National College of Ireland

Configuration Manual

MSc Research Project
Msc in Artificial Intelligence

Parkhi Bhardwaj
Student ID: 23163861

School of Computing
National College of Ireland

Supervisor: Kislay Raj

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Parkhi Bhardwaj
Student ID:	23163861
Programme:	Msc in Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	Kislay Raj
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	700
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:	Parkhi Bhardwaj
Date:	12th December 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

Parkhi Bhardwaj
23163861

1 Requirements Guide

In order to run the code, please follow all the steps written in this configuration manual. This document describes all the necessary requirements like software and hardware requirements, environment setup, and more.

2 Machine Hardware requirements

The following are the hardware requirements that will be needed primarily for the project to work. The configurations of the machine were: 16gb RAM, 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz, 64-bit OS, Windows 11.

3 Machine Software requirements

The following software requirements are used to run the code. Jupyter Notebook is the environment used for the code compiling. Python 3.12 is used as the language for this project. Google drive/Gmail account is used to link to notebook. Microsoft excel is used to store data as csv file. Overleaf was used to write the research project report and this configuration manual.

4 Environment set up

Anaconda Navigator environment setup is used here. The next step is to Launch Jupyter Notebook. The steps are also followed by images to help in better understanding of the steps to be taken. Figure 1 describes the environment setup of Anaconda Navigator.

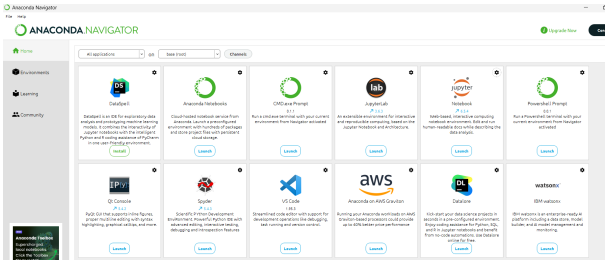


Figure 1: Anaconda Navigator

5 Data Selection Process

The dataset used for this research project is from the open source dataset website https://physionet.org/. The dataset is called MIT-BIH Arrhythmia Database. Figure2 shows the overview of the dataset page on physionet.

Figure 2: MIT-BIH Arrhythmia Dataset

6 Install Libraries

The following libraries need to be installed for the research project. The results may vary depending on the fact that some of the libraries have not been properly installed. The below list includes all the libraries used. Snapshot of python file is given in Figure 3.

1. Pandas
2. Tensorflow
3. Numpy
4. Scikit-learn
5. Matplotlib
6. Seaborn
7. Plotly
8. Mlxtend

```
In [28]: import pandas as pd
         import numpy as np
         import seaborn as sns
         import matplotlib.pyplot as plt
         import plotly.express as px

         import tensorflow as tf
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import SimpleRNN, GRU, LSTM, Dense, Dropout
         from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

         from sklearn.preprocessing import MinMaxScaler
         from sklearn.model_selection import train_test_split

         from sklearn.metrics import classification_report, confusion_matrix
         from mlxtend.plotting import plot_confusion_matrix

         from sklearn.metrics import roc_curve, auc
         from sklearn.preprocessing import label_binarize

         from sklearn.metrics import mean_squared_error
```

Figure 3: Install Libraries

7 Implementation and using the code files

A quick breakdown of the specific files that are in the folder. There are 4 jupyter notebooks in python for all the four models used. Each file performs data preprocessing primarily then specific models are applied therefore, each jupyter notebook file is named according to the architecture, see figure4. Below is the detail description of each file.

- RNN- This file builds RNN model and measure performance metrices.
- LSTM- This file builds LSTM model and measure performance metrices.
- GRU- This file builds GRU model and measure performance metrices.
- CNN-LSTM- This file builds hybrid CNN-LSTM model and measure performance metrices.

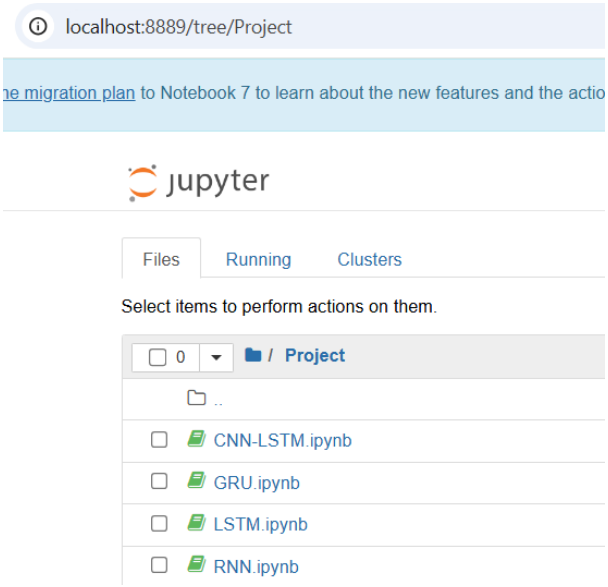


Figure 4: Project folder containing code files.

Download the dataset and upload in to the same folder. Two .csv files containing dataset is shown in figure5.

Figure 5: Dataset in the project folder

In order to run the code, open the python notebook files one by one and run the code. Each code contains all the preprocessing steps like data importing, data preprocessing, data balancing, data scaling, data visualization code. Below are the few snapshots of data preprocessing.

```
In [4]: # Load dataset
train_data = pd.read_csv('mitbih_train.csv', header=None)
test_data = pd.read_csv('mitbih_test.csv', header=None)

In [5]: print(f"Number of samples in train data: {train_data.shape[0]}")
print(f"Number of samples in test data: {test_data.shape[0]}")

Number of samples in train data: 87554
Number of samples in test data: 21892

In [6]: # Dataset balancing
train_data[187]=train_data[187].astype(int)
equilibre=train_data[187].value_counts()
print(equilibre)

187
0    72471
4     6431
2     5788
1     2223
3      641
Name: count, dtype: int64

Arrhythmia Dataset¶
Number of Samples: 109446
Number of Categories: 5
Sampling Frequency: 125Hz
Data Source: Physionet's MIT-BIH Arrhythmia Dataset
Classes: ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]
```

Figure 6: Data Loading and balancing

```
In [7]: # Explore the dataset
train_data.head()

Out[7]:
```

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049020	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082621	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045946	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058011	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.150235	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

```
5 rows × 188 columns

In [8]: print(f"Missing values in train data: {train_data.isna().sum().sum()}")
print(f"Missing values in test data: {test_data.isna().sum().sum()}")

Missing values in train data: 0
Missing values in test data: 0

In [9]: print(f"Duplicated values in train data: {train_data.duplicated().sum()}")
print(f"Duplicated values in test data: {test_data.duplicated().sum()}")

Duplicated values in train data: 0
Duplicated values in test data: 0

In [10]: print (train_data.iloc[:, -1].value_counts())  # Check class distribution in train data

187
```

Figure 7: Data Scaling and splitting

```
In [7]: # Explore the dataset
train_data.head()

Out[7]:
```

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049020	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082621	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045946	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058011	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.150235	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

```
5 rows × 188 columns

In [8]: print(f"Missing values in train data: {train_data.isna().sum().sum()}")
print(f"Missing values in test data: {test_data.isna().sum().sum()}")

Missing values in train data: 0
Missing values in test data: 0

In [9]: print(f"Duplicated values in train data: {train_data.duplicated().sum()}")
print(f"Duplicated values in test data: {test_data.duplicated().sum()}")

Duplicated values in train data: 0
Duplicated values in test data: 0

In [10]: print (train_data.iloc[:, -1].value_counts())  # Check class distribution in train data

187
```

Figure 8: Exploring dataset

In [12]:
```
test_classes = test_data.iloc[:, -1].unique()
train_classes = train_data.iloc[:, -1].unique()
labels = {
    0: "Normal",
    1: "Artial Premature",
    2: "Premature ventricular contraction",
    3: "Fusion of ventricular and normal",
    4: "Fusion of paced and normal"
}
```

In [13]:
```
train_counts = train_data.iloc[:,-1].value_counts().rename(labels)

plt.figure(figsize=(8, 6))
ax = sns.barplot(x=train_counts.index, y=train_counts.values)

# Annotate each bar with the count
for i, p in enumerate(ax.patches):
    ax.annotate(f'{train_counts[i]}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom',
                fontsize=10)

plt.title('Number of images per class in train data')
plt.xlabel('Classes')
plt.ylabel('Number of samples')
plt.xticks(rotation=90)
plt.show()
```

Figure 9: Data Visualization

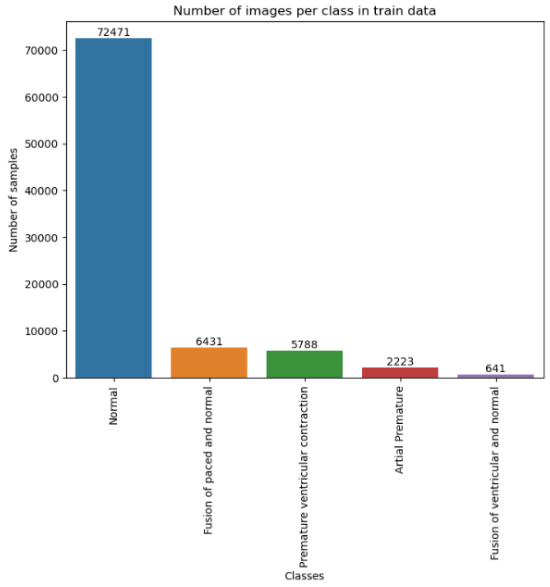


Figure 10: Visualization graph

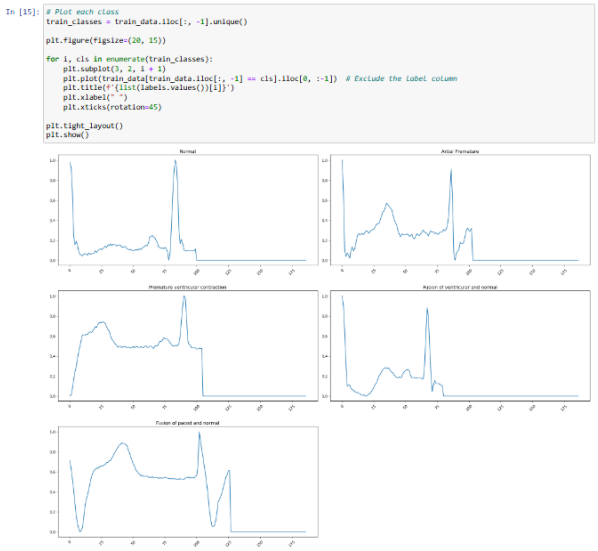


Figure 11: Classes

7.1 Model Training

Each code file contains the implementation of the model. Starting with training the data set and then validating and finally testing. Performance metrics like accuracy, precision, recall and F1-score are then evaluated. Below is the snapshot of the RNN model implemented.

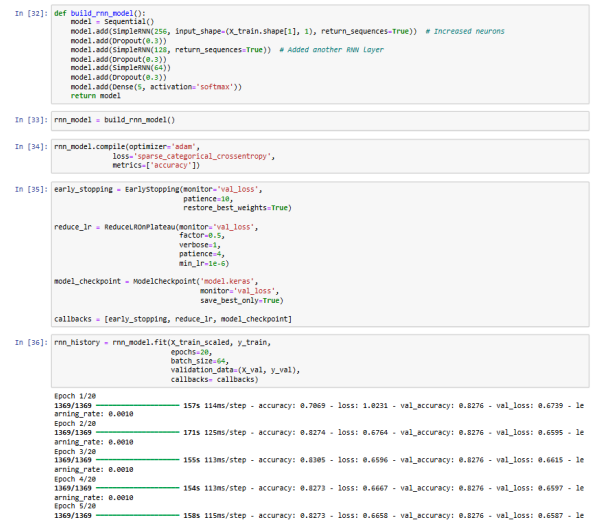
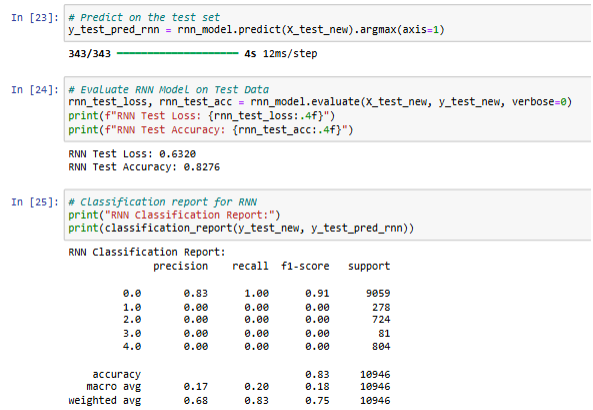


Figure 12: RNN Model

Results are given below:



```
In [23]: # Predict on the test set
         y_test_pred_rnn = rnn_model.predict(X_test_new).argmax(axis=1)

         343/343 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step

In [24]: # Evaluate RNN Model on Test Data
         rnn_test_loss, rnn_test_acc = rnn_model.evaluate(X_test_new, y_test_new, verbose=0)
         print(f"RNN Test Loss: {rnn_test_loss:.4f}")
         print(f"RNN Test Accuracy: {rnn_test_acc:.4f}")

         RNN Test Loss: 0.6320
         RNN Test Accuracy: 0.8276

In [25]: # Classification report for RNN
         print("RNN Classification Report:")
         print(classification_report(y_test_new, y_test_pred_rnn))

         RNN Classification Report:
                       precision    recall  f1-score   support

                  0.0       0.83      1.00      0.91      9059
                  1.0       0.00      0.00      0.00       278
                  2.0       0.00      0.00      0.00       724
                  3.0       0.00      0.00      0.00        81
                  4.0       0.00      0.00      0.00       804

             accuracy                           0.83     10946
            macro avg       0.17      0.20      0.18     10946
         weighted avg       0.68      0.83      0.75     10946
```

Figure 13: RNN model result

The same has to be done for other python notebooks- LSTM, GRU, CNN-LSTM in order to obtain desired results. Once these steps are done the code will execute well and make it possible to consider the model performances and its evaluation.
Thank you for reading.

References