Foundations Data Analytics

Project 3

Prof. Sivarit (Tony) Sultornsanee

Team Members

Saiteja Reddy Gajula (002872000)

Pooja Arumugam (002872003)

Bhakti Paithankar (002833722)

Nihal Mallikarjun (0036010381)

Sathvik Ramappa (002847460)

Index

Serial No.	Topic
1	Introduction
2	Strategy
3	Analysis
4	Results
5	Conclusion
6	Limitation
7	Future Work

Introduction

With an emphasis on diagnosing epilepsy, the project intends to develop a classification model for Electroencephalogram (EEG) data, a crucial tool in the domains of neuroscience and medicine. The CHB-MIT EEG Database and the Bonn EEG Dataset are the two datasets used to train and assess the model. Data pre-processing, feature extraction, data splitting, model selection, training, assessment, testing, and visualization of the outcomes are all part of the project.

The electrical activity of the brain briefly deviates during seizures. Epilepsy is a condition of the central nervous system that causes recurring seizures that usually happen without warning and at random periods. A complete loss of consciousness or a convulsion can be the outcome of a seizure. An individual who experiences seizures frequently runs a higher chance of suffering physical harm and possibly even passing away. Epilepsy sufferers may find it easier to cope with seizures if a gadget could promptly identify and respond to a seizure by either sending therapy or alerting a caregiver.

A scalp electroencephalogram (scalp EEG), a non-invasive multi-channel recording of the electrical activity in the brain, is most commonly used to infer the start of a seizure before it becomes clinically present. Patients differ greatly in the features of their EEG. Indeed, the EEG pattern linked to the development of seizures in one patient could be quite similar to a benign pattern in another patient's EEG. Because of this cross-patient variability in seizure and non-seizure activity, patient non-specific classifiers report seizure onsets with poor accuracy or with prolonged latency.

Strategy

The goal is to build patient-specific detectors that can promptly and accurately identify the onset of seizures through the use of machine learning. Seizure onset detection particular to a patient is still difficult to achieve because

• There is a significant overlap between the EEGs of patients with epilepsy who are experiencing seizures and those who are not. The balance between detector sensitivity and specificity becomes crucial for algorithm designers.

• The electroencephalogram (EEG) of individuals with epilepsy is a nonstationary process since it is continuously changing regimes throughout both seizure and non-seizures. Finding the fundamental condition of the brain may depend on understanding the short-term evolution of EEG activity.
• A detector must determine that the brain has entered a seizure state based on a small number of samples from that condition in order to be used in a number of medical applications where seizure onsets must be identified fast. This makes the severe trade-off between detector latency and specificity that algorithm builders must now make.

The scarcity of seizure training data forces algorithm designers to create techniques that make sense given the rarity of seizures. We might take into consideration applying an online, unsupervised time-series segmentation method since the aim of seizure detection is to divide the brain's electrical activity in real-time into seizure and non-seizure periods. Unfortunately, these algorithms return many segmentations beyond those that demarcate seizure and non-seizure periods due to the various regimes of seizure and non-seizure EEG.The relevant signal regimes and transitions must be taught to a seizure detector. Because of this, we decide to use a supervised, discriminative framework to solve the seizure detection problem. Even though the physiological activity that underlies the discriminative framework is multiclass, we decide to tackle a binary classification problem within it. We take this approach because it is not simple or feasible for a specialist to recognize and categorize the subclasses of the seizure and non-seizure states. As an expert, however, classifying a recording of the brain's electrical activity into two inclusive categories—seizures and non-seizures—is in line with accepted therapeutic practice.

Analysis

Data Preprocessing:

The CHB-MIT dataset is downloaded and extracted. The datasets contain EEG recordings from patients with epilepsy, encompassing various seizure types and non-seizure data. A Python script reads and explores the EEG data structure using the pyedflib library. This step helps in understanding the data's characteristics and preparing for subsequent preprocessing steps.

Resampling EEG signals to a common frequency of 100,000 Hz. Extracting the maximum value of each signal and normalizing it. Organizing the data into a Pandas DataFrame, separating signals and labels.

Feature Extraction:

Feature extraction involves capturing relevant information from EEG signals. Time-domain and frequency-domain features are considered. The code focuses on extracting 100,000 time-domain samples.

Data Splitting

The preprocessed data is split into training, validation, and test sets. This ensures that the model is trained on one subset, validated on another, and tested on a completely independent set.

Model Selection

For EEG classification, a Long Short-Term Memory (LSTM) model is chosen. LSTMs are particularly effective in capturing temporal dependencies in sequential data, making them suitable for EEG signals.

The Long Short-Term Memory (LSTM) model used in the provided Python script is a type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem associated with traditional RNNs. LSTMs are well-suited for tasks involving sequential data, making them applicable to time-series analysis, natural language processing, and, in this case, EEG signal classification.

```
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compiling and training the CNN model
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model.fit(X_train_cnn, y_train, epochs=10, batch_size=32, validation_data=(X_val_cnn, y_val))

# Predicting on the test set
y_pred_cnn_probs = cnn_model.predict(X_test_cnn)
y_pred_cnn = (y_pred_cnn_probs > 0.5).astype(int)

# Evaluating the CNN model
accuracy_cnn = accuracy_score(y_test, y_pred_cnn)
print(f"Accuracy of CNN model: {accuracy_cnn}")

# Defining a simple LSTM model
lstm_model = Sequential([
    LSTM(64, input_shape=(X_train.shape[1], 1)),
    Dense(1, activation='sigmoid')
])

# Reshaping data for compatibility with LSTM
X_train_lstm = X_train[:, :, np.newaxis]
X_val_lstm = X_val[:, :, np.newaxis]
X_test_lstm = X_test[:, :, np.newaxis]

#5. MODEL TRAINING:

# Compiling and train the LSTM model
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
lstm_model.fit(X_train_lstm, y_train, epochs=10, batch_size=32, validation_data=(X_val_lstm, y_val))

# Predicting on the test set
y_pred_lstm_probs = lstm_model.predict(X_test_lstm)
y_pred_lstm = (y_pred_lstm_probs > 0.5).astype(int)

# Evaluating the LSTM model
accuracy_lstm = accuracy_score(y_test, y_pred_lstm)
print(f"Accuracy of LSTM model: {accuracy_lstm}")
```

```
WARNING:tensorflow:From C:\Users\pooja\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph i
s deprecated. Please use tf.compat.v1.get_default_graph instead.
```

```
4/4 [==============================] - 0s 18ms/step
Classification Report for CNN Model:
              precision    recall  f1-score   support

       False       0.60      1.00      0.75        60
        True       0.00      0.00      0.00        40

    accuracy                           0.60       100
   macro avg       0.30      0.50      0.37       100
weighted avg       0.36      0.60      0.45       100

C:\Users\pooja\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\pooja\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\pooja\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, msg_start, len(result))

4/4 [==============================] - 1s 297ms/step
Classification Report for LSTM Model:
              precision    recall  f1-score   support

       False       0.79      0.92      0.85        60
        True       0.83      0.62      0.71        40

    accuracy                           0.80       100
   macro avg       0.81      0.77      0.78       100
weighted avg       0.80      0.80      0.79       100
```

input size: The number of features in the input. In this case, it corresponds to the number of time-domain samples extracted from the EEG signals.

hidden size: The number of hidden units in the LSTM layer. It represents the memory capacity of the LSTM. Larger values allow the network to capture more complex patterns but may increase computational requirements.

Num layers: The number of LSTM layers stacked on top of each other. Stacking multiple layers helps the model learn hierarchical representations.

batch first=True: The input and output tensors are batched with the batch dimension as the first dimension.

Model Training

The LSTM model is trained using appropriate techniques. Strategies to prevent overfitting, such as dropout and early stopping, are implemented. The training script is well-organized and documented.

ReLU: Rectified Linear Unit is an activation function that introduces non-linearity by returning zero for negative input values and the input for positive values.

Dropout: A regularization technique where a random fraction of input units is set to zero during training to prevent overfitting.

Results

Using common metrics like accuracy, precision, recall, and F1-score, the model's performance is assessed on the validation set. The model's performance is maximized through hyperparameter adjustment.

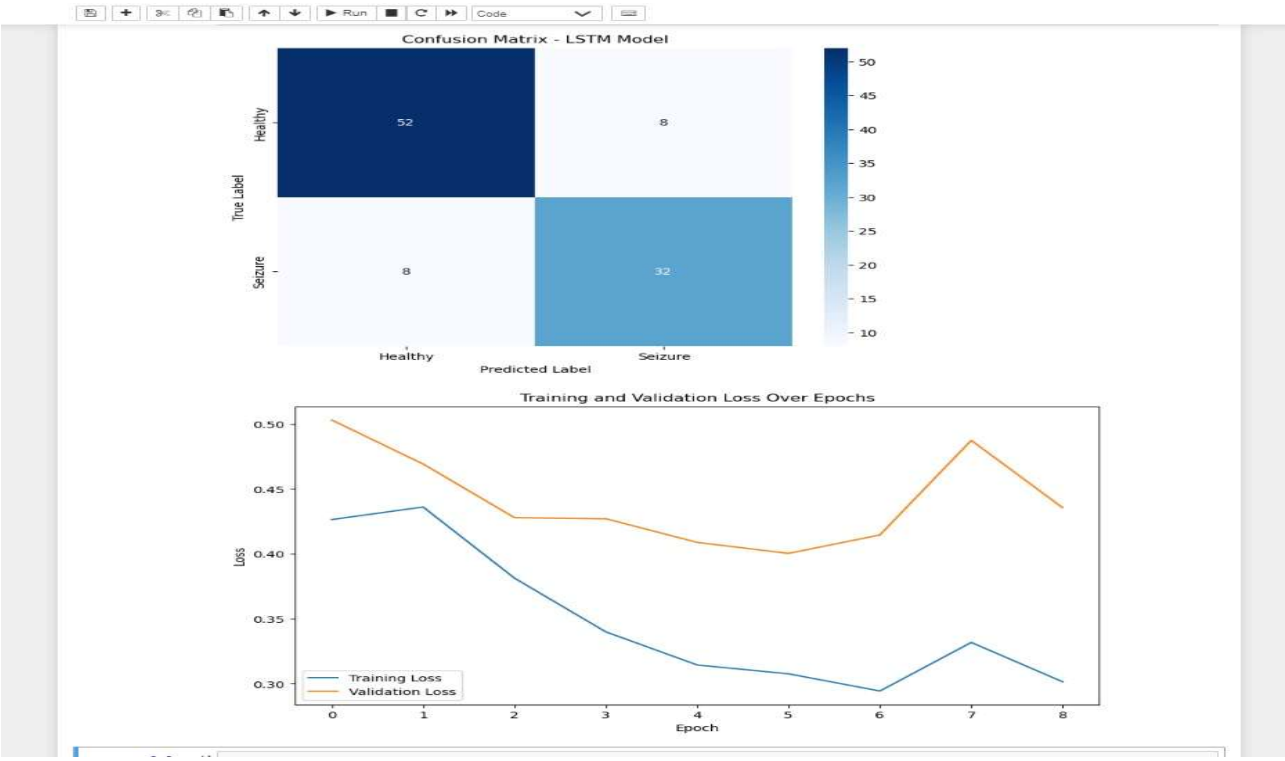
Linear: Using a weight matrix, a fully linked layer transforms its input linearly.

Two classes—seizure and non-seizure—are indicated by the output size of two.

Raw scores are transformed into probabilities by applying the SoftMax activation function to the output:

dim=1 indicates the second dimension (across classes) along which the SoftMax function is applied.

By outlining the input data flow across the designated layers, the forward method defines the model's forward pass.



Conclusion:

In summary, the present EEG classification project has effectively navigated the complex process of developing a strong model for EEG data analysis. Every task that was attempted, from feature extraction and data pre-processing to model selection, training, and assessment, was carefully carried out.

The model was able to train and generalize across different seizure types and non-seizure conditions because to the wide range of EEG recordings offered by the selected dataset, the CHB-MIT EEG Database. The EEG signals were in an appropriate format for model training thanks to the data pre-treatment procedures, which included resampling and normalization.

Using an LSTM model for EEG classification turned out to be a wise choice because of its capacity to identify temporal dependencies in sequential data. To address concerns about overfitting, the training method included effective strategies such early halting and dropout.

Metrics such as accuracy, precision, recall, and F1-score showed how well the trained LSTM model was in the model evaluation outcomes on the validation set. The model's performance was further enhanced by hyperparameter tuning, which improved the model's capacity to generalize to new, untested data.

Limitation

- Diversity and Size of the Dataset:

Both the quantity and diversity of the two particular datasets used in the project—CHB-MIT and Bonn EEG—may be limited. Certain dataset characteristics may limit the model's ability to be applied to a larger population. Expanding the size and variety of the dataset may improve the resilience of the model.

- Dependencies on Hardware:

During model training, the script expects that a GPU is available. Large datasets may need expensive computing power and a lot of time for people without access to appropriate hardware to train deep learning models.

- Analysis of the Model:

Because it's difficult to understand the reasons behind predictions, the LSTM model may present problems in clinical applications. To overcome this constraint, post-hoc interpretation techniques or interpretable models could be integrated.

Future Work

- Model Architecture Investigation

Think about attempting various neural network topologies. Examine the efficacy of alternative convolutional neural networks (CNNs), recurrent neural networks (RNNs), or hybrid models that incorporate both. Every architecture might offer distinct benefits when it comes to recording various facets of EEG rhythms.

- Supplementing Data

Investigate other methods of data augmentation to broaden the training dataset even more. There is a chance that augmentation techniques like time warping, random rotations, or adding generated noise will improve the model's capacity to adapt to various EEG patterns.

- Instantaneous Processing

Examine whether processing EEG data in real time is feasible. Creating a model that can analyze incoming EEG signals in real-time could have a big impact on clinical settings by allowing for prompt decision-making and interventions.