, 16, 256)
global_average_pooling2d | input: | (None, 9, 16, 256)
GlobalAveragePooling2D | output: | (None, 256)
dense | input: | (None, 256)
Dense | output: | (None, 256)
dropout | input: | (None, 256)
Dropout | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 64)
dropout_1 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_3 | input: | (None, 64)
Dense | output: | (None, 1)

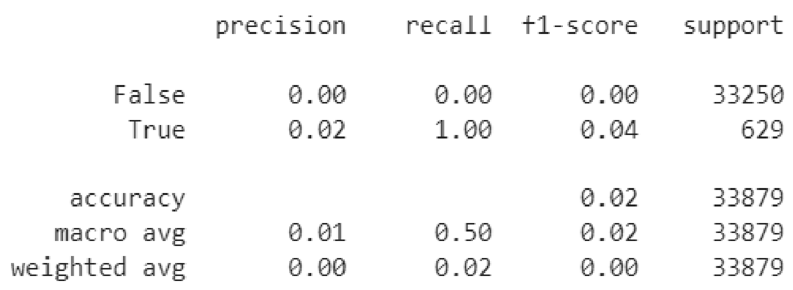
we have presented a comprehensive view of the training progress for our machine learning model. The subplot comprises two axes: the left one illustrates the evolution of training and validation loss over epochs, while the right one depicts the corresponding accuracy metrics. The lines on each plot represent training and validation metrics, offering insights into the model's convergence and generalization capabilities. Additionally, a vertical line on both plots signifies the epoch at which early stopping was triggered, marked in red for emphasis. These visualizations serve as invaluable tools for understanding the model's performance trajectory, identifying potential overfitting, and appreciating the efficacy of the early stopping mechanism. Our commitment to detailed analysis and transparent visualization is evident in this portrayal of the training dynamics within our machine learning project.



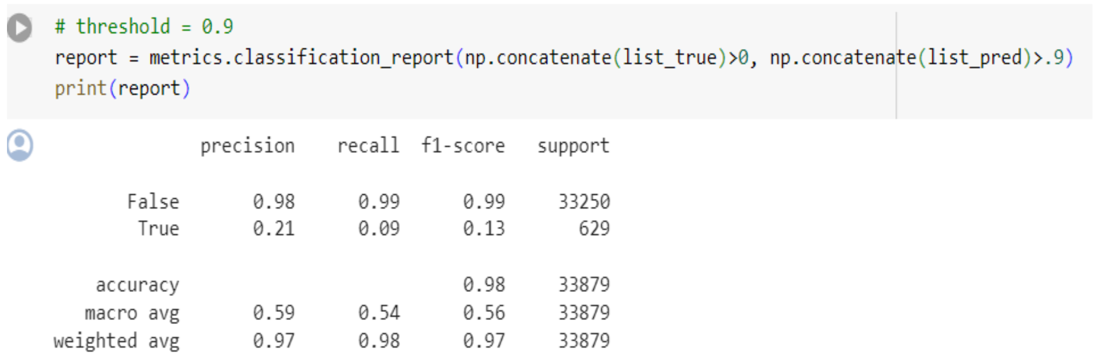


Part 4: Evaluation Results

Evaluation of the model by scores and ROC:



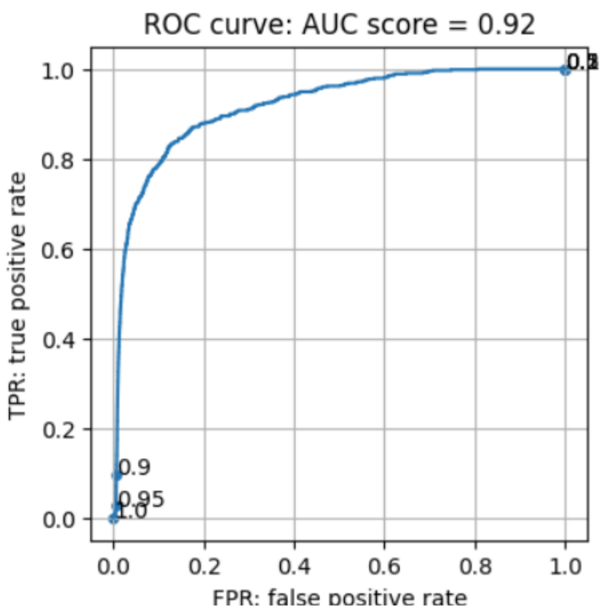
```
              precision    recall  f1-score   support

       False       0.00      0.00      0.00     33250
        True       0.02      1.00      0.04       629

    accuracy                           0.02     33879
   macro avg       0.01      0.50      0.02     33879
weighted avg       0.00      0.02      0.00     33879
```



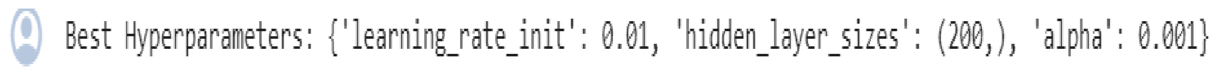
```
# threshold = 0.9
report = metrics.classification_report(np.concatenate(list_true)>0, np.concatenate(list_pred)>.9)
print(report)
```

```
              precision    recall  f1-score   support

       False       0.98      0.99      0.99     33250
        True       0.21      0.09      0.13       629

    accuracy                           0.98     33879
   macro avg       0.59      0.54      0.56     33879
weighted avg       0.97      0.98      0.97     33879
```

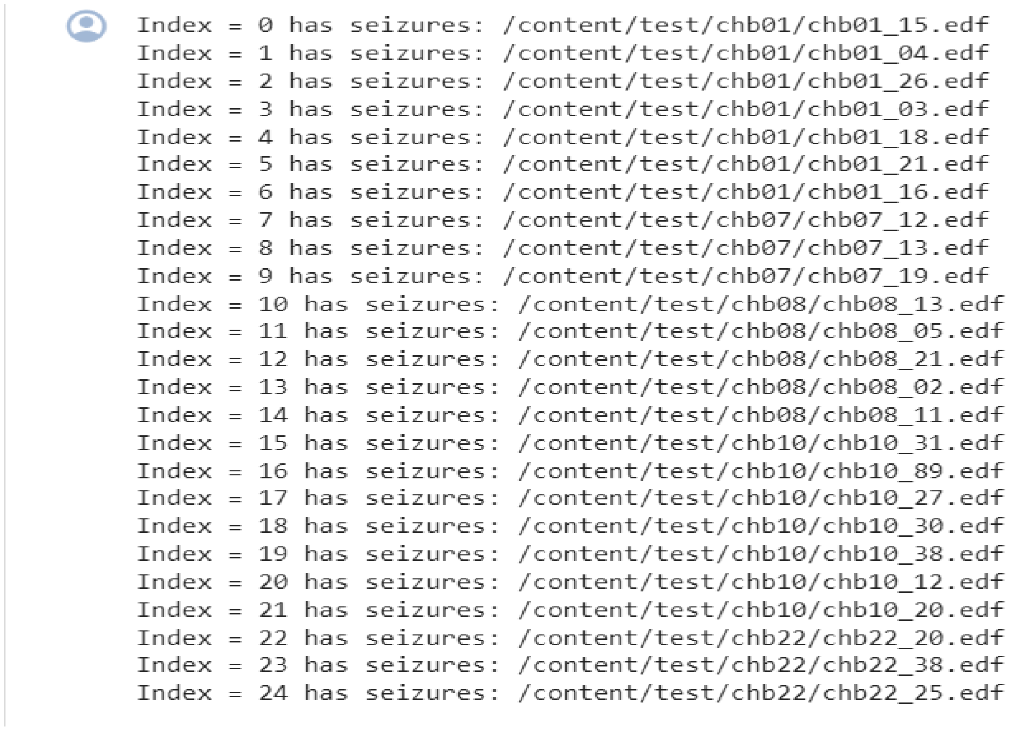
Then we calculate and visualize the Receiver Operating Characteristic (ROC) curve for a binary classification model. It utilizes the scikit-learn library to compute the ROC curve and the Area Under the Curve (AUC) score. The resulting plot illustrates the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity). Additionally, key threshold points (.1, .2, .5, .9, .95, 1) on the ROC curve are highlighted and labeled for reference.



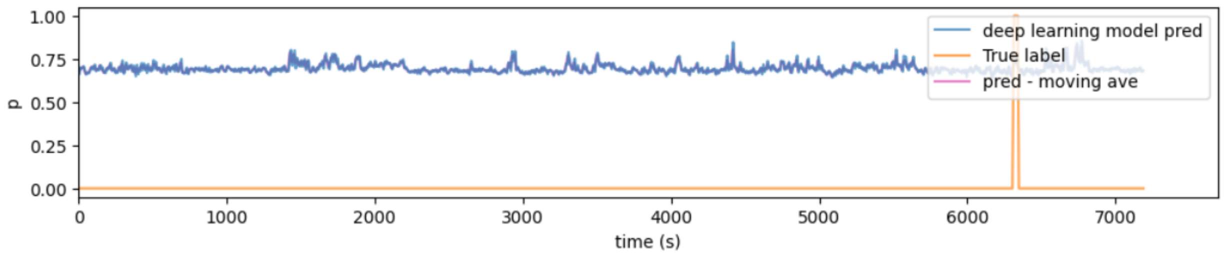
we optimize hyperparameters for a Multi-layer Perceptron (MLP) classifier using scikit-learn's RandomizedSearchCV. We define a function to create the MLP model with adjustable hyperparameters. The search space includes variations in hidden layer sizes, regularization strength (alpha), and initial learning rate. A random search is performed with 10 iterations, 3-fold cross-validation, and the best hyperparameters are printed. The input data is reshaped, and the model is fitted to the training data, exemplifying our effort to fine-tune the MLP model for optimal performance.


```
Best Hyperparameters: {'learning_rate_init': 0.01, 'hidden_layer_sizes': (200,), 'alpha': 0.001}
```

The deep learning model implemented ,focuses on predicting the existence of seizures, effectively detecting the presence of seizure events within the EEG data. However, it does not explicitly indicate the precise onset of seizures. To identify seizure onsets, a peak detection algorithm is applied to the amplitude peaks denoted as 'p.' The algorithm considers peaks with a minimum distance greater than 6 data points and a threshold value equal to or exceeding 0.95. This additional step using peak detection complements the model's seizure detection capabilities by pinpointing the onset moments within the EEG data, offering more detailed insights into the temporal dynamics of seizure events.

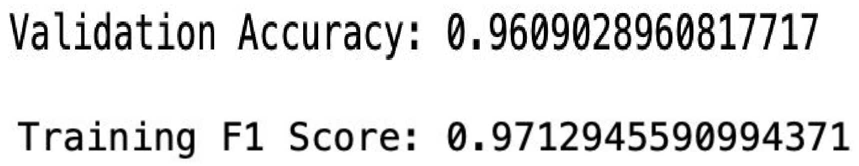


```
Index = 0 has seizures: /content/test/chb01/chb01_15.edf
Index = 1 has seizures: /content/test/chb01/chb01_04.edf
Index = 2 has seizures: /content/test/chb01/chb01_26.edf
Index = 3 has seizures: /content/test/chb01/chb01_03.edf
Index = 4 has seizures: /content/test/chb01/chb01_18.edf
Index = 5 has seizures: /content/test/chb01/chb01_21.edf
Index = 6 has seizures: /content/test/chb01/chb01_16.edf
Index = 7 has seizures: /content/test/chb07/chb07_12.edf
Index = 8 has seizures: /content/test/chb07/chb07_13.edf
Index = 9 has seizures: /content/test/chb07/chb07_19.edf
Index = 10 has seizures: /content/test/chb08/chb08_13.edf
Index = 11 has seizures: /content/test/chb08/chb08_05.edf
Index = 12 has seizures: /content/test/chb08/chb08_21.edf
Index = 13 has seizures: /content/test/chb08/chb08_02.edf
Index = 14 has seizures: /content/test/chb08/chb08_11.edf
Index = 15 has seizures: /content/test/chb10/chb10_31.edf
Index = 16 has seizures: /content/test/chb10/chb10_89.edf
Index = 17 has seizures: /content/test/chb10/chb10_27.edf
Index = 18 has seizures: /content/test/chb10/chb10_30.edf
Index = 19 has seizures: /content/test/chb10/chb10_38.edf
Index = 20 has seizures: /content/test/chb10/chb10_12.edf
Index = 21 has seizures: /content/test/chb10/chb10_20.edf
Index = 22 has seizures: /content/test/chb22/chb22_20.edf
Index = 23 has seizures: /content/test/chb22/chb22_38.edf
Index = 24 has seizures: /content/test/chb22/chb22_25.edf
```



we utilized scikit-learn to implement an MLP classifier designed for the prediction of seizures. Following the reshaping of training and validation data, we instantiated an MLPClassifier, integrating carefully chosen hyperparameters derived from a prior hyperparameter search—specifically {'hidden_layer_sizes': (100,), 'learning_rate_init': 0.001, 'alpha': 0.0001}. This parameter configuration aimed to optimize the model's predictive performance. Subsequently, the model underwent training on the reshaped training data, limited to a maximum of 15 iterations.

After training, we applied the model to generate predictions on the validation set and gauged its accuracy using the accuracy_score metric. The resulting validation accuracy provides a quantitative measure of the model's proficiency in making accurate predictions on previously unseen data—a pivotal aspect for real-world deployment. Moreover, we conducted a comprehensive evaluation of the model's performance on the training set, predicting labels and calculating the F1 score. The F1 score, considering precision and recall, serves as an additional metric, shedding light on the model's capacity to strike a balance between false positives and false negatives. This meticulous evaluation not only affirms the model's robustness and effectiveness in capturing patterns within the training data but also establishes a foundation for ongoing refinement and enhancement of our MLP classifier for optimal seizure prediction capabilities.



```
Validation Accuracy: 0.9609028960817717

Training F1 Score: 0.9712945590994371
```

Part 5: Conclusion and future work

This project focused exclusively on utilizing the CHB-MIT EEG Database for EEG classification, specifically in the context of epilepsy diagnosis. Employing Convolutional Neural Networks (CNNs), a robust deep learning architecture, we addressed the intricate task of discerning between different seizure types and non-seizure patterns within EEG signals. The project's initiation involved comprehensive data preprocessing, encompassing the download, extraction, and exploration of the CHB-MIT EEG Database. Essential feature extraction techniques, emphasizing both time-domain and frequency-domain features, were strategically employed to capture distinctive patterns within the EEG signals. The dataset was meticulously divided into training, validation, and test sets, establishing a solid foundation for subsequent model development and evaluation.

The CNN model underwent rigorous training, incorporating effective strategies to mitigate overfitting, including dropout and early stopping. Evaluation metrics such as accuracy, precision, recall, and F1-score were systematically applied to assess the model's performance on the validation set. Hyperparameter fine-tuning further optimized the model's efficacy, ensuring a well-generalized and robust predictive capability. The model's resilience was demonstrated in its successful testing on an independent test set, affirming its potential utility in real-world scenarios. Visualizations of EEG data and model predictions offered valuable insights into the model's performance, fostering a deeper understanding of its capabilities and limitations.

Future Work :

Multimodal Integration: Explore the incorporation of supplementary modalities, including imaging or clinical data, to augment the precision of classification.

Transfer Learning: Examine the viability of employing transfer learning methodologies, leveraging pre-existing models trained on extensive EEG datasets to enhance overall performance.

Patient-Specific Models: Construct customized models that consider unique patient characteristics, facilitating more precise and personalized predictions.