National College of Ireland

Dynamic Time Warping Enhanced CNN-LSTM for Robust Seizure Prediction in EEG Data

MSc Research Project
Data Analytics

Bharath Subramani
Student ID: x22240721

School of Computing
National College of Ireland

Supervisor: Mr. Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Bharath Subramani
Student ID:	x22240721
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Mr. Bharat Agarwal
Submission Due Date:	12/08/2024
Project Title:	Dynamic Time Warping Enhanced CNN-LSTM for Robust Seizure Prediction in EEG Data
Word Count:	1358
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Bharath Subramani
Date:	11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Dynamic Time Warping Enhanced CNN-LSTM for Robust Seizure Prediction in EEG Data

Bharath Subramani
x22240721

1 Introduction

This configuration manual offers step-by-step procedures and guidelines on how to set up and execute the identified project on epileptic seizure prediction using a CNN-LSTM hybrid model with DTW. The project focuses on the data pre-processing, feature engineering, model developing and model assessing based on deep learning methods.

2 System Configuration

As for the matters of implementation and execution of the given project, needs are mandatory with regard to the equipment as well as the programs. The characteristics of the hardware are: manufacturer: Intel (R) Extended Brand Marketing, Brand Title: Intel(R) Core(TM) i5-10300H @ 2. 50GHz processor, with presently, it has 8.00 GB of installed memory (7. 84 GB usable) and has affinity for 64 bit operating system with x64 based processor. For quicker training particularly for deep learning models, it is advised to use NVIDIA graphic card supporting CUDA though it's not mandatory.

3 Data Collection

The data set applied in this project is known as the 'Epileptic Seizure Recognition,' data set which the author downloaded from the Kaggle website; a site commonly used in offering data sets and machine learning competitions. In this dataset, there are five classes of signal that are all picked up through the EEG and represent various conditions of the brain. To achieve this particular project, the dataset is fetched from Kaggle source in CSV file format. Another aspect is to check if the data is compacted and safely stored from the unauthorized uses, but at the same time accessible for the further data preparation activity. It is constituted by such a large dataset that will aid in training and even validation of all the machine learning models under this project.

4 Environment Setup

First of all, you have to download the Python and all required libraries for this project. First, it is necessary to install Anaconda, which can make the work with environment and Python installation easier. Once installation of Anaconda is done, then create another Conda environment with Python 3. 7 and activate it. Then you need to install

```
#Importing required Libraries and Packages
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from scipy.signal import butter, lfilter
from fastdtw import fastdtw
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D, Flatten, BatchNormalization
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
from numpy.fft import fft, fftfreq
```

Figure 1: Importing python libraries

the necessary libraries: TensorFlow, NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, SciPy, FastDTW with the help of Conda and pip. This setup is particularly important so as to guarantee that all necessary setups in the environment are suited for the project.

5 Data Pre-processing

This activity can thus be described as one of the essential preprocessing steps of the data set to work with. It includes data pre-processing in which the data is made ready for training Data pre-processing may include formulation, transformation or normalizing of data as per the training process. In this project, the following pre-processing steps were undertaken:This project consisted of the following pre-processing steps as a prior of the analysis:

5.1 Handling Missing Values

The absence of some observations was also wanted in the set since they may have an influence on the performance of the used machine learning algorithms. Thankfully in the current dataset scrutin did not have to venture out for any missing values therefore was not in a position to do imputation of the values or even the deletion of the values. As such, it helps in minimization of wastage of data in that all the data will be used on modeling and checking of the model.

```
#Checking for null values
missing_values = data.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 Unnamed    0
X1          0
X2          0
X3          0
X4          0
           ..
X175        0
X176        0
X177        0
X178        0
y           0
Length: 180, dtype: int64
```

Figure 2: Handling Null values

5.2 Feature Scaling

Due to the fact that EEG data is presented in various scales depending on the feature, normalization of the features was required. Standardization was done to bring out the fact that in each of the features there must be a mean equal to 0 and a standard deviation of 1. This step is especially important for models like a neural network as such models' performance depends directly on the scale of the inputted data. Standardization is a way of making certain that all features of the medium have the same impact on learning.

```
# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_filtered)
```

Figure 3: Standardization

5.3 Splitting the Data

The pre-processing was done and based this data 70% was used for training the model while 30% was used for testing the model. This was also applied while training and tuning hyperparameters of the model, 20% of the used training data were checked using model check in a bid to minimize over fitting. Some of the training procedure that was used included early stopping or reduction of the learning rate to define the time and rate that training should occur.

```
# Splitting the data into training and testing sets into 70:30 respectively
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

Figure 4: Splitting the data

5.4 Applying Butterworth bandpass filter

A Butterworth bandpass filter is applied to reduce the unwanted noise from the EEG signals. The filtered dataset is named X_{filtered} .

```
# Function to extract DTW features
def extract_dtw_features(X, reference_sequence, sequence_length):
    dtw_features = []
    reference_sequence = np.asarray(reference_sequence, dtype=np.float64).ravel()

    for sequence in X:
        sequence = np.asarray(sequence, dtype=np.float64).ravel()

        # Compute the DTW distance
        distance, _ = fastdtw(sequence, reference_sequence, dist=2)

        # Create fixed-size sequences for CNN input
        dtw_sequence = np.full(sequence_length, distance)
        dtw_features.append(dtw_sequence)

    return np.array(dtw_features)
```

Figure 5: Butterworth bandpass filter

6 Exploratory Data Analysis

6.1 Verifying Class Distribution

Exploratory Data Analysis (EDA) is a process where the given data is analyzed in order to know about the data, patterns in the data, outliers and to check assumptions. It helps in the determination of the data preprocessing steps that need to be undertaken as well as the most appropriate model to be used.

```
# Plotting class distribution
plt.figure(figsize=(8, 5))
sns.countplot(x=y)
plt.title('Distribution of Classes')
plt.xlabel('Class')
plt.ylabel('Frequency')
plt.show()
```

Figure 6: Class Distribution

7 Data Modelling

To address the feature learning of spatial and temporal dynamics of EEG data, the CNN-LSTM model is applied in this step. For spatial features of the data the CNN

```
# Adding convolutional layers
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(sequence_length, 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(BatchNormalization())
model.add(Dropout(0.2))
```

```
# Adding LSTM layers
model.add(LSTM(64, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dropout(0.2))
```

Figure 7: Enter Caption

layers are employed to learn local patterns within the time series data. These features are then passed into the LSTM layers which are special types of layers that are capable of understanding the sequential information. Therefore, the presented approach combines the strengths of CNNs and LSTMs and can be used for epileptic seizure prediction in EEG signals. The model is then compiled with the appropriate optimizer and the loss function is used and training is controlled by checkpoints such as, early stopping and reducing the learning rate to avoid overfitting.

8 Feature Extraction

The feature extraction process in this project is based on DTW to compare time-series data which is EEG sequences in this case. The extract_dtw_features function is defined to calculate the DTW distance from the given dataset with all EEG sequences to a reference sequence, which may be the average of all training sequences. This DTW distance considers the temporal shifts as well as differences between the sequences, which can be a good characteristic that depicts the temporal behavior of the EEG signals. The extracted DTW distances are then converted into fixed-size sequences as it is imperative to feed the CNN in the following modeling step. This way, the model is capable to recognize non-linear time dependencies in the EEG signals and, therefore, improve the accuracy of the epileptic seizure prediction.

```python
# Function to extract DTW features
def extract_dtw_features(X, reference_sequence, sequence_length):
    dtw_features = []
    reference_sequence = np.asarray(reference_sequence, dtype=np.float64).ravel()

    for sequence in X:
        sequence = np.asarray(sequence, dtype=np.float64).ravel()

        # Compute the DTW distance
        distance, _ = fastdtw(sequence, reference_sequence, dist=2)

        # Create fixed-size sequences for CNN input
        dtw_sequence = np.full(sequence_length, distance)
        dtw_features.append(dtw_sequence)

    return np.array(dtw_features)
```

Figure 8: Dynamic Time Warping

9 Model Training

For the training of the CNN-LSTM hybrid model, the training loop with callbacks that are specifically designed for early stopping and reducing the learning rate is used. Another form of regularization used is early stopping whereby the training is carried out until the model's performance on the validation set begins to decline. On the same note, the learning rate reduction callback is used to decrease the learning rate once the validation loss flattens in order to keep the model learning even as it nears convergence. This combination of the techniques can be effective in training the model in the best way possible that is the model can be made to generalize as much as possible while using as much of the computational power as possible.

```python
# Compiling the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```python
# Callbacks for learning rate adjustment and early stopping
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.0001)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Figure 9: Training the model

10 Model Evaluation

The last stage of the project is the testing of the proposed CNN-LSTM model on the test set in order to check the model's efficiency on new, yet unseen data. This entails determining measures such as the accuracy, precision, recall, and F1 score that help in assessing the model's performance systematically. Also, graphs including confusion matrices and accuracy/loss learning curves are used to reveal the capability that a model has for classifying classes and the performance of the model when training. It is important to conduct these evaluations to determine the model's performance and where possible improvements can be made.

```
#Importing required Libraries and Packages
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from scipy.signal import butter, lfilter
from fastdtw import fastdtw
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D, Flatten, BatchNormalization
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
from numpy.fft import fft, fftfreq
```

```
# Predictions
y_pred = model.predict(X_test_dtw)
y_pred = (y_pred > 0.5).astype(int)
```

```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
# Plotting confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
# Performance metrics
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Figure 10: Evaluating the model