TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

DEEP LEARNING APPROACH FOR HEART RATE PREDICTION USING PPG SIGNAL

SUBMITTED BY:

NAYAN BASTAKOTI (PUL075BEI018)
SUMAN LAMSAL (PUL075BEI042)
SURAJ POUDEL (PUL075BEI044)
YUKTA SHARMA (PUL075BEI028)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING
LALITPUR, NEPAL

April, 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Deep learning approach for heart rate prediction using PPG signal"** submitted by **Nayan Bastakoti**, **Suman Lamsal**, **Suraj Poudel**, and **Yukta Sharma** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
Anku Jaiswal
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
Person B
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner
Person C
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgement

In bringing out the report of our major project undertaken in our final year, We recognize all the teachers whose contributions have been undeniably helpful and their valuable assistance in preparation of this project has been of great importance. We would like to thank the Department of Electronics and Communication Engineering, IOE, Pulchowk campus and academic authorities for providing necessary guidelines, suggestions and support. We especially recognise our supervisor, **Asst. Prof. Anku Jaiswal**, for her priceless and meticulous supervision, advice and vital contribution during the research of this project.

Further on, we also acknowledge the role played by the seniors in guiding us through earlier projects and setting the benchmark which motivates us. We have not just learned the content of the project, but also learned group work and problem tackling among the group members while researching for the project. Therefore, we thank all the concerned authorities for their valuable contributions.

Abstract

Photoplethysmography (PPG) is a low-cost optical device that measures changes in blood volume in the microvascular tissue bed from the skin's surface. It has been used in commercial medical devices to gauge peripheral vascular disease and autonomic function by monitoring blood pressure, heart rate, and oxygen saturation. Due to the presence of motion artifact during exercises, there arises difficulty in measuring heart rate from PPG signal. A machine learning based approach is used to monitor heart rate (HR) using wrist-type photoplethysmography (PPG) signals in this paper. By combining 1D CNN and a bidirectional LSTM, the model get benefit from the strengths of both architectures, capturing both local and long-term patterns in the input data. The proposed model exhibits average absolute error of less than 1.5 bpm for all the training and test datasets. The model shows the promising result with less than 300 thousands network parameters.

Keywords: *PPG, ECG, CNN, LSTM, Motion Artifact, Sequence models*

Contents

List of Figures

List of Tables

List of Abbreviations

ANC	Adaptive noise cancellation
BPM	Beats Per Minute
CNN	Convolutional neural network
DFT	Discrete Fourier Transform
ECG	Electrocardiography
EMD	Empirical mode decomposition
FFT	Fast Fourier Transform
GRU	Gated Recurrent Unit
HR	heart rate
ICA	Independent component analysis
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
LED	Light emitting diode
LOSO	Leave One Session Out
LSTM	Long-short term memory
MA	Motion artifact
MAE	Mean Absolute Error
NAG	Nesterov accelerated gradient
Nadam	Nesterov-accelerated Adaptive Moment Estimation
NLL	Negative log likelihood
PPG	Photoplethysmography
ReLU	Rectified Linear Unit
RNN	Recurrent neural network
SPC	Signal Processing Cup
SVD	Singular value decomposition
SVM	Support vector machine
WFPV	Wiener Filtering and the Phase Vocoder

1. Introduction

Continuous monitoring of heart rate in natural environments can provide clinicians with valuable information about a patient's health status, especially for patients with heart diseases. Many wearable devices such as Samsung Gear Fit, Atlas Fitness Tracker, and Mio Alpha Heart Rate Sport Watch have the capability to monitor heart rate in real-time using Photoplethysmography (PPG) signals obtained from the wearer's wrists. The PPG signals can be sensed and measured from various body parts, e.g., the finger, ear, wrist, arm, etc [9] using optical sensors that consist of a light transmitter (LED) and a receiver (photodetector). These sensors can operate in transmission or reflection modes, and they detect changes in blood volume by measuring the intensity of the light reflected from the skin.

1.1 Background

Over the past several decades, the capability of wearable technology has gradually increased. With their numerous internal and external sensors, contemporary wearable technology provide a wide range of practical fitness tracking functions,including step, calorie, sleep, and other tracking metrics. One of these capabilities is the ability to measure heart rate (HR) during exercise using photoplethysmography (PPG). HR monitoring, which is incorporated into smartwatches or wristbands, can direct exercisers in adapting their training load to better meet their training objectives. PPG signals have gained popularity as an alternative to conventional HR estimation methods based on Electrocardiography (ECG).

In this study, we focus on using deep learning techniques to estimate heart rate from PPG signals obtained from the wrist. By combining a 1D CNN and a bidirectional LSTM, the model can benefit from the strengths of both architectures, capturing both local and long-term patterns in the input data. The 1D CNN can extract features that are specific to local regions of the input sequence, while the bidirectional LSTM can capture long-term dependencies and contextual information across the entire sequence. Our approach is evaluated on a publicly available dataset, and the results demonstrate the effectiveness of our method in accurately estimating heart rate from PPG signals.

1.2 Problem statements

The main challenge of incorporating a reflectance type photoplethysmography sensor device is its inaccuracy during intensive physical activities of the subject. While measuring the HR, a wide range of parameters are needed to be taken in consideration such as– different activities, body movements, body types, and wearable-device form factors, i.e., smartwatches, earbuds, wristbands, etc. There are five foundational challenges that impact the accuracy: Optical Noise, Sensor Location, Skin Tone, The Crossover Problem, Low Perfusion. It is difficult to detect the heartbeat accurately, since the signals are contaminated by extremely strong motion artifacts caused by subjects' hand movements and the sensor is sensitive to it. Regardless of the methods adopted in order to overcome the effects of MA, it is not possible to get accurate results in the measurement of HR of a subject when the power corresponding to the HR in the measured PPG is low. The PPG's signal-to-noise ratio(SNR) is a challenging aspect of getting an accurate HR measurement. Low SNR usually occurs when the subject is indulged in intensive physical exercise.

1.3 Objectives

- To detect Heart Rate(HR) accurately minimizing the inaccuracy caused by motion artifact(MA).

1.4 Scope

This project will find an effective method to predict heart rate and will incorporate several wearable heart rate monitors. The product's utility will optimize athletics performance by providing real-time data. It will also help in early diagnosis of cardiac arrhythmia (Atrial Fibrillation) and Arterial stiffness. It can also be used for biometric identification. The potential applications of our work are vast, ranging from improving the accuracy of wearable health monitoring devices to aiding in the diagnosis and treatment of cardiovascular diseases.

2. Literature Review

2.1 Related work

Several studies and scientific research have been carried out in heart rate(HR) estimation. HR monitors are becoming increasingly common on portable devices such as smartwatches and fitness trackers. In these devices, the pulse is detected by an optical sensor built into the wrist unit's watchband or case back. Many different groups have tried to reduce the power dimension and improve the accuracy of HR. Several Digital Signal Processing(DSP) techniques as well as Machine learning (ML) algorithms have been used to solve this issue.

PPG technology is plagued by two major issues: motion artifact (MA) noise and power consumption. MA removal methodologies are important for obtaining accurate HR.HR estimation from artifact-induced PPG signals has received a lot of attention recently.

Several methods including adaptive filtering [18], independent component analysis (ICA) [8], frequency-domain analysis, empirical mode decomposition (EMD), wavelet-based denoising, SVD decomposing and other decomposition models, Novel algorithm(WFPV) using wiener filtering, windows adaptive noise cancellation(ANC), spectral subtraction, and Kalman filtering have been used for removing MA. All algorithms' performances were evaluated in relation to ground truth HR, which was obtained concurrently from the ECG signal. The three-stage TROIKA method, based on signal decomposition, sparsity-based high-resolution spectrum estimation, spectral peak tracking, and verification, has largely influenced HR estimation from wrist-worn PPG [19].

To detect heartbeats from a PPG signal, several supervised learning approaches were proposed as an alternative to traditional techniques. A variety of neural network-based frameworks have been employed to estimate HR with the least amount of inaccuracy. Several algorithms were used for the classification of spectral peaks of PPG for heart rate tracking. [20] used linear regression for HR prediction. [16] used SVM classifier for classification based on spectral peak separation and spectral peak ratio. [1] used CNN and LSTM where images and signals were given as input to CNN for the training of network.

Low-complexity processing has recently gained attention in the context of real-time HR estimation in resource-constrained environments. In [3], a methodology based on the Fast Fourier transform (FFT) spectrum of short windows of PPG and tri-axial accelerometer signals was proposed.

The latest research paper published in 2022 [5], used the hybrid of signal processing and machine learning techniques which characterized signal using statistical, time, and frequency features and learned to estimate heart rate through a Convolutional-Recurrent Regressor.

2.2 Related theory

2.2.1 PPG signal

The cardiac cycle is the series of events that take place between the start and finish of a heartbeat. The ventricular diastole and the ventricular systole are the two primary phases of the heart cycle. Blood flows to the auricles during the diastole, also known as the relaxation phase, which causes a reduction in blood vessel pressure. The blood is pumped during the systole, or contraction phase out of the ventricles and dispersed throughout the body, increasing blood pressure.

PPG uses the optical biomonitoring method by detecting changes in the reflected light due to change in skin reflectivity during contraction and dilation of blood vessels. And later this reflected light on the photodiode causes changes in current flow which then is changed to a digital signal for further processing. A photo-emitter of infrared light is coupled to a photo-receiver, using as the medium of light propagation, the body segment in which it is desired to register the PPG signal as shown in [Fig. 2.1][10]. The pulsatile signal of the blood volume (pulse wave) is detected by the photo-transistor.

Further processing can be done in two approaches i.e. time domain and frequency domain where signals are fed into the neural network for the training process to determine the spectral peak of ventricular contraction and dilation.

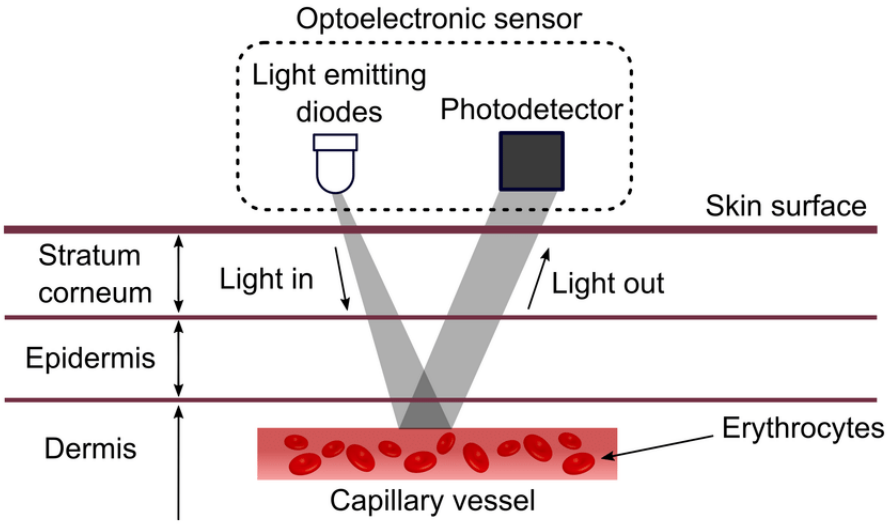


Figure 2.1: Working principle of PPG sensor

2.2.2 Deep Neural Network

Deep Neural Network is a subset of machine learning that mimics the way the brain learns. This technology is used by scientists and engineers to provide machines with some degree of "thinking" ability similar to that of the human mind. The networks process and reprocess data, gradually refining the analysis and results to accurately recognize, classify, and describe objects within the data. It is a subset of artificial neural networks (ANNs) that has multiple layers of interconnected nodes or neurons, which allow it to model complex non-linear relationships between input and output variables.

In a DNN, the input layer receives the raw data, such as an image or audio file, and then passes it through a series of hidden layers, each of which applies a set of learned weights and biases to the input to transform it into a more abstract representation. Finally, the output layer produces the predicted result, such as a classification or regression value.

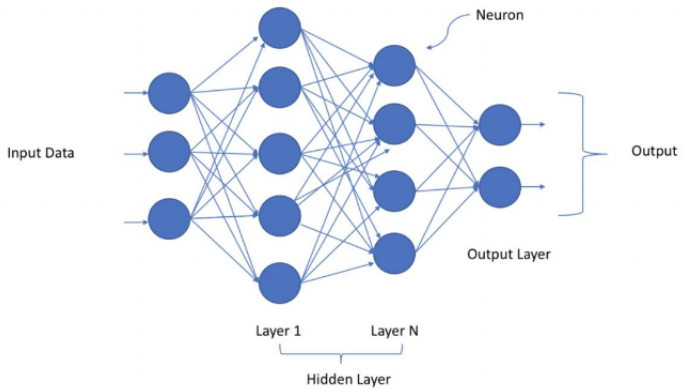


Figure 2.2: Deep Neural Networks

There are various types of deep neural networks (DNNs) that are used in different applications. Here are some of the most common types:

1. Feedforward Neural Network(FNNs): FNNs are type of neural network where information flows only in one direction, from the input layer through the hidden layers to the output layer. In FNNs, there are no cycles or loops in the network architecture. In this network, the input layer receives input data, which is then passed through one or more hidden layers where the data is transformed using nonlinear activation functions. The output of the final hidden layer is then fed into the output layer, which produces the final output of the network. The weights and biases of the network are learned during training using an optimization algorithm, such as backpropagation.

2. Convolutional Neural Networks (CNNs): CNNs are mainly used for image and video analysis. They consist of convolutional layers that detect and extract features from the input image, followed by pooling layers that reduce the spatial dimensions of the features. CNNs have achieved state-of-the-art performance in many computer vision tasks, such as image classification, object detection, and segmentation.

3. Recurrent Neural Networks (RNNs): RNNs are designed for sequential data analysis, such as natural language processing and speech recognition. They have loops in their architecture that allow them to process information from previous time steps, which makes them suitable for tasks that involve temporal dependencies. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are popular variants of RNNs that can handle long-term dependencies.

2.2.3 Butterworth Filter

The Butterworth filter is a type of signal processing filter that is designed to have a frequency response that is as flat as possible in the passband, which is the range of frequencies that the filter allows to pass through without significant attenuation. The main advantage of the Butterworth filter is that it has a maximally flat magnitude response in the passband, which means that it does not introduce any ripples or distortions in the frequency response. This makes it ideal for applications where a flat frequency response is important, such as in audio processing, image processing, or biomedical signal processing. The roll-off rate of the Butterworth filter can be controlled by adjusting the order of the filter. A higher-order filter will have a steeper roll-off rate but will also introduce more phase distortion in the passband. [Fig. 2.3] shows the filter passes the frequency component of 0.5-4.5Hz attenuate other.

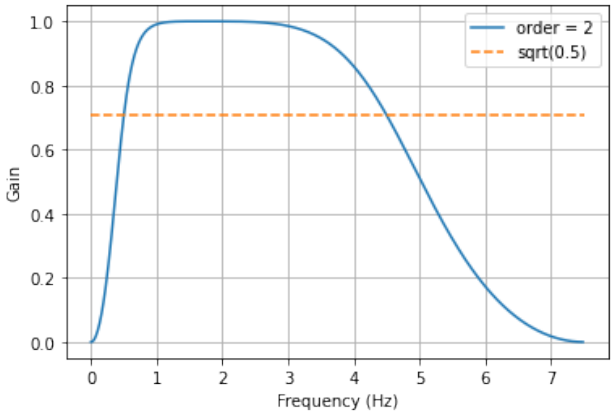


Figure 2.3: Butterworth frequency response

3. Methodology

We used publicy available dataset of PPG signals collected during treadmill exercise, which was preprocessed through sliding window, bandpass filter, resampling, and z-score normalization. We trained a deep learning model using a combination of convolutional neural networks (CNN) and bidirectional long short-term memory (LSTM) networks to predict heart rate from PPg signals.

3.1 Datasets

The datasets for our project:

3.1.1 IEEE Signal Processing Cup (SPC)

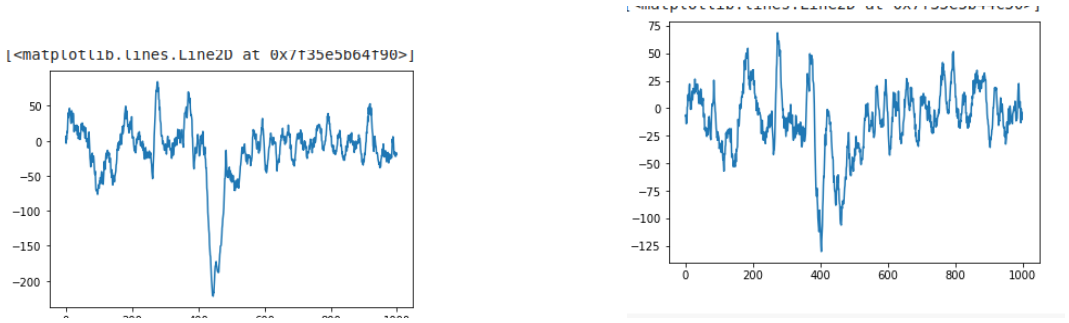
Its training data consists of 12 male subjects, yellow skin, 18-35 years for dataset collection. The wristband embedding consists of 2-channel PPG(pulse oximeter with 609 nm green LED), 3-axis acceleration, ECG recorded simultaneously from the chest (used as reference for HR). All signals sampled at 125 Hz and subjects are walked or ran on a treadmill in order:

- 1-2 km/h for 0.5 min
- 6-8 km/h for 1 min
- 12-15 km/h for 1 min
- 6-8 km/h for 1 min
- 12-15 km/h for 1 min
- 1-2 km/h for 0.5 min

Subjects used the hand to pull clothes, wipe sweat on forehead, and push buttons on the treadmill, in addition to freely swing. The test dataset consists of 11 subjects, aged 19-58 years whose wrist PPG was acquired using a pulse oximeter with green LEDs, 3-axis accelerometer, and ECG simultaneously recorded from chest. All signals were sampled at 125 Hz. It consists of five minutes recording of intensive arm movements. Subjects performed various commonly used arm rehabilitation exercises, running, jump, push-up and boxing.[19]

The training dataset is of 81.4MB in 'ts' file format. The 2-channel PPG and 3-axis acceleration and ECG value are separated with ':'. Each channel PPG signal is recorded for

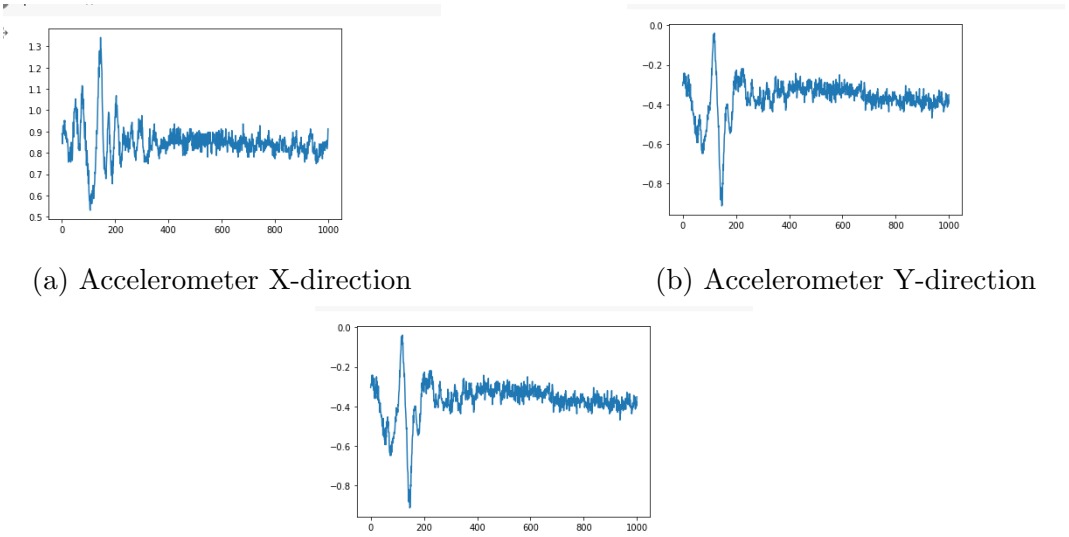
1000 times at a sampling rate of 125Hz for 8 sec. [Fig. 3.1] shows two channel PPG signal visualized using matplotlib and [Fig. 3.2] shows three-axis accelerometer data with captured heart rate of 75BPM using ECG.



(a) Channel 1 PPG signal

(b) Channel 2 PPG signal

Figure 3.1: PPG signal (75BPM)



(a) Accelerometer X-direction

(b) Accelerometer Y-direction

(c) Accelerometer Z-direction

Figure 3.2: Accelerometer X,Y,Z (75BPM)

3.1.2 BAMI-II

The BAMI-II dataset contains wrist PPGs recorded during stay, walking, and running. In this dataset, 23 subjects were taken in the exercise protocol which included

- 2 min of walking at 3–4 km/h
- 4 min of walking at 3–4 km/h
- 4 min of running at 6–8 km/h
- 1 min of rest to cool down

The session was designed to reflect cardiac rehabilitation exercise for cardiac patients with poor exercise ability, in which they normally walk or run by holding a treadmill bar. The subjects were 17 men and 6 women with an average age of 22.0 $\pm$ 1.7 years. The entire exercise process was performed on a treadmill. The signal of accelerometers and gyroscopes are collected to remove the motion artifact from the PPGs. A reference chest ECG is included to allow a benchmark comparison of heart rate during treadmill exercise. The PPG, accelerometer and gyroscope signal are sampled at 50 Hz wile ECG is sampled at 125Hz.[2].

3.2 Explored Approaches

Estimating heart rate from a photoplethysmography (PPG) signal can be done using different approaches. One common approach we tried was to first apply Fast Fourier Transform (FFT) to the PPG signal and extract the heart rate frequency component. In addition to using FFT of PPG signal as input for heart rate estimation, we used FFT of accelerometer data as an input. This is because motion artifact can interfere with the accuracy of heart rate estimation from PPG signals. By using accelerometer data as an additional input we account for the motion artifact which would have improved the accuracy of heart rate estimation. The FFT of accelerometer data can be used to identify the frequency components corresponding to the motion artifact. This frequency spectrum is then used as input to a convolutional neural network (CNN) instead of the raw PPG signal and raw accelerometer data. This allows the CNN to learn directly from the frequency domain features of the signal, which can potentially improve the accuracy of heart rate estimation. The CNN was trained using a dataset that includes the FFT of PPG signals, FFT of accelerometer and their corresponding heart rates.

While using FFT of accelerometer data as an input to the CNN can be a useful, it didn't gave the expected results. There could be several reasons for this, such as the CNN architecture may not be suitable for the data. The best MAE we were able to achieve was 12.633. [Fig. 3.3] shows the architecture of the CNN model we initially tried and [Fig. 3.4] shows the hyperparameter search for the CNN model architecture. We can see that the best set of hyperparameters i.e. batch size:8, conv2d layer 1 filters: 128, conv2d layer 2 filters: 256, conv2d layer 3 filters: 64, conv2d layer 4 filters: 1024, conv2d layer 5 filters: 8, dense units: 1024, in between activation: ReLU, learning rate: 0.0001 and output activation: tanh gave MAE of 12.633. So we, dropped the idea of using CNN and frequency domain approach and shifted to using CNN and LSTM with time domain approach

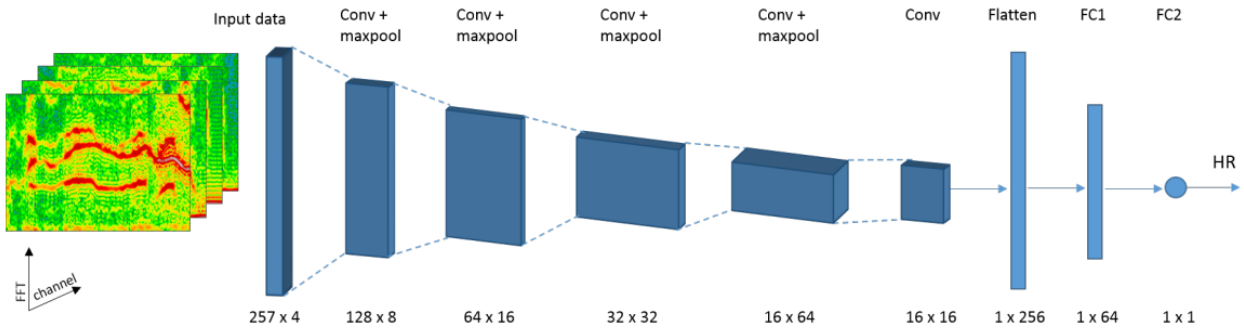


Figure 3.3: Tried CNN Architecture

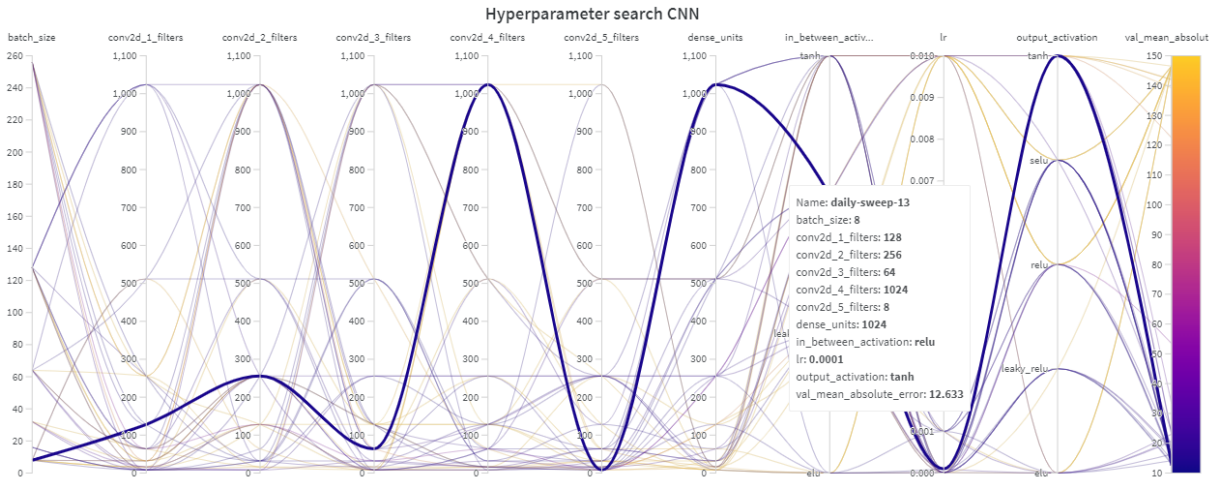


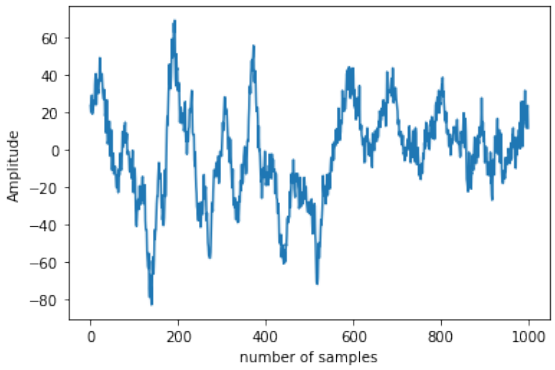
Figure 3.4: CNN hyperparameter tuning

3.3 Preprocessing

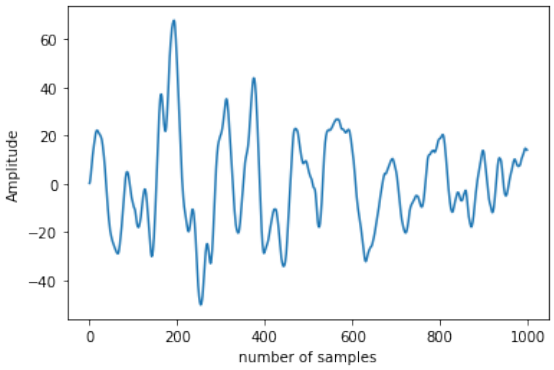
We passed signal into series of preprocessing techiques before training our deep learning model to ensure that it is in a suitable format for machine learning. The preprocesing techniques includes butterworth bandpass filter, resampling, sliding window and z-score normalization.

3.3.1 Bandpass Filter

Bandpass filter is performed to remove components of the PPG signal that do not reflect heart rate by allowing the signals of specific range and attenuating signal outside of range. The accelerometer and PPG signals were passed into 2nd order Butterworth bandPass Filter with cutoff frequencies of 0.5-4.5Hz which effectively removed signal components outside the range of cardiac activity.



(a) Original PPG signal



(b) Butterworth Filtered PPG signal

3.3.2 Resampling

Resampling reduces the amount of data while still retaining the relevant information about the cardiac waveform. The resampling works by applying a Fourier transform to the input signal and then interpolating the Fourier coefficients to the desired sampling rate. The resulting signal is then obtained by applying an inverse Fourier transform to the interpolated coefficients. The signals were then re-sampled to 64 Hz. After resampling the number of samples was reduced to 512 samples. This process can be broken down into the following steps:

1. Compute the discrete Fourier transform (DFT) of the input signal using the fast Fourier transform (FFT) algorithm.
2. Calculate the new sampling rate based on the desired resampling frequency and the original sampling rate.
3. Interpolate the Fourier coefficients to the new sampling rate using an up-sampling or down-sampling technique, depending on the new sampling rate relative to the original sampling rate.
4. Compute the inverse Fourier transform (IFFT) of the interpolated Fourier coefficients to obtain the resampled signal.

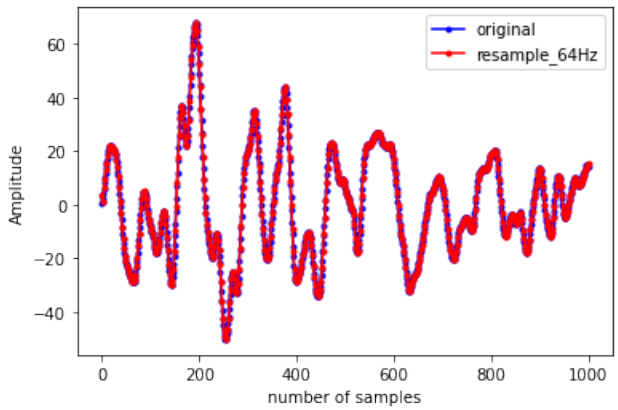


Figure 3.6: Resampling of signal

3.3.3 Sliding Window

The sliding window approach is used to segment the continuous PPG signal into smaller windows or segments of fixed duration. Each segment is then processed independently to estimate the heart rate within that segment. By sliding the window along the signal with some overlap, we can estimate the heart rate at multiple time points throughout the signal. For each subject, a sliding window approach was applied to the signals with a window length of 8 seconds and a 2 second slide. This means each window contained 8 seconds of PPG data, and adjacent windows overlapped by 6 seconds.

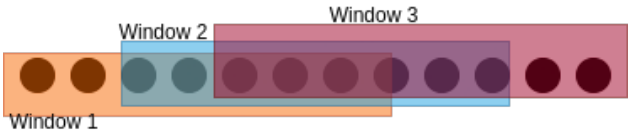


Figure 3.7: Sliding Window

3.3.4 Z-score Normalization

The final preprocessing step include z-score normalization, which is a common technique used to standardize the data and remove the effect of different scales in the input features. Z-score normalization ensure that the signals have a mean of zero and a standard deviation of one, which makes them more comparable across different signals and less sensitive to variations in the amplitude or baseline of the signal. It is calculated by using the simple formula

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where

- z is the z-score
- x is a particular data point
- μ is the mean of the input data
- σ is the standard deviation of the input data

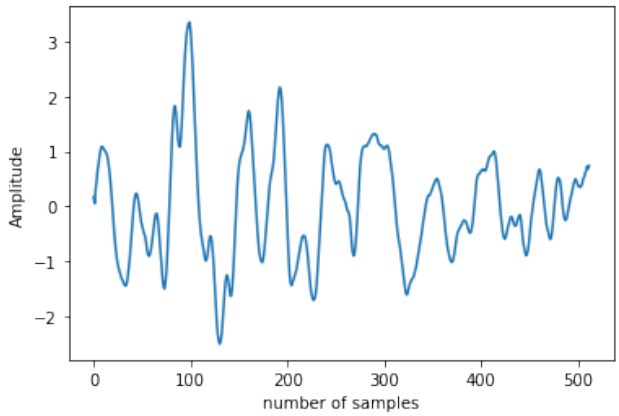


Figure 3.8: Normalized signal

3.4 Model Architecture and Implementation

In this paper, deep learning based heart rate prediction model consists of mainly two architecture i.e. Convolution Neural Network (1D- CNN) and Bidirectional LSTM. The preprocessed data of PPG signals and Accelerometer signals are used as separate input layer later they were merged for overall feature extraction using CNN. As, traditional neural network lacks correlation with time and each step of each parameters is independent. But, in heart rate prediction, change in physiological signals are closely related to time. So, we adopted time related Bidirectional LSTM as one of our architecture module. Later, output of Bi-LSTM is convolved, flattened and dense for final prediction.

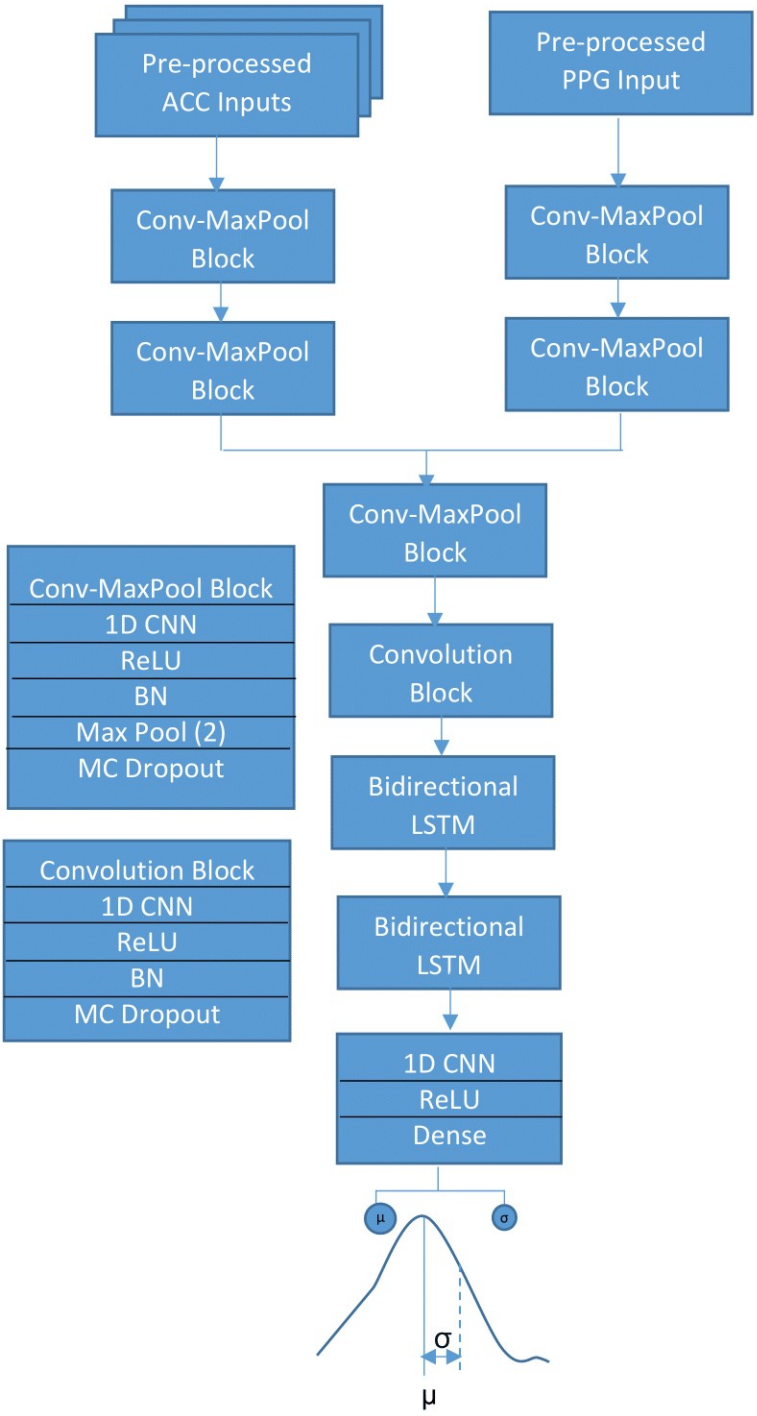


Figure 3.9: Model summary

3.4.1 Intuition behind each blocks of architecture:

Convolution layer for sensor data

The convolution block in sensor data extracts local interaction with each sensing modality. The input size of (512, 1) for PPG signal and (512, 3) for accelerometer sizhang2014troikagnals are convolved separately with filter size of 32. The activation function used in this block is ReLU. Then obtained output is batch normalized, averaged pooled as required and passed to the merging module. Several dropout layer are also used for preventing overfitting sensor data during feature extraction.

Convolution layer for concatenated data

The above obtained sensor specific features are then merged together in this module for extracting global features from both PPG and accelerometer signals. As like in first block, the obtained concatenated output is convolved, batch normalized and average pooled for obtaining global feature. Later, the output obtained from this block is passed to time based module.

Time- based module(Bi-directional LSTM module)

The global feature are passed to bidirectional LSTM for obtaining time domain feature that reflects the behavior of the signal over time. The number of LSTM units used is 32.The temporal features are then passed to the prediction module.

Prediction module

The temporal features are passed to the prediction module which contains a convolutional layer to reduce the dimensionality of the features for the fully connected layer.

3.4.2 Operational blocks used in architecture

Several unit blocks are used in model architecture for various functions which are described below:

1D-Convolution

One-dimensional convolution(1D-Convolution) is a mathematical operation commonly used in signal processing and machine learning. It involves sliding a small window, called a kernel or filter, over a one-dimensional input signal and computing a weighted sum of the values within the window at each position.

The basic idea behind 1D convolution is to extract local features from the input signal by analyzing the values in a small neighborhood around each point. The kernel defines the shape of this neighborhood and the weights assigned to each value in the neighborhood control how much influence it has on the output. The process of 1D convolution can be broken down into the following steps:

1.Define a kernel of size k and weights w1, w2, ..., wk.

2.Slide the kernel over the input signal x, starting from the leftmost position.

3.At each position, compute the dot product of the kernel and the values in the current neighborhood of size k centered at that position.

4.Store the result in a new output signal y at the corresponding position.

5.Repeat steps 2-4 for every position in the input signal, stopping when the kernel reaches the rightmost position.

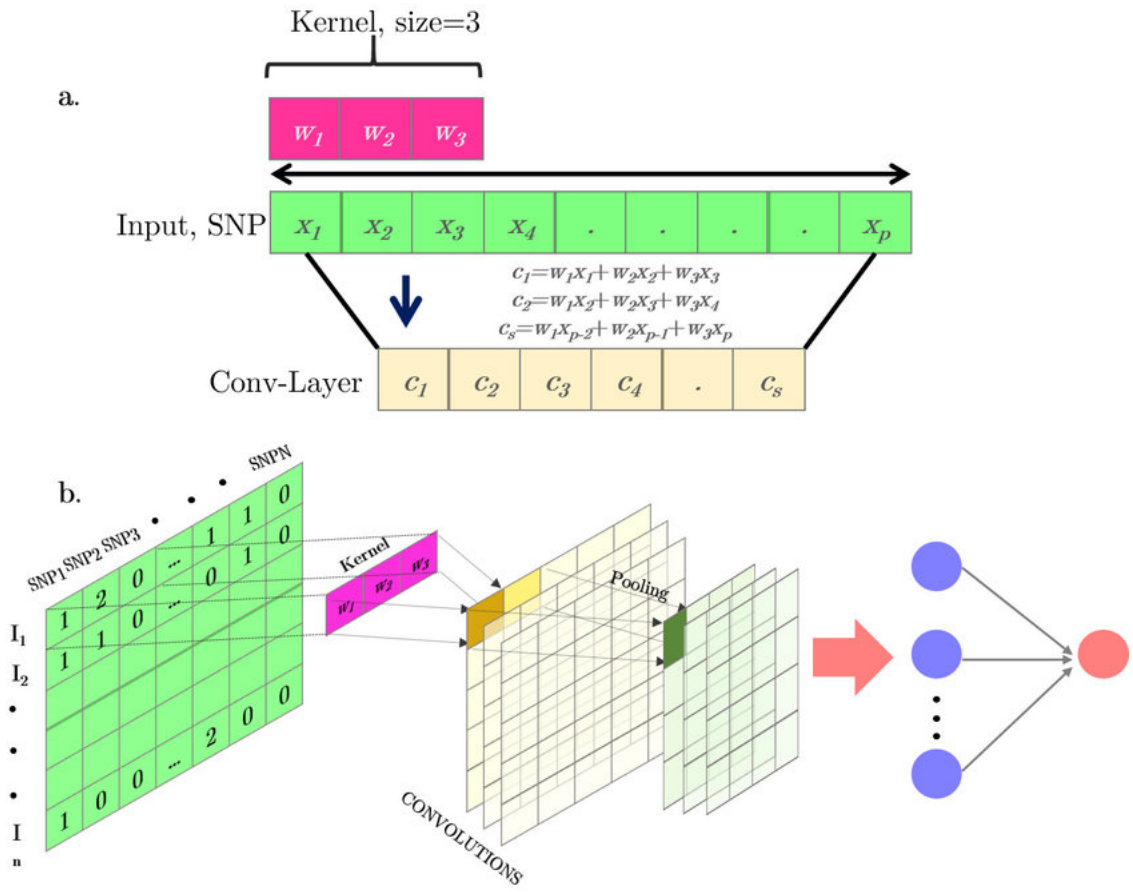


Figure 3.10: Operation of 1D-Convolution

Max pooling

Max pooling is a type of operation commonly used in convolutional neural networks (CNNs) for image recognition tasks. It is a way to reduce the spatial size (height and width) of the input feature map while retaining its depth (number of channels).

In max pooling, a window (often called the kernel or filter) is applied to the input feature map, and the output at each location is the max of the values within that window. The size of the window is typically a hyperparameter that is chosen based on the size of the input feature map and the desired output size. Max pooling is a way to reduce the dimensionality of the input feature map while retaining important features, and it can help to prevent overfitting and improve the efficiency of the network.

Activation Function

Here, in our architecture, we have used Rectified Linear Unit(ReLU) as an activation function. It can be calculated as:

$$f(x) = MAX(0, x) \tag{3.2}$$

In other words, the output of the function is the maximum of the input x and 0. The ReLU function is a non-linear function, which means that it can introduce non-linearities into the output of a neural network, allowing it to learn more complex relationships between inputs and outputs.

The main benefit of the ReLU function is that it is computationally efficient to evaluate and differentiate, which makes it a popular choice in neural network architectures. Additionally, ReLU has been shown to perform well in practice and is less prone to the vanishing gradient problem that can occur with other activation functions like the sigmoid function.[Fi]

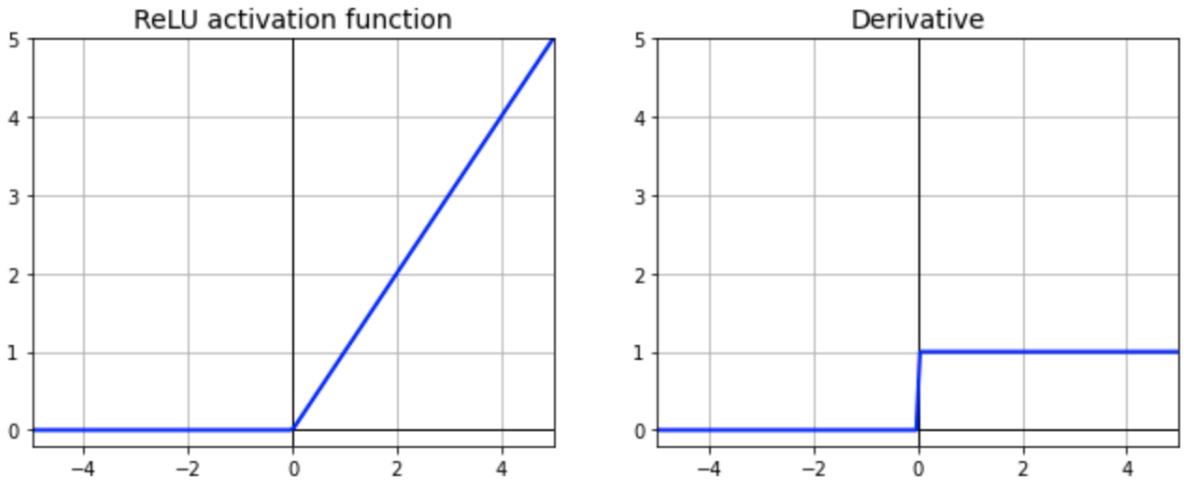


Figure 3.11: ReLU and its derivative

Batch Normalization

Batch normalization is a technique used in deep neural networks to normalize the inputs of a layer. It aims to improve the speed, performance, and stability of the training process.
The normalization process involves centering and scaling the inputs to a layer by subtracting the mean and dividing by the standard deviation of the batch of inputs. This normalization is applied independently to each feature dimension (i.e., each channel in a convolutional neural network).
It works on following ways:

1. Compute the mean and variance of the mini-batch:

Given a mini-batch of inputs, batch normalization computes the mean and variance of each feature dimension (i.e., channel in a convolutional neural network). This is done independently for each feature dimension using the following equations:

$$mean = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.3}$$

$$variance = \frac{1}{m}\sum_{i=1}^{m}(x_i - mean)^2 \tag{3.4}$$

Here, m is the number of inputs in the mini-batch and x_i is the i-th input in the mini-batch.

2. Normalize the inputs:

Batch normalization normalizes the inputs by subtracting the mean and dividing by the standard deviation. The normalized input
z_i is computed as follows:

$$z_i = \frac{x_i - mean}{\sqrt{(variance + \epsilon)}} \tag{3.5}$$

where ϵ is a small constant (e.g., 10^-5) added to the denominator for numerical stability.

3.Scale and shift:

Batch normalization introduces two learnable parameters, gamma and beta, which allow the network to learn the optimal scale and shift for each feature dimension. The normalized and transformed input
y_i is computed as follows:

$$y_i = \gamma * z_i + \beta \tag{3.6}$$

where γ and β are learnable parameters that are updated during training via backpropagation.

4.Apply non-linearity: Finally, the transformed input y_i is passed through a non-linear activation function (e.g., ReLU) to produce the output of the batch normalization layer.

Long Short Term Memory (LSTM)

An enhanced Recurrent Neural Network (RNN) and sequential network, called LSTM network, permits information to stay. It is capable of resolving the RNN's vanishing gradient issue. It processes data passing on information as it propagates forward as in RNN. The differences are the operations within the LSTM's cells.These operations are used to allow

the LSTM to keep or forget information. The LSTM have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

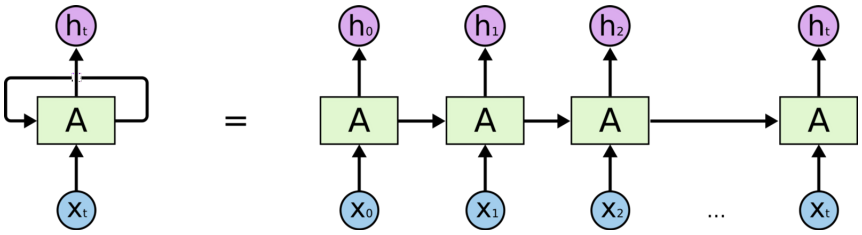


Figure 3.12: Architecture of RNN

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.[Fig. 3.13] shows the architecture of LSTM.

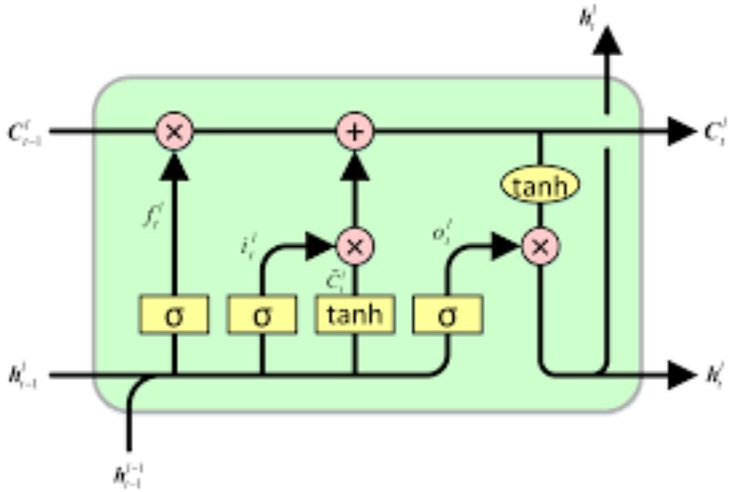


Figure 3.13: Architecture of LSTM

Components of LSTMs

The LSTM cell contains the following components:

- Forget Gate "f" (a neural network with sigmoid).
- Candidate layer "C"(a NN with Tanh)
- Input Gate "I" (a NN with sigmoid)

- Output Gate "O"(a NN with sigmoid)
- Hidden state "H" (a vector)
- Memory state "C" (a vector)

Inputs to the LSTM cell at any step are X_t (current input) , H_{t-1} (previous hidden state) and C_{t-1} (previous memory state). Outputs from the LSTM cell are H_t (current hidden state) and C_t (current memory state).

Working of gates in LSTMs

First, LSTM cell takes the previous memory state C_{t-1} and does element wise multiplication with forget gate (f) to decide if present memory state C_t. If forget gate value is 0 then previous memory state is completely forgotten else f forget gate value is 1 then previous memory state is completely passed to the cell (Remember f gate gives values between 0 and 1).

- $C_t = C_{t-1} * f_t$
- Calculating the new memory state:
  $C_t = C_t + (I_t * C'_t)$
- Now, we calculate the output:
  $H_t = tanh(C_t)$

Dense layer

In a convolutional neural network (CNN), a dense layer (also known as a fully connected layer) is a type of layer that connects every neuron in the previous layer to every neuron in the current layer. Unlike convolutional layers, which only connect neurons in local regions of the input, dense layers can capture global patterns in the input and used at the end of a CNN to produce the final output. In a CNN, the output of the last convolutional layer is often flattened (i.e., reshaped into a 1D vector) before being passed through one or more dense layers. This allows the dense layers to capture global patterns in the input and make a final prediction.

Overall, dense layers in CNNs can help to improve the expressiveness of the network and

make it more capable of capturing complex relationships between inputs and outputs. However, they also increase the number of parameters in the network, which can make it more prone to overfitting if not properly regularized.

3.4.3 Method used for overcoming overfitting

DropOut

Dropout is a regularization technique commonly used in neural networks to prevent overfitting. Overfitting occurs when a model learns to fit the training data too closely, resulting in poor generalization to new, unseen data.

Dropout works by randomly dropping out (setting to zero) a fraction of the neurons in a layer during training. The dropout rate is typically set between 0.1 and 0.5, meaning that each neuron has a probability of being dropped out during each training step. When neurons are dropped out, the remaining neurons have to step up and take over their roles. This encourages the network to learn more robust and general features, rather than relying on specific neurons to always be present. Dropout also prevents neurons from co-adapting, which can lead to overfitting.

During inference, all neurons are used, but their outputs are scaled down by the dropout rate to account for the fact that not all neurons were active during training. By using dropout, a neural network can learn to generalize better and avoid overfitting, which can lead to improved performance on new, unseen data. However, it is important to note that using too much dropout can hurt performance, so it is important to find the right balance between regularization and model capacity.

Early stopping

Early stopping is another regularization technique commonly used in machine learning to prevent overfitting. It works by monitoring the performance of the model on a validation set during training and stopping the training process when the performance on the validation set stops improving.

The basic idea behind early stopping is that the model is likely to be overfitting the training data if its performance on the validation set starts to degrade while the performance on the

training set continues to improve. By stopping the training process at this point, the model is prevented from continuing to fit the noise in the training data and is instead encouraged to generalize better to new, unseen data. To implement early stopping, the training process is typically monitored after each epoch, and the model with the best performance on the validation set is saved. If the performance on the validation set does not improve for a certain number of epochs, the training process is stopped and the saved model is used for inference.In our case, we have used patience(number of epochs with no improvement after which training will be stopped) of 30.

The key advantage of early stopping is that it is easy to implement and can be effective at preventing overfitting, especially when combined with other regularization techniques such as dropout. However, it is important to choose the right stopping criteria and to avoid stopping too early, as this can result in underfitting and poor performance on the validation and test sets.

3.4.4 Model checkpoint

Model checkpointing is a technique used to periodically save the weights and other parameters of a model during training, so that the training process can be resumed from the last saved point if it is interrupted or crashes.The saved model checkpoints can also be used to perform inference on new data or to fine-tune the model on additional data later on.

The main benefit of model checkpointing is that it allows you to save time and resources by avoiding the need to restart the training process from scratch if it is interrupted or if you want to resume training from a certain point. It can also be used to compare the performance of different models trained on the same data, as well as to perform model selection by choosing the model checkpoint with the best performance on a validation set. In our case, we have used saveBestOnly=True, it only saves when the model is considered the "best" and the latest best model is considered on the basis of minimum validation loss.

3.4.5 Loss function

We have used Negative log likelihood(NLL) as our loss function.It measures the error between the predicted probability distribution and the actual probability distribution of the target variable. Intuitively, the NLL loss function penalizes the model more heavily when it makes

confident but incorrect predictions, and less heavily when it makes uncertain predictions or correct predictions. This makes it a suitable loss function for models that are designed to output probabilities. It can be mathematically given as for normal distribution as:

$$NLL = -\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)\right) \tag{3.7}$$

where, σ = standard deviation and μ = mean

3.4.6 Optimizer

We have used Nadam as our optimizer.Nadam stands for "Nesterov-accelerated Adaptive Moment Estimation".It combines the benefits of two other optimization methods: Nesterov accelerated gradient (NAG) and Adam. Like NAG, Nadam uses a momentum term to accelerate the gradient descent process and overcome the problem of oscillation and slow convergence in regions with high curvature. However, Nadam also incorporates the adaptive learning rate and bias correction techniques of Adam, which allow for efficient learning rates that adapt to the shape of the loss surface and scale the learning rates based on the running averages of the first and second moments of the gradients.

Nadam performs well in a variety of deep learning applications, particularly in models with large amounts of data or complex loss surfaces. It has been shown to converge faster and achieve better generalization than other optimization methods, such as Adagrad and RMSprop.The update rule for Nadam can be described as follows:

1. Compute the gradient of the loss function with respect to the model parameters.

2. Compute the exponentially decaying average of the first and second moments of the gradients.

3. Correct the first and second moments for bias.

4. Compute the Nesterov accelerated gradient using the corrected first moment and momentum term.

5. Update the model parameters using the Nesterov accelerated gradient and the learning rate.The update rule is of the form:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\left(\beta_1\hat{m}_t + \frac{(1-\beta_t)g_t}{1-\beta_1^t}\right) \tag{3.8}$$

4. System design

[Fig. 4.1] shows the general workflow of our system. Firstly, we collect the data from PPG to determine the spectral peaks. As the signal mostly contains noisy signal, so it is combined with the signal obtained from accelerometer to remove the MA effect. The available PPG and accelerometer signals undergoes preprocessing steps which includes filtering(Band pass filter), resampling and normalization to obtain fine data for feeding to our model. Then obtained signals are separately applied to our deep learning architecture which includes 1D-convolution operation, pooling and batch normalization, later obtained ppg and accereometer signals are merged together and feed to our LSTM for finalizing our our training part. After training phase, we cross-validate(LOSO cross-validation) our model in order to obtain more accurate heart rate prediction. During prediction, mean provides predicted heart rate and variance provides uncertainity in prediction.

4.1 Software Requirement

The software tools used in our project include TensorFlow, NumPy, SciPy, and Pandas. TensorFlow is an open-source machine learning framework widely used for building deep learning models. NumPy is a Python library for numerical computing, providing support for arrays, matrices, and mathematical functions. SciPy is a scientific computing library that offers tools for signal processing, optimization, and statistical analysis. Pandas is a library for data manipulation and analysis, providing data structures and functions for cleaning, exploring, and transforming data. Together, these tools provide a powerful and flexible environment for implementing and testing deep learning models. We used kaggle for training our model.

Tensorflow	Numpy	Pandas
Sklearn	Matplotlib	Seaborn
Scipy	Kaggle	keras

Table 4.1: Software tools

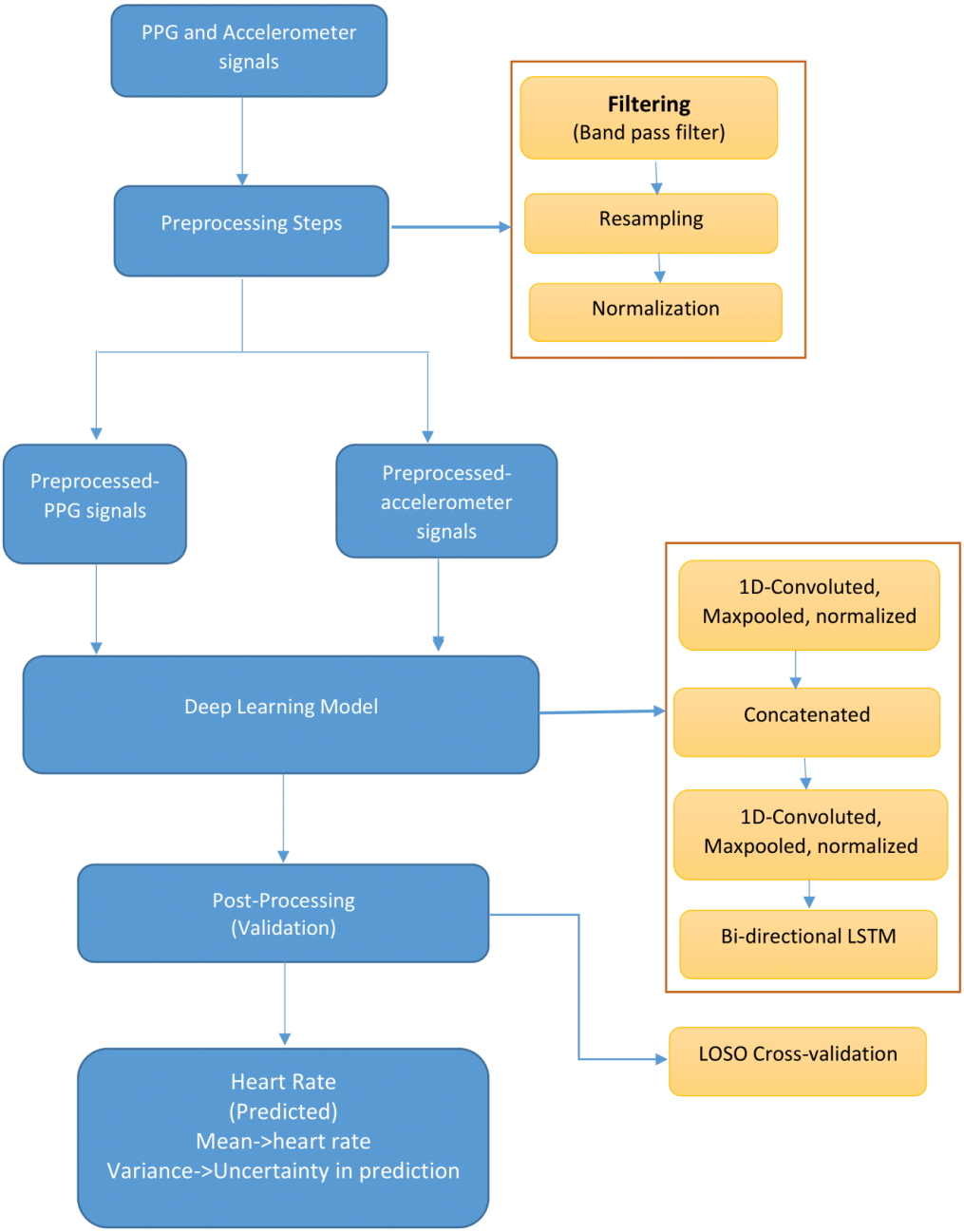


Figure 4.1: System Workflow

5. Results and Discussion

The approach and model that we adopted for the project must be evaluated to determine its performance on unseen data. It must be able to estimate the heart rate not only on the data it was trained on, but also on the new data. There are various different techniques for model evaluation. We have used leave-one-session-out (LOSO) cross-validation[13].

5.1 Leave-One-Session-Out Cross Validation (LOSO CV)

It is a model evaluation technique where parameter optimisation is performed on all data except of one session, and the left-out session is used as test data. This procedure is repeated so that each session is used as test data exactly once. Thus, if a data set has adequate variety, results reported with LOSO cross-validation can reflect the generalisation capabilities of the developed algorithms.

The reason for using LOSO cross-validation is mainly due to the fact that the optimisation of model parameters to a specific subject or even a specific session is not possibly useful in daily life. It should perform well in all situations. For heart rate estimation a single subject is left out in a session for testing purpose and remaining data is used for training and validation. [Fig. 5.1] shows the concept of LOSO cross-validation

Sessions \ Subjects										
1	S_1	S_2	S_3	..	..	..	..	S_{n-2}	S_{n-1}	S_n
2	S_1	S_2	S_3	..	..	..	..	S_{n-2}	S_{n-1}	S_n
.	.								.	.
.	.								.	.
.	.								.	.
$n-1$	S_1	S_2	S_3	..	..	..	..	S_{n-2}	S_{n-1}	S_n
n	S_1	S_2	S_3	..	..	..	..	S_{n-2}	S_{n-1}	S_n

Figure 5.1: LOSO cross-validation

5.2 Evaluation Metrics

Evaluation metrics are used to measure the quality and performance of the machine learning model. Evaluating machine learning models is essential for any project. There are many different types of evaluation metrics available to test a model. The problem we are dealing is a Regression problem so, the metrics for regression are used particularly Mean Absolute Error(MAE)

5.2.1 Mean Absolute Error (MAE)

MAE is a model evaluation metric used with regression models. Absolute Error is the amount of error in the measurements. It is the absolute difference between the true value and predicted value i.e. the true heart rate recorded from ECG and predicted heart rate by the model. To obtain the overall MAE, the MAE of individual sessions were averaged

It is mathematically represented as:

$$MAE = \frac{1}{S}\sum_{s=1}^{S}\Big(\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|\Big) \tag{5.1}$$

$$where, S = number\,of\,sessions$$
$$n = number\,of\,samples$$
$$y = True\,value$$
$$\hat{y} = Predicted\,value$$

5.2.2 Uncertainty

In the context of machine learning, uncertainty refers to the lack of confidence in one's model estimates.Uncertainty is an important consideration in many machine learning applications, and in particular, it may be crucial to quantify the amount of uncertainty for any given prediction referred to as predictive uncertainty. When high stakes decisions are being made based on predictions of machine learning models, e.g. in health care, it is vital to know how much confidence to have in the prediction. Since, our application domain is health care for the estimation of heart rate, the consideration of uncertainty is vital.

Predictive uncertainty refers to the uncertainty in a prediction made about some target variable of interest. Predictive uncertainty exists whenever one has uncertainty in making a given prediction, and to express this uncertainty, we have made distributional predictions,

instead of point predictions.Therefore, instead of predicting that the heart rate of person is 128 BPM(a point prediction), our model predicts that the heart rate is approximately distributed according to a Gaussian distribution, with mean of 128 BPM and standard deviation of 1.2 BPM.

The uncertainty can be divided into two categories: aleatoric uncertainty (inherent uncertainty of the system) and epistemic uncertainty (uncertainty about the choice of model).

Aleatoric Uncertainty

Aleatoric uncertainty is a type of uncertainty that arises from the randomness or variability of a system or phenomenon because of information that cannot be measured (i.e. noise). It is also known as stochastic uncertainty or data uncertainty.When this noise is present, aleatoric uncertainty cannot be eliminated even by more data or knowledge and even if the number of samples collected tends towards infinity. We can assume the aleatoric uncertainty, the inherent randomness, to be either constant (homoscedastic) or variable (heteroscedastic), as a function of the input explanatory variables. Here, we have considered heteroscedastic aleatoric uncertainty.

It can be mathematically represented as:

$$u_a(x_i) = \frac{1}{T}\sum_{t=1}^{T} \hat{\sigma}_{i,t}^2 \tag{5.2}$$

$$where, u_a(x_i) = aleatoric\, uncertainty$$
$$T = number\, of\, predictions$$
$$\hat{\sigma} = predicted\, standard\, deviation$$
$$x = input\, window$$

Epistemic Uncertainty

Epistemic uncertainty refers to the uncertainty of the model and is often due to a lack of training data. Epistemic uncertainty is the uncertainty that comes from being unsure about one's model choice. Epistemic uncertainty of a trained model will decrease as the size of training data increases and might also be affected by the suitability of model architecture.As

opposed to aleatoric uncertainty, epistemic uncertainty can in principle be reduced on the basis of additional information. If one is doing modelling with a neural network and is given a finite number of samples to train on, the uncertainty of what the weights in the network should be is epistemic uncertainty. However, as the number of samples being trained on tends to infinity, the epistemic uncertainty tends towards zero as the correct model is able to be identified.

It can be mathematically represented as:

$$u_e(x_i) = \frac{1}{T}\sum_{t=1}^{T}\mu_{i,t}^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\mu_{i,t}\right)^2 \tag{5.3}$$

$$where, u_e(x_i) = epistemic\, uncertainty$$

$$T = number\, of\, predictions$$

$$\mu = predicted\, mean\, value$$

$$x = input\, window$$

[Fig. 5.2] shows the Aleatoric and Epistemic Uncertainty for simple hypothetical data. We can see for values between -3 and -2 there is high aleatoric uncertainty and is low for range about 2.5 to 3.5. The Epistemic uncertainty is high where the data points are not present in the traing data. The model is highly uncertain in this region and its prediction is merely a random guess.

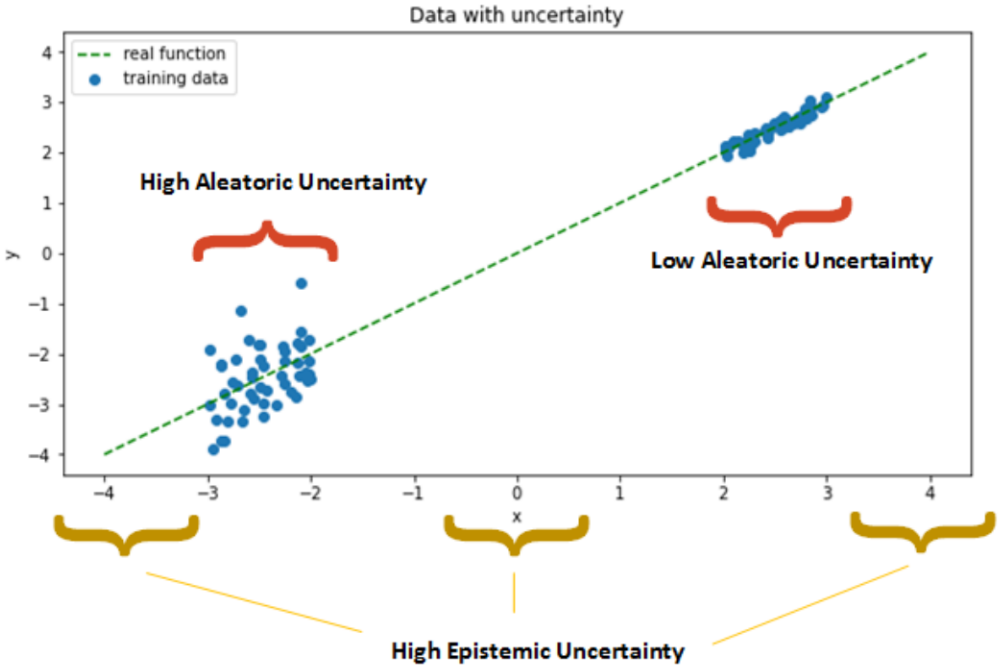


Figure 5.2: Uncertainty

Sharpness

Sharpness is a measure of how narrow, concentrated, or peaked the predictive distribution is. Sharpness is evaluated solely based on the predictive distribution, and neither the datapoint nor the ground truth distribution are considered when measuring sharpness.Sharpness is a valuable property in the predictive distribution as sharper the distributional, more is the model confident in its predictions.

[Fig. 5.3] shows, two Gaussian distributional predictions, one with mean 0 and variance 0.1 and another with mean 0 and variance 1. First distribution is sharper than a second with mean 0 and variance 1 and hence first is more confident than the second one.

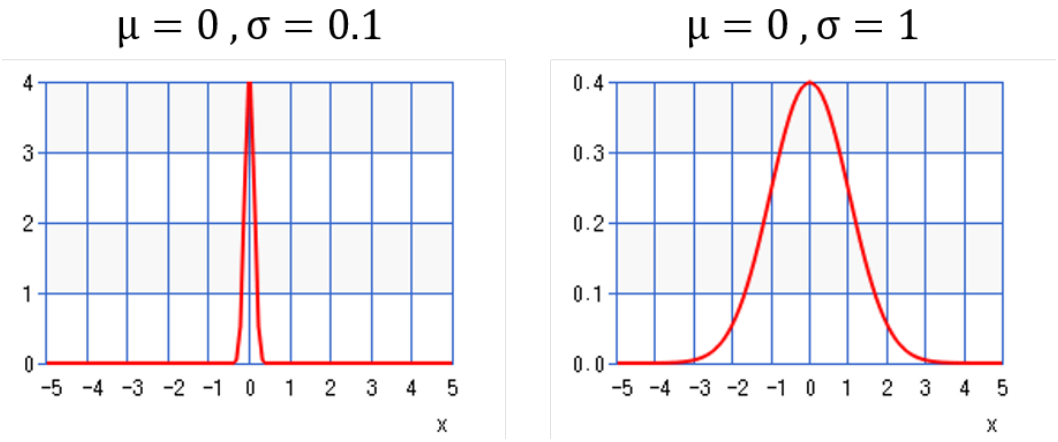


Figure 5.3: Sharpness

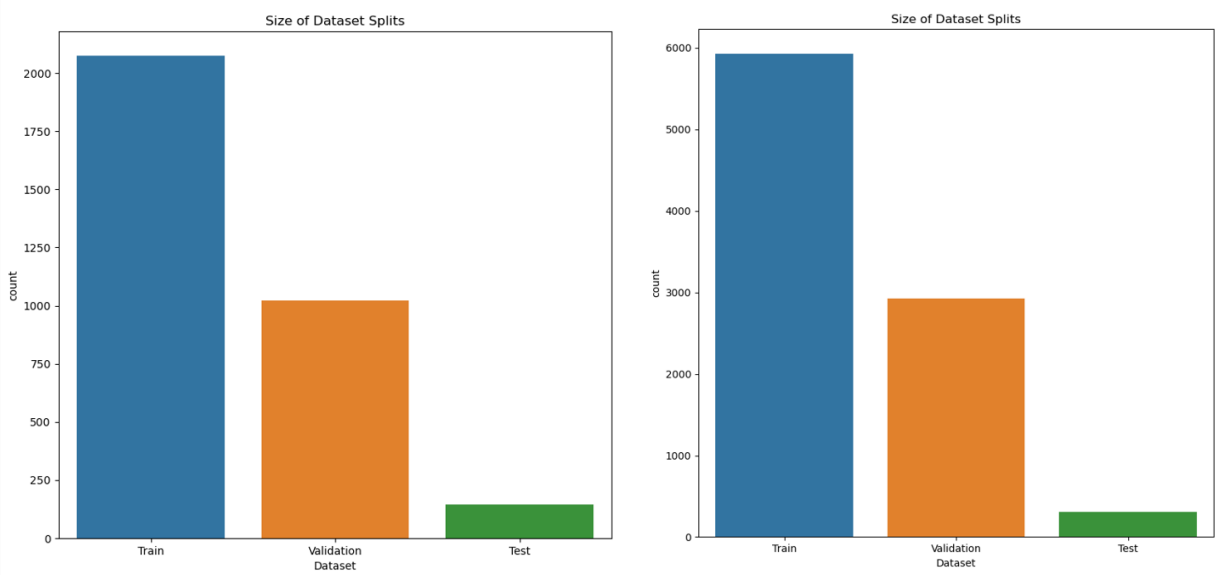
5.3 Model Performance

We have trained and evaluated our model on two publicly available datasets; IEEE SPC dataset and BAMI II dataset. The performance of our model was comparable to State of the Art techniques with fewer number of parameters and model size.

As the model is trained and evaluated using LOSO CV, the performance of model for different folds were evaluated. The performance on BAMI dataset was a little lower as compared to IEEE dataset

5.3.1 Dataset distribution for particular fold

[Fig. 5.4a] and [Fig. 5.4b] shows the amount of datasets in each of the training, validation and test sets for a particular fold for IEEE and BAMI dataset respectively. It shows that



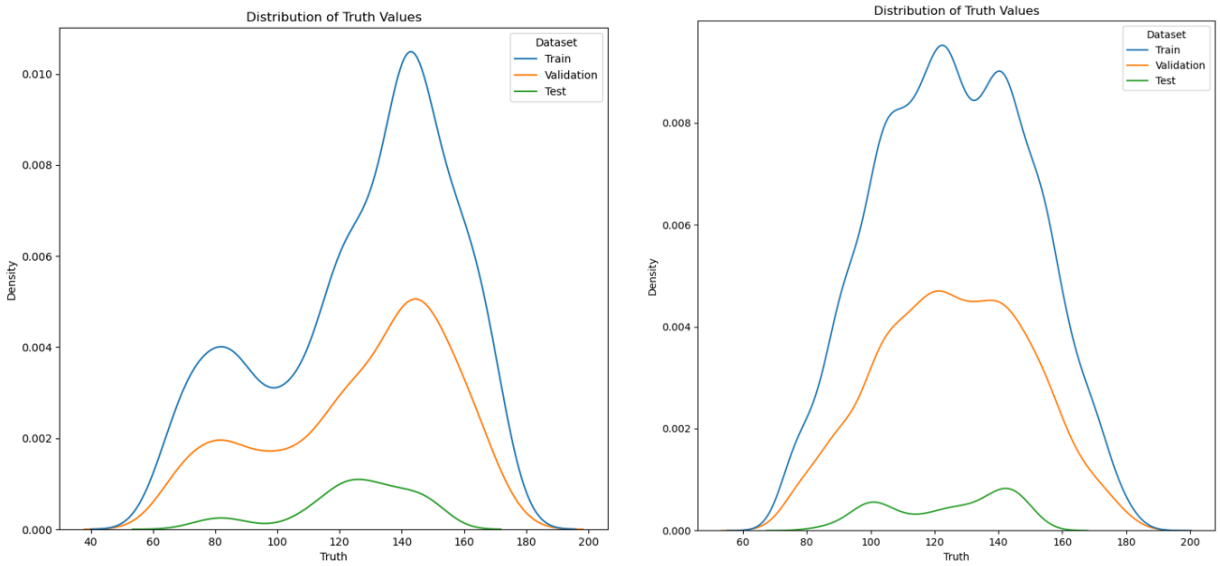
(a) IEEE (b) BAMI

Figure 5.4: Dataset split proportions

roughly around 2,200 windows are present in training set, about 1,000 in validation set and about 1,00 in test set for IEEE and roughly around 5,800 windows are present in training set, about 3,000 in validation set and about 1,00 in test set for BAMI. Each window consists of 8s data with 512 samples. Stratified sampling was performed for partitioning the data into train and validation sets so as to preserve the original distribution of data.

5.3.2 Distribution of true Heart Rate

[Fig. 5.5a] and [Fig. 5.5b] shows the distribution of ground truth value of Heart Rate (HR) in each of the training, validation and test sets for a particular fold for IEEE and BAMI dataset respectively. It shows that the distribution is similar for train set and validation set. This is mainly due to the use of stratified sampling for partitioning the data into train and validation sets so as to preserve the original distribution of data for the validation.

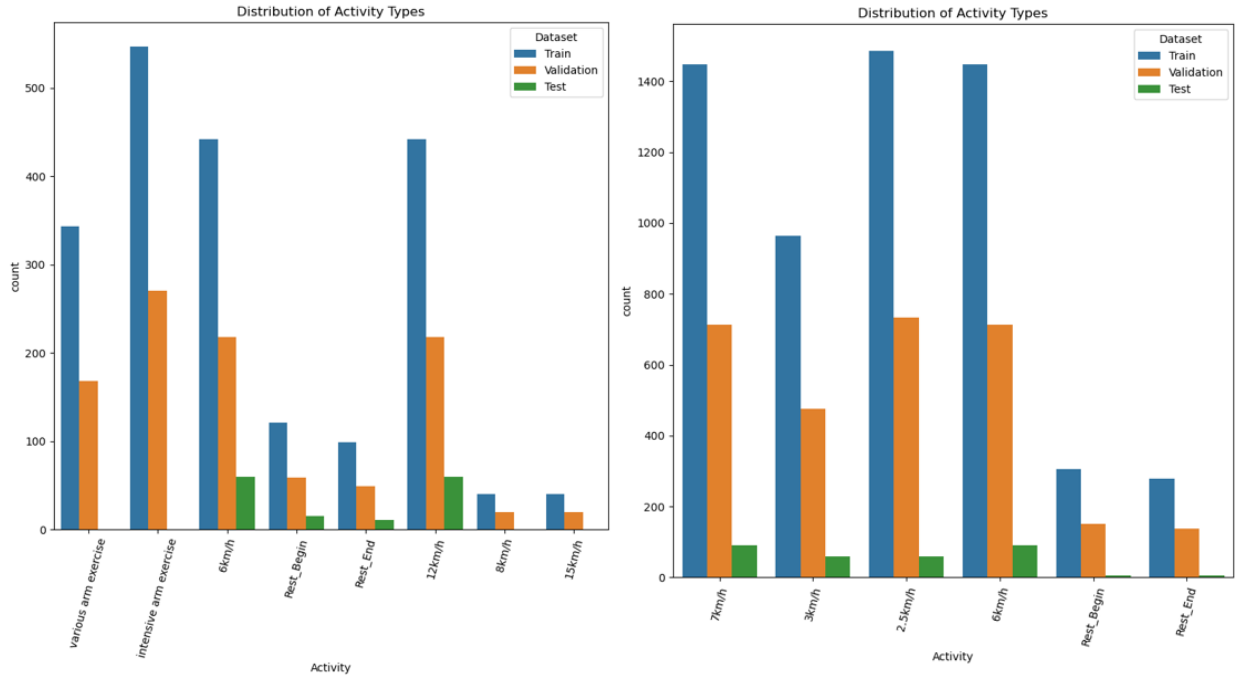


(a) IEEE (b) BAMI

Figure 5.5: True HR distribution

5.3.3 Distribution as per activity

[Fig. 5.6a] and [Fig. 5.6b] shows the distribution of dataset as per the activity done during data collection for training, validation and test sets for a particular fold for IEEE and BAMI dataset respectively. It shows that the activities various arm exercise, intensive arm exercise, running on treademill at 8 Km/hr and 15 Km/hr donot have any data in test test for IEEE.This is due to LOSO CV where a particular subject selected for testing doesn't have data for the mentioned activites. For BAMI all activities are included in all sets.



(a) IEEE (b) BAMI

Figure 5.6: Activity types distribution

Learning Curve

[Fig. 5.7] and [Fig. 5.8] shows the learning curve for the dataset IEEE and BAMI respectively. It shows that the MAE decreases with the increase in the number of epochs for both training and validation set. For IEE, the minimum MAE for validation set is obtained at epoch number 128 where the model saved and used for the evaluation of test set. On evaluation at test set it gave MAE of 1.686. Similarly, for BAMI the minimum MAE for validation set is obtained at epoch number 77 where the model saved and used for the evaluation of test set. On evaluation at test set it gave MAE of 1.782.

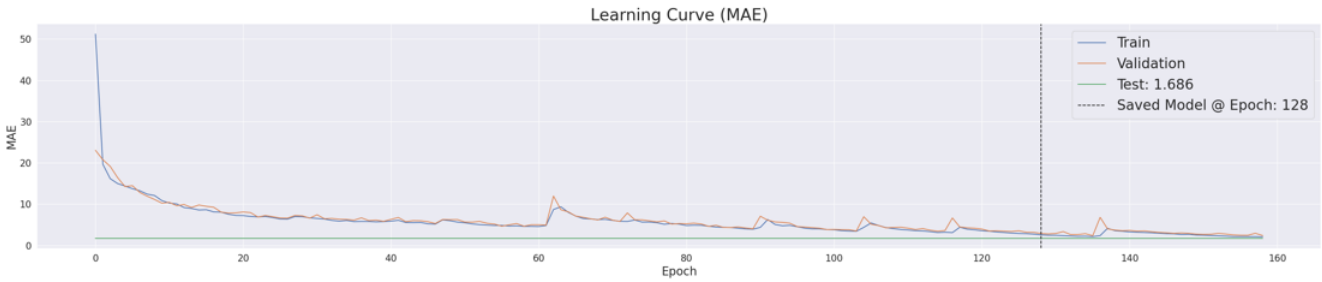


Figure 5.7: Learning Curve(MAE) IEEE

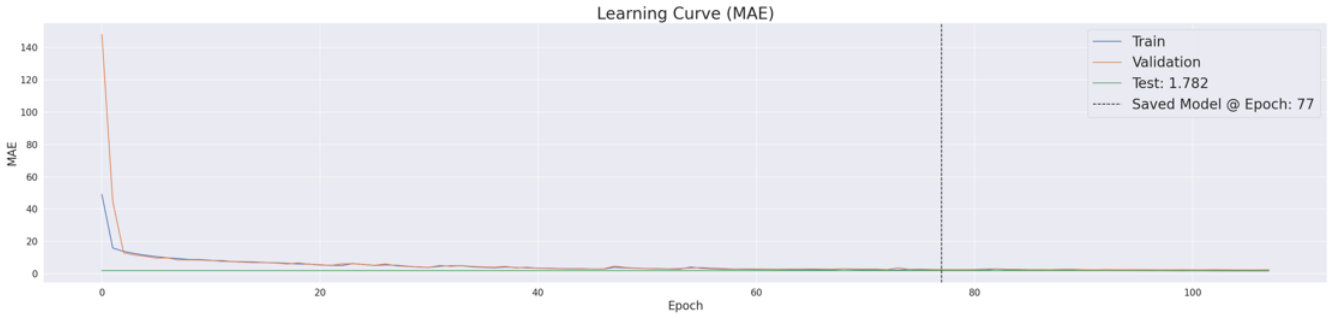
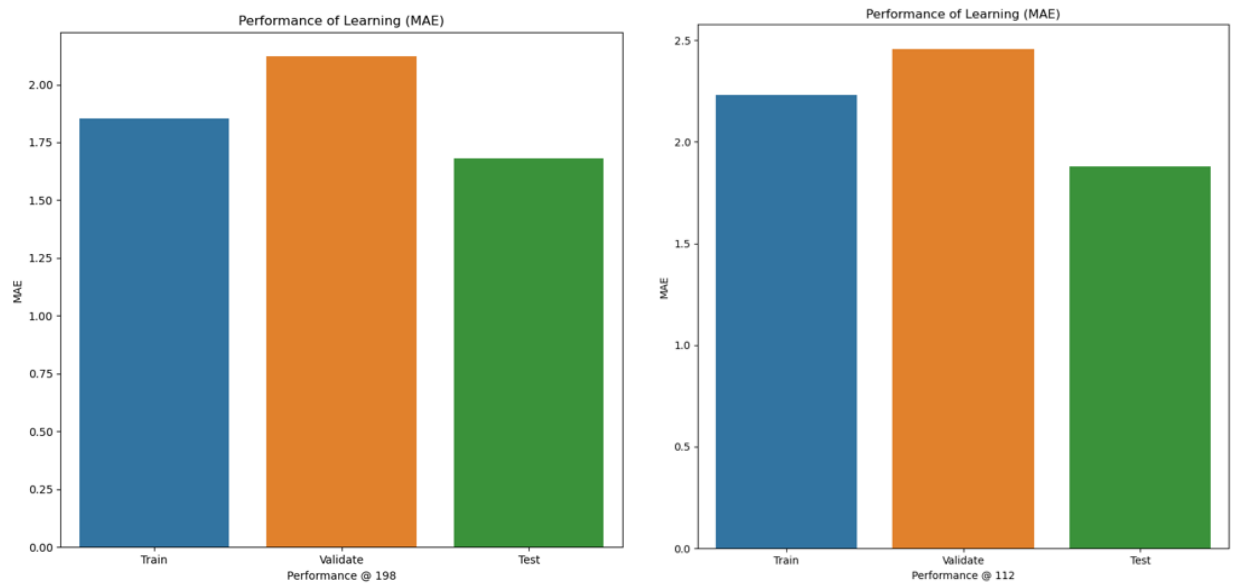


Figure 5.8: Learning Curve(MAE) BAMI

[Fig. 5.9a] and [Fig. 5.9b]shows the count plot of MAE and different sets of data for IEEE and BAMI dataset. It shows that the MAE on train set is about 1.8, validation set is about 2.2 and on test set is 1.68 for IEEE.Similarly, for BAMI the MAE on train set is about 2.2, validation set is about 2.4 and on test set is 1.88. From the count plot, we can see model has similar performance on all three sets i.e. it doesn't overfit or underfit the dataset.



(a) IEEE (b) BAMI

Figure 5.9: MAE as per dataset

Loss function

[Fig. 5.10] and [Fig. 5.11] shows the loss for the dataset against number of epochs for IEEE and BAMI respectively. It shows that the Loss decreases with the increase in the number of epochs for both training and validation set. For IEE, the minimum Loss for validation set is obtained at epoch number 128 where the model saved and used for the evaluation of test

set. On evaluation at test set it gave Loss of 1.725. Simialrly, for BAMI the minimum Loss for validation set is obtained at epoch number 77 where the model saved and used for the evaluation of test set. On evaluation at test set it gave Loss of 1.825.

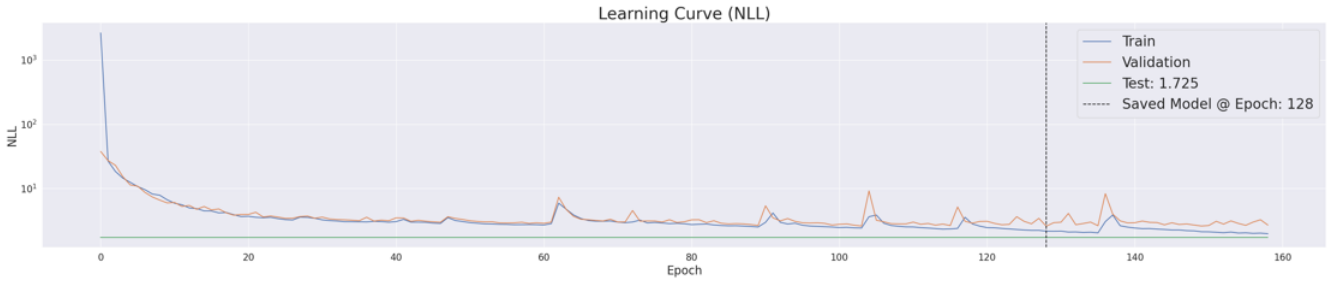


Figure 5.10: Learning Curve(NLL) IEEE

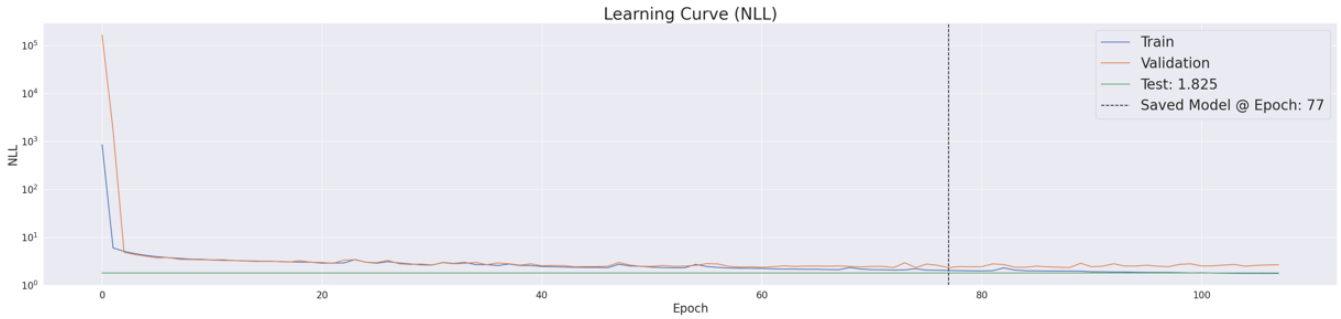
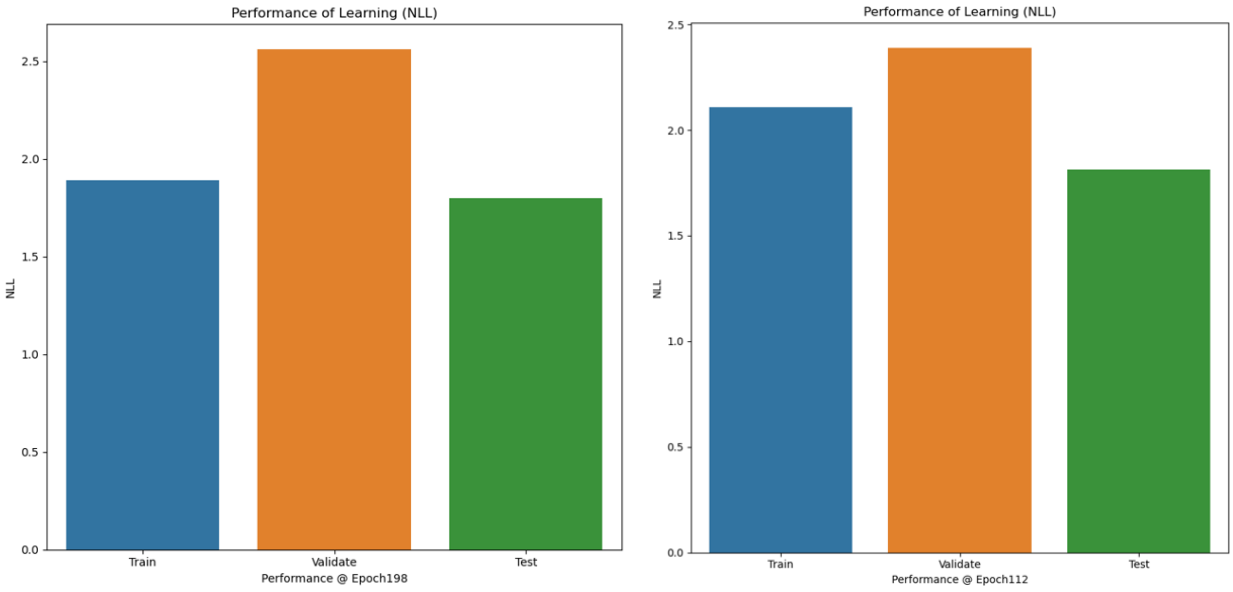


Figure 5.11: Learning Curve(NLL) BAMI

[Fig. 5.12a] [Fig. 5.12b] and shows the count plot of Loss and different sets of data for IEEE and BAMI respectively. It shows that the Loss on train set is about 1.8, validation set is about 2.6 and on test set is 1.799 for IEEE and about 2.1 on train set, about 2.4 on validation set and 1.813 on test set for BAMI. From the count plot, we can see model has similar performance on all three sets.

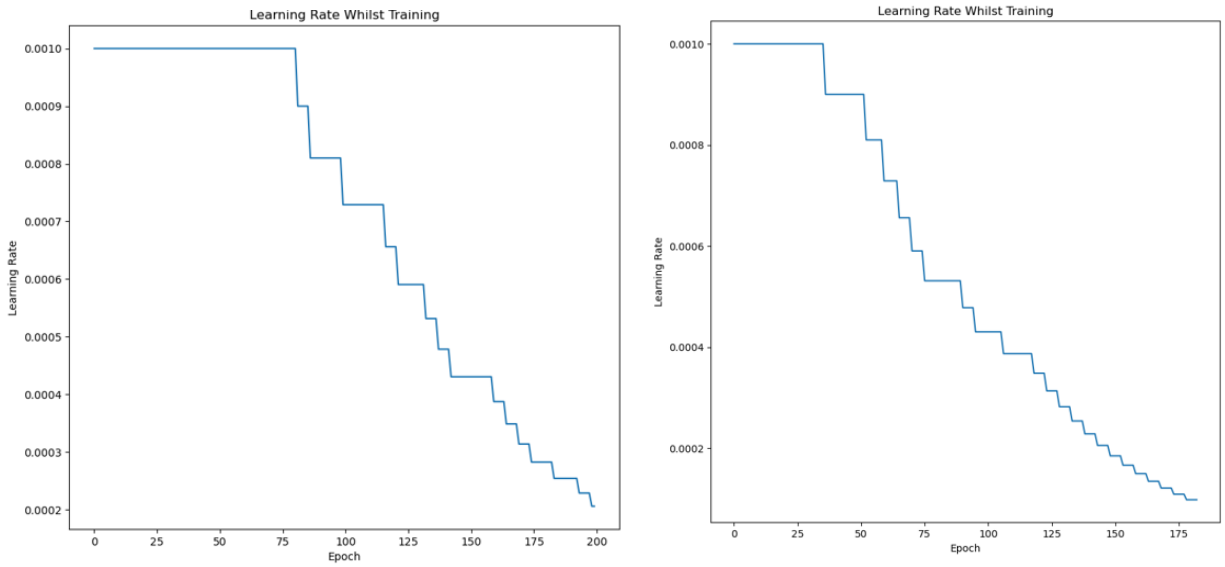


(a) IEEE (b) BAMI

Figure 5.12: NLL as per dataset

Learning rate

[Fig. 5.13a] [Fig. 5.13b] shows the change in learning rate as number of epochs increases for IEEE and BAMI respectively. Reducing the learning rate on a plateau is a common technique used in deep learning to improve the performance of a model during training. The idea is to gradually decrease the learning rate when the model stops making significant progress in reducing the loss on the training data.Here, we have started the learning rate from 0.001 which is gradually decreases.The learning rate is reduced by of 0.9 if the Loss doesn't improve for 5 epochs.

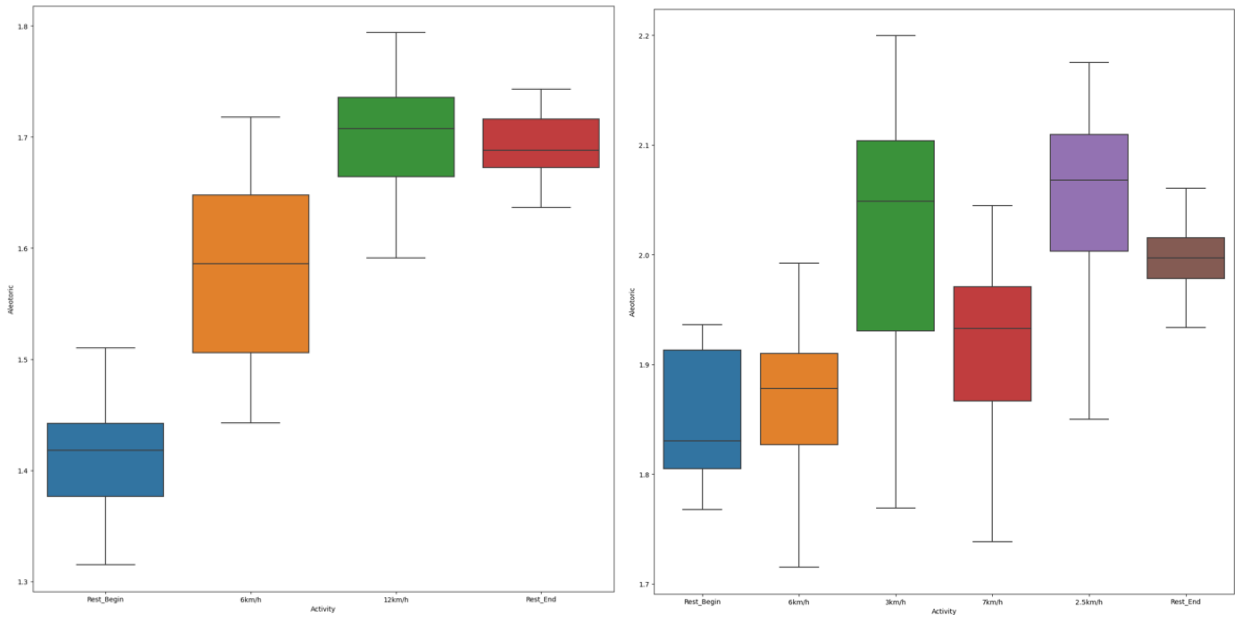


(a) IEEE (b) BAMI

Figure 5.13: Learning Rate while training

Aleatoric uncertainty

[Fig. 5.14a] and [Fig. 5.14b]shows box plot of the aleatoric uncertainty for the test set for IEEE and BAMI respectively. It is the uncertainty that exists inherently in the dataset and can't be removed. From the plot we can see that running on treadmill at 12km/hr activity has high amount of uncertainty as compared to other activities for IEEE and 2.5km/hr in case of BAMII. The least amount of uncertainty is on rest begin for both IEEE and BAMI.

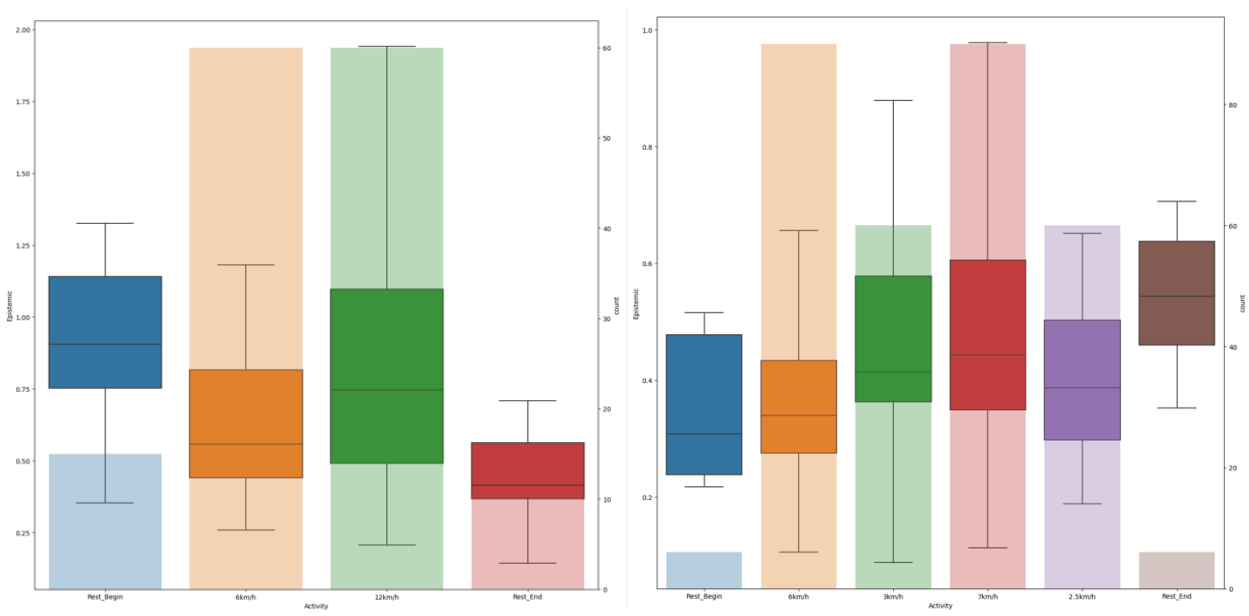


(a) IEEE (b) BAMI

Figure 5.14: Aleatoric uncertainty

Epistemic Uncertainty

[Fig. 5.15a] and [Fig. 5.15b] shows box plot of the epistemic uncertainty for the test set for IEEE and BAMI respectively. It is the uncertainty that exists due to the model and can be reduced. The shaded region of the plot represents the band of epistemic uncertainty.

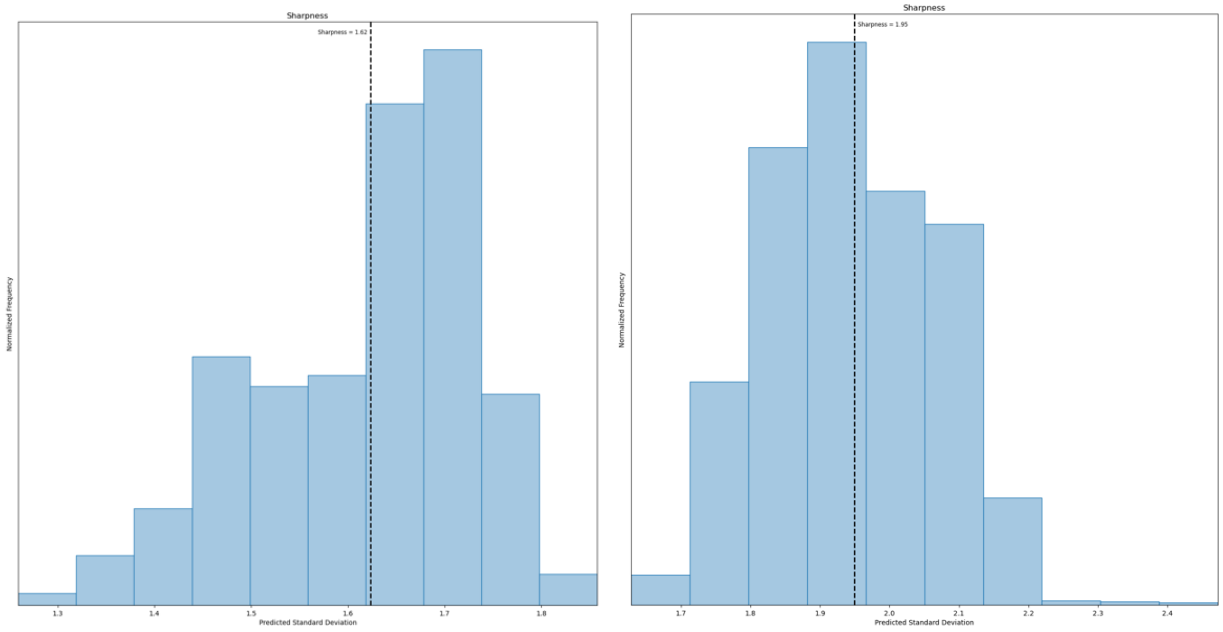


(a) IEEE (b) BAMI

Figure 5.15: Epistemic uncertainty

Sharpness

[Fig. 5.16a] and [Fig. 5.16b] shows the sharpness of the prediction for IEEE and BAMI respectively. The sharp is the prediction, the more confident a model is in its prediction. From the plot we can see that the mean sharpness is 1.62 for IEEE and 1.95 for BAMI.



(a) IEEE (b) BAMI

Figure 5.16: Sharpness

Overall Performance

For IEEE dataset, [Fig. 5.18] shows the MAE for overall sessions which is 1.3 BPM. Similarly, [Fig. 5.18] shows MAE for all session as per the activity. The mean MAE for activities is 1.27 BPM.

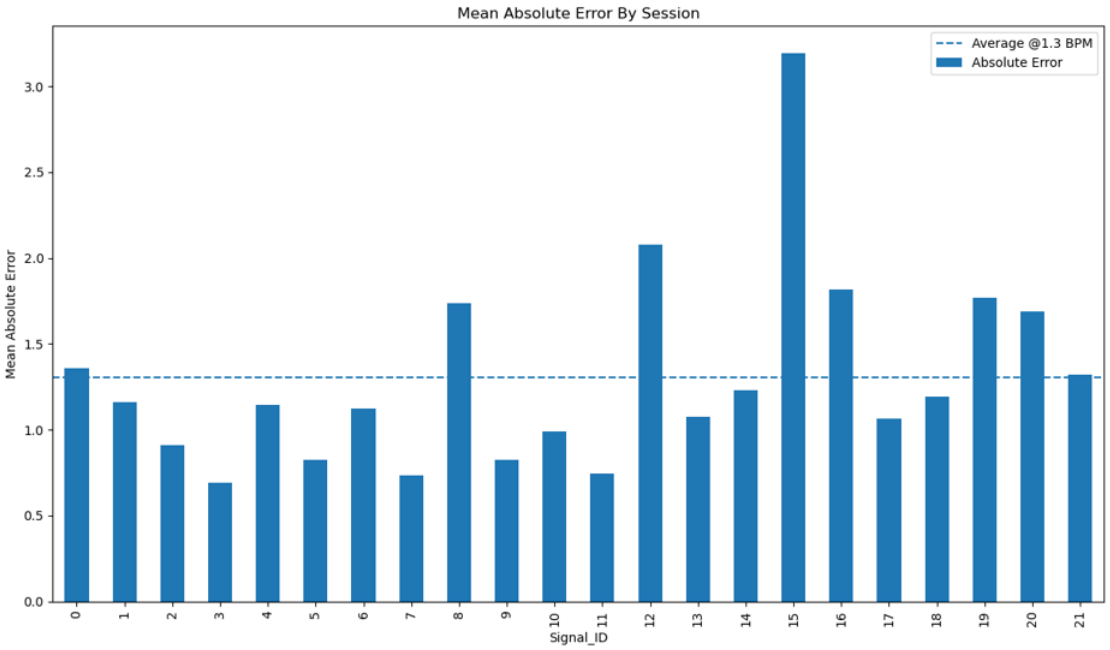


Figure 5.17: MAE by session IEEE

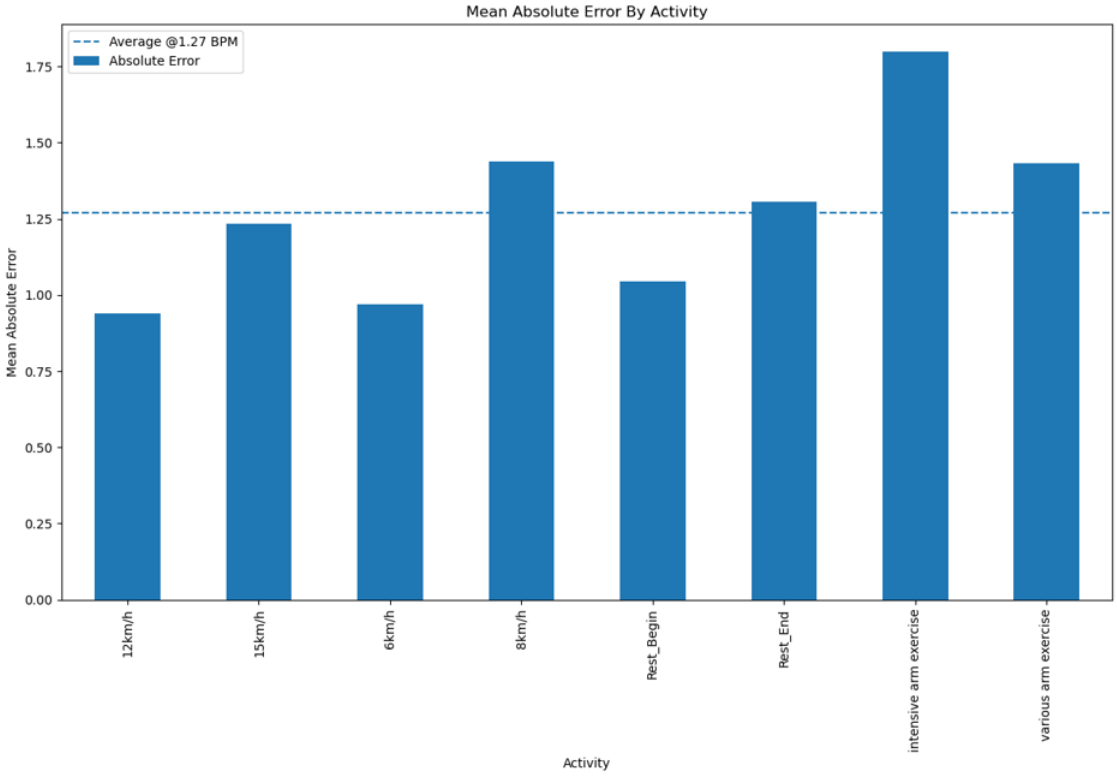


Figure 5.18: MAE by Activity IEEE

For BAMI dataset, [Fig. 5.19] shows the MAE for overall sessions which is 1.54 BPM. Similarly, [Fig. 5.20] shows MAE for all session as per the activity. The mean MAE for

activities is 1.69 BPM.

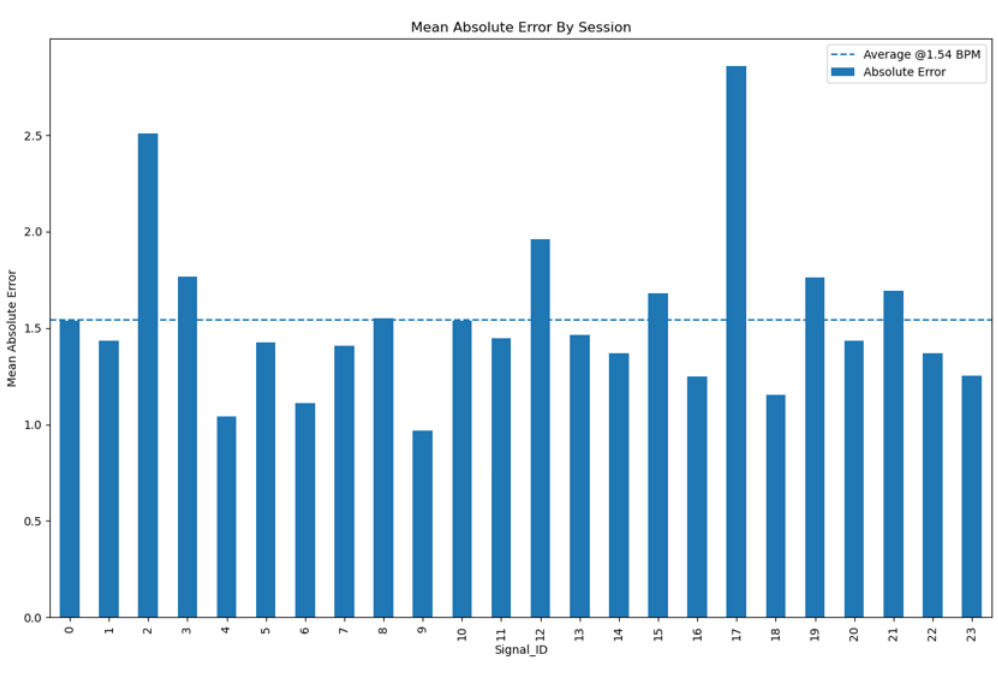


Figure 5.19: MAE by session BAMI

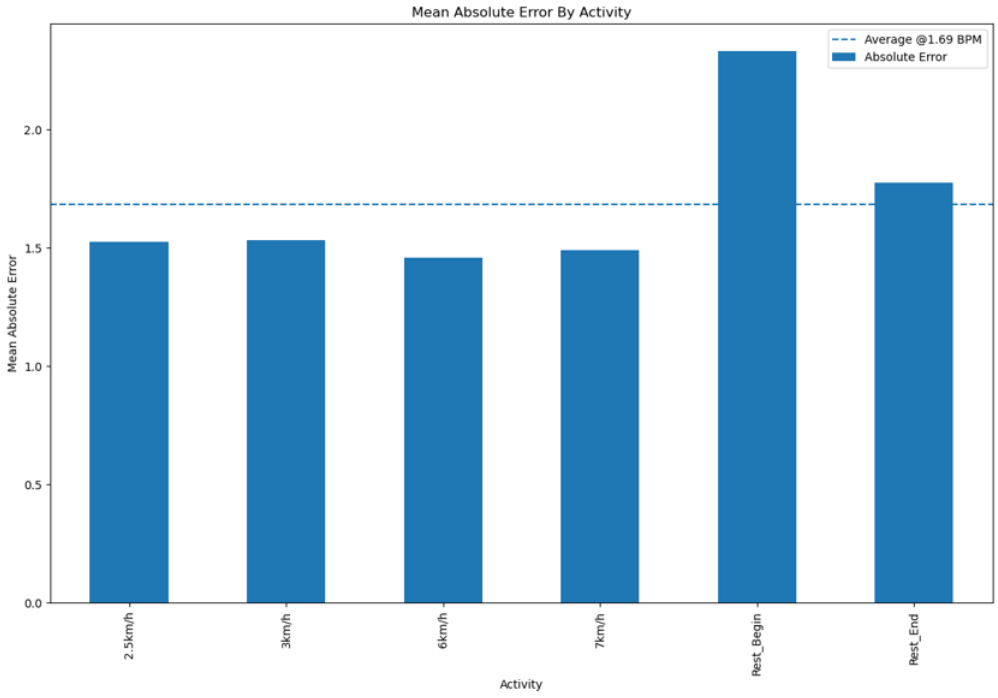


Figure 5.20: MAE by Activity BAMI

5.4 Result Comparison

According to the [Table 5.1] provided, Our model performs well compared to other methods for pulse rate estimation on both IEEE and BAMI datasets. Our model has an MAE value of 1.30 on the IEEE dataset and 1.54 on the BAMI dataset which is comparable to the MAE values of DeepPulse and Chung et al., the two best performing methods on both datasets.Compared to other methods, Our model has the advantage of using only a single PPG signal, smaller number of parameters and a low sampling rate, which can reduce implementation costs and make the system more efficient.

Methods	Datasets	
	IEEE	BAMI
Deep PPG	4.00 ±5.40	N/A
CorNET (LOSO)	4.67 ±3.71	6.61 ±5.35
Binary CorNET	6.20 ±4.95	N/A
PPGnet	3.36 ±4.10	12.48 ±14.45
Chung et al.	**0.67 ±0.50**	**1.46 ±1.23**
DeepPulse	1.16 ±0.47	1.65 ±0.40
Our model	1.30 ±0.56	1.54 ±0.42

Table 5.1: Comparison of MAE among different models

6. Conclusion

We successfully implemented deep learning approach to predict HR from PPG signal along with accelerometer signal. The result shows that our approach is comparatively better than other convention approach. We have used only one PPG signal and a low sampling rate can be an effective way to reduce the cost of implementing a pulse rate estimation system. Traditionally, pulse rate estimation systems used multiple PPG signals and high sampling rates to capture detailed information about the pulse waveforms. However, using multiple PPG sensors and high sampling rates can be expensive and may require complex signal processing algorithms to extract meaningful information from the data. The model shows the good result with less than 300 thousands network parameters. These results indicate the potential of this deep learning approach in monitoring heart rate using PPG signals, which could have important implications in healthcare and wearable technology.

7. Limitations and Future Enhancements

7.1 Limitations

The limitations of our project are:

- Limited dataset: There is limited dataset used for training and testing the model. Some of these sources such as skin tone, skin temperature, age, sex and BMI have not been fully considered in the datasets used. Additionally, the dataset only covers a limited range of motions and exercises, which may not be representative of all possible scenarios in real-life use. As a result, the generalizability of the model may be limited.
- Challenge in Real-time monitoring: Next limitation is its hardware implementation and computational time, which can be challenging in real-time monitoring applications.
- Inherent Limitation: It shows lower accuracy for heart rate detection in individuals with darker skin tones due to the inherent limitations of photoplethysmography technology.

7.2 Future Enhancements

Based on the findings and limitations of this study, several avenues for future research can be pursued. Firstly, the development of a larger dataset covering a wider range of motions and exercises can help improve the accuracy and generalizability of the model. Additionally, exploring different deep learning architectures and optimization techniques can further enhance the performance of the model. Furthermore, future research can focus on the implementation of the model on portable and low-power hardware devices for real-time heart rate monitoring applications.

References

[1] Dwaipayan Biswas et al. "CorNET: Deep learning framework for PPG-based heart rate estimation and biometric identification in ambulant environment". In: *IEEE transactions on biomedical circuits and systems* 13.2 (2019), pp. 282–291.

[2] Heewon Chung et al. "Deep learning for heart rate estimation from reflectance photoplethysmography with acceleration power spectrum and acceleration intensity". In: *Ieee Access* 8 (2020), pp. 63390–63402.

[3] Elisabetta De Giovanni et al. "Ultra-low power estimation of heart rate under physical activity using a wearable photoplethysmographic system". In: *2016 Euromicro Conference on Digital System Design (DSD)*. IEEE. 2016, pp. 553–560.

[4] Loc Nguyen Huynh, Rajesh Krishna Balan, and Youngki Lee. "Deepsense: A gpu-based deep convolutional neural network framework on commodity mobile devices". In: *Proceedings of the 2016 workshop on wearable systems and applications*. 2016, pp. 25–30.

[5] Shahid Ismail, Imran Siddiqi, and Usman Akram. "Heart rate estimation in PPG signals using Convolutional-Recurrent Regressor". In: *Computers in Biology and Medicine* 145 (2022), p. 105470.

[6] Delaram Jarchi and Alexander J Casson. "Description of a database containing wrist PPG signals recorded during physical exercise with both accelerometer and gyroscope measures of motion". In: *Data* 2.1 (2016), p. 1.

[7] A Reşit Kavsaoğlu, Kemal Polat, and M Recep Bozkurt. "A novel feature ranking algorithm for biometric recognition with PPG signals". In: *Computers in biology and medicine* 49 (2014), pp. 1–14.

[8] Rajet Krishnan, Balasubramaniam Natarajan, and Steve Warren. "Two-stage approach for detection and reduction of motion artifacts in photoplethysmographic data". In: *IEEE transactions on biomedical engineering* 57.8 (2010), pp. 1867–1876.

[9] Inho Lee et al. "Systematic review on human skin-compatible wearable photoplethysmography sensors". In: *Applied Sciences* 11.5 (2021), p. 2313.

[10] Jermana L Moraes et al. "Advances in photopletysmography signal analysis for biomedical applications". In: *Sensors* 18.6 (2018), p. 1894.

[11] Daniel Ray, Tim Collins, and Prasad Ponnapalli. "Deep Neural Network Architecture Search for Wearable Heart Rate Estimations." In: *Studies in health technology and informatics* 281 (2021), pp. 1106–1107.

[12] Daniel Ray, Tim Collins, and Prasad VS Ponnapalli. "DeepPulse: An Uncertainty-aware Deep Neural Network for Heart Rate Estimations from Wrist-worn Photoplethysmography". In: *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE. 2022, pp. 1651–1654.

[13] Attila Reiss et al. "Deep PPG: large-scale heart rate estimation with convolutional neural networks". In: *Sensors* 19.14 (2019), p. 3079.

[14] Toshiyo Tamura et al. "Wearable photoplethysmographic sensors—past and present". In: *Electronics* 3.2 (2014), pp. 282–302.

[15] Jie Tang et al. "Enabling deep learning on IoT devices". In: *Computer* 50.10 (2017), pp. 92–96.

[16] Jiping Xiong et al. "SVM-based spectral analysis for heart rate from multi-channel WPPG sensor signals". In: *Sensors* 17.3 (2017), p. 506.

[17] Jianchu Yao and Steve Warren. "A short study to assess the potential of independent component analysis for motion artifact separation in wearable pulse oximeter signals". In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. IEEE. 2006, pp. 3585–3588.

[18] Rasoul Yousefi et al. "A motion-tolerant adaptive algorithm for wearable photoplethysmographic biosensors". In: *IEEE journal of biomedical and health informatics* 18.2 (2013), pp. 670–681.

[19] Zhilin Zhang, Zhouyue Pi, and Benyuan Liu. "TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise". In: *IEEE Transactions on biomedical engineering* 62.2 (2014), pp. 522–531.

[20] Lianning Zhu et al. "Heart rate monitoring during physical exercise from photoplethysmography using neural network". In: *IEEE sensors letters* 3.1 (2018), pp. 1–4.

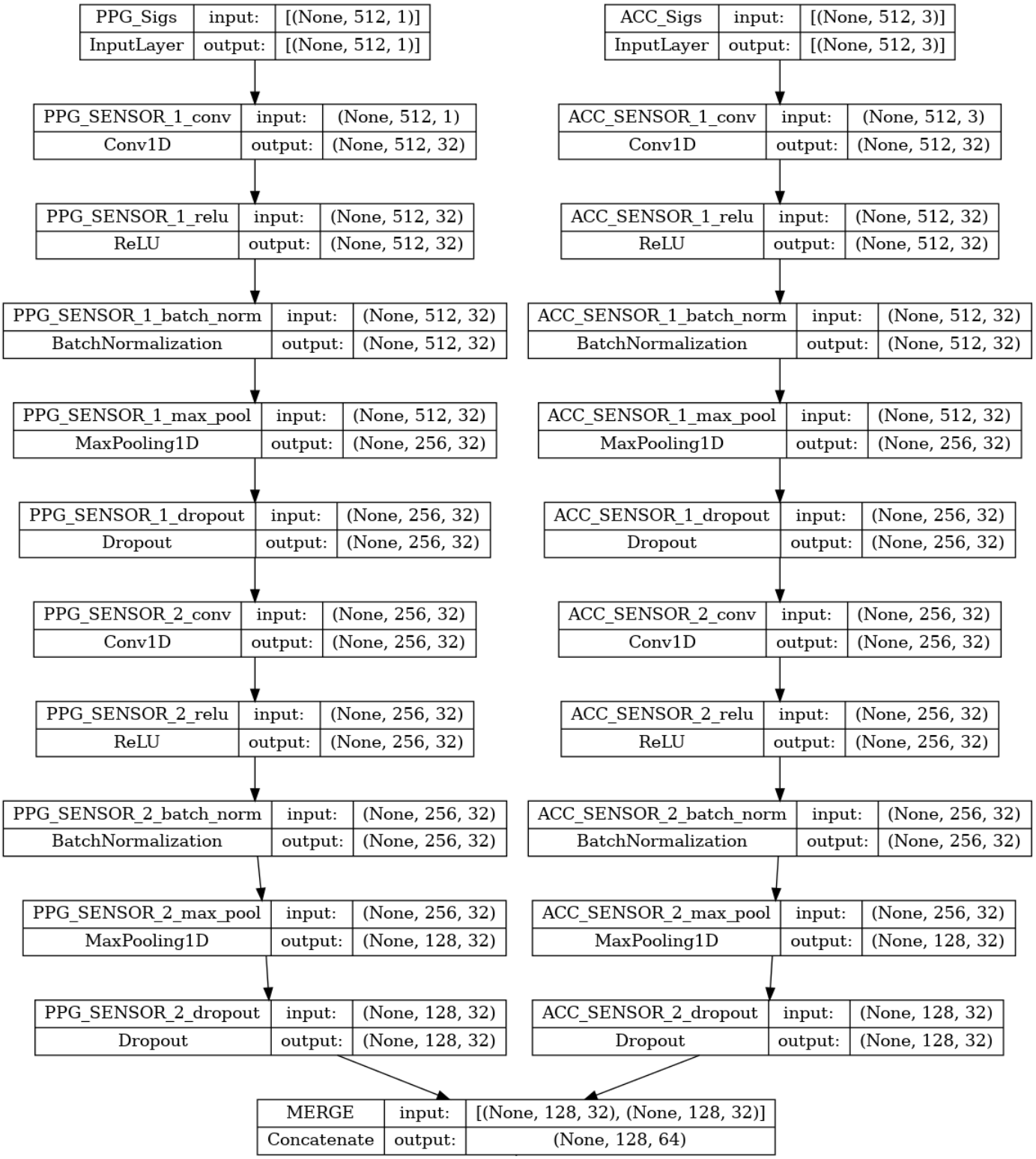
[21] Chengzhi Zong and Roozbeh Jafari. "Robust heart rate estimation using wrist-based PPG signals in the presence of intense physical activities". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, pp. 8078–8082.

Appendix

The table 7.1 presents the model parameters for our machine learning model. The parameters are grouped into different categories, including Band-pass Filter Parameters, Convolution 1D Parameters, LSTM Parameters, and Other Parameters. The table lists the specific parameter values for each category, such as the lower and upper frequencies for the band-pass filter, the size and number of filters for the sensor and concatenate layers, the dropout rates for different layers, the resampling frequency, the optimizer, and the loss function.

Band-pass Filter Parameters	
Lower Frequency	0.5 Hz
Upper Frequency	4.5 Hz
Order	2
Convolution 1D Parameters	
Sensor filter Size	16
Sensor number of filters	32
Sensor dropout	0.15
Concatenate filter size	16
Concatenate number of filters	64
Concatenate dropout	0.15
Pool size	2
LSTM Parameters	
Units	32
Dropout	0.15
Other Parameters	
Resampling frequency	64
Optimizer	Nadam
Epochs	200
Batch size	32
Loss Function	Negative Log Likelihood(NLL)

Table 7.1: Model Parameters


PPG_Sigs | input: | [(None, 512, 1)]
InputLayer | output: | [(None, 512, 1)]
ACC_Sigs | input: | [(None, 512, 3)]
InputLayer | output: | [(None, 512, 3)]
PPG_SENSOR_1_conv | input: | (None, 512, 1)
Conv1D | output: | (None, 512, 32)
ACC_SENSOR_1_conv | input: | (None, 512, 3)
Conv1D | output: | (None, 512, 32)
PPG_SENSOR_1_relu | input: | (None, 512, 32)
ReLU | output: | (None, 512, 32)
ACC_SENSOR_1_relu | input: | (None, 512, 32)
ReLU | output: | (None, 512, 32)
PPG_SENSOR_1_batch_norm | input: | (None, 512, 32)
BatchNormalization | output: | (None, 512, 32)
ACC_SENSOR_1_batch_norm | input: | (None, 512, 32)
BatchNormalization | output: | (None, 512, 32)
PPG_SENSOR_1_max_pool | input: | (None, 512, 32)
MaxPooling1D | output: | (None, 256, 32)
ACC_SENSOR_1_max_pool | input: | (None, 512, 32)
MaxPooling1D | output: | (None, 256, 32)
PPG_SENSOR_1_dropout | input: | (None, 256, 32)
Dropout | output: | (None, 256, 32)
ACC_SENSOR_1_dropout | input: | (None, 256, 32)
Dropout | output: | (None, 256, 32)
PPG_SENSOR_2_conv | input: | (None, 256, 32)
Conv1D | output: | (None, 256, 32)
ACC_SENSOR_2_conv | input: | (None, 256, 32)
Conv1D | output: | (None, 256, 32)
PPG_SENSOR_2_relu | input: | (None, 256, 32)
ReLU | output: | (None, 256, 32)
ACC_SENSOR_2_relu | input: | (None, 256, 32)
ReLU | output: | (None, 256, 32)
PPG_SENSOR_2_batch_norm | input: | (None, 256, 32)
BatchNormalization | output: | (None, 256, 32)
ACC_SENSOR_2_batch_norm | input: | (None, 256, 32)
BatchNormalization | output: | (None, 256, 32)
PPG_SENSOR_2_max_pool | input: | (None, 256, 32)
MaxPooling1D | output: | (None, 128, 32)
ACC_SENSOR_2_max_pool | input: | (None, 256, 32)
MaxPooling1D | output: | (None, 128, 32)
PPG_SENSOR_2_dropout | input: | (None, 128, 32)
Dropout | output: | (None, 128, 32)
ACC_SENSOR_2_dropout | input: | (None, 128, 32)
Dropout | output: | (None, 128, 32)
MERGE | input: | [(None, 128, 32), (None, 128, 32)]
Concatenate | output: | (None, 128, 64)

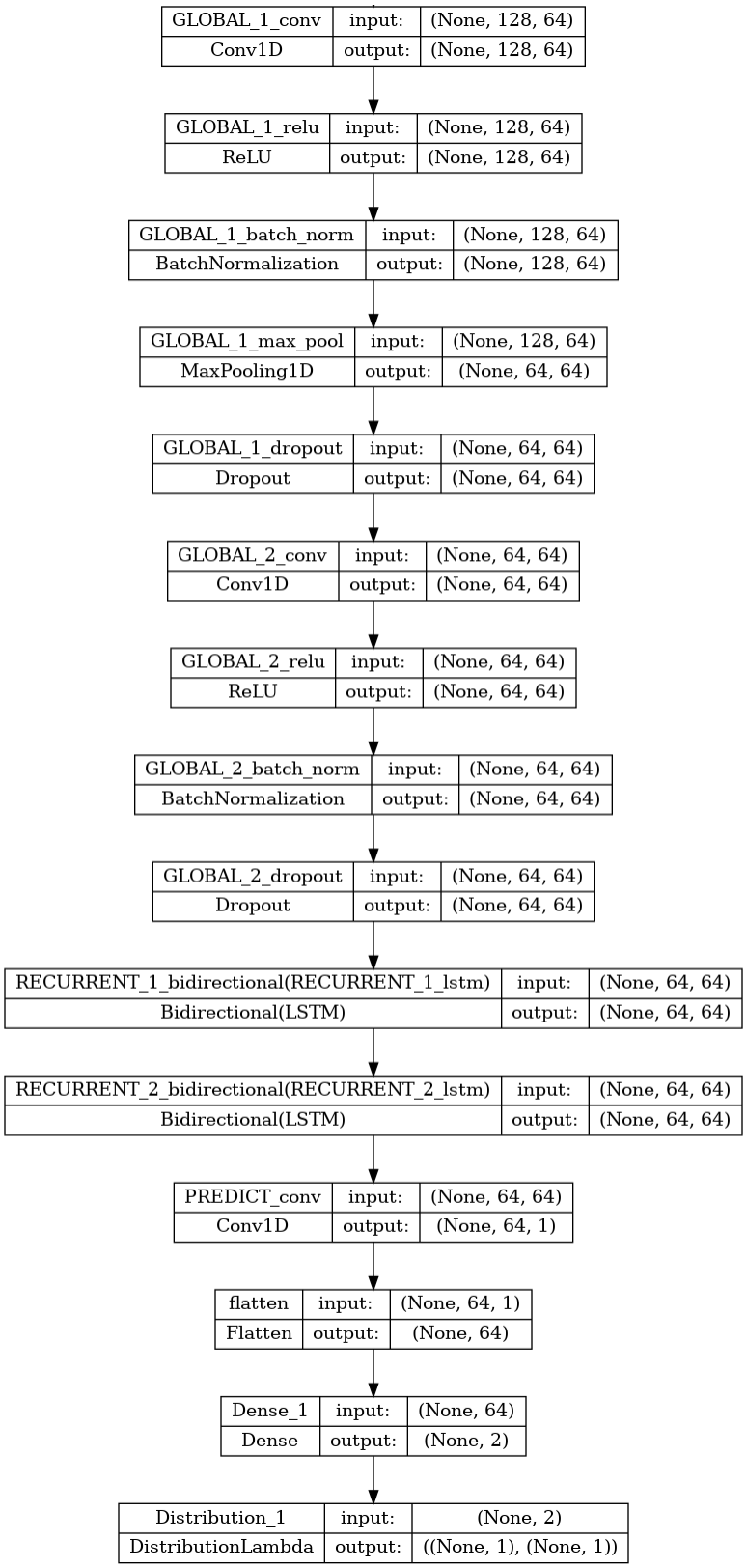


Figure 7.1: Model Architecture

[Fig. 7.1] shows the layers in the overall architecture of our developed model.

[Fig. 7.2] shows PPG signal, accelerometer signals,true heart rate and predicted mean and standard deviation of heart rate for 8s window.

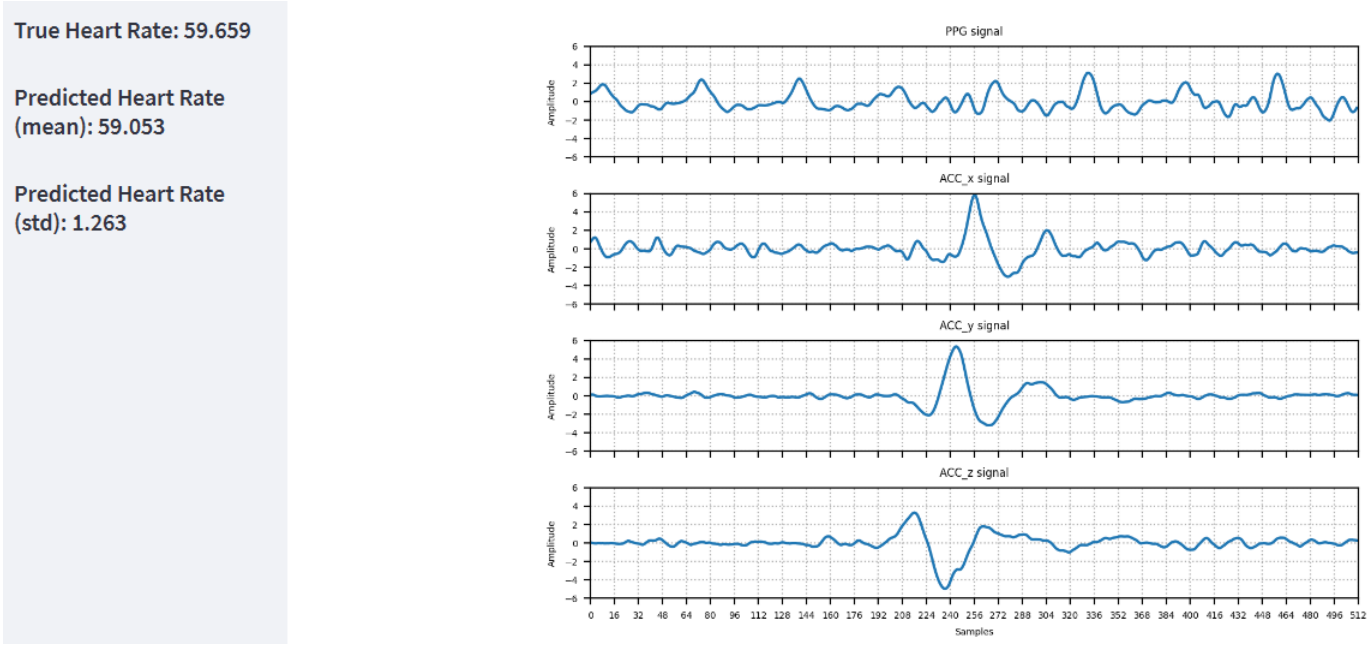


Figure 7.2: Prediction on test data