Prediction of ABP and ECG signal from PPG signal using deep learning
Configuration Manual

MSc Research Project
Data Analytics

Sarthak Sinha
Student ID: x21178321

School of Computing
National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sarthak Sinha
Student ID:	x21178321
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Teerath Kumar Menghwar
Submission Due Date:	18/09/2023
Project Title:	Prediction of ABP and ECG signal from PPG signal using deep learning
Word Count:	782
Page Count:	8

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:	Sarthak Sinha
Date:	17th September 2023

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):	

Prediction of ABP and ECG signal from PPG signal using deep learning

Sarthak Sinha
21178321

1 Introduction

The Configuration Manual offers a step-by-step guide outlining the essential hardware and software prerequisites for conducting modelling and running code, covering the entire implementation process from the preparation of data through its execution. This report acts as a thorough reference to aid in the replication of the study titled "Prediction of ABP and ECG signal from PPG signal using deep learning."

2 System Configuration

2.1 Hardware Configuration

Cloud-based hardware and local-based hardware components and specifications will be addressed in this section.

Local-Based System: A MacBook machine is used to conduct the initial stages of the research which involved data exploration and data preprocessing. The Macbook machine had a RAM of 16 GB and Apple M1 as CPU with a total of 8 cores.

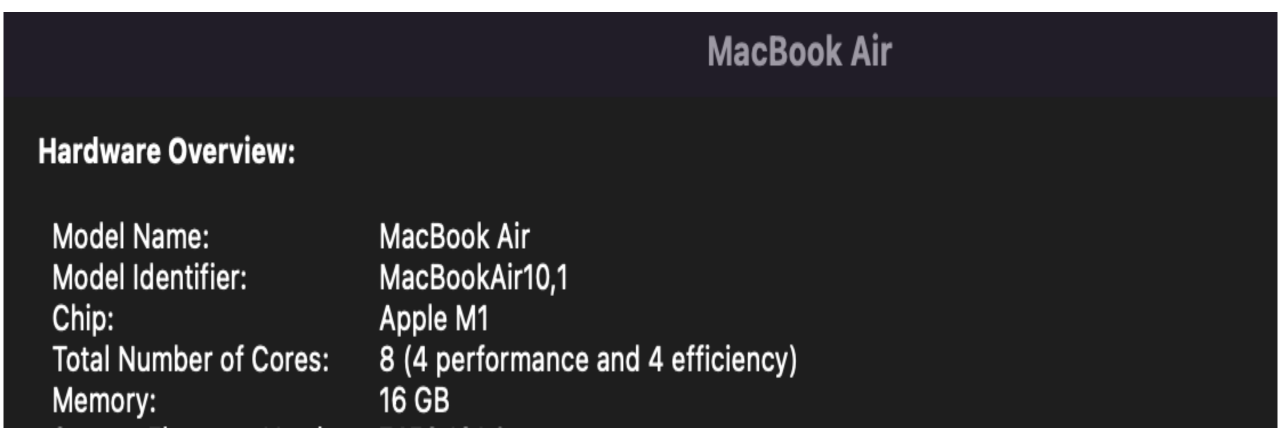

Figure-1 Local Machine Specification (MacBook)

Cloud-Based System: To carry out the data-intensive task of the research such as deep learning model training, Amazon Web Services (AWS) with cloud-based EC2 service have been used which was provided by the National College of Ireland[1]. AWS EC2 services provide virtual machines or instances with user-preferred operating systems. The instances have various instance types which can be found in this [link](). For the purpose of this research Memory Optimized based AWS EC2 instance type had been configured due to the volume of the dataset.

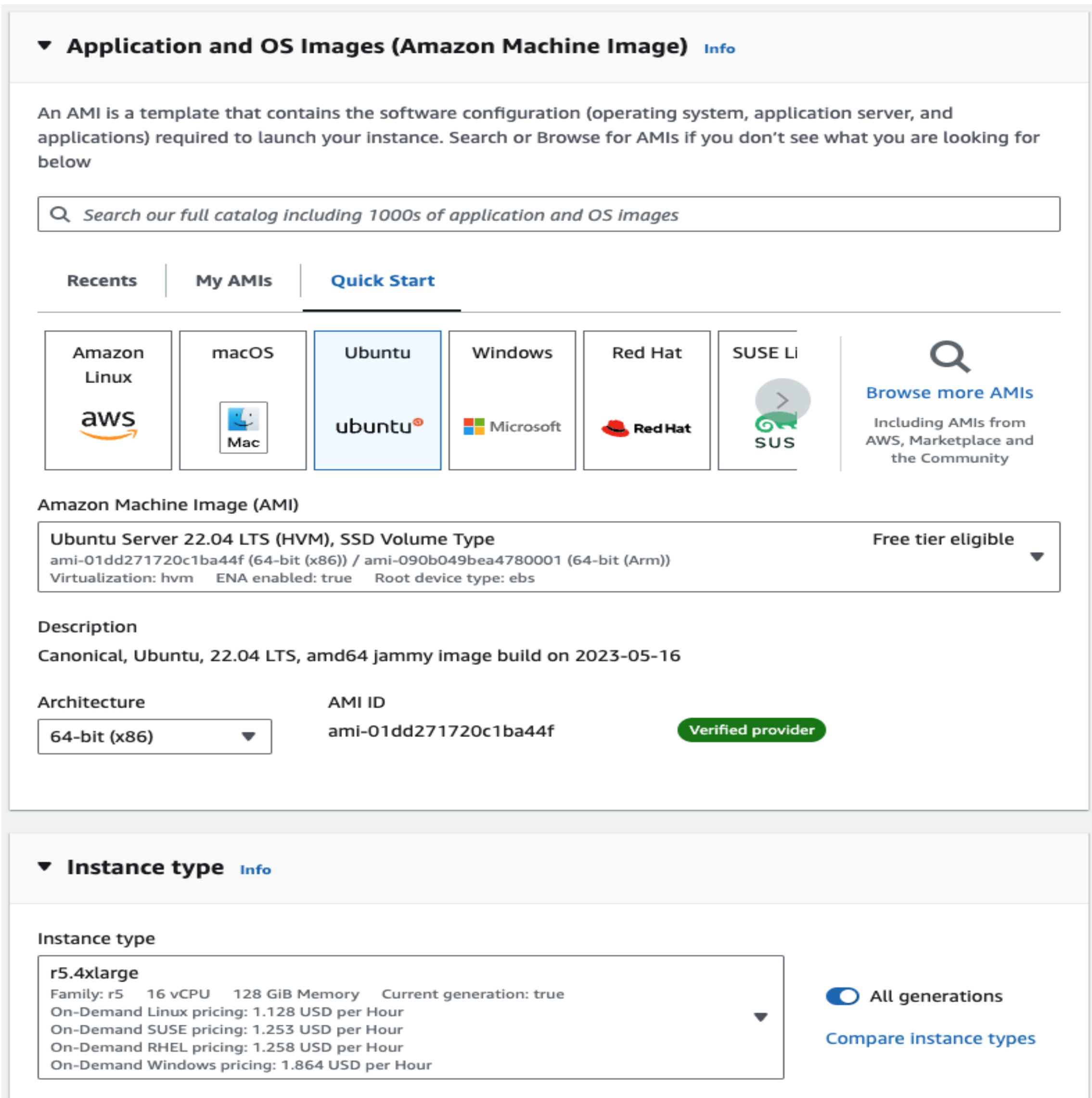

Figure-2 AWS EC2 Instance

[1] https://cloud.ncirl.ie/

An EC2 instance with instance type of family r5.4xlarge has been chosen for this research. It had 16 CPUs and 128 GB RAM. Ubuntu OS with SSD volume type had been used. The pricing can be referred to in the above screenshot.

2.2 Software Configuration

In this section, we will go through the tools, frameworks and libraries used as software components in this research. The code for this research has been developed in Python 3.11.4 with MacOS Command Line Interface (CLI). Jupyter Notebooks[2] has been used which is a web-based interactive development environment for conducting data science workflows. The jupyter notebook is installed using "pip" which is a package installer in Python using the following command "pip3 install notebook".

Setting Up of Virtual Environment: A virtual environment was created using the venv[3] module. Using the following command.

- python3 -m venv <name_of_virtualenv>
- source <name_of_virtualenv>/bin/activate, the following command is used to activate the virtual environment.

Python Packages/ libraries had been installed which can be found in the "requirements.txt" file along with their specific versions. These packages can be installed in the system using the following command "pip3 install -r requirements.txt".

In order to set up the Jupyter Notebook kernel with the virtual environment following command have been used "python -m ipykernel install --user --name=<name_of_virtualenv>"

3 Project Development

Once the previous steps are executed, create a new Jupyter Notebook file for the project. Using the "jupyter notebook" command in the CLI will open the interface of the notebook. Next, create a new notebook with the preferred kernel and import all the necessary libraries into the notebook.

---

[2] https://jupyter.org/
[3] https://docs.python.org/3/library/venv.html

Importing Libraries

```python
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import torch
import os
import warnings
import tensorflow as tf
from keras import layers
from keras import backend as K
from keras import optimizers
from scipy import signal
from sklearn.model_selection import KFold, train_test_split
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Reshape, Conv1D, MaxPooling1D, Flatten, Dense, Bidirectional, LSTM, Dropc
warnings.filterwarnings('ignore')
```

Figure-3 Importing Libraries

3.1 Reading the Dataset

There are 8 ".mat" files available on the Kaggle[4] website and need to be stored in the project directory. Mention the dataset folder name in the 'datapath' variable name, in this research, the dataset folder name was "archive1". A function is created to load all the ".mat" files from the dataset folder and combined using os[5] and SciPy[6] module python.

Defining Functions to load full and partial list of mat files

```python
def load_data_partial(filename):
    mat_contents = sio.loadmat(filename)
    return mat_contents
```

```python
def load_data(fileDir, exercise):
    word = exercise.lower()
    file_path_list = []
    valid_file_extensions = [".mat"]
    valid_file_extensions = [item.lower() for item in valid_file_extensions]


    for file in os.listdir(fileDir):
        extension = os.path.splitext(file)[1]
        if extension.lower() not in valid_file_extensions:
            continue
        file_path_list.append(os.path.join(fileDir, file))

    Data = []
    for path in file_path_list:
        base=os.path.basename(path)
        base = os.path.splitext(base)[0]
        if word in base:
            print(fileDir+'/%s'%(base))
            mat_contents = sio.loadmat(fileDir+'/%s'%(base))
            val = mat_contents['p']
            total_array = val[0,:] #assigning an array
            Data.append(total_array)

    return Data
```

Figure-4 Load Dataset

[4] https://www.kaggle.com/datasets/mkachuee/BloodPressureDataset
[5] https://docs.python.org/3/library/os.html
[6] https://scipy.org/

3.2 Feature Extraction

Features like PPG, ECG and ABP signals are extracted from the combined dataset and represented as a numpy[7] array as shown below in Figure-5.

Feature Extraction: Extracting PPG ,ECG and ABP values from total_data

```
PPG = []
ABP = []
ECG = []
for i in range((len(total_data))):
    for j in range(len(total_data[i])):
        k = len(total_data[i][j][0,:])
        for n in range(k//1000):
            ppg = (total_data[i][j][0,(n*1000):(n*1000)+1000]) # Extracting PPG values
            abp = (total_data[i][j][1,(n*1000):(n*1000)+1000]) #Extracting ABP values
            ecg = (total_data[i][j][2,(n*1000):(n*1000)+1000]) #Extracting ECG values
            PPG.append(ppg)
            ABP.append(abp)
            ECG.append(ecg)

# Converting list of PPG,ECG and ABP as array
PPG = np.asarray(PPG)
ABP = np.asarray(ABP)
ECG = np.asarray(ECG)


```

Figure-5 Feature Extraction

3.3 Data Visualization

Plotting graphs for PPG,ECG and ABP signals

```
# plotting sample  ppg, ecg and bp signals
fig, ax = plt.subplots(3,1, figsize=(6,6), sharex=True)
y = 1000
ax[0].set_title('Photoplethysmography (PPG) graph', fontsize=12)
ax[0].set_ylabel('Signal Value')
ax[0].plot(PPG[y,:], c = 'dodgerblue')

ax[1].set_title('Electrocardiogram (ECG) graph', fontsize=12)
ax[1].set_ylabel('Signal Value')
ax[1].plot(ECG[y,:], c='darkorange')

ax[2].set_title('Arterial Blood Pressure (ABP) graph', fontsize=12)
ax[2].set_ylabel('Signal Value')
ax[2].set_xlabel('Sample size')
ax[2].plot(ABP[y,:], c = 'red')
```

```
[<matplotlib.lines.Line2D at 0x298411db0>]
```

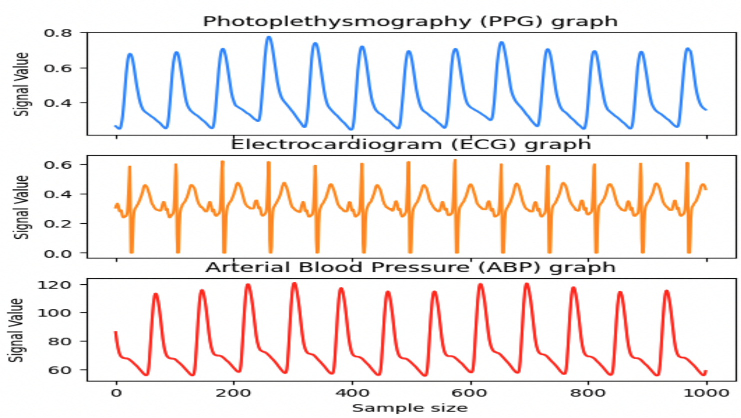

Figure-6 Dataset Visualization

To perform the data visualization matplotlib[8] package was used to infer the signals.

[7] https://numpy.org/doc/stable/reference/generated/numpy.array.html
[8] https://matplotlib.org/

3.4 Data Preprocessing

Defining Function to Normalize the PPG and ABP values

```
def normalise(x):
    normalised = (x-min(x))/(max(x)-min(x))
    return normalised

def scale_abp(x):
    normalised = x/200
    return normalised

def normalise_abp(abp, x_max, x_min):
    normalised = (abp-x_min)/(x_max-x_min)
    return normalised

def abp_maxmin_value(x):
    max_x = []
    min_x = []
    for i in range(len(x)):
        for j in range(len(x[i])):
            max_x.append(max(x[i][j][1,:]))
            min_x.append(min(x[i][j][1,:]))
    x_max = max(max_x)
    x_min = min(min_x)
    return x_max, x_min
```

Figure-7 Dataset Normalization

To keep the range between 0 and 1 the dataset normalization is performed using a min-max scaler function.

3.5 Data Preparation

Splitting the normalized data into 70% train and 30% test

Input : *Normalized PPG values*

Output: *Normalized ABP and ECG values*

```
X_train_PPG_N, X_test_PPG_N, y_train_ABP_N, y_test_ABP_N, y_train_ECG_N, y_test_ECG_N = train_test_split(
    PPG_N, ABP_N, ECG_N, test_size=0.30)
```

```
X_train_PPG_N_reshape = np.reshape(X_train_PPG_N, (X_train_PPG_N.shape[0], X_train_PPG_N.shape[1], 1))
```

```
X_test_PPG_N_reshape =np.reshape(X_test_PPG_N, (X_test_PPG_N.shape[0] ,X_test_PPG_N.shape[1],1))
```

Figure-8 Dataset Preparation

The dataset is split into train and test split with a ratio of 70 to 30 using the scikit-learn[9] package and the split signal is reshaped as per the deep learning model required dimensional shape which is 3 dimensions using the numpy module.

3.6 Model Building

Using the deep learning framework TensorFlow[10] the models were built using TensorFlow functional API[11].

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, MaxPooling1D, Flatten, Dense
```

```
input_shape = (X_train_PPG_reshape.shape[1],1)
inputs = Input(shape=input_shape)

# Add the first convolutional layer
x = Conv1D(filters=256, kernel_size=3, activation='relu')(inputs)
x = MaxPooling1D(pool_size=2)(x)

# Add additional convolutional layers as needed
x = Conv1D(filters=128, kernel_size=3, activation='relu')(x)
x = MaxPooling1D(pool_size=2)(x)

# Add additional convolutional layers as needed
x = Conv1D(filters=64, kernel_size=3, activation='relu')(x)
x = MaxPooling1D(pool_size=2)(x)

# Flatten the output for further processing
x = Flatten()(x)

# Branch 1 for output 1
bp_output = Dense(units=1000, activation='linear',name='bp_out')(x)

# Branch 2 for output 2
ecg_output = Dense(units=1000, activation='linear',name='ecg_out')(x)

model = Model(inputs=inputs, outputs=[bp_output, ecg_output])
```

```
Metal device set to: Apple M1
```

Figure-9 Model Building

3.7 Model Evaluation

The model is evaluated using two metrics Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) using the TensorFlow library as shown below in Figure-10.

---

[9] https://scikit-learn.org/stable/
[10] https://www.tensorflow.org/
[11] https://www.tensorflow.org/guide/keras/functional_api

Evaluating CNN model with Normalized ABP values

```python
#Predicting on the test set using the LSTM model
CNN_predictions_N = model.predict(X_test_PPG_N_reshape)
rmse = tf.keras.metrics.RootMeanSquaredError()
rmse.update_state(y_test_ABP_N, CNN_predictions_N[0])
print(f'CNN Model RMSE for Normalized ABP: {rmse.result().numpy()}')

# MAE for LSTM Model
MAE= tf.keras.metrics.MeanAbsoluteError()
MAE.update_state(y_test_ABP_N, CNN_predictions_N[0])
print(f'CNN Model MAE for Normalized ABP: {MAE.result().numpy()}')
```

```
 10/301 [..............................] - ETA: 3s
```

```
2023-08-09 10:01:26.399143: I tensorflow/core/grappler/optimizers/custom_graph_opti
mizer for device_type GPU is enabled.
```

```
301/301 [==============================] - 4s 12ms/step
CNN Model RMSE for Normalized ABP: 0.11169151216745377
CNN Model MAE for Normalized ABP: 0.08471555262804031
```

Figure-10 Model Evaluation

NOTE: There will be three different files which are created based on three deep learning models: Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN) and Hybrid CNN-LSTM model.