Seizure Prediction with Deep Learning and Machine Learning on EEG Data

EEG Classification Model

Table of Contents

Introduction

Seizures, temporary deviations in the brain's electrical activity, are a hallmark of epilepsy. Individuals with epilepsy experience recurrent, unpredictable seizures, which can manifest as a brief loss of attention or full-body convulsions. These seizures pose risks of physical injuries and, in severe cases, may lead to death. Swift detection and response to seizures through devices or interventions can significantly mitigate associated challenges.

The dataset comprises EEG recordings from pediatric subjects, with 23 cases grouped into 22 subjects (5 males, ages 3–22; 17 females, ages 1.5–19). Subjects were monitored for several days post-medication withdrawal to evaluate seizures and surgical candidacy. The data was collected at the Children's Hospital Boston, with subjects monitored following the withdrawal of anti-seizure medication to characterize their seizures and assess the possibility of surgical intervention.

Recordings include 9 to 42 continuous .edf files per case, with gaps due to hardware limitations. Privacy is maintained by replacing protected health information (PHI) with surrogate data. The files, sampled at 256 samples per second, contain EEG signals using the International 10-20 system. Some records include additional signals, such as ECG or vagal nerve stimulus.

The dataset includes 664 .edf files, with 129 containing seizures. Information on subjects' gender and age is available. A RECORDS file lists all files, and a RECORDS-WITH-SEIZURES file details those with seizures. Annotation files indicate seizure start and end times. Usage notes highlight the dataset's potential for developing seizure onset detection algorithms.

The primary goal is to create a machine learning or deep learning model that can effectively analyze patterns within EEG images to predict the onset of seizures. The model should provide sufficient lead time for medical professionals or patients to take preventive actions, optimizing seizure management strategies. The practical application involves real-time monitoring of EEG images, allowing the model to forecast an incoming seizure. This use case is particularly valuable for individuals with epilepsy, as it enables timely medical intervention, potentially preventing or minimizing the impact of seizures and enhancing overall quality of life.

The project involves preprocessing and feature extraction from EEG images, followed by the development of a predictive model. Both machine learning algorithms (e.g., SVM, RF, GBM) and deep learning techniques (specifically CNNs) are explored and compared. Extensive parameter tuning and evaluation using relevant metrics is conducted to ensure the model's reliability and accuracy.

Data Preprocessing & Feature Extraction

Preprocessing for ML

Label Encoding: Categorical labels were converted into numerical format through label encoding. This process ensures that machine learning algorithms can appropriately interpret and utilize categorical information. Each unique label was assigned a unique numerical identifier, facilitating model training.

Data Normalization: Feature columns underwent normalization to ensure consistent scales across variables. This process involved transforming numerical values to a standard scale, typically between 0 and 1. Normalization prevents certain features from dominating the learning process due to their larger scale, promoting fair influence from all features during model training.

Data Splitting: The dataset was divided into training and testing sets to assess model performance. The common 80-20 split was employed, where 80% of the data was allocated for training and 20% for testing. This separation allows for model training on a substantial portion of the data while evaluating its generalization on unseen instances.

Preprocessing for CNN

Our data preprocessing journey commenced with individual text files containing Electroencephalogram (EEG) recordings for each class, denoted as Z, S, F, N, and O. We systematically transformed this raw data into a format suitable for input into machine learning models:

- **Data Extraction:** Extracted numerical values from the EEG recordings, capturing the essence of brain activity patterns.
- **Label Attachment:** Introduced class labels to each data point, creating a list of lists where each inner list represents an individual data point along with its corresponding class label.
- **PyTorch Tensor Conversion:** Converted the data to PyTorch tensors, aligning our data with the chosen deep learning framework and enabling seamless integration into the neural network architecture.

To efficiently manage and preprocess the data during model training and evaluation, we implemented a custom **EEGDataset** class with the following key functionalities:

- **Batching:** Facilitated batch processing by organizing the data into batches, a crucial optimization for accelerated training and reduced memory overhead.
- **Data Splitting:** Divided the dataset into distinct subsets for training, validation, and testing, establishing a robust foundation for unbiased model evaluation.
- **Data Type Conversion:** Converted EEG values to the float type, aligning the data with the numerical requirements of machine learning algorithms.
- **GPU Acceleration:** Leveraged available GPU resources to expedite training and computation, enhancing the overall efficiency of the machine learning pipeline.

Normalization of EEG values played a pivotal role in preparing the data for model ingestion:

- **Mean and Standard Deviation Calculation:** Computed the mean and standard deviation of each column to facilitate standardization, a crucial step in enhancing the model's ability to discern relevant patterns.
- **Within-Column Normalization:** Ensured uniform treatment of all features by normalizing EEG values within each column, mitigating potential biases introduced by varying scales.

This comprehensive data preprocessing pipeline lays the groundwork for a robust and well-structured machine learning workflow, setting the stage for effective model training and subsequent performance evaluation.

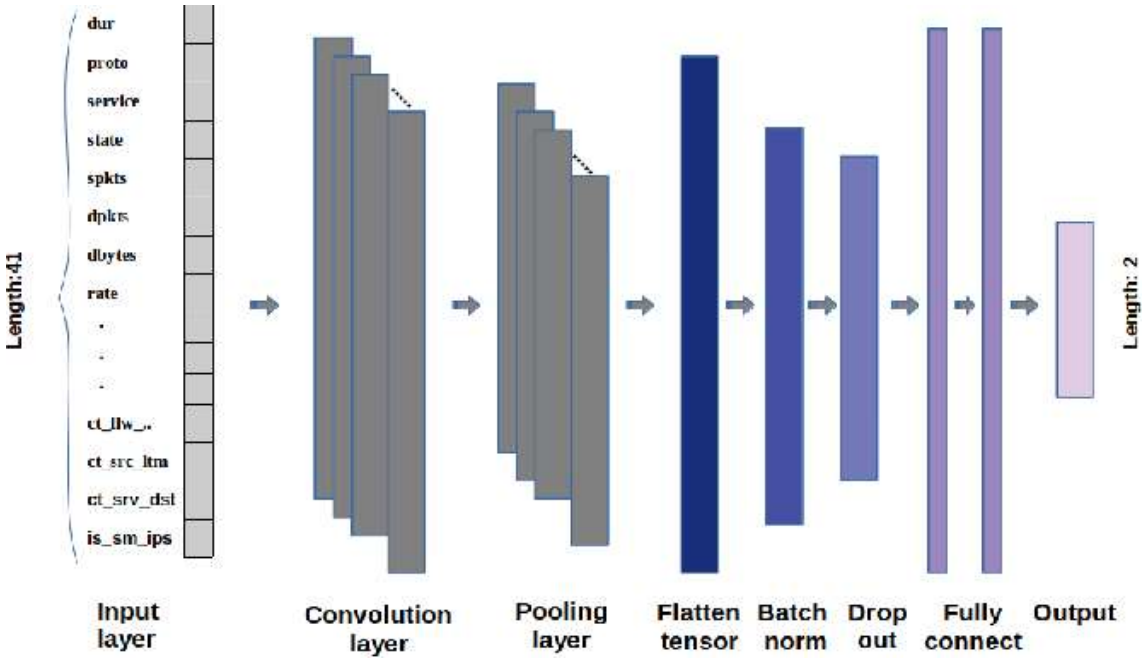
Architecture of Models

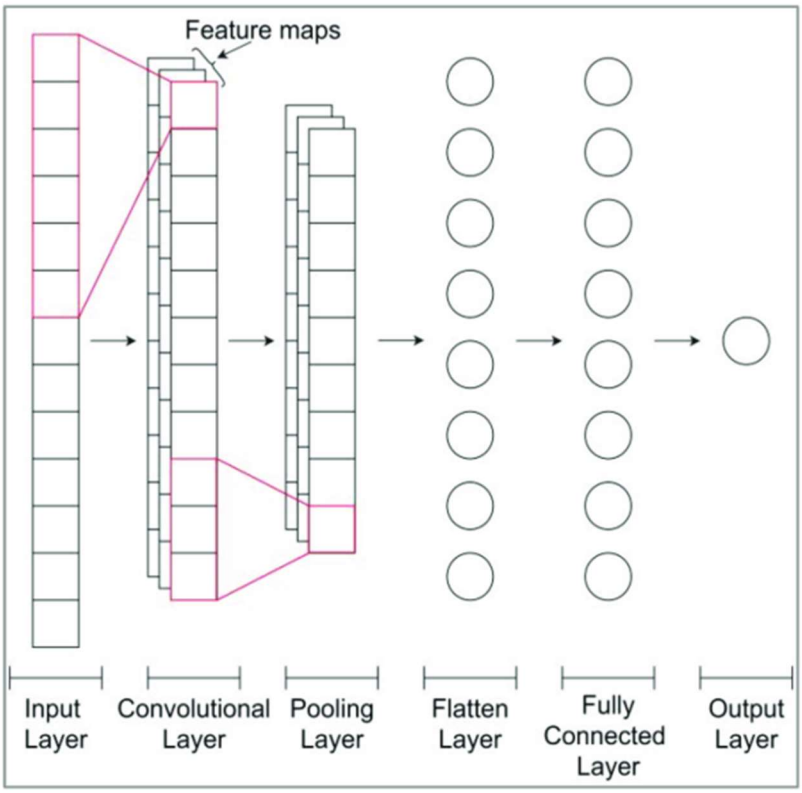
We have used two separate models for our task: SVM, Random Forest and Gradient Boosted Model for machine learning, and Convolutional Neural Network for deep learning.

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. It finds the optimal hyperplane that separates data into classes, maximizing the margin between them.

Random Forest is an ensemble learning method that constructs multiple decision trees in parallel during training and outputs the mode of the classes for classification or the average prediction for regression.

Gradient Boosted Model is made using an ensemble technique where weak decision trees are sequentially added to correct errors of the previous models. It combines their predictions to improve overall performance in both regression and classification tasks.





On the other hand, our deep learning approach involved the construction of a hierarchical Convolutional Neural Network (CNN) comprising three distinct convolutional blocks, each followed by two fully connected layers. This sophisticated architecture leverages a combination of convolutional operations, activation functions, batch normalization, and dropout layers to enhance its overall performance.

Architecture

We constructed a hierarchical CNN model with the following convolutional blocks:

1. First Convolution Block:
    - Convolutional Layer 1:
        - Kernel Size: 16
        - Stride: 4
        - Number of Channels: 4
    - Batch Normalization
    - Dropout with Probability: 0.5

2. Second Convolution Block:
    - Convolutional Layer 2:
        - Kernel Size: 8
        - Stride: 2
        - Number of Channels: 2
    - Batch Normalization
    - Dropout with Probability: 0.5

3. Third Convolution Block:
    - Convolutional Layer 3:
        - Kernel Size: 4
        - Stride: 1
        - Number of Channels: 4
    - Batch Normalization
    - Dropout with Probability: 0.5

These convolutional blocks are designed to extract hierarchical features from the input data, facilitating the model's ability to discern intricate patterns and representations within the dataset.

The CNN architecture is further complemented by two fully connected layers, providing a seamless transition from the convolutional layers to the final output. These linear layers contribute to the extraction of high-level features and facilitate the model's ability to make complex decisions based on the learned representations.

The integration of Rectified Linear Unit (ReLU) activation functions, batch normalization, and dropout layers in each convolutional block serves to enhance the model's non-linearities, improve training convergence, and mitigate overfitting.

Training details

The dataset was divided into three subsets: a training set for model parameter learning, a validation set for fine-tuning and hyperparameter optimization, and a test set for evaluating the model's ultimate performance.

Optimization Techniques: During training, we employed the Adam optimizer and the Cross Entropy loss criterion. The Adam optimizer, known for its efficiency in handling sparse gradients, played a crucial role in adjusting the model's parameters to minimize the loss function, thereby enhancing predictive accuracy.

Hyperparameter Tuning: We explored various channel configurations for the convolutional layers, experimenting with different combinations to identify the configuration that yielded the highest accuracy on the validation set.

By systematically training models with diverse configurations, we sought to strike a balance between model complexity and generalization. The identified optimal configuration not only reflects the nuances of the dataset but also ensures the model's adaptability to a broader range of scenarios.

Results & Discussion

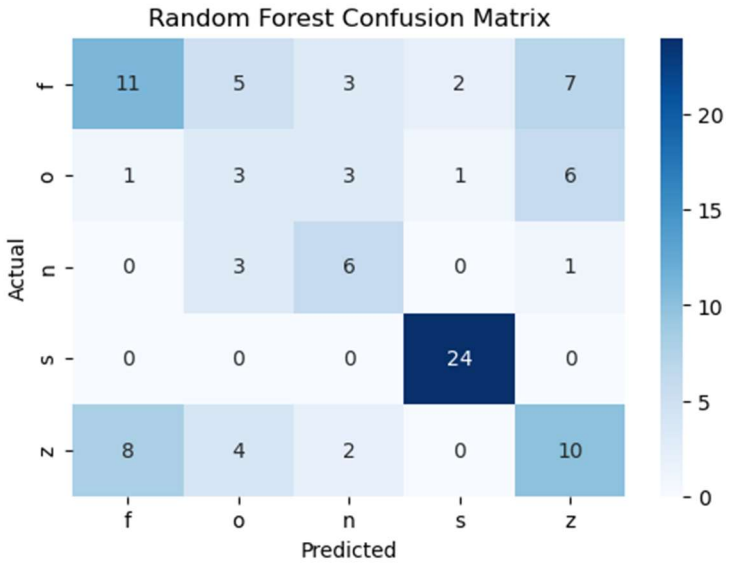
Our model evaluation process was meticulously designed to gauge the performance of various models, considering different hyperparameters, and identify the optimal configuration for the task at hand:

- **Loading Best-Performing Model:** We loaded the model that exhibited the highest test accuracy during the initial evaluation phase.
- **Hyperparameter Exploration:** We evaluated several models with varying hyperparameters on the test dataset. And we systematically examined different configurations to understand their impact on model performance.
- **Identification of Optimal Model:** We identified the model that achieved the highest number of correct predictions on the test dataset. And we selected the optimal configuration for subsequent analysis and reporting.

For visualization, we generated a confusion matrix using the predicted and actual labels from the test dataset. Utilized the Seaborn library to create a visually informative representation of the confusion matrix. Then we a detailed classification report, encompassing precision, recall, and F1-score metrics. Calculated metrics for each class, offering insights into the model's performance on individual categories.

ML Results

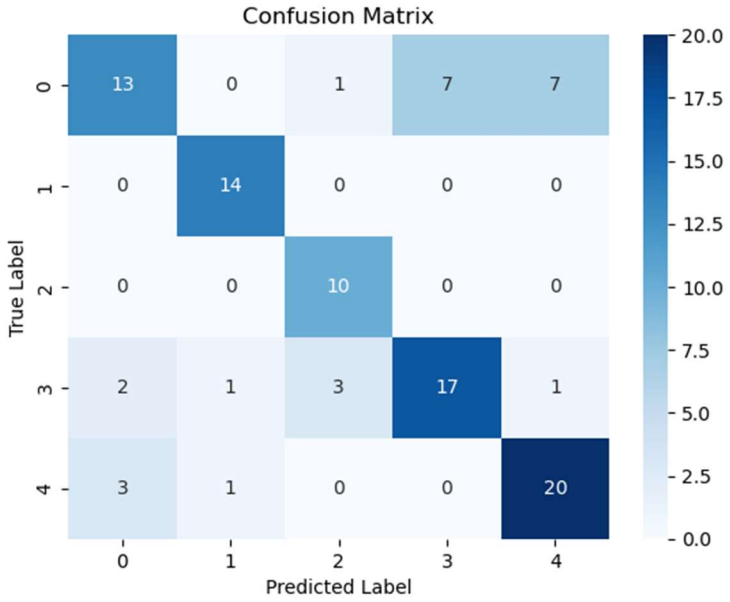
- **Confusion Matrix:**



- **Classification Report:**
  - Accuracy: Gradient Boosting exhibited the highest accuracy among the models, showcasing its ability to make correct predictions.
  - Precision: SVM demonstrated the highest precision, indicating its efficiency in accurately identifying positive instances.
  - Recall (Sensitivity): Gradient Boosting and Random Forest models tied for the highest recall, showcasing their strength in correctly identifying actual positive cases.
  - F1 Score: The balancing precision and recall, was highest for Gradient Boosting, underlining its overall effectiveness.

CNN Results

- **Confusion Matrix:**



- **Classification Report:**
  - The best-performing model was characterized by 4, 2, and 4 channels in the three convolutional layers.
  - Achieved an impressive overall test accuracy of 74%, signifying its competence in classifying EEG recordings.

The confusion matrix and classification report supplied in-depth insights into the model's strengths and areas for improvement. These visualizations and metrics contribute to a comprehensive understanding of the model's performance on a per-class basis.

Conclusion

The comprehensive evaluation of four distinct models—Support Vector Machine (SVM), Random Forest (RF), Gradient Boosted Model (GBM), and Convolutional Neural Network (CNN)—revealed insightful findings. Each model underwent rigorous parameter tuning to optimize performance, and the assessment was facilitated by the use of confusion matrices.

While all models demonstrated improved performance post-tuning, the CNN emerged as the standout performer with the highest overall efficacy. This outcome aligns with expectations, considering the CNN model's inherent capability to capture intricate patterns and hidden information within data, particularly in image-related tasks.

The significance of parameter tuning underscored the importance of meticulous model configuration for achieving optimal results across diverse algorithms. The utilization of confusion matrices provided a nuanced understanding of model performance, elucidating not only accuracy but also the intricacies of true positives, false positives, true negatives, and false negatives.

In summary, the CNN model's superior performance reinforces the advantage of deep learning in tasks where intricate features are paramount. However, the tuning process for all models serves as a reminder of the critical role parameter optimization plays in unlocking the full potential of machine learning and deep learning algorithms. The choice of the most suitable model ultimately depends on the specific characteristics of the data and the nature of the task at hand.