Configuration Manual

MSc Research Project
MSc Data Analytics

Mohammad Tabish
Student ID: x22205705

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mohammad Tabish
Student ID:	x22205705
Programme:	Msc Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Vikas Tomer
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	1153
Page Count:	6

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:	
Date:	16th September 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):	

Configuration Manual

Mohammad Tabish
x22205705

1 Introduction

Detailed configuration for the project Hybrid Deep Learning Strategies for Epileptic Seizure Detection. The present work is intended to provide a stable platform for the proper detection of epileptic seizures using deep learning models within EEG data. The guide details the system configuration, project workflow, and scripts to reproduce all applicable tables/figures from this paper.

2 System Configuration

2.1 Hardware Requirements

For the successful running of the project, you need to have these Hardware Requirements.

- **Processor:** A powerful CPU, such as an Intel i7 or equivalent, so that the computational load whilst training and evaluating deep-learning models can be handled.
- **RAM:** 16 GB or more – Deep learning algorithms have a lot of operations and data to deal with, so you need enough memory.
- **Storage:** Fast storage, at least 256 GB SSD – this is a necessary condition for quick accessing of datasets and reading-writing.
- **GPU:** NVIDIA GPU with CUDA support (Highly Recommended) — If you have a CUDA-enabled GPU, it will greatly speed up the training of deep learning models, especially on large datasets and complex architectures.

2.2 Software Requirements

The software environment applied within this project enclosure:

- **Operating System:** A stable operating system, such as Ubuntu 20.04 LTS or Windows 10, that supports the necessary software and libraries.
- **Python 3.8 or higher** – The programming language used for developing and running the deep learning models.
- **Jupyter Notebook** – An interactive environment for developing, testing, and sharing code.

- **TensorFlow 2.x** – The primary deep learning framework used for building and training the models.
- **Scikit-learn** – A library for preprocessing data, evaluating models, and handling machine learning tasks.
- **Matplotlib** – A plotting library used for creating static, interactive, and animated visualizations.
- **Seaborn** – An extension of Matplotlib for statistical data visualization.
- **Plotly** – A graphing library used for creating interactive visualizations.
- **Joblib** – A library for saving and loading machine learning models.

```
import os
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import joblib
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 1: Importing Libraries

3 Project Implementation

4 Data Collection

Publicly available datasets of EEGs were used in gathering the data for this project. These sets are for three different conditions: healthy, interictal (between seizures), and during a seizure. Both of these sets contain EEG recordings that are used for training and evaluating the deep learning models.

	0	1	2	3	4	5	6	7	8	9	...	4088	4089	4090	4091	4092	4093	4094	4095	4096	Type
80	36.0	27.0	24.0	12.0	6.0	-11.0	-22.0	-21.0	-10.0	-3.0	...	32.0	65.0	97.0	100.0	88.0	80.0	66.0	59.0	6.0	Healthy
84	-44.0	-63.0	-85.0	-73.0	-72.0	-69.0	-55.0	-44.0	-38.0	-34.0	...	-29.0	-30.0	-14.0	11.0	41.0	40.0	20.0	-25.0	15.0	Healthy
33	-10.0	-9.0	-33.0	-24.0	-61.0	-52.0	-41.0	-4.0	16.0	14.0	...	-12.0	-5.0	-6.0	-5.0	-17.0	-37.0	-29.0	-18.0	-51.0	Healthy
81	20.0	27.0	-5.0	-32.0	-38.0	-1.0	41.0	65.0	73.0	49.0	...	39.0	43.0	31.0	-12.0	-73.0	-114.0	-111.0	-71.0	-9.0	Healthy
93	-45.0	-62.0	-76.0	-98.0	-108.0	-106.0	-83.0	-65.0	-48.0	-41.0	...	-8.0	10.0	20.0	26.0	25.0	9.0	-24.0	-56.0	49.0	Healthy

5 rows × 4098 columns

Figure 2: Dataset Structure

5 Feature Selection

Before proceeding with modeling, feature selection is done for pre-processing the EEG data. The features were extracted from the raw EEG signals as it is, mainly concentrating on two main observations:

- **Amplitude:** This is the amplitude of an EEG signal, which reveals how much brain activity there was.
- **Temporal Dynamics:** Temporal dynamics, which capture patterns of the signal over time that are characteristic of particular brain states.

It is also important to incorporate these features because routine EEGs capture the signature of epileptic activity in similar frequency bands—a necessary requirement for distinguishing between healthy, interictal, and ictal states.

6 Data Pre-processing

EEG preprocessing steps also needed to be taken in order for the data fed into our deep learning models to have consistency and reliability.

- **Segmentation:** The fixed-length segments were made of EEG signals to normalize the input data.
- **Padding/Truncating:** Shorter signals were zero-padded, and longer ones were truncated.
- **Normalization:** Signals were scaled to a common range, as it can accelerate the convergence of models during training.

7 Feature Engineering

Feature Engineering for Improving Epileptic Seizures Detection Model: An ensemble of hybrid deep learning models was engineered that consisted of the following:

- **CNN:** We utilized Convolutional Neural Networks for extracting spatial features from the EEG data.
- **LSTM:** The signals were then processed using Long Short-Term Memory networks for capturing the temporal dependencies in them.
- **GRU:** Simpler Gated Recurrent Units, which are not as powerful to model long-term dependencies like LSTM, were also tried for modeling temporal information.
- **Attention Mechanisms:** The inclusion of Attention layers enabled the models to attend to important aspects for interpreting EEG signals.

8 Exploratory Data Analysis

An Initial Exploratory Data Analysis (EDA) to know the distribution and overview of EEG data. The data was visualized using diverse types of plots:

The histogram summarizes the consistent component of amplitude distributions for groups, and a box plot adds information about both range and dissemination values of amplitudes.

```python
fig = px.histogram(df_sample, x=0, color='Type', nbins=100, title="Amplitude Distribution")
fig.update_layout(xaxis_title="Amplitude", yaxis_title="Count")
fig.show()

fig = px.box(df_sample, x='Type', y=0, title="Amplitude Distribution by Type")
fig.update_layout(xaxis_title="Type", yaxis_title="Amplitude")
fig.show()

fig = go.Figure()

# Adding lines for each type
for dtype, data in [('Healthy', df_A), ('Interictal', df_C), ('Ictal', df_E)]:
    sample_indices = np.random.choice(data.shape[0], size=5, replace=False)
    for index in sample_indices:
        fig.add_trace(go.Scatter(y=data.iloc[index, :-1], mode='lines', name=f"{dtype} - Sample {index}"))

fig.update_layout(title="EEG Signals for Selected Samples",
                  xaxis_title="Time",
                  yaxis_title="Amplitude")
fig.show()

fig = px.scatter(df_sample, x=0, y=1, color='Type', title="Scatter Plot of EEG Signals")
fig.update_layout(xaxis_title="Amplitude at Time 0", yaxis_title="Amplitude at Time 1")
fig.show()

# Density plot for the amplitude values
fig = px.density_contour(df_sample, x=0, y=1, color='Type', title="Density Contour of EEG Signals")
fig.update_layout(xaxis_title="Amplitude at Time 0", yaxis_title="Amplitude at Time 1")
fig.show()
```

Figure 3: EDA

9 Data Transformation

The input to the convolutional neural network architectures was transformed EEG data. This involved reforming the data and making it model-ready.

10 Modelling

Several hybrid deep learning models were implemented in this project:

- **CNN with LSTM:** CNN for extracting spatial features but LSTM to help model the sequences.
- **CNN with GRU:** This is similar to CNN with LSTM but uses GRU for temporal modeling and is also computationally cheap.
- **CNN with Attention:** Improved the focusing of CNN-LSTM to only consider target information by integrating attention mechanisms.

```python
# Define the hybrid model
model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    Conv1D(128, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    LSTM(128, return_sequences=True),  # Added LSTM layer
    LSTM(64),  # Additional LSTM layer
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])

model.summary()

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy*100:.2f}%')
```

Figure 4: CNN with LSTM

```python
# Define the model
model = Model(inputs=inputs, outputs=outputs)

model.summary()

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=128, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy*100:.2f}%')

# Predictions
y_pred = np.argmax(model.predict(X_test), axis=1)

# Classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=['Healthy', 'Interictal', 'Ictal']))
```

Figure 5: CNN with LSTM and Attention Mechanism

```python
# Define the model
model = Model(inputs=inputs, outputs=outputs)

model.summary()

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=128, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy*100:.2f}%')

# Predictions
y_pred = np.argmax(model.predict(X_test), axis=1)

# Classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=['Healthy', 'Interictal', 'Ictal']))

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Healthy', 'Interictal', 'Ictal'], yticklabels=['Healthy', 'Interictal', 'Ictal'])
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```

Figure 6: CNN with GRU

	Model	Accuracy	Loss	Precision (Macro Avg)	Recall (Macro Avg)	F1-Score (Macro Avg)
0	CNN	0.8792	0.3984	0.90	0.87	0.88
1	CNN with LSTM	0.5847	0.8951	0.64	0.58	0.55
2	CNN with LSTM and Attention	0.7932	1.1814	0.83	0.75	0.76
3	CNN with GRU	0.4918	0.9513	0.54	0.43	0.37

Figure 7: Model Comparision

11 Evaluation

Models were evaluated individually on multiple metrics like accuracy, loss, precision, recall, and F1 score. We compare the results of the evaluation to know which model performs best.

12 Conclusion

We aim to offer a supporting comprehensive guide for the configuration and actual deployment of hybrid deep learning strategies in epileptic seizure detection based on this protocol's results. The project includes figures with code examples that will help anyone replicate the research and comprehend some of its mechanisms.