POLITECNICO DI TORINO

Master's degree
in ICT FOR SMART SOCIETIES

Master's Degree Thesis

Non-invasive cuff-less blood pressure estimation using LSTM-based recurrent neural networks

Supervisors

Prof. DANILO DEMARCHI

Prof. GUIDO PAGANA

Candidate

Mojtaba KAZEMI

July 2023

Abstract

Modern technology significantly contributes to improving living conditions and decreasing disease prevalence. Wearable health tools have been at the forefront of key breakthroughs in the healthcare sector. The fact that these tools can be used in both normal activities and therapeutic applications has led to significant advancement in this field.

Hypertension, characterized by periodic high blood pressure (BP), is the principal risk factor for Cardiovascular diseases (CVDs) that are among the leading causes of mortality. Hypertension often goes unnoticed by individuals, consequently, there is a pressing need for a monitoring device capable of continuously tracking blood pressure in various conditions of everyday life. Furthermore, nowadays to achieve continuous blood pressure monitoring is necessary to use invasive devices. However, these invasive methods pose limitations and additional challenges for patients.

To address these concerns, the European SINTEC project, with the aim of developing a wearable health device (WHD) for continuous BP monitoring. The improvement of WHDs focuses on three key areas: enhancing sensor accuracy, implementing advanced machine learning algorithms, and facilitating seamless communication between subsystems and other devices. The accuracy of WHDs relies heavily on sensor technology. Advances in sensors have significantly improved the precision and reliability of measurements. Machine learning (ML) algorithms play a vital role in data analysis and interpretation in WHDs. These algorithms process the collected physiological data, identify patterns, and generate actionable insights. By leveraging large database, ML algorithms can continuously improve and adapt their performance. They enable personalized health recommendations and early detection of abnormal health conditions, empowering individuals to make informed decisions about their lifestyle and improve their health. WHDs now feature sophisticated communication skills that make it possible for them to be easily integrated with smartphones and PCs. This enables individuals to keep track of and share their health information with medical providers. The purposes of this thesis are to enhance the accuracy of BP monitoring devices and to eliminate noise and situational dependency. Due to the fact that ECG and PPG signals provide a safer and comfortable monitoring experience for people than invasive techniques, this is accomplished by using them to improve the accuracy of blood pressure estimation. While ECG measures the electrical activity of the heart using electrodes placed on the skin, PPG analyzes changes in blood volume using light sensors.

The MIMIC-III database is used for training and testing the algorithm because consist of desired signals of diverse population. Signal cleaning techniques are

employed, followed by feature extraction. Machine learning clustering algorithms are utilized to remove outliers from the extracted features. Finally, a novel neural network model incorporating the Long Short-Term Memory (LSTM) layer is developed to predict continuous systolic and diastolic blood pressure values.
WHDs have witnessed remarkable advancements, driving the continuous improvement of non-invasive BP monitoring. These devices offer accurate and convenient solutions for individuals to monitor their cardiovascular health in real-time, both in daily life and medical settings. With the integration of advanced sensors, ML algorithms, and seamless communication capabilities, wearable health devices.

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude to my loving wife, whose unwavering support, patience, and encouragement have been instrumental throughout this journey.

I am also deeply indebted to my supervisors and Dr. Figini for their guidance and mentorship. Their invaluable expertise and constant belief in my abilities have shaped me into a better researcher. I am grateful for their patience, constructive feedback, and the opportunities they have provided for my growth.

Table of Contents

List of Tables

List of Figures

Acronyms

ABP
Arterial Blood Pressure

AI
Artificial Intelligence

BP
Blood Pressure

CVDs
Cardiovascular Diseases

DBP
Diastolic Blood Pressure

ECG
Electrocardiogram

HR
Heart Rate

LSTM
Long short-term memory

NARX
nonlinear autoregressive exogenous

NN
Neural Network

PPG

Photoplethysmogram

PPT

Pulse Transit Time

SBP

Systolic Blood Pressure

SD

Standard Deviation

SINTEC

Soft Intelligence Epidermal Communication platform

Chapter 1

Introduction

Cardiovascular disease (CVD) is the most common cause of death globally. Approximately, 17 million deaths occur worldwide due to cardiovascular disease, which makes up almost one-third of all deaths. High blood pressure called hypertension is a vital factor to the development of cardiovascular disease (CVD) [1, 2]. Therefore, reduce the number of people with hypertension is one of the global targets for noncommunicable diseases.

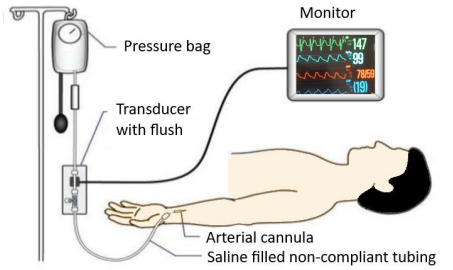
However, 1.28 billion persons between the ages of 30 and 79 are estimated to have hypertension, with the majority (two-thirds) residing in low- and middle-income nations, and nearly 46 percent of them are unaware that they have it [1]. In some situations, headaches, blurred vision, dizziness, and chest pain are symptoms of very high blood pressure, but in the majority of hypertension cases, there are no symptoms.

The blood pressure is presented with two values, one presents the systolic blood pressure(SBP) and another one demonstrates diastolic blood pressure (DBP). In the past, only way to measure the BP is used of sphygmomanometer devices but now Smart medical devices have been expanded to measure the BP continuously, in this way Links foundation introduce a SINTEC medical device to measure it so this thesis endeavour to find an algorithm to predict blood pressure more accurately.

1.1 State of the art devices

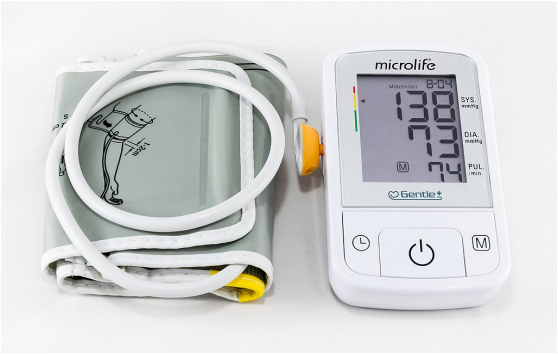
Measurement devices can be classified into two categories: invasive and non-invasive. Invasive blood pressure monitoring is crucial during aortic resection procedures as it allows real-time assessment of the patient's vascular competence [3], as illustrated in Figure 1.1-(a). On the other hand, non-invasive blood pressure measurement is commonly performed using a cuff and sphygmomanometer in a clinical setting, as shown in Figure 1.1-(b). The digital sphygmomanometer, depicted in Figure

1.1-(c), is the next-generation device that combines a cuff with digital calculations instead of relying on auscultation. These digital devices are user-friendly and can be easily used at home without the need for specialized training [4]. However, they are not suitable for continuous monitoring in intensive care units (ICUs) or for unstable or critically ill patients.



(a) Invasive blood pressure monitoring [5].

(b) Non-Invasive blood pressure monitoring [4].



(c) The digital sphygmomanometer that used for measuring the BP [4].

Figure 1.1: Blood pressure monitoring.

To address the limitations of traditional blood pressure measurement methods, researchers have focused on developing cuffless and continuous monitoring

techniques using wearable health devices (WHDs) [6]. Recent technological advancements, particularly in wearable devices and non-invasive sensors, have paved the way for cuffless blood pressure monitoring systems. These systems utilize physiological signals such as the electrocardiogram (ECG), photoplethysmogram (PPG), and pulse transit time (PTT) to estimate blood pressure.

However, a variety of factors, such as noise, variability, and individual differences, may interfere with the calculation of blood pressure from these physiological data. One approach to address these challenges is to use machine learning techniques such as neural networks to model the relationship between physiological signals and blood pressure. Neural networks have been successfully applied in various fields, such as image and speech recognition, natural language processing, and biomedical engineering.

Continuous and reliable non-invasive blood pressure monitoring holds great potential in preventing and managing cardiovascular diseases (CVDs), with hypertension being a primary risk factor [7]. By leveraging advancements in technology and machine learning, researchers aim to develop innovative methods for cuffless blood pressure monitoring that offer convenience, accuracy, and the ability to continuously monitor blood pressure, ultimately contributing to improved cardiovascular health outcomes.

1.2 Main idea

In this thesis, our objective is to develop a neural network model for the continuous prediction of blood pressure without the need for a cuff. The proposed model will utilize various physiological signals, including ECG, PPG, in addition to PTT and heart rate, to estimate blood pressure in real-time. The performance of the model will be assessed using a dataset collected from both healthy individuals and patients with hypertension.

The structure of this thesis is as follows. Chapter 1 offers a comprehensive literature review, covering topics such as cuffless blood pressure monitoring, the utilization of physiological signals for blood pressure estimation, and the application of machine learning techniques for blood pressure prediction. This chapter provides the necessary background and context for the subsequent chapters.

Chapter 2 delves into the dataset employed in this study, outlining the data collection process and describing the preprocessing steps undertaken to ensure data quality. Additionally, the chapter covers the methods employed for feature extraction, which involves extracting relevant information from the physiological signals.

In Chapter 3, we present the proposed neural network model and provide detailed information about its architecture. This chapter outlines the design choices and

considerations made in developing the model, ensuring its effectiveness in accurately predicting blood pressure.

Chapter 4 focuses on the experimental results and performance evaluation of the proposed model. We present and analyze the outcomes of the experiments conducted using the dataset, assessing the model's accuracy and reliability in predicting blood pressure.

Finally, in Chapter 5, we conclude the thesis by summarizing the contributions made, highlighting the limitations of the study, and proposing potential avenues for future research and improvements in the field of continuous cuffless blood pressure prediction.

Through this thesis, we aim to advance the understanding and capabilities of predicting blood pressure using non-invasive and continuous monitoring techniques. The development of an accurate and reliable neural network model holds promise for enhancing healthcare practices and improving patient outcomes in the field of blood pressure management.

Chapter 2

Psychological signals

2.1 Arterial Blood Pressure

The Arterial Blood Pressure (ABP) signal represents a pressure wave that travels through our arteries. As it moves along different sections, its speed and shape change. Imagine a wave traveling through a flexible pipe it gradually becomes weaker and slower due to its interaction with the pipe's material [8]. However, if the pipe divides into smaller branches of varying sizes, something interesting happens. The wave's signal gets stronger because of reflection phenomena. The exact nature of these phenomena varies depending on the blood vessel. When it comes to recording ABP, we use the aorta, which is the least rigid vessel, and where reflections are negligible [9].

When the ventricle propels blood into the aorta, initiating a cascade of events. Initially, the aortic pressure rises, almost matching the ventricle's pressure. However, this surge in aortic pressure is ephemeral. During the diastolic phase, when the heart rests, the flow of blood into the aorta comes to a halt. Consequently, the pressure gradually diminishes, reaching its nadir just moments before the subsequent heartbeat.

Blood pressure is typically represented by two values: systolic blood pressure (SBP), which corresponds to the highest pressure in the arterial system during ventricular contraction, and diastolic blood pressure (DBP), which represents the lowest pressure during ventricular relaxation [10]. The average arterial pressure throughout the cardiac cycle, known as Mean Arterial Pressure (MAP), holds considerable significance when calculated with formula 2.1.

$$MAP = \frac{SBP + (2 * DBP)}{3} \tag{2.1}$$

It is possible to get the ABP waveform invasively or non-invasively. A catheter linked to a pressure transducer and placed into an artery, usually the radial or femoral artery, is used to measure invasive ABP. An aneroid sphygmomanometer, sometimes referred to as a blood pressure cuff, is a non-invasive measurement is shown in figure 2.1.

To measure blood pressure using an aneroid sphygmomanometer, the first step

(a) Dial arrangement in aneroid type BP measurement technique [4].

(b) Wrist blood pressure monitor [11].

Figure 2.1: Aneroid sphygmomanometers.

involves securely wrapping the cuff around the upper arm. By squeezing the rubber bulb, the cuff is inflated, exerting pressure on the brachial artery and causing its compression. As the cuff is gradually deflated, blood flow resumes within the artery, and the pressure in the cuff is gradually released. The point at which the first sound, known as the Korotkoff sound, is heard as blood begins to flow is recorded as the systolic blood pressure. Conversely, the pressure at which the sound disappears entirely is recorded as the diastolic blood pressure [7]. It is presented on figure 2.2 Aneroid sphygmomanometers find extensive use in medical environments such as hospitals, clinics, and doctor's offices. They are also popular among individuals for monitoring blood pressure at home. These devices offer portability and operate without the need for electricity, rendering them convenient for use in diverse settings. It is essential to regularly calibrate and maintain aneroid sphygmomanometers to ensure accurate measurements.

The level of arterial blood pressure (ABP) is directly linked to factors such as cardiac output, arterial elasticity, and peripheral vascular resistance. Blood pressure can be easily manipulated and influenced by various activities [13]. It is crucial to keep your blood pressure within normal ranges. Stage 1 hypertension is defined as having a blood pressure between 140/80 mmHg and 159/99 mmHg. Stage 2 hypertension is classified as having a BP between 160/100 mmHg and

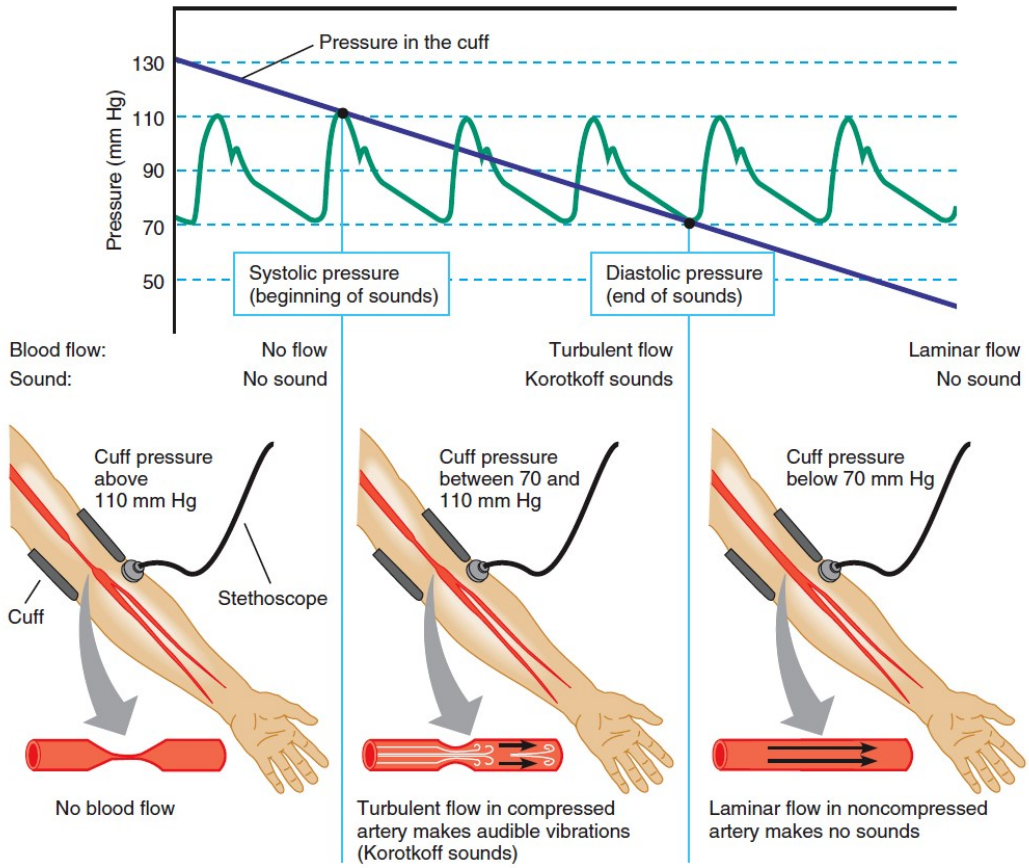


Figure 2.2: Waveform and measurement of arterial blood pressure [12].

179/109 mmHg [14]. Hypertensive emergency refers to a very high blood pressure that causes potentially life-threatening symptoms and end-organ damage, while hypertensive urgency specifies a blood pressure more than 180/120 mmHg. Contrarily, hypotension is defined as a blood pressure less than 90/60 mmHg [15].
The mean arterial pressure (MAP) is approximately 83.3 mm Hg for a healthy adult [12]. The diastolic pressure carries more significance in this calculation. This is because the aortic pressure reaches its maximum level for a shorter duration during a single heartbeat compared to the minimum level, which lasts about twice as long. By considering this weighted mean, the MAP provides a more comprehensive representation of the overall pressure within the arteries. It serves as a vital parameter for assessing cardiovascular health and plays a significant role in ensuring proper blood perfusion to various organs and tissues throughout the body [7].

The relationship between ABP and other physiological signals, such as electrocardiogram (ECG) and photoplethysmogram (PPG), can provide valuable insights

into cardiovascular function. ECG records the electrical activity of the heart, revealing information about heart rate, rhythm, and cardiac abnormalities. Changes in blood pressure can influence the ECG waveform, particularly in conditions such as hypertension or hypotension, where alterations in heart rate and rhythm may occur as a compensatory response.

PPG, on the other hand, measures blood volume changes in the microvascular bed of tissue using light. PPG signals can be influenced by changes in blood pressure as variations in arterial volume affect light absorption and reflection. Therefore, PPG signals can indirectly reflect blood pressure changes, providing information about vascular tone and peripheral perfusion.

By integrating ABP, ECG, and PPG signals, a more comprehensive understanding of the cardiovascular system can be achieved. This synergistic approach allows for the assessment of cardiac electrical activity, blood pressure dynamics, and vascular responses, enabling the identification and management of cardiovascular disorders. Therefore, investigating the relationship between ABP, ECG, and PPG signals holds great potential for advancing our knowledge of cardiovascular health and improving diagnostic and therapeutic approaches.

2.2 Importance of PPG and ECG

The non-invasiveness of ECG and PPG signals is a key benefit. PPG uses light sensors to assess changes in blood volume, whereas ECG uses electrodes applied to the skin to measure the electrical activity of the heart. ECG and PPG signals offer a safer and more comfortable monitoring experience for people than invasive techniques like artery catheterization. This feature's non-invasiveness lowers the possibility of problems, encourages patient compliance, and allows for long-term monitoring.

The widespread use of ECG and PPG signals in wearable technology is another benefit. ECG and PPG sensors may now be found in wearable gadgets like smartwatches, fitness trackers, and mobile health applications thanks to recent developments in sensor technology and downsizing. These gadgets offer a practical and discrete way to continuously check blood pressure throughout the day. The wearable technology enables continuous data collecting and real-time trend analysis of blood pressure patterns.

Real-time monitoring and dynamic evaluation of blood pressure are made possible by ECG and PPG signals. A more thorough understanding of blood pressure trends, including fluctuations caused by various activities, postural changes, and reactions to stresses, is possible with continuous data recording. The quick diagnosis of anomalies or variations in blood pressure is made possible by real-time analysis of ECG and PPG data, enabling prompt treatments and individualized healthcare.

To improve blood pressure monitoring, ECG and PPG signals can be combined with other physiological indicators. In addition to blood pressure, ECG data can offer influential knowledge on heart rate variability, cardiac function, and arrhythmias. The pulse transit time, which has been connected to variations in blood pressure, may be derived from PPG data. A more complete and accurate image of cardiovascular health may be produced by combining different signals, which helps with the diagnosis and treatment of various cardiovascular disorders. Due to its non-invasive nature and accessibility in wearable devices, the use of ECG and PPG signals for continuous blood pressure monitoring offers a number of benefits. These signals make it possible to conveniently track blood pressure patterns in real-time, enabling individualized treatment and the early identification of anomalies. The evaluation of cardiovascular health is improved overall when ECG and PPG data are combined with other physiological indicators. Utilizing these benefits, researchers and doctors may create prediction models for continuous blood pressure monitoring that are more precise and effective, improving healthcare outcomes.

2.3 Photoplethysmogram

PPG is a non-invasive technique for measuring the quantity of light that is absorbed or reflected by blood vessels in living tissue, and it was first investigated in the 1930s [10].PPG is a composite name made out of the words "photo," which stands for light, "plethysmo," which means volume, and "graphy," which stands for recording [16]. The PPG signal responds to variations in blood volume rather than blood vessel pressure because the degree of optical absorption or reflection relies on the amount of blood that is present in the optical path. In other words, PPG records the volume of blood in the sensor coverage area using a photoelectric method, whether transmissive or reflective, to detect changes in blood volume and produce a PPG signal [17]. In fact, the sensor coverage area covers multiple capillaries, as well as veins and arteries. As a result, the PPG signal is a complicated blend of the cardiovascular system's arteries and veins' blood flow. Typically, pulsatile and non-pulsatile blood volume are included in a raw PPG signal [18]. A photodetector and a light-emitting diode (LED) make up a PPG device, which emits light and detects it. Depending on where the LED and photodetector are located, the device can be categorized as transmissive or reflecting. Configurations for a photoplethysmogram measurement equipment are shown in 2.3. In the transmissive kind, skin tissues are found between the photodetector and the LED, which is on the other side of the device. The photodetector is placed next to the LED in the reflecting type. The transmissive type is typically used to measure PPG in the distal area of the body, where skin tissues, including those of fingers, toes, and earlobes, are thin.

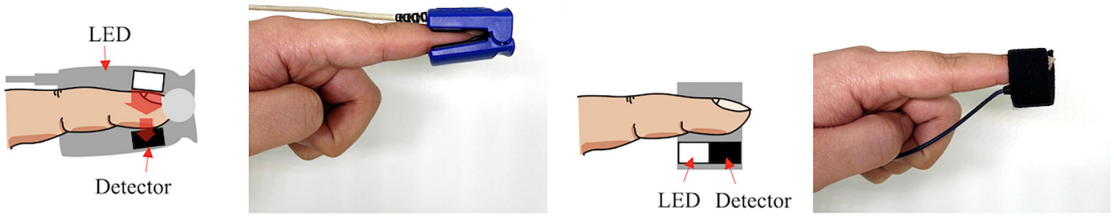


Figure 2.3: Set up of photoplethysmography measuring: Left: transmissive type - right: reflective type [16].

This is since the transmissive type measures attenuated light intensity after the light passes through skin tissues. The transmission-type PPG sensor shows more stable PPG measurement performance than the reflective type [16].

The amount of light that the sensor can detect decreases as more light is absorbed by the tissue as blood circulates through it. The result is a waveform that shows how the blood volume has changed over time. The PPG waveform typically has a number of characteristics, such as a sharp upstroke caused by the arterial pulse, a peak that corresponds to systolic blood pressure, and a dicrotic notch that is connected to the closing of the aortic valve.

Figure 2.4 illustrates how the PPG waveform is created by subtracting the quantity of light reflected or transmitted by human tissue from the light intensity measured with a photodetector. In general, a pulsatile component and a non-pulsatile component make up the PPG waveform [17]. The pulsatile component, sometimes referred to as the alternating current (AC) component, is connected to variations in arterial blood volume. It is connected to vasodilation, vasomotor, and vascular tones and is timed to the cardiac cycle. The PPG waveform's other components, omitting the pulsatile component, are referred to as the non-pulsatile component, also known as the direct current (DC) component. The biological properties of the measurement site, such as the tissue composition and basic blood volume, as well as external variables, such as the measuring device's specifications and ambient light, all have an impact on non-pulsatile components.

The PPG waveform changes according to cardiac activity. Additionally, it could alter as a result of breathing, autonomic nervous system activity, vascular and venous activity. The PPG waveform has two curves: a rising curve for capillary blood volume increases caused by ventricular contraction, and a descending curve for capillary blood volume decreases caused by heart dilatation. According to heart activity, it is repeated. At that point, the PPG waveform's rising curve is referred to as its systolic phase, and its falling curve as its diastolic phase [7]. The PPG waveform of a single pulse and several feature points are shown in Figure 2.4. The beginning of pulsation is known as the pulse onset, and it occurs when blood volume is at its lowest before the systolic phase. The maximum blood volume is

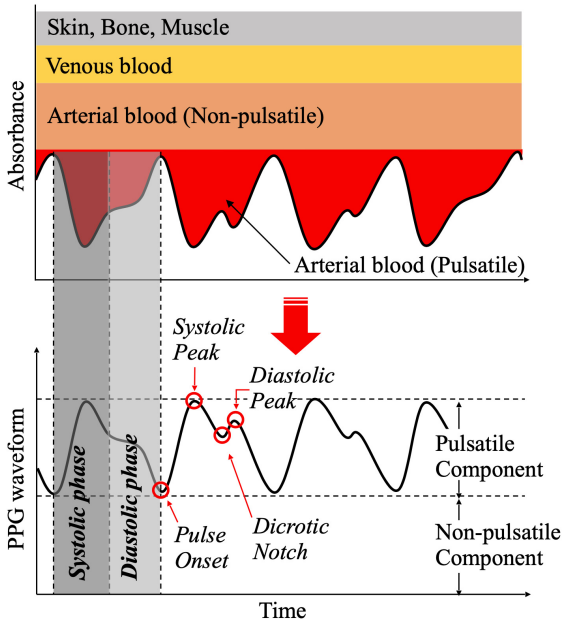


Figure 2.4: Set up of photoplethysmography measuring: Left: transmissive type - right: reflective type [16].

where systolic peak is determined. Just before the aortic valve shuts, transient rising and falling of the PPG waveform during diastole occurs as blood volume in capillaries briefly increases again due to the presence of a pressure gradient in the opposite direction to the blood flow.

2.4 Electrocardiography

Electrocardiography (ECG) is a crucial and widely used diagnostic tool in clinical medicine. It serves as a valuable screening tool in both inpatient and outpatient settings due to its affordability and ease of acquisition [19]. The ECG is instrumental in diagnosing various heart conditions, including previous myocardial infarction, current cardiac ischemia, conduction abnormalities such as atrial fibrillation, and life-threatening tachycardias. Additionally, ECG findings help determine the appropriate type of implanted cardiac defibrillator for the treatment of advanced heart failure [20].

Beyond cardiac conditions, ECGs can also provide insights into noncardiac diseases. Certain noncardiac conditions, such as electrolyte imbalances and adverse effects of medications, can manifest as unique patterns on an ECG due to their impact on conduction patterns. This further underscores the versatility and importance of ECGs in clinical practice [21, 22].

The electrocardiogram (ECG) is a time-dependent recording of the electrical activity that occurs within the heart throughout the cardiac cycle. The heart's electrical signals are highly coordinated, leading to synchronized electrical potentials that can be detected even at locations distant from the heart's source [22]. These electrical potentials corresponding to different phases of the cardiac cycle can be captured by electrodes placed on the surface of the skin. The presence of bodily fluids in the body facilitates the conduction of electrical activity generated by nerve or muscle tissue, allowing it to propagate throughout the body [23].

The ECG waveform typically has amplitudes on the order of millivolts (mV), with values exceeding 0.5 mV. The maximum amplitude observed in a normal ECG waveform ranges between 2 mV and 3.0 mV [7].

During an ECG test, electrodes are placed on specific locations of the patient's chest, arms, and legs, as shown in Figure 2.5. These electrodes are used to monitor the electrical impulses generated by the beating heart. The resulting ECG signal provides a visual representation of the heart's electrical activity over time [24].

The ECG is a valuable diagnostic tool that allows healthcare professionals to assess the electrical functioning of the heart, identify abnormalities or irregularities in the cardiac rhythm, and aid in the diagnosis and management of various cardiac conditions.

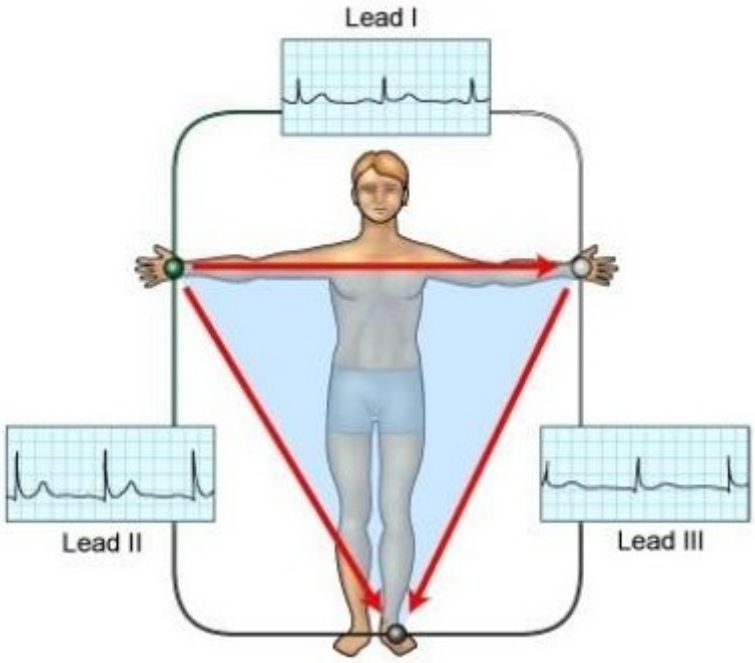


Figure 2.5: Einthoven's triangle [25].

The ECG signals consist of various peaks and regions as illustrated in figure 2.6

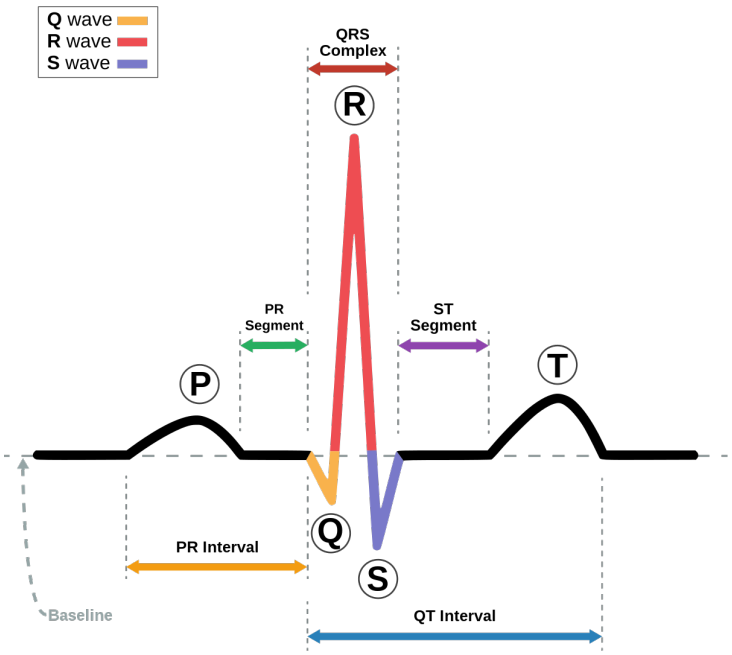


Figure 2.6: ECG signal [26].

that provide important information about the electrical activity of the heart [27, 25]. Here is an explanation of the key peaks and regions:

- P wave: The P wave represents the depolarization (contraction) of the atria, the upper chambers of the heart. It indicates the initiation of an electrical impulse in the sinoatrial (SA) node, which triggers atrial contraction.
- QRS complex: The QRS complex is a group of waves that represents the depolarization of the ventricles, the lower chambers of the heart. It consists of three distinct components:
  - Q wave: The first downward deflection after the P wave represents initial ventricular depolarization.
  - R wave: The upward deflection following the Q wave indicates further ventricular depolarization.
  - S wave: The downward deflection following the R wave represents the final phase of ventricular depolarization.

  The QRS complex signifies the contraction of the ventricles, which pumps blood out of the heart.

- T wave: The T wave represents the repolarization (recovery) of the ventricles. It reflects the electrical resetting of the heart muscle to prepare for the next heartbeat.
- ST segment: The ST segment is the region between the end of the QRS complex and the beginning of the T wave. It represents the interval when the ventricles are fully depolarized before repolarization begins. Deviations or abnormalities in the ST segment can indicate myocardial infarction (heart attack) or other cardiac conditions.
- PR interval: The PR interval is the time from the beginning of the P wave to the start of the QRS complex. It reflects the conduction time from the atria to the ventricles and can provide information about the health of the atrioventricular (AV) node and conduction pathways [27, 24, 28].

These peaks and regions, when analyzed collectively, can help healthcare professionals diagnose various heart conditions, including arrhythmias, conduction abnormalities, ischemia, and ventricular hypertrophy[29].

Although blood pressure is not directly measured by ECG signals, they can provide valuable information about the cardiovascular system that is relevant to blood pressure assessment. The ECG waveform can reveal abnormalities in heart rate and rhythm, which may be associated with cardiovascular conditions, including high blood pressure. Additionally, changes in the amplitude and duration of the ECG waveform can indicate variations in blood pressure or volume [30].

By analyzing the ECG signals, healthcare professionals can gain insights into a person's overall cardiovascular health, including their blood pressure status. However, it is important to note that blood pressure is typically measured using separate techniques, such as a blood pressure cuff, as it is a distinct physiological parameter from the electrical activity of the heart that the ECG captures [31, 27].

Therefore, while ECG signals can provide valuable information about the cardiovascular system and offer clues related to blood pressure, they are not a direct measurement of blood pressure itself. Combining ECG findings with dedicated blood pressure measurements enables a more comprehensive assessment of an individual's cardiovascular health.

Chapter 3

Materials and methods

This project comprises five main sections: data reading, data cleaning, feature extraction, model design, and model testing, as depicted in Figure 3.1. Each section plays a crucial role in the overall process of the project.

The first section, "Data Reading" involves downloading the necessary data from the appropriate server and selecting the desired signals for analysis. This step ensures that the project has access to the required data for further processing and modeling.

The "Data Cleaning" section focuses on preparing the dataset for analysis by addressing any inconsistencies, errors, or missing values present in the signal data. This step is essential to ensure data quality and accuracy throughout the project.

In the "Feature Extraction" step, the relevant characteristics and information are extracted from the cleaned data. This involves identifying and computing features that are informative and significant for the model design and subsequent analysis.

The "Model Design" section involves developing the model architecture and selecting suitable algorithms and techniques to train the model. This step incorporates the extracted features from the previous section to build a model that can effectively predict the desired outcome or make accurate estimations.

Finally, in the "Model Testing" phase, the performance and effectiveness of the designed model are evaluated. Standard evaluation factors, such as accuracy, precision, recall, or other suitable metrics, are employed to assess the model's performance and determine its suitability for the intended purpose.

Each section of the project roadmap consists of specific tasks, methods, and materials that will be explained and implemented in detail throughout the project. By following this roadmap, the project aims to progress systematically through each section, ensuring comprehensive data analysis, feature extraction, and model development, ultimately leading to reliable and accurate predictions or estimations.

H

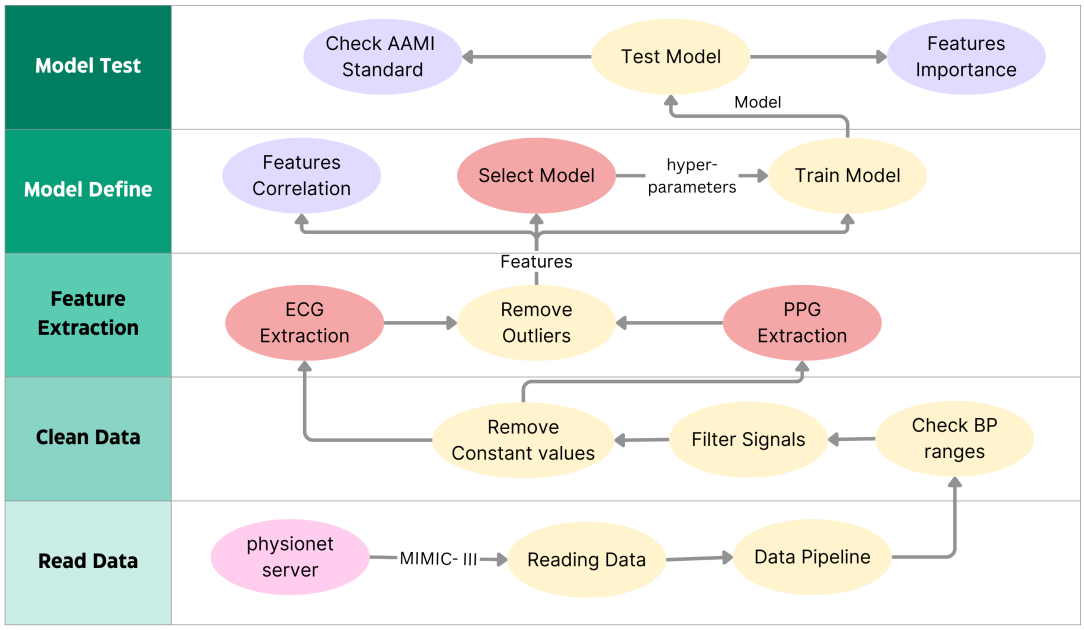


Figure 3.1: Road-map of project.

3.1 MIMIC-III database

This thesis focused on designing an effective neural network prediction model, which requires a substantial amount of data for training, validation, and testing. The Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC) database, collected by the MIT lab, served as a valuable resource for this purpose.

The MIMIC-III Waveform Database consists of over 60,000 record sets from approximately 30,000 patients in the intensive care unit (ICU), although the demographic information is not included [32]. Each record set contains digitized signals such as electrocardiogram (ECG), arterial blood pressure (ABP), respiration, fingertip photoplethysmogram (PPG), and other signals, as well as periodic measurements stored in a "numerics" record. These record sets provide quasi-continuous recordings of vital signs for a single patient throughout their ICU stay, typically spanning a few days or even several weeks. It should be noted that not all signals are available for all patients at all times due to occasional signal disconnections [33].

Access to the MIMIC database can be obtained through tools available on the Phyisonet website [34]. The Lightwave web application offers visualization capabilities for the signals, while the WFDB (WaveForm database) software package provides downloading functionality. The WFDB package is compatible with Python

and will be utilized in the subsequent sections of this thesis to evaluate the predictive model.

By leveraging the MIMIC-III database and the wfdb package, this thesis aims to leverage real-world patient data to develop and assess the effectiveness of the proposed prediction model. These resources offer valuable insights into the physiological signals and their dynamics during ICU stays, enabling the development of robust and reliable predictive models for various medical applications.

3.2 Python

Python has gained immense popularity as a programming language for machine learning due to its inherent advantages and robust ecosystem. Both researchers and practitioners in the field of machine learning recognize Python as one of the most user-friendly, readable, and expressive programming languages available [35, 36].

Python's simplicity and readability make it easier for users to understand and write code, enabling efficient development and experimentation in the machine learning domain [36]. Its clean syntax and extensive libraries, such as NumPy, Pandas, and scikit-learn, provide powerful tools for data manipulation, analysis, and model development.

Furthermore, Python's versatility allows seamless integration with other technologies and frameworks commonly used in machine learning. The creation and training of neural network models is facilitated by TensorFlow, a well-known library in this area. Additionally, well-known libraries that provide effective tools for machine learning applications include PyTorch, scikit-learn, and Keras. Users may concentrate on model creation and experimentation by using these libraries to abstract away complicated implementation concerns [37].

Machine learning has advanced significantly thanks in large part to the dynamic and active Python community. Through online forums, tutorials, and in-depth documentation, researchers, developers, and practitioners actively create libraries, share expertise, and offer help [38]. This collaborative setting encourages creativity and guarantees easy access to resources for machine learning projects.

Overall, Python's combination of simplicity, readability, extensive libraries, and community support makes it an ideal choice for machine learning practitioners and academics. Its user-friendly nature and rich ecosystem contribute to the accelerated growth and advancements in the field of machine learning.

3.3 Literature review

The field of continuous blood pressure monitoring has seen the application of various machine learning algorithms, ranging from traditional methods like logistic and linear regression to more sophisticated techniques such as artificial neural networks (ANN) with their diverse architectures and characteristics [39]. These machine learning models are developed to provide medical experts with a valuable tool for supporting clinical decision-making [40].

As discussed in Section 2, the PPG and ECG signals have been extensively utilized in the estimation of blood pressure. Table 3.1 provides a summary of previous works in this domain, including the algorithms employed and the corresponding results. These studies have employed a range of machine learning techniques to develop models capable of accurately estimating blood pressure based on the analysis of PPG and ECG signals.

The table presents an overview of the various approaches and their outcomes, providing valuable insights into the performance and effectiveness of different machine learning algorithms for continuous blood pressure estimation. These studies serve as a foundation for the current research, guiding the selection and evaluation of suitable algorithms in developing an accurate and reliable blood pressure prediction model.

Kachuee et al. conducted a study in which they collected data from the MIMIC-II database, specifically ECG, PPG, and ABP signals. They applied certain limitations on the data, such as restricting the systolic blood pressure (SBP) to be between 80 and 180 mmHg, and the diastolic blood pressure (DBP) to be between 60 and 130 mmHg. They also filtered out signals with a duration of less than 10 minutes, resulting in a final database of 3,663 record segments [41].

To ensure invariance to changes in sampling frequency, the input signals were resampled at a constant frequency of 1 kHz during the preprocessing stage. The signals were then decomposed into their component parts using a discrete wavelet transform (DWT) with a Daubechies 8 (db8) mother wavelet and 10 decomposition levels. After removing coefficients associated with ultrahigh and extremely low frequencies, the remaining coefficients underwent traditional wavelet denoising. The signals were reconstructed after this cleansing process.

In terms of feature selection, five features were extracted from the ECG and PPG signals. These features included the ECG R-Peak, three points on the PPG signal (PATp, PATf, and PATd), heart rate (HR), augmentation index (AI), large artery stiffness index (LASI), and inflection point area ratio (IPA) [41].

For their machine learning algorithm, Kachuee et al. utilized several regression techniques, including Regularized Linear Regression, Decision Tree Regression, Support Vector Machine (SVM), Adaptive Boosting (AdaBoost), and Random

Forest Regression (RFR). Among these techniques, AdaBoost yielded the best results, as indicated in Table 3.1 [42].

These findings highlight the effectiveness of the AdaBoost technique in predicting blood pressure using the extracted features from ECG and PPG signals. It demonstrates the potential of machine learning approaches in the field of continuous blood pressure monitoring and offers valuable insights for the development of accurate and reliable prediction models.

Z. Li et al. developed a novel method for blood pressure estimation, which they applied to the MIMIC database as well as experimental subjects. The database was filtered based on specific criteria, including a systolic blood pressure (SBP) range of 80 to 180 mmHg, a diastolic blood pressure (DBP) range of 50 to 130 mmHg, and a range of absolute differences between SBP and DBP of 20 to 70 mmHg. As a result of this filtering process, they obtained a dataset comprising 120,684 segments from the MIMIC database [43].

To estimate blood pressure, the researchers designed two neural networks: a feature-net and a regression-net. These networks were constructed using deep neural network architectures, and they were interconnected to enable information flow between them. The feature-net was responsible for extracting relevant features from the input data, while the regression-net performed the actual blood pressure estimation [44]. By connecting these networks, they aimed to leverage the learned features to improve the accuracy of the regression-net in predicting blood pressure.

The performance of their method was evaluated using the mean absolute error (MAE), as shown in Table 3.1. The Intra-patient dataset was utilized for training the model, and reference inputs were used for calibration purposes. The MAE values in the table provide an indication of the model's performance in accurately estimating blood pressure based on the given dataset and experimental setup [43].

The work of Z. Li et al. represents a novel approach to blood pressure estimation, employing deep neural networks and utilizing a filtered dataset. Their method shows promise in achieving accurate blood pressure estimation and contributes to the field of continuous blood pressure monitoring.

Slapničar et al. conducted a study in which they designed a convolutional neural network (CNN) prediction model to estimate blood pressure from the photoplethysmogram (PPG) signal. The features extracted from the PPG signal included PPG, PPG', and PPG" in both the temporal and frequency domains. The aim was to capture relevant information from the PPG signal that could contribute to accurate blood pressure estimation [45].

To address the challenge of training very deep networks with the vanishing gradient problem, the researchers proposed a developed ResNet model. This approach utilized shortcut connections, also known as residual connections, between larger blocks of layers. By incorporating these connections, the issue of decreased backward error propagation and inadequate weight updates in the initial layers of

deep networks was mitigated [45].

The performance of the developed ResNet model was evaluated by testing it on a dataset comprising 510 subjects from the MIMIC-III database. The results of the evaluation are presented in the corresponding table [45]. These results provide insights into the model's performance in accurately estimating blood pressure based on the PPG signal.

By leveraging the capabilities of CNNs and incorporating residual connections within the ResNet architecture, Slapničar et al. demonstrated a promising approach for blood pressure estimation from the PPG signal. Their study contributes to the advancement of continuous blood pressure monitoring and highlights the potential of deep learning techniques in this field.

Senturk et al. achieved one of the lowest mean absolute errors (MAE) in blood pressure estimation by designing the NARAX neural network model and testing it on the MIMIC-II dataset. Their model utilized electrocardiogram (ECG), photoplethysmogram (PPG), and arterial blood pressure (ABP) signals as inputs.

In the preprocessing stage, the researchers applied min-max normalization to normalize the signals. Then, a low-pass filter with a cutoff frequency of 40 Hz was employed to remove high-frequency noise from the PPG and ABP signals, with a 5 Hz cutoff frequency. Additionally, a median filter was used to correct low-frequency baseline shifts in the signals.

For feature extraction, Senturk et al. extracted time, frequency, and chaotic features from both the ECG and PPG signals. These features captured relevant information from the signals, enabling the model to make accurate blood pressure predictions [46].

To train, validate, and test the model, the dataset was divided into three portions: 60% for training, 15% for validation, and 25% for testing the model's performance.

The proposed NARAX neural network model, combined with the preprocessing steps and feature extraction techniques, demonstrated promising results in blood pressure estimation. Senturk et al.'s study contributes to the field of continuous blood pressure monitoring and highlights the effectiveness of neural network models in this domain [46]. Panwar et al. proposed a model called "PP-Net" for the simultaneous estimation of diastolic blood pressure (DBP), systolic blood pressure (SBP), and heart rate. Their approach involved utilizing the Long-term Recurrent Convolutional Network (LRCN) and eliminating the need for explicit feature extraction. The model was tested on the MIMIC-II database, and promising results were obtained.

The PP-Net model was designed by combining convolutional neural network (CNN), long short-term memory (LSTM), and fully connected layers. The CNN component served as a feature extractor and consisted of two 1D convolutional layers with interleaved rectified linear unit (ReLU) activations, max-pooling layers, and drop-out layers. The output features extracted from the CNN were then fed

into the LSTM model [47].

The LSTM model, composed of two LSTM layers, utilized the tangent activation function and included a dropout layer for regularization. The LSTM layers were responsible for capturing temporal dependencies in the input signals and providing context for predicting the physiological parameters.

Finally, the output from the LSTM model was passed through a fully connected layer, which was responsible for predicting the DBP, SBP, and heart rate values simultaneously.

By leveraging the LRCN architecture and combining CNN, LSTM, and fully connected layers, Panwar et al. demonstrated the effectiveness of their PP-Net model in estimating multiple cardiovascular parameters. The model's performance on the MIMIC-II database showcased its potential for accurate and simultaneous prediction of DBP, SBP, and heart rate [47].

Baker et al. proposed a prediction model for diastolic blood pressure (DBP) and systolic blood pressure (SBP) by combining temporal convolutional layers with long short-term memory (LSTM) layers. Their approach did not involve explicit feature selection, and they directly used electrocardiogram (ECG) and photoplethysmogram (PPG) signals as inputs for the network.

In addition, they employed a bidirectional LSTM model, which considers the data in both the original and reversed order. Bidirectional LSTMs (BiLSTMs) have the advantage of learning from past and future values within the sequence, thereby capturing more temporal dependencies in the data [48].

By combining temporal convolutional layers, LSTM layers, and the bidirectional architecture, Baker et al. achieved acceptable results in the prediction of DBP and SBP. Their model leveraged the information present in ECG and PPG signals to make accurate predictions for blood pressure parameters [48].

Chih-TA et al. proposed a cascade neural network model for the estimation of diastolic blood pressure (DBP), systolic blood pressure (SBP), and heart rate (HR). Their model consisted of multiple stages, with each input (PPG and ECG) being processed by a convolutional layer to extract relevant features.

The extracted features from the PPG and ECG signals were then concatenated and passed through a long short-term memory (LSTM) layer. The LSTM layer was responsible for capturing the temporal dependencies and patterns in the combined features.

By utilizing this cascade neural network architecture, Chih-TA et al. were able to avoid explicit feature extraction, simplifying the overall algorithm. Instead, the network automatically learned and extracted the relevant features from the input signals during the training process.

This approach allowed for the simultaneous estimation of DBP, SBP, and HR using PPG and ECG signals as inputs. The cascade neural network model provided an effective solution for blood pressure and heart rate estimation without the need

for separate feature extraction steps [49].

Figini et al. presented a novel approach for feature extraction in the context of blood pressure estimation. They employed time windows to clean and refine the extracted features. The proposed method was evaluated using either the MIMIC-III dataset or data collected from the SHIMMER database [50].

To estimate blood pressure, Figini et al. employed various linear machine learning algorithms, including Support Vector Regression (SVR), Random Forest, Ridge Regressor, and Linear Regressor. Their evaluation demonstrated significant results, indicating the effectiveness of their approach.

One notable advantage of their algorithm is its quickly and simplicity, making it suitable for implementation on wearable devices. The proposed method offers promise for real-time blood pressure estimation, leveraging feature extraction techniques and linear machine learning algorithms [50].

3.4 Data accquisition

3.4.1 Pre-processing

In the preprocessing stage of this study, several steps were undertaken to ensure data quality and suitability for further analysis. First, signals that exhibited constant values and contained missing values (NaN) were removed from the dataset. This step aimed to eliminate signals that did not provide meaningful information for analysis.

Next, the arterial blood pressure (ABP) signals were not processed but rather used directly to train, validate, and test the model for predicting systolic blood pressure (SBP) and diastolic blood pressure (DBP), which were the focus of this study.

However, in the case of the electrocardiogram (ECG) and photoplethysmogram (PPG) signals, a preprocessing step was applied. These signals were filtered using a 5th-order Butterworth filter with upper and lower cutoff frequencies (f_H and f_L) set at 1 Hz and 10 Hz, respectively. The purpose of this filtering was to remove any unwanted noise and artifacts present in the signals.

Furthermore, the ECG and PPG signals were verified for missing values and the presence of consecutive constant values. If any ECG or PPG signal contained missing values or had more than 20 percent of consecutive constant values, those signals were eliminated from the dataset. This step aimed to ensure the reliability and quality of the remaining signals for subsequent analysis and modeling.

By performing these preprocessing steps, the dataset was prepared by removing signals with constant values, missing values, and unreliable ECG and PPG signals. This ensured that the data used for training and testing the predictive model was of high quality and suitable for accurate blood pressure estimation.

Authors	**Year**	**Database**	**ML algorithm**	**Performance SBP-DBP**	**Number**
Kachuee et al. [42]	2017	MIMIC-III	Adaptive Boosting	MAE 11.17-5.35	3663
Z. Li et al. [43]	2019	subjects	CNN	MAE 4.44-3.29	120684
Slapničar et al.[45]	2019	MIMIC-III	ResNet-GRU	MAE 9.43 - 6.88	510
Senturk et al. [46]	2020	MIMIC-II	NARX-NN	MAE 3.25 - 1.73	4500
Panwar et al.[47]	2020	UCI dB	LRCN	MAE 3.97 - 2.30	304
Baker et al. [48]	2021	MIMIC-III	CNN-LSTM	MAE 4.41 - 2.91	...
Chih-TA et al. [49]	2022	MIMIC-III	CNN+GRU	MAE 2.44 - 1.40	1551
Figini et al. [50]	2022	MIMIC-III /SHIMMER	Linear Regressor	MAE 3.16 - 2.01	90
Figini et al. [50]	2022	MIMIC-III /SHIMMER	Random Forest	MAE 3.16 - 1.95	90
Figini et al. [50]	2022	MIMIC-III /SHIMMER	Ridge Regressor	MAE 3.20 - 2.01	90
Figini et al. [50]	2022	MIMIC-III /SHIMMER	SVR	MAE 3.20 - 1.83	90
Pin-You et al. [51]	2022	MIMIC-III	CNN	MAE 3.63 - 2.5	300

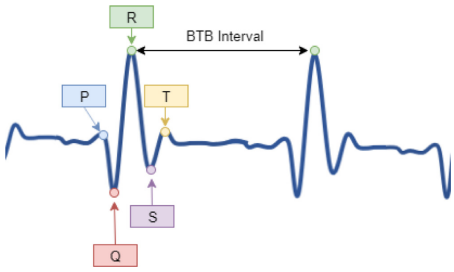
Table 3.1: Summary

3.4.2 Feature extraction

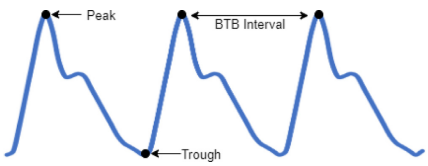
In the feature extraction stage, significant features were extracted from the cleaned data over a time period equivalent to approximately two cardiac cycles (T = 1.5 seconds). Specifically, various time intervals between specific points in the electrocardiogram (ECG) signal were computed. These points included the R-peaks, Q, P, T, Q, and S, as illustrated in Figure 3.2-a.

The extraction of these time intervals provides valuable information about the temporal characteristics and patterns within the ECG signal. These features can capture important aspects of the cardiac cycle, such as the duration of specific segments and the intervals between significant points. Analyzing these time intervals can offer insights into the electrical activity and timing of different phases of the heart's contraction and relaxation.

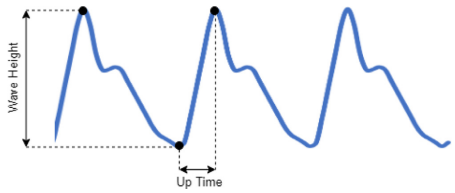
By extracting these significant features from the ECG signal, we can capture essential temporal information that may contribute to the accurate prediction of blood pressure. These features serve as inputs to the predictive model, allowing it to learn and identify patterns in the data that correlate with blood pressure changes. The selection and computation of these specific time intervals provide relevant information for modeling and analyzing the relationship between the ECG signal and blood pressure estimation.



(a) Typical ECG waveform.



(b) Typical PPG waveform.



(c) Calculation of additional PPG features.

Figure 3.2: Features of ECG and PPG[52]

The feature selection process from the ECG signal began with the detection of R-peaks. To accomplish this, a software program was developed using Python,

which employed the *scipy.signal.find_peaks()* function. This function identified all local maxima in the signal, with a distance parameter set to 60. The time interval between two consecutive R-peaks was then designated as the BTB_{ECG} series.

Each value in the BTB_{ECG} series was divided in half, resulting in a value called $\Delta/2$. This value was used to extract additional maximums and minimums from the signal. Specifically, the maximum and minimum values between the R-peak and its preceding $\Delta/2$ were identified and defined as T and S, respectively. Additionally, the period between the R-peak and its prior $\Delta/2$ was calculated.

Furthermore, within this time window, the highest and lowest values for P and Q were determined. In cases where multiple maximum or minimum values were present between each pair of adjacent peaks, the signal qualities were reevaluated. If more than two values were found, the highest value closest to the R-peak was selected as T, while the next highest value closest to the next R-peak was chosen as P. The same procedure was applied to identify the minimum values.

The results of this feature selection process, including the identification of T, P, and S points, as well as the corresponding maximum and minimum values, are depicted in Figure 3.4-a.

By carefully analyzing the characteristics and properties of the ECG signal, this feature selection methodology allows for the extraction of relevant features that capture important aspects of the cardiac cycle. These features will subsequently be utilized in the predictive modeling stage to improve the accuracy and effectiveness of blood pressure estimation.

In the process of identifying the S-peaks of the PPG signal, it was observed that several spurious peaks were present. To ensure accurate detection of only the genuine S-peaks, the amplitude of the peaks underwent kernel density estimation (KDE) analysis [50]. By analyzing the distribution of peak amplitudes, the program identified the minimum value between the two peaks and retained only the peaks with amplitudes above that threshold. This step helped eliminate false peaks and enhance the precision of S-peak detection in the PPG signal.

Two additional features were extracted from the PPG signal: the time interval between two consecutive peaks and the height of the trough to the S-peak. These features provide valuable information about the temporal characteristics and the relative magnitude of specific points within the PPG signal. Figure 3.4-b illustrates the extraction of these features from the PPG signal.

Moving on to the blood pressure estimation, as depicted in Figure 3.4-c, the systolic blood pressure (SBP) was determined as the maximum value in the arterial blood pressure (ABP) signal, representing the peak pressure during each cardiac cycle. Conversely, the diastolic blood pressure (DBP) was determined as the minimum value in the ABP signal, corresponding to the lowest pressure between two consecutive cardiac cycles.

By incorporating these features and accurately identifying the S-peaks in the

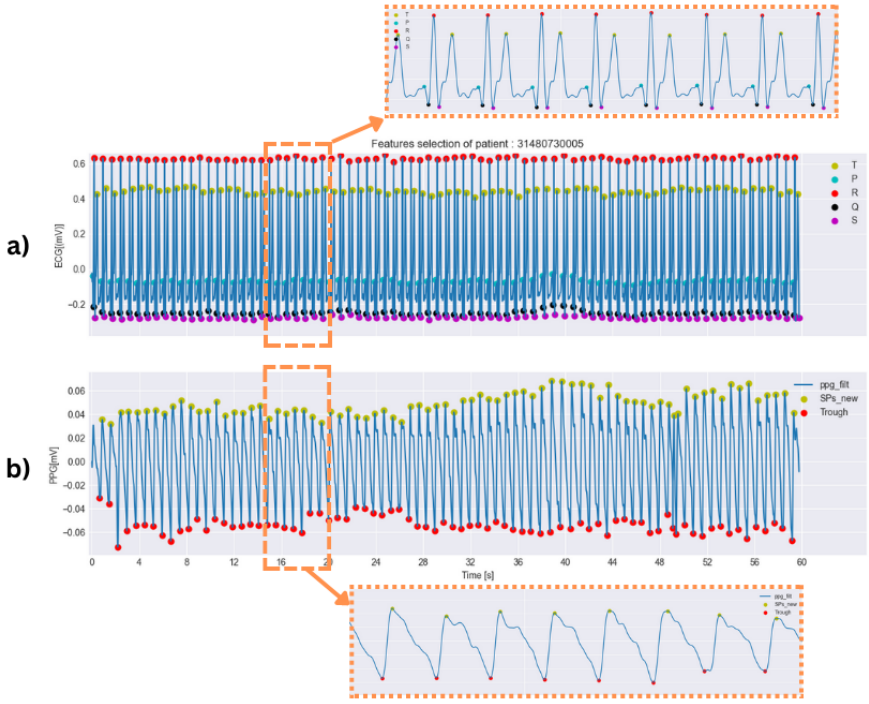


Figure 3.3: a) Features of ECG signal. b) Features of PPG signal.

PPG signal, along with the determination of SBP and DBP from the ABP signal, the study aims to enhance the accuracy and reliability of blood pressure estimation. These features provide valuable insights into the dynamics of the PPG and ABP signals, enabling the development of a robust predictive model for blood pressure assessment.

3.4.3 Remove outliers

The DBSCAN algorithm was employed to detect outliers of the features that extracted of the PPG and ECG signals. It works by grouping together points in a high-density region and separating them from points in low-density regions. The DBSCAN algorithm was chosen for outlier detection because it can handle noise in the data and is less sensitive to parameter selection than other approachs, such as the k-means algorithm.

The DBSCAN algorithm requires two parameters: epsilon (ϵ) and min-samples. The epsilon value determines the radius of the neighborhood around each point, while the min-samples value specifies the minimum number of points required

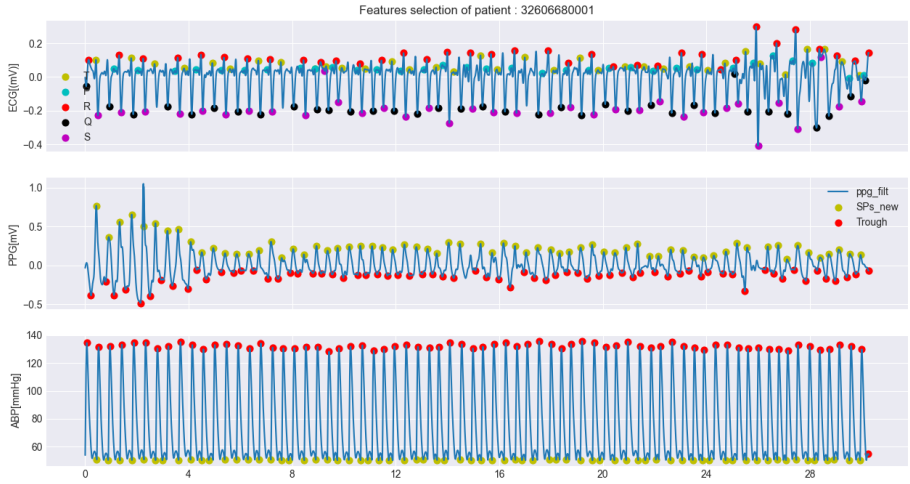


Figure 3.4: a) Features of ECG signal. b) Features of PPG signal.

to form a dense region. In this study, the epsilon value was set to the standard deviation of the signal, which is a common method used to determine this parameter.

The min-samples value was chosen through a trial and error process. Initially, a range of values were tested for min-samples, and the results were compared to the ground truth values to determine the optimal parameter value. We found that setting the minsamples value to 4% of the feature values resulted in the most accurate outlier detection.

Overall, the DBSCAN algorithm was an effective method for detecting outliers in the PPG and ECG signals. The selection of appropriate parameters, such as epsilon and min-samples, was crucial to the success of this method. By using the standard deviation of the signal to determine the epsilon value and setting the min-samples value to 4% of the feature values, we were able to accurately identify outliers in the signals.

3.5 LSTM Neural Network

The developed neural network model comprised a convolutional neural network and a Long short-term memory (LSTM) layer that used P, Q, R, S, and T points and the time interval between R points of ECG signals and S-peaks, troughs, wave heights, up-times, and the time interval between S-peaks, as well as heart rate and PTT. The architecture of the model illustrated in Figure 3.5.

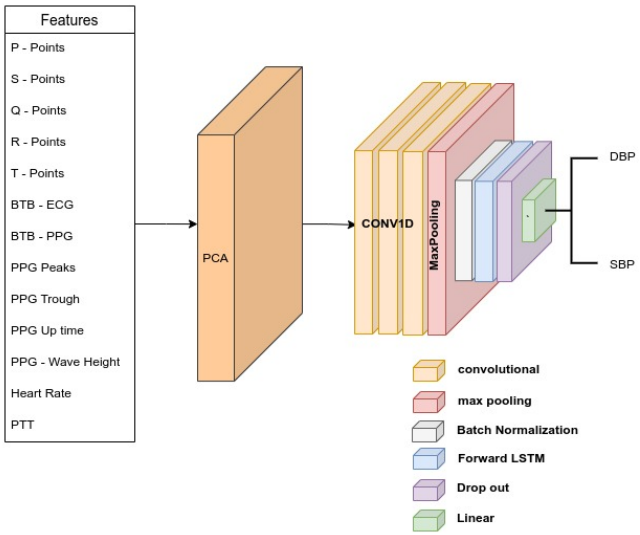


Figure 3.5: Features selected from ECG and PPG and extract the DBP and SBP form ABP.

Before performing the neural network model on the data, the Principal Component Analysis (PCA) technique was used to reduce the dimensionality of datasets to avoid overfitting and increased computational complexity while preserving the maximum amount of information. PCA serves as a technique for transforming a potentially correlated set of variables into a new set of uncorrelated variables known as principal components. These components, which are linear combinations of the original features, are ordered based on the amount of variation they explain within the data. By harnessing the power of PCA, the correlated features undergo a transformation into a lower-dimensional space, where their interdependencies are no longer present.

The primary objective of applying PCA in this particular context is to capture the most significant information from the original features while minimizing the effects of correlation. By reducing the dimensionality of the feature space, PCA enables a more efficient representation of the data, effectively eliminating redundancy stemming from high correlations.

In the developed model, the number of principal components was determined to be 95% of the variance in the data. This was achieved by calculating the percentage of variance explained by each principal component and selecting the minimum number of components required to reach the desired level of variance. The resulting set of principal components is employed as inputs to the neural network model.

As shown in Figure 3.5, the developed model passes the output of the PCA into network comprised of convolutional layers utilized a kernel size of 2 with a stride of

1. The rectified linear unit (ReLU) , a popular activation function in deep learning models, is the activation function utilized in this layer.

The maximum pooling 1D layer is then added to the model, which down-samples the input data along the time dimension in order to lower the dimensionality of the output from the Conv1D layer. With a pool size of 2 and a stride of 2, this layer combines each pair of neighboring time steps into a single output. This lowers the computing cost of the model and aids in helping it discover more broad patterns in the data.

Following the MaxPooling1D layer, the BatchNormalization layer is added, normalizing the activations of the preceding layer across the batch size. This lessens the impact of internal covariate shift and aids in the model's faster convergence during training.

The model is then given the LSTM layer, which is made up of a number of memory cells with the capacity to store data over time. The number of memory cells was considered as hyperparameters of the model, which can be adjusted to strike a compromise between model complexity and performance. The hyperbolic tangent (tanh) is the activation function used in this layer.

After the LSTM layer, a Dropout layer is introduced, which randomly removes some of the layer's units during training. This enhances the model's generalization capabilities and prevents overfitting.

The model is then given a Dense layer with two output units that predicts the values of the systolic and diastolic blood pressure. Because this layer uses a linear activation function, the output values are not restricted to a certain range. Mean squared error is the loss function utilized in the model, while Adam is the optimizer with a learningRate-specified learning rate.

In conclusion, the developed model is an LSTM-based deep learning model that predicts blood pressure values using 13 features taken from ECG, PPG, HR, and PTT signals. Conv1D, MaxPooling1D, BatchNormalization, LSTM, Dropout, and a Dense layer with two output units are the layers that make up the model. With a predetermined learning rate, the model is trained using mean squared error loss and Adam optimizer.

Chapter 4

Results

In this chapter, are presented the results of the study that focused on utilizing the proposed LSTM neural network model for the prediction of continuous blood pressure (BP).

The study aimed to develop a reliable and accurate model for estimating continuous BP values based on the input data. It has been trained and evaluated the LSTM neural network using a comprehensive dataset, which included various physiological signals and relevant features.

The results of this study demonstrate the effectiveness of the proposed LSTM model in predicting continuous BP. Through rigorous evaluation and analysis, promising performance metrics are observed, including low mean absolute error (MAE) and high correlation coefficients, indicating the model's ability to accurately estimate BP values.

Additionally, the LSTM model showcased its capability to capture temporal dependencies and patterns within the data, enabling accurate predictions even for challenging scenarios with complex physiological dynamics. This highlights the significance of leveraging LSTM architectures in handling sequential data for accurate BP estimation.

4.1 Cleaning data

After filtering the data from the MIMIC-III database, the essential features were extracted from the ECG and PPG signals. However, due to the presence of noise, some outliers were identified in the data. To address this issue, the DBSCAN algorithm was employed for outlier removal as part of the pre-processing stage.

In Figure 4.1, it can be observed that the ECG signal at approximately 36 seconds exhibited a large maximum value. However, the DBSCAN algorithm did not identify it as an R-peak and instead chose the average value of its neighboring

Figure 4.1: Features selection without performing DBSCAN.

points, as shown in Figure 4.2. This approach helps to mitigate the impact of outliers and maintain the consistency of the feature.

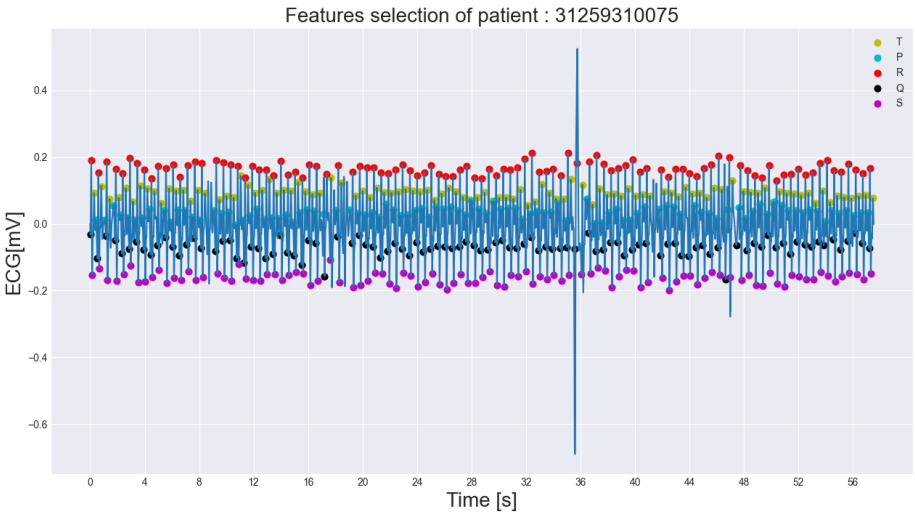


Figure 4.2: Features selection after performing DBSCAN.

The DBSCAN algorithm utilizes two important parameters, epsilon and min-samples, for clustering and identifying outliers. In this case, the epsilon value was

determined as the standard deviation of the signal, while min-samples were set to 4 percent of the feature length. By setting these parameters appropriately, the algorithm can capture the fluctuation of the feature while removing values that fall outside the defined range.

In summary, the DBSCAN algorithm was employed to remove outliers from the extracted ECG and PPG signals. This approach effectively addresses the presence of noise and ensures that the selected features maintain their reliability. By utilizing the epsilon and min-samples parameters, the algorithm preserves the fluctuation of the feature while eliminating values that deviate beyond the defined range.

4.2 Features correlation

One crucial step in the feature selection stage of data pre-processing is feature correlation. The statistical link between several variables or features in a dataset is referred to as feature correlation. Feature correlation helps identify the most relevant and informative features within the ECG and PPG signals. By examining the relationship between different features, researchers can pinpoint the ones that have a strong association with blood pressure. This enables them to focus on the most meaningful and influential features during the modeling process.

Positive correlation indicates that when feature A changes, feature B also changes, and vice versa when feature A changes, feature B also changes. Both properties have a linear connection and move together. A feature A's growth causes feature B's reduction, and vice versa, according to a negative correlation.

Every one of the correlation types has a range of values from 0 to 1, with mildly or strongly positive correlation characteristics being around 0.5 or 0.7. A correlation score value of 0.9 or 1 indicates the outcome when there is a strong and perfect positive connection. A value of -1 will be used to indicate a significant negative association.

Analyzing feature correlations can provide valuable insights into the physiological mechanisms governing blood pressure regulation. Understanding how certain features relate to each other helps unravel the complex interplay between cardiovascular dynamics and the ECG and PPG signals. These insights contribute to a deeper understanding of the physiological processes underlying blood pressure fluctuations.

In Figure 4.3, a striking correlation emerges between the S-peak and trough of the PPG signal and its height. Moreover, the S-peaks and troughs of the signals display a predictability factor. On the subject of ECG signals, a noticeable correlation exists between heart rate and the time interval between two consecutive R-peaks. However, the correlation between other features is comparatively weaker. To address the impact of feature correlations, Principal Component Analysis (PCA)

has been employed to mitigate their influence.

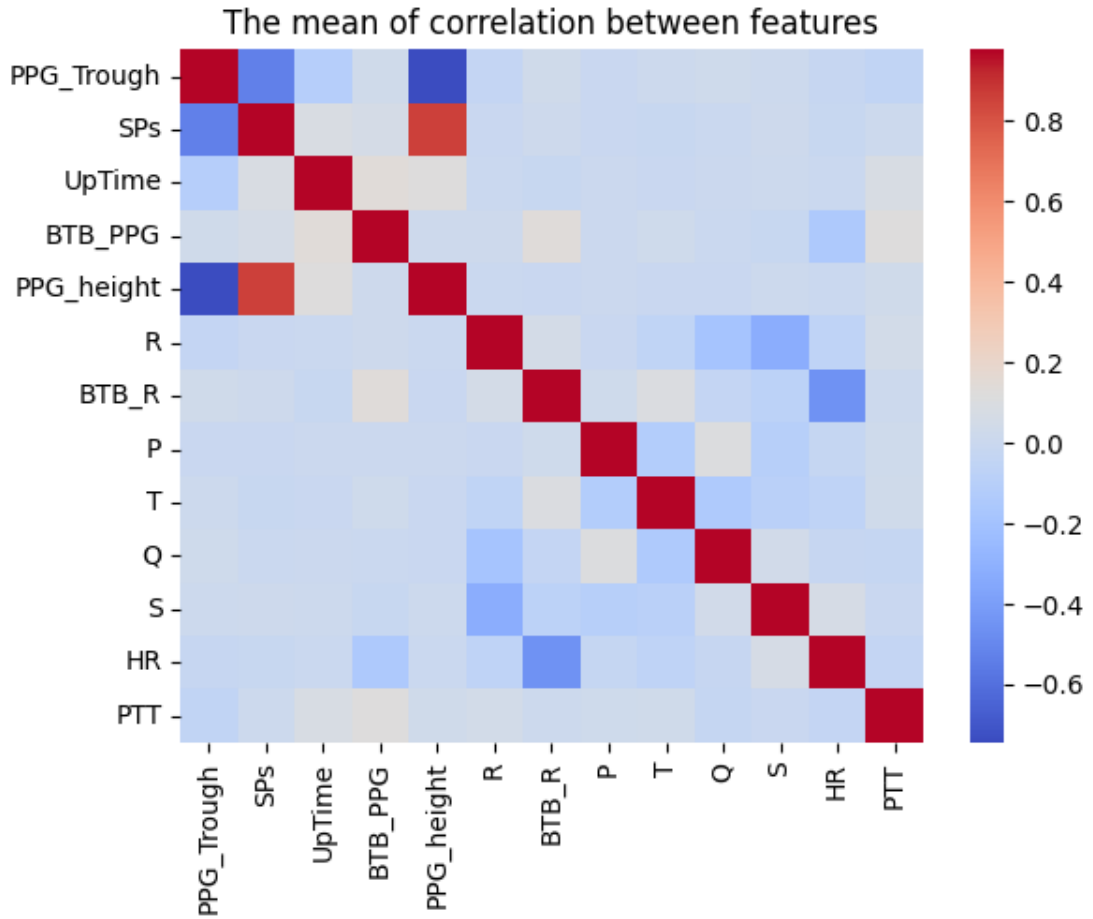


Figure 4.3: The mean of correlation between features.

By employing Principal Component Analysis (PCA), it becomes possible to extract a subset of uncorrelated components that retain the most relevant information from the original features. This application of PCA offers several advantages, including more accurate and robust modeling. The selected components are independent of each other and less influenced by inter-feature correlations, leading to improved predictive capabilities.

To summarize, the observed correlation between specific features in the PPG signal and ECG signal, along with the potential impact on blood pressure estimation, has necessitated the use of PCA. By reducing the influence of correlated features, PCA plays a vital role in enhancing the accuracy and interpretability of predictive models.

4.3 Defined hyperparameters

In the next level for design the model, the hyperparameters was defined with grid search for ten patients to find the best values for hyperparameters. In this part the learning rate and the neurons of convolution and LSTM layer of model consider as hyperparameters the result of grid search is expressed in the table 4.1. As can be seen, all samples reach the best score with a learning rate value of 0.001 and neurons in LSTM layer 128. Moreover, for the value of neurons in the convolution layer, most of the samples reached 128. These values have finally been taken into account for the neural network model.

Patient	Best score	learning rate	CONV neurons	LSTM neurons
31041590003	-0.0048	0.001	128	128
31029120002	-0.0072	0.001	128	128
31034130005	-0.0024	0.001	64	128
31047600001	-0.0095	0.001	64	128
31047600001	-0.0091	0.001	128	128
31061520003	-0.0030	0.001	128	128
3403850	-0.0056	0.001	128	128
3605744	-0.0062	0.001	64	128
3130355001	-0.0081	0.001	128	128

Table 4.1: Grid search results to find hyperparameters of model

4.4 Prediction results

In the evaluation process of the model, a total of 19,676 data points were initially extracted from the MIMIC-III database. However, during the pipeline section, only 3,565 data points were identified as having all the required signals. Subsequently, after the pre-processing step, the dataset was further reduced to 780 instances that could be utilized for evaluating the proposed model.

To improve the model's prediction performance, the *StandardScaler()* function is applied to normalize the data. This normalization process ensures that all the features are on a similar scale, preventing any particular feature from dominating the model's learning process.

For training the model, 65% of the data is considered the training set. This subset was used to train the model, allowing it to learn the underlying patterns and relationships in the data.

Furthermore, 15% of the data was allocated as the validation set. This set was utilized to assess the model's performance during the training phase. It enabled us

to fine-tune the model's hyperparameters and prevent overfitting by monitoring its performance on unseen data. In training phase, 30 epochs are used to train the model and as can be seen that in the figure 111 the loss value in both training and validation data reduced to the value under 0.1 and then it was be constant so that shows the training of the mode not go to overfitting and not underfitting.

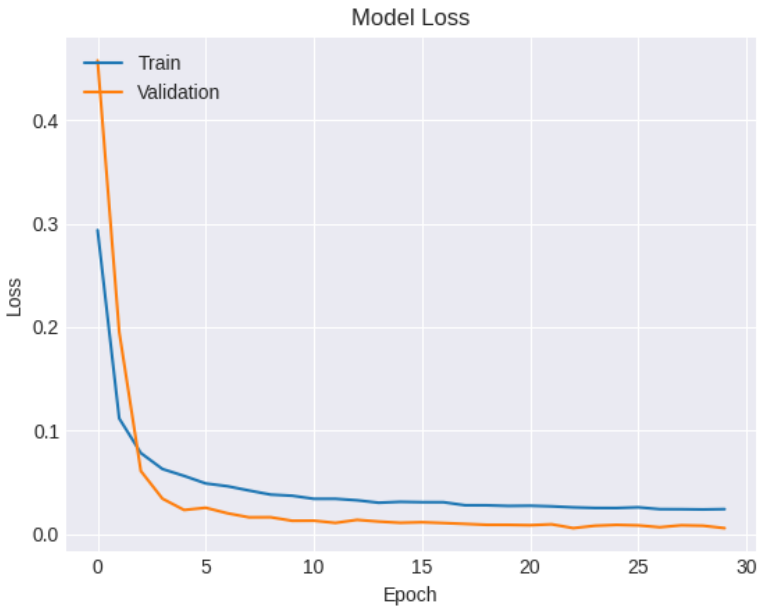


Figure 4.4: Training and validation loss for output.

Finally, the remaining 25% of the data was assigned as the testing set. This set served as an independent evaluation set to assess the final performance of the trained model. Testing the model on this unseen data allowed us to evaluate its generalization ability and its accuracy in making predictions.

To illustrate the prediction results,a sample of patients was selected to presents the estimations of diastolic blood pressure (DBP) and systolic blood pressure (SBP) for both the training and test phases in figure 4.5. Additionally, Figure 4.6 showcases the regression analysis between the real values and the predictive values, providing insights into the relationship and accuracy of the predictions.

The selected sample presented how well the model performs in estimating DBP and SBP. The figures demonstrate the closeness between the predicted values and the real values, indicating the model's ability to capture the underlying patterns and make accurate blood pressure predictions. The regression analysis further validates the reliability of the model, as it shows a strong correlation between the predicted and actual values.

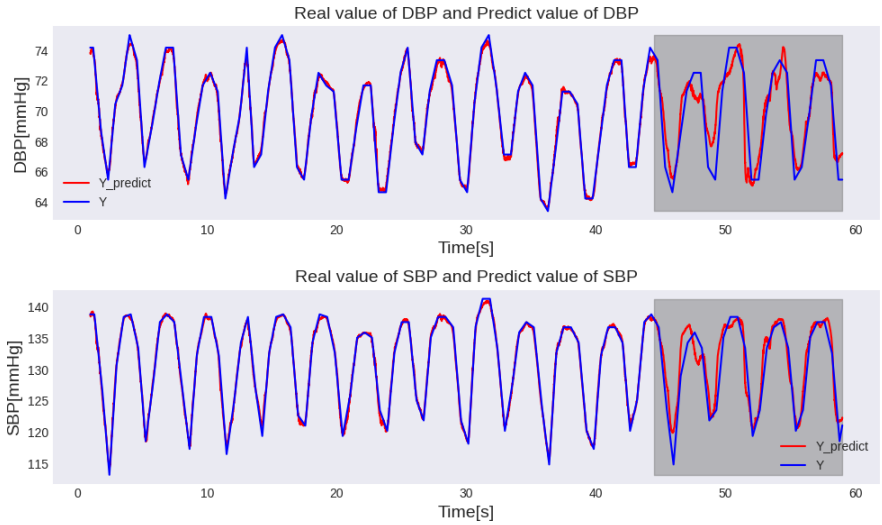


Figure 4.5: Real value and Prediction values for DBP and SBP.

Furthermore, Figure 4.7 presents the histogram of mean errors for the patients. This histogram provides a visual representation of the distribution of errors between the predicted and real values. Analyzing the histogram allows for an assessment of the overall accuracy and consistency of the model's predictions. A well-distributed histogram with a majority of errors close to zero indicates that the model has achieved a high level of accuracy in estimating blood pressure.

The results of the mean absolute error (MAE) for diastolic blood pressure (DBP) are depicted in Figure 4.8. It is evident from the histogram that over 90% of the instances exhibit an MAE below 5 mmHg. This indicates that the model's performance in estimating DBP is generally accurate, with a majority of the predictions falling within a 5 mmHg margin of error.

Similarly, the histogram of MAE for systolic blood pressure (SBP) is illustrated in Figure 4.9. In this case, more than 80% of the instances demonstrate a mean absolute error below 5 mmHg. This implies that the model's predictions for SBP also exhibit a high level of accuracy, with the majority of estimates being within 5 mmHg of the ground truth values.

Mean Arterial Pressure (MAP) is a vital hemodynamic parameter that plays a crucial role in assessing cardiovascular health. It represents the average pressure in the arteries throughout one complete cardiac cycle. MAP is an essential indicator of perfusion pressure, which ensures that organs and tissues receive an adequate blood supply to meet their metabolic demands. Calculating MAP involves considering both the systolic and diastolic blood pressure values. This is because the heart spends a longer duration in diastole (relaxation) compared to systole (contraction).

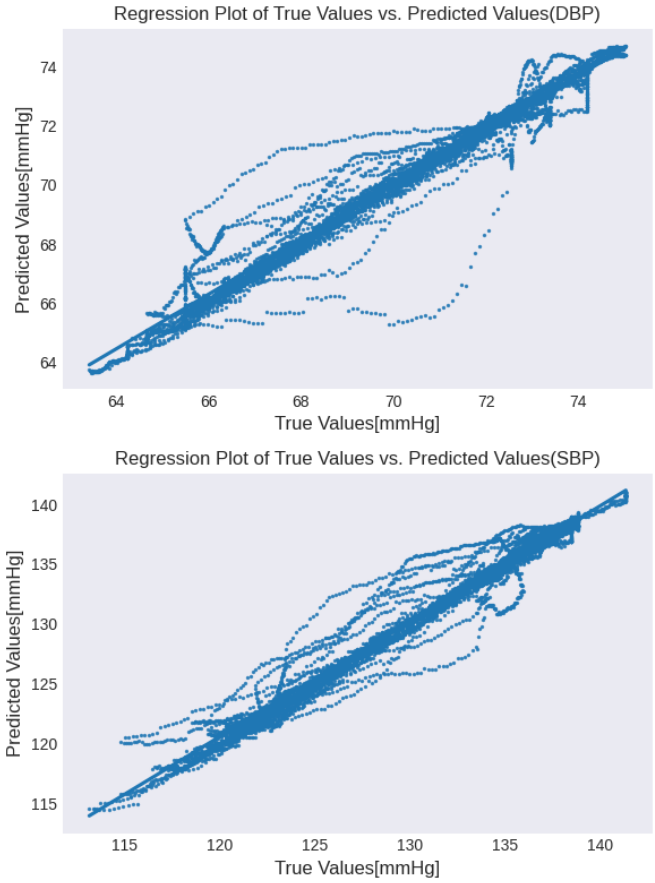


Figure 4.6: Regression between Real value and predict value for DBP and SBP

By incorporating both systolic and diastolic pressures, MAP provides a comprehensive estimation of the driving force for blood flow during the entire cardiac cycle. It offers valuable insights into the pressure experienced by organs and tissues during both relaxation and contraction of the heart. Maintaining an optimal MAP is crucial for ensuring adequate blood perfusion to various organs, including the brain, heart, kidneys, and other vital tissues. Deviations from the normal range of MAP can indicate underlying cardiovascular conditions or disruptions in blood volume regulation.

To assess the accuracy of the Mean Arterial Pressure (MAP) predictions, the Mean Absolute Error (MAE) of MAP has been calculated. The corresponding histogram of the MAE values is presented in Figure 4.10. Notably, it can be observed that over 90% of all errors fall below 5 mmHg.

According to the AAMI standard [53], medical devices are required to meet specific criteria for mean difference (mean absolute error) and standard deviation

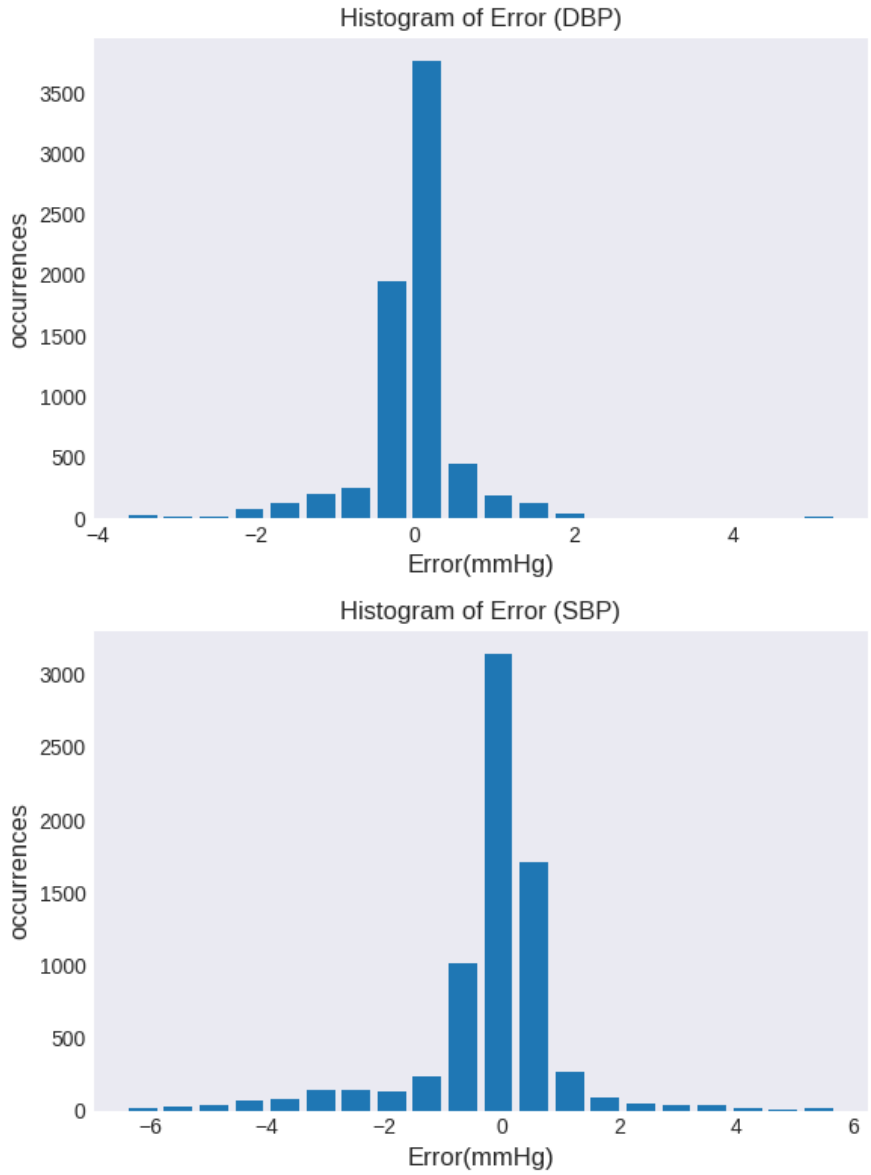


Figure 4.7: Histogram of error for DBP and SBP.

when compared to gold standard measurements. To receive a "Pass" grade, a device must have a mean difference ≤ 5 mmHg and a standard deviation ≤ 8 mmHg. Conversely, if these criteria are not met, the device is assigned a "Fail" grade.

In Table 4.2, the performance of the algorithms is presented, and they comfortably achieve "Pass" grades based on the AAMI criteria. The algorithms demonstrate impressively low mean absolute errors (MAEs) and standard deviations (SDs) in

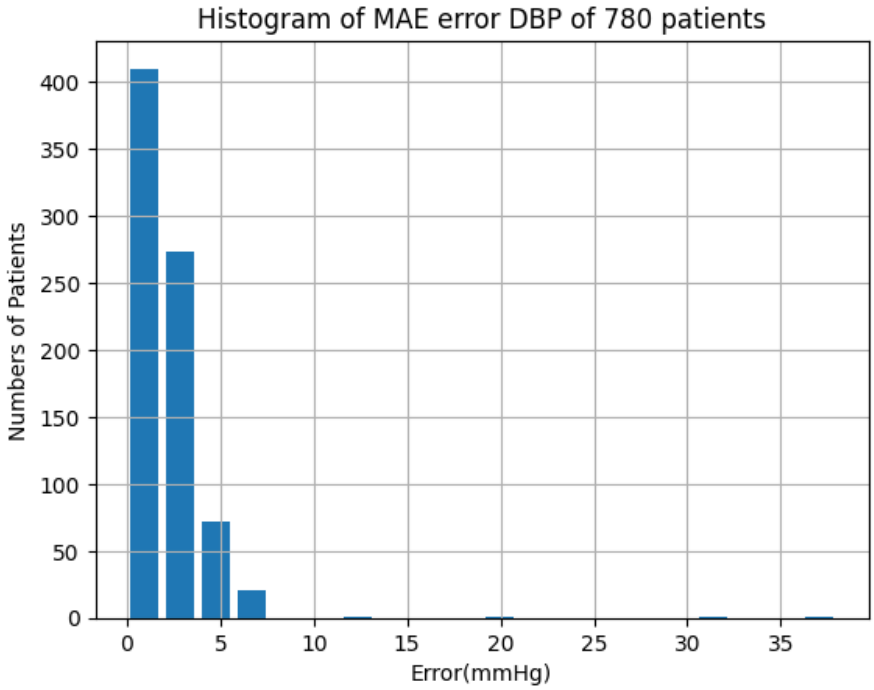


Figure 4.8: DBP Histogram of MAE.

estimating all blood pressure parameters. These results indicate that the algorithms are highly accurate in predicting blood pressure and are suitable for implementation in healthcare settings.

By meeting the stringent requirements set by the AAMI standard, the algorithms provide reliable and precise estimations of blood pressure. The low MAEs and SDs underscore their effectiveness and potential for practical use in healthcare applications.

	MAE(mmHg)	**S**D(mmHg)	Grade
DBP	1.85	2.45	Pass
SBP	3.18	3.48	Pass
MAP	2.01	2.34	Pass

Table 4.2: Assessment of algorithm based on AAMI standard.

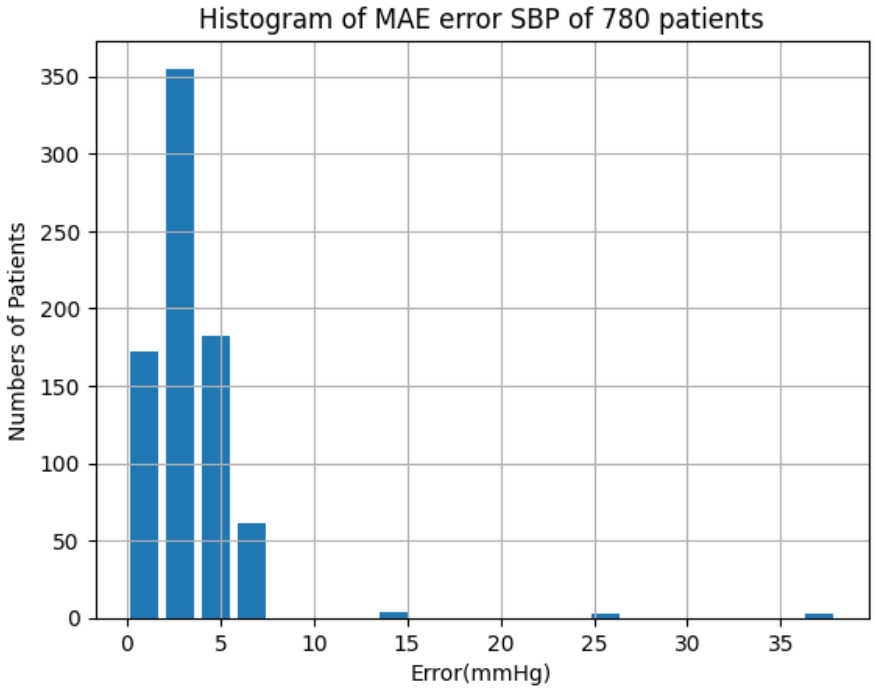


Figure 4.9: SBP Histogram of MAE.

4.5 Features importance

To determine the feature importance of an LSTM model in predicting blood pressure, one approach is to use permutation importance with a scoring metric of negative mean absolute error. Permutation importance measures the decrease in model performance when the values of a specific feature are randomly shuffled while keeping other features unchanged. By calculating the decrease in the mean absolute error (MAE) as a result of permuting each feature, their importance in the LSTM model can be assessed.

The permutation importance function can be applied to the LSTM model with the scoring metric set to 'neg mean absolute error'. This approach involves the following steps:

- Prepare the dataset: Organize the dataset into features (input variables) and target variables (blood pressure values).
- Train the LSTM model: Build and train the LSTM model using the prepared dataset.
- Calculate baseline MAE: Obtain the baseline MAE by evaluating the model's performance on the validation or test set.

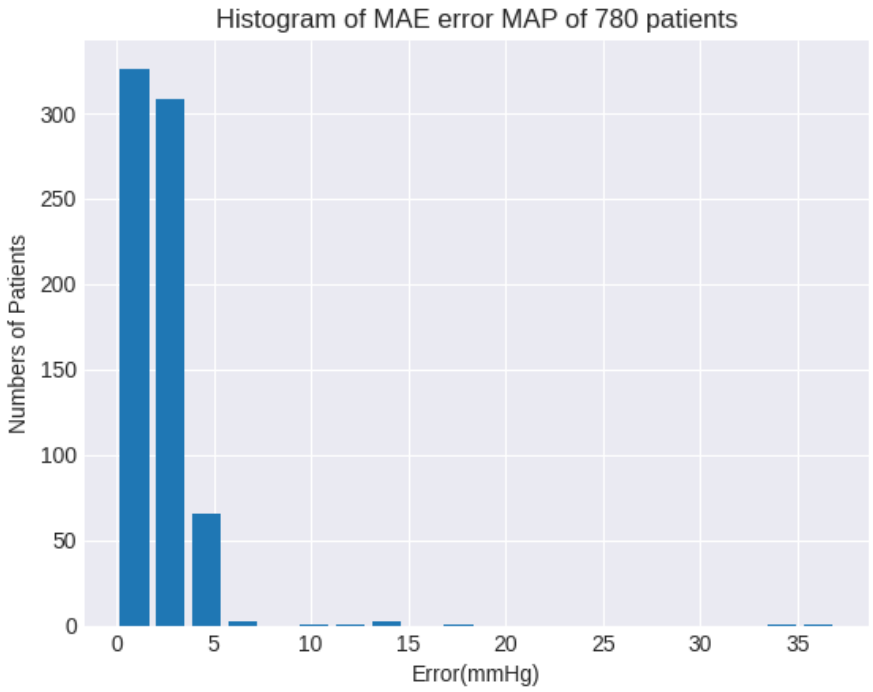


Figure 4.10: MAP Histogram of MAE.

- Compute feature importances: Utilize the permutation_importance function to calculate the feature importances based on the decrease in MAE when each feature is permuted. This analysis measures the impact of each feature on the model's predictive performance.
- Rank the feature importances: Rank the feature importances in descending order to identify the most important features for predicting blood pressure using the CNN-LSTM model.

By following the mentioned steps and applying permutation importance with the 'neg_mean_absolute_error' scoring metric, the results are illustrated in Figure 4.11. The figure showcases the importance scores for each feature, highlighting the variability in importance when the values of the features are permuted. A higher fluctuation of scores indicates that the importance of certain features may vary significantly depending on the permutation of their values.

Furthermore, from the feature importance scores, it can be observed that the duration of the PPG signal and heart rate (HR) values have a more positive effect on the accuracy of the model. This is indicated by their larger mean importance values compared to other features. These results suggest that variations in the duration of the PPG signal and HR values have a significant impact on the model's

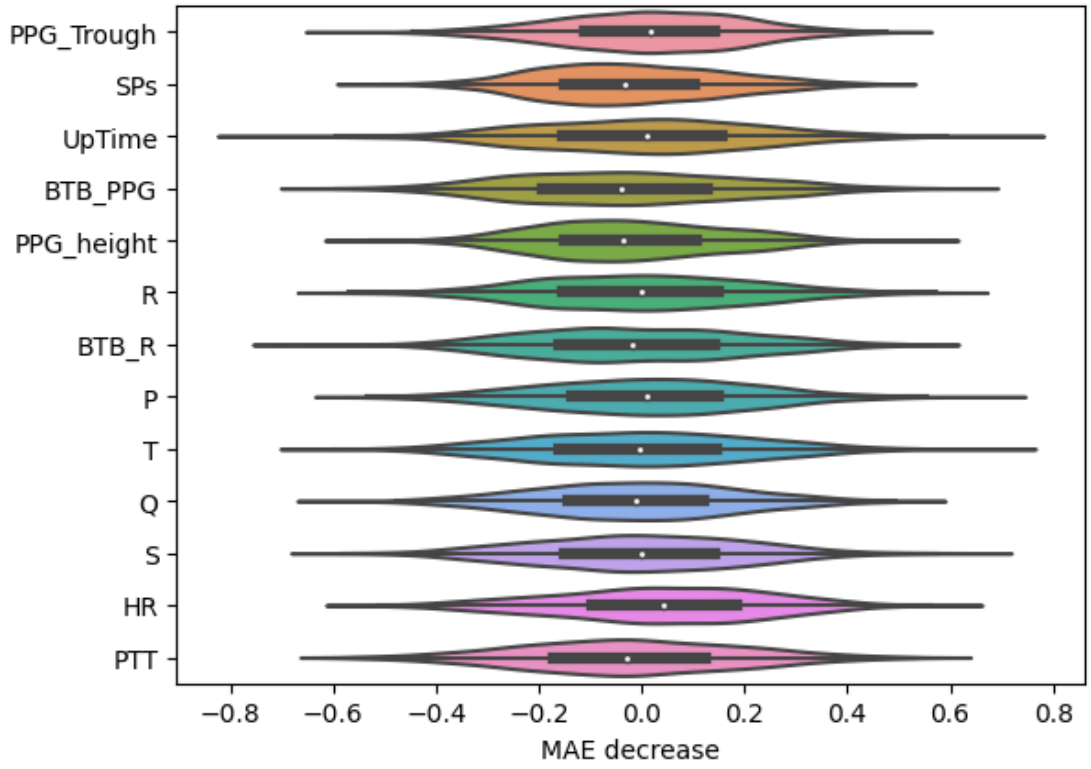


Figure 4.11: Importance Feature.

predictive performance for blood pressure estimation.

In summary, Figure 4.11 presents the feature importance scores obtained through permutation importance analysis with the 'neg_mean_absolute_error' scoring metric. The scores highlight the variability in importance among the features, with the duration of the PPG signal and HR values demonstrating a more positive effect on the model's accuracy due to their larger mean importance values.

The outcomes of the study provide valuable insights into the potential of utilizing CNN-LSTM for continuous BP prediction. These findings hold implications for improving clinical decision-making, enhancing patient monitoring systems, and potentially reducing the need for invasive BP measurement methods.

Chapter 5

Conclusions

In this study, a simple and robust model was developed for the estimation of systolic blood pressure (SBP), diastolic blood pressure (DBP), and mean arterial pressure (MAP) using features extracted from ECG and PPG waveforms. The proposed LSTM neural network model demonstrated strong performance in predicting these parameters, achieving low mean absolute error (MAE) and standard deviation (SD).

The model's performance was evaluated against the AAMI standard, and it was found to meet the criteria, with MAE values below the threshold for both SBP and DBP. This indicates that the predictions made by the model are accurate and reliable, making it suitable for blood pressure estimation in clinical settings.

Additionally, the correlation between the features and their influence on the model's predictions was investigated. By calculating feature importance, the most influential features in predicting SBP and DBP were identified. This information provides valuable insights into the underlying physiological mechanisms and can guide future research and feature selection techniques.

The findings of this study highlight the potential of the proposed LSTM neural network model for continuous and non-invasive blood pressure monitoring. The model's accuracy and robustness make it a promising tool for early detection and management of cardiovascular diseases. Furthermore, the integration of blood pressure monitoring into everyday devices can enable convenient and proactive health monitoring, leading to improved preventive care.

In conclusion, this study has contributed to the field of non-invasive blood pressure estimation and has laid the foundation for further advancements in physiological signal analysis. The developed model has demonstrated its effectiveness and holds promise for future applications in clinical practice and personal health monitoring.

5.1 Future work

There are several aspects in which this project can be further developed:

1.Integration of Additional Features: In future research, the integration of additional physiological features into the LSTM model for estimating continuous blood pressure can be explored. Features such as respiratory rate, or arterial stiffness measurements can provide valuable information about the cardiovascular system. By incorporating these features into the model, it may enhance the accuracy and robustness of blood pressure estimation. This expanded feature set can potentially capture more comprehensive information about the physiological state and improve the overall performance of the predictive model.

2.Real-time Signal Application: The implemented algorithm can be applied to signals acquired in real time, which would enable its effective use in wearable devices. This would provide users with continuous and convenient monitoring of their blood pressure, allowing for timely intervention and management of cardiovascular health.

2. Long-term Monitoring: Continuous monitoring of blood pressure over an extended period is crucial for the prevention of cardiovascular diseases. Future research can focus on developing methods that can reliably and comfortably monitor blood pressure over a longer duration, ensuring patient comfort and minimizing any potential discomfort or damage.

3. Feature Selection Optimization: Further analysis has revealed that removing the least important features can improve prediction time without significantly affecting the accuracy of diastolic blood pressure (DBP) estimation. However, it was observed that reducing the number of features for systolic blood pressure (SBP) estimation led to higher error rates. Future work can explore advanced feature selection techniques to optimize the balance between prediction time and accuracy for both SBP and DBP.

4. Low-power Wearable Device: The development of a low-power wearable device that incorporates the computationally efficient LSTM algorithm can be a promising avenue. Such a device would enable individuals to monitor their blood pressure continuously, even during their daily activities. Clinical trials can be conducted to assess the performance of the device in various applications, such as fitness tracking and telehealth.

By focusing on these aspects, the project can be extended to provide more robust and user-friendly solutions for blood pressure estimation, contributing to the advancement of preventive healthcare and improved management of cardiovascular health.

Appendix A

Python Code

A.1 Read data from server

```
    import wfdb, os
    import pprint
    import pandas as pd
    import numpy as np
    import json
    import matplotlib.pyplot as plt
    from wfdb import processing

    class readMIMIC_III():
        def __init__(self,file='RECORDS', save_dir = 'patient',
    root_dir = './THESIS DATABASE'):
            with open(file,'r') as f:
                self.dbs = [line.strip() for line in f.readlines()]
            self.Num_p , self.n= len(self.dbs) , 0
            self.Num_file = 0
            self.root_dbs = self.dbs[0][0:2]
            self.desired_signal = ['II','PLETH','ABP']
            fs , _duratian = 125 , 60
            self.sig_len = fs * _duratian
            self.sig_len_max = 10 * self.sig_len
            self.root_dir = root_dir
            if not os.path.exists(f'{self.root_dir}/{save_dir}'):
                os.mkdir(f'{self.root_dir}/{save_dir}')
            self.save_dir = save_dir

        def get_record(self,database, seg, patient_dir):
             if '_' in seg and len(seg.split('_')[1]) == 4:
                seg_header = wfdb.rdheader(seg, pn_dir = f'mimic3wdb
    /{self.root_dbs}/{database}')
```

```
28                  if seg_header.__dict__['sig_len'] > self.sig_len and
    all(elem in seg_header.__dict__['sig_name'] for elem in self.
    desired_signal):
29
30                      if seg_header.__dict__['sig_len'] <= self.
    sig_len_max:
31                          record = wfdb.rdrecord(
32                              seg,
33                              pn_dir = f'mimic3wdb/{self.root_dbs}/{
    database}',
34                              channel_names= self.desired_signal)
35                      else :
36                          record = wfdb.rdrecord(
37                              seg,
38                              pn_dir = f'mimic3wdb/{self.root_dbs}/{
    database}',
39                              sampto = self.sig_len_max,
40                              channel_names= self.desired_signal)
41
42                      diffs = np.diff(record.p_signal[:,0])
43                      non_zero_diffs = len([i for i in diffs if round(i
    ,3) == 0])
44                      if non_zero_diffs < self.sig_len :
45                          raise ValueError('the 1 signal are constant')
46
47                      diffs = np.diff(record.p_signal[:,1])
48                      non_zero_diffs = len([i for i in diffs if round(i
    ,3) == 0])
49                      if non_zero_diffs < self.sig_len:
50                          raise ValueError('the 2 signal are constant')
51
52                      diffs = np.diff(record.p_signal[:,2])
53                      non_zero_diffs = len([i for i in diffs if round(i
    ,3) == 0])
54                      if non_zero_diffs < self.sig_len:
55                          raise ValueError('the 3 signal are constant')
56                      df = record.to_dataframe()
57
58                      df.to_csv(f'{self.root_dir}/{self.save_dir}/{seg
    }.csv')
59                      print(f'Number of Patient that was downloads : {
    len(os.listdir(f"{self.root_dir}/{self.save_dir}/"))}/{self.
    Num_desired_pat}')
60                      self.Num_file += 1
61                      print(f'File {seg} was downloaded')
62
63          def run(self, Num_desired_pat = 40):
64              self.Num_desired_pat = Num_desired_pat
65              print(f'Number of patient: {self.Num_p}')
```

```
66              for db in self.dbs[200:]:
67                  try:
68                      if len(os.listdir(f'{self.root_dir}/{self.
    save_dir}/')) < self.Num_desired_pat:
69                          self.n+= 1
70                          layout_hea = wfdb.rdheader(f'{db[:-1]}_layout
    ', pn_dir=f'mimic3wdb/{self.root_dbs}/{db}')
71                          if all(elem in layout_hea.__dict__['sig_name'
    ] for elem in self.desired_signal):
72                              print(f'{self.n}/{self.Num_p} \nThere are
     the desired signals in layout {db[:-1]}')
73
74                              record_hea = wfdb.rdheader(f'{db[:-1]}',
    pn_dir=f'mimic3wdb/{self.root_dbs}/{db}')
75
76                              for seg in record_hea.__dict__['seg_name'
    ]:
77                                  pat_dir = seg.split('_')[0]
78                                  if not os.path.exists(f"{self.
    root_dir}/{self.save_dir}/{pat_dir}"):
79                                      self.get_record(database = db,
    seg = seg, patient_dir=pat_dir)
80
81              except Exception as e:
82                  print(f'Error in {db[:-1]} the error is:-->>\n{e}
    ')
83
84          print(f'Number of file was downloaded  {self.Num_file}')
85
86      def plot_record(self, record_name, sig_start = 0, sig_end =
    None):
87          record = wfdb.rdrecord(
88                      record_name,
89                      pn_dir = f'mimic3wdb/{record_name[0:2]}/{
    record_name.split("_")[0]}',
90                      sampfrom = sig_start,
91                      sampto = sig_end,
92                      channel_names= self.desired_signal)
93
94          record_hea = wfdb.rdheader(record_name, pn_dir=f'
    mimic3wdb/{record_name[0:2]}/{record_name.split("_")[0]}')
95
96          pprint.pprint(f'fs = {record_hea.__dict__}',width=1)
97
98
99          df = record.to_dataframe()
100
101  class read_csv():
102      def __init__(self, file_name):
```

```
103             self.file_name = file_name
104             self.df = pd.read_csv(file_name)
105 
106         def run(self):
107             columns = self.df.columns
108             self.df = self.df.rename(columns={columns[0]: 'Time'})
109             self.df['Time'] = self.df['Time'].apply(lambda x: x.split
    (':')[-1][0:-3])
110 
111             df_new = self.df[(self.df['ABP'] > 50) & (self.df['ABP']
    < 200)]
112 
113             if df_new['ABP'].isnull().all() or df_new['ABP'].nunique
    () == 1:
114                 raise ValueError('There is not Signal ABP')
115 
116             if df_new['II'].isnull().all()  or df_new['II'].nunique()
     == 1:
117                 raise ValueError('There is not Signal ECG')
118 
119             if df_new['PLETH'].isnull().all()  or df_new['II'].
    nunique() == 1:
120                 raise ValueError('There Is not Signal PPG or it is
    constant')
121 
122             fig, axs = plt.subplots(3,1)
123             self.df.plot(ax=axs[0],y='II')
124             self.df.plot(ax=axs[1],y='ABP')
125             self.df.plot(ax=axs[2],y='PLETH')
126             axs[0].set_title(f'File name : {self.file_name}')
127             plt.legend()
128             plt.show()
129             # plt.savefig(f'./onworking/plots/{self.file_name.split
    ("/")[-1].split(".")[0]}.png')
130             plt.close()
131 
132 
133     if __name__=='__main__':
134         file = './DATABASE/ver_2/records/RECORDS-38'
135         data1 = readMIMIC_III(file = file, save_dir= 'patient-38',
    root_dir = './THESIS DATABASE/ver_2')
136         data1.run(Num_desired_pat = np.inf)
```

A.2 Pre-processing of dataset

```
    import matplotlib.pyplot as plt
    import numpy as np
    import pandas as pd
    import scipy
    import os
    from sklearn.cluster import DBSCAN
    from statsmodels.tsa.stattools import adfuller

    from SintecProj import SintecProj
    class PreProcess_Sintec():

        def __init__(self,file = '3400715_pat.csv',pat_dir =  './Patients', result_dir = '.', showplot= False):
            self.showplot = showplot
            self.fs = 125
            self.figsize = (15,9)
            self.patient_path = pat_dir
            self.check_dir(result_dir)
            self.plt_Feat_path = f'{result_dir}/Plots/Plots_features'
            self.regr_path = f'{result_dir}/Dataset'
            self.lower_limit_ABP = 50
            self.upper_limit_ABP = 200
            self.preprocess_data(file)

        def read_data(self,file):
            pat_name = file.split('_')[0]
            if file.split('_')[1] == 'pat.csv':
                df = pd.read_csv(f'{self.patient_path}/{file}', quotechar="'",sep=',',skiprows=[1])
                if df.iloc[0][0][0] == '"':
                    df.columns = [x.replace('"',"") for x in df.columns]
                    df.columns = [x.replace("'","") for x in df.columns]

                    df['Time'] = df['Time'].apply(lambda x: x[3:-2])
                    df[df.columns[-1]] = df[df.columns[-1]].apply(lambda x: x[:-1])
                    df = df.replace('-', np.nan)
                    df.index = df['Time']

                    def_columns = []
                    for x in df.columns:
                        if 'ABP' in x or x=='II' or 'PLETH' in x:
                            def_columns.append(x)
                    df = df[def_columns]
                    df = df.astype(float)
                else:
                    df['Time'] = df['Time'].apply(lambda x: x[1:-1])
```

```
45                      df.index = df['Time']
46                      df = df.replace('-', np.nan)
47                      df = df[['ABP','PLETH','II']]
48                      df = df.astype(float)
49                  if pat_name == '3601980': df = df[0:5021]
50              else:
51                  df = pd.read_csv(f'{self.patient_path}/{file}')
52                  columns = df.columns
53                  df = df.rename(columns={columns[0]:'Time'})
54                  df['Time'] = df['Time'].apply(lambda x: x.split(':')
    [-1][0:-3])
55                  df = df.replace('', np.nan)
56                  df = df.astype(float)
57                  pat_name = pat_name + file.split('_')[1][0:4]
58
59              df = self.cleaning_data(df, pat_name)
60              return df, pat_name
61
62          def cleaning_data(self, df, pat_name):
63
64              df_new = df[(df['ABP'] > self.lower_limit_ABP) & (df['ABP
    '] < self.upper_limit_ABP)]
65
66              if df_new['ABP'].isnull().all() or df_new['ABP'].nunique
    () == 1:
67                  raise ValueError('There is not Signal ABP or the
    signal is constant')
68
69              if df_new['II'].isnull().all()  or df_new['II'].nunique()
     == 1:
70                  raise ValueError('There is not Signal ECG or the
    signal is constant')
71
72              if df_new['PLETH'].isnull().all()  or df_new['PLETH'].
    nunique() == 1:
73                  raise ValueError('There Is not Signal PPG or or the
    signal is constant')
74
75              fig, axs = plt.subplots(3,2)
76              df.plot(ax=axs[0,0],y='II')
77              df.plot(ax=axs[1,0],y='ABP')
78              df.plot(ax=axs[2,0],y='PLETH')
79
80              axs[0,0].set_title(f'Initial Signals of patient: {
    pat_name}')
81
82              df = df.dropna(subset=['ABP'])
83              # df.reset_index(inplace=True)
84
```

```
85              for sig in df.columns[1:]:
86                  diff = df[sig].diff(periods=2)
87                  df = df.loc[diff != 0]
88
89              df.reset_index(inplace=True)
90              df.plot(ax=axs[0,1],y='II')
91              df.plot(ax=axs[1,1],y='ABP')
92              df.plot(ax=axs[2,1],y='PLETH')
93
94
95              axs[0,0].set_title(f'removed constant Signals of patient: {pat_name}')
96              if self.showplot: plt.show()
97
98              return df_new
99
100         def preprocess_data(self, file):
101             self.df, self.patient = self.read_data(file)
102             self.df.index = range(0, len(self.df))
103             # Filtering the signal
104             b, a = scipy.signal.butter(N=5,
105                                        Wn=[1, 10],
106                                        btype='band',
107                                        analog=False,
108                                        output='ba',
109                                        fs=self.fs
110                                        )
111             self.ecg_filt = scipy.signal.filtfilt(b, a, self.df['II'])
112             self.ecg_diff = np.gradient(np.gradient(self.ecg_filt))
113             self.ppg_filt = scipy.signal.filtfilt(b, a, self.df['PLETH'])
114
115             if np.isnan(self.ppg_filt).all():
116                 raise ValueError('After filtering the PPG signal got Nan')
117
118             if np.isnan(self.ecg_filt).all():
119                 raise ValueError('After filtering the ECG signal got Nan')
120
121         def check_dir(self,result_dir):
122             if not os.path.exists(f'{result_dir}/Dataset'):
123                 os.mkdir(f'{result_dir}/Dataset')
124             if not os.path.exists(f'{result_dir}/Plots'):
125                 os.mkdir(f'{result_dir}/Plots')
126             if not os.path.exists(f'{result_dir}/Plots/Plots_features'):
127                 os.mkdir(f'{result_dir}/Plots/Plots_features')
```

```
128
129         def find_dbp_sbp(self):
130             # find DBP/SBP points
131             DBPs, _ = scipy.signal.find_peaks(-self.df['ABP'],
    prominence=.5, distance=30, width=10)
132             SBPs, _ = scipy.signal.find_peaks(self.df['ABP'],
    prominence=.5, distance=30, width=10)
133             return DBPs,SBPs
134
135         def ppg_feature(self):
136             """
137                 Find featurs of PPG Signal: Peak(SP) - Trough(Tr) -
    Up Time - BTB Interval - PPG Height
138             """
139             # find SP peaks/time of Trough (min of PPG) and UpTime
140             SPs, _ = scipy.signal.find_peaks(self.ppg_filt,
    prominence=.05, width=10)
141             SP = SintecProj()
142             SPs_new,_ = SP.PPG_peaks_cleaner(self.ppg_filt, SPs)
143             Trs, _ = scipy.signal.find_peaks(-self.ppg_filt,
    prominence=.05)
144             # print(f'len Trs:{len(Trs)}')
145             # print(f'len SPs new: {len(SPs_new)}--- len Trs: {len(
    Trs)}')
146             # Trs cleaning
147             for i in range(len(SPs_new)-1):
148                 elements = [x for x in Trs if SPs_new[i] < x <
    SPs_new[i+1]]
149                 if len(elements) > 1:
150                     elements.remove(max(elements))
151                     Trs = np.setdiff1d(Trs, elements)
152                 elif len(elements)==0:
153                     tr_l,_ = scipy.signal.find_peaks(-self.ppg_filt[
    SPs_new[i]: SPs_new[i+1]])
154                     Trs.append(SPs_new[i]+max(tr_l))
155
156
157             for index in range(min(len(Trs),len(SPs_new))):
158                 if Trs[index] > SPs_new[index]:
159                     if index ==0:
160                         Trs = np.insert(Trs,index, 0)
161                     else:
162                         tr_l,_ = scipy.signal.find_peaks(-self.
    ppg_filt[SPs_new[index-1]: SPs_new[index]])
163                         Trs = np.insert(Trs,index, SPs_new[index-1]+
    tr_l)
164
165             Trs = self.clean_list(SPs_new,Trs,params='max')
```

```
166             self.ppg_filt = self.removeoutliers(self.ppg_filt,SPs_new
    )
167             self.ppg_filt = self.removeoutliers(self.ppg_filt,Trs)
168             # calculate UpTime
169             UpTime , UpTime_f=[] , []
170             dist={}
171 
172             for i in SPs_new:
173                 dist[i]=[]
174                 for j in Trs:
175                     dist[i].append(i-j)
176 
177             for sp,d in dist.items():
178                 try:
179                     uptime = min([i for i in d if 0<i<40])
180                 except:
181                     uptime =np.nan
182                 UpTime_f.append(uptime)
183 
184             # when there isn't uptime because couldn't find SP or Tr,
     mean of them will be replace by NaN
185             mean_UpTime = int(np.round(np.mean([i for i in UpTime_f
    if not np.isnan(i)])))
186 
187             for i in UpTime_f:
188                 if np.isnan(i):
189                     i = mean_UpTime
190                 UpTime.append(i)
191 
192             # print(f'len UpTime : {len(UpTime)}')
193 
194             # calculate BTB for PPG
195             BTB_ppg = []
196             for index, i in enumerate(SPs_new):
197                 if index < (len(SPs_new)-1):
198                     BTB_ppg.append(SPs_new[index+1] - i)
199             BTB_ppg.insert(0, round(np.mean(BTB_ppg[0:10])))
200 
201             # print(f'len BTB_ppg : {len(BTB_ppg)} - len SPs_new : {
    len(SPs_new)}')
202 
203             # calculate PPG height
204             PPG_h = []
205             loc_PPG_h = []
206             for index,i in enumerate(Trs):
207                 try:
208                     if i<SPs_new[index]:
209                         PPG_h.append(self.ppg_filt[SPs_new[index]]-
    self.ppg_filt[i])
```

```
                        loc_PPG_h.append(SPs_new[index])
                except:
                    break
            return SPs_new, Trs, UpTime, PPG_h, BTB_ppg, loc_PPG_h

        def find_smallest_greater(self, arr, i):
            min_val, min_index = None, None
            for index, val in enumerate(arr):
                if val > i and (min_val is None or val < min_val):
                    min_val = val
                    min_index = index
            return min_val, min_index

        def ecg_feature(self):
            """
                Find featurs of ECG Signal: R,T,P,Q,S series - BTB Interval
            """
            # find time of R,T,P,Q,S series
            Rs, _ = scipy.signal.find_peaks(self.ecg_filt, prominence=.05, distance=60)
            RTs, _ = scipy.signal.find_peaks(self.ecg_filt, prominence=.05, distance=20)
            RTPs, _ = scipy.signal.find_peaks(self.ecg_filt, prominence=.05, distance=10)
            T_LMin, _ = scipy.signal.find_peaks(-self.ecg_filt, prominence=.16, distance=10)  #.16

            BTB_R = []
            for index, i in enumerate(Rs):
                if index < (len(Rs)-1):
                    BTB_R.append(Rs[index+1] - i)
            try:
                BTB_R.insert(0, round(np.mean(BTB_R[0:10])))
            except:
                pass

            Ts = [i for i in RTs if i not in Rs]
            Ps = [i for i in RTPs if i not in RTs]
            # print(f'len Ts {len(Ts)} - len RTs: {len(RTs)} - len Rs: {len(Rs)} - len RTPs:{len(RTPs)} - len Ps: {len(Ps)}')

            # define delta_T for each R_peak to find Q and S
            pre_i = 0
            Rs_dT = []
            for index, i in enumerate(Rs):
                if index == 0:
                    delta_T = abs(i - Rs[index+1])/2
                    Rs_dT.append((i, (0, i+delta_T)))
```

```
253                     pre_i = i
254
255                 elif index == len(Rs)-1:
256                     Rs_dT.append((i, (i-int((i - pre_i)/2), len(self.
    ecg_filt))))
257
258                 else:
259                     Rs_dT.append((i, (i-int((i - pre_i)/2), i+abs(i -
     Rs[index+1])/2)))
260                     pre_i = i
261             # print(f'T_LMin{len(T_LMin)}')
262             for rs,dt in Rs_dT:
263                 existP = self.existParam(Ts,lowLim=rs,upperLim=dt[1])
264                 if not existP:
265                     # print(type(self.ecg_filt))
266                     # print(rs,int(dt[1]))
267                     if int(dt[1])>len(self.ecg_filt):
268                         _series = self.ecg_filt[rs:-1]
269                     else:
270                         _series=self.ecg_filt[rs:int(dt[1])]
271                     Ts_new, _ = scipy.signal.find_peaks(_series,
    prominence=.01, distance=5)
272                     try:
273                         Ts.append(rs+min(Ts_new))
274                     except:
275                         pass
276                     # print(rs+Ts_new)
277                 existP = self.existParam(Ps,lowLim=dt[0],upperLim =
    rs)
278                 if not existP:
279                     _series=self.ecg_filt[int(dt[0]):rs]
280                     Ps_new, _ = scipy.signal.find_peaks(_series,
    prominence=.01, distance=5)
281                     try:
282                         Ps.append(dt[0]+max(Ps_new))
283                     except:
284                         pass
285
286
287             # Find Qs, Ss
288             Qs, Ss = [], []
289             for i in T_LMin:
290                 for rs, dt in Rs_dT:
291                     if dt[0] <= i <= dt[1]:
292                         if i < rs:
293                             Qs.append(i)
294                         else:
295                             Ss.append(i)
296
```

```
297             for rs,dt in Rs_dT:
298                 exist_Qs = self.existParam(Qs,lowLim=dt[0],upperLim=
    rs)
299                 if not exist_Qs:
300                     _series=self.ecg_filt[int(dt[0]):rs]
301                     q_min, _ = scipy.signal.find_peaks(-_series,
    prominence=.01, distance=1)
302                     try:
303                         Qs.append(dt[0] + max(q_min))
304                     except Exception as e:
305                         # print('fall', e)
306                         pass
307                 exist_ss = self.existParam(Ss,lowLim=rs,upperLim=dt
    [1])
308                 if not exist_ss:
309                     _series = self.ecg_filt[rs:int(dt[1])]
310                     s_min,_ = scipy.signal.find_peaks(-_series,
    prominence=.01, distance=5)
311                     try:
312                         Ss.append(rs + min(s_min))
313                     except Exception as e:
314                         # print('fall', e)
315                         pass
316
317             Ts = self.clean_list(Rs,Ts, params='min')
318             Ps = self.clean_list(Rs,Ps, params='max')
319             Qs = self.clean_list(Rs,Qs, params='max')
320             Ss = self.clean_list(Rs,Ss, params='min')
321             return Rs, Ts, Ps, Qs, Ss, BTB_R
322
323         def values_between(self, lst, a, b):
324             result = []
325             for value in lst:
326                 if a <= value <= b or b <= value <= a:
327                     result.append(value)
328             return result
329
330         def clean_list(self,main_list,chck_list,params):
331             for i in range(len(main_list)-1):
332                 results = self.values_between(chck_list, main_list[i
    ], main_list[i+1])
333                 if len(results)>1:
334                     if params == 'max':
335                         value = max(results)
336                     elif params == 'min':
337                         value = min(results)
338                     results.remove(value)
339                     chck_list = [x for x in chck_list if x not in
    results]
```

```
340                     # chck_list.remove(rm_value)
341             return chck_list
342
343         def existParam(self,_list,lowLim,upperLim):
344             existP = False
345             for param in _list:
346                 if lowLim< param<= upperLim:
347                     existP = True
348                     break
349             return existP
350
351         def Build_DataFrame(self):
352
353             SPs_new, Trs, UpTime, PPG_h, BTB_ppg, loc_PPG_h = self.
    ppg_feature()
354
355             Rs, Ts, Ps, Qs, Ss, BTB_R = self.ecg_feature()
356
357             self.ecg_filt = self.removeoutliers(self.ecg_filt,Rs)
358             self.ecg_filt = self.removeoutliers(self.ecg_filt,Ts)
359             self.ecg_filt = self.removeoutliers(self.ecg_filt,Qs)
360             self.ecg_filt = self.removeoutliers(self.ecg_filt,Ss)
361             self.ecg_filt = self.removeoutliers(self.ecg_filt,Ps)
362
363             DBPs,SBPs = self.find_dbp_sbp()
364
365             time = np.arange(0,(max(max(Rs), max(SPs_new), max(Ts),
366                                     max(Qs), max(Ss), max(Ps),max(Trs
    ))+1),1)
367             real_time = time/self.fs
368             # print(f'len real_time: {len(real_time)}')
369             # print(f'max(Rs): {max(Rs)}\nmax(SPs_new): {max(SPs_new)
    }\nmax(T): {max(Ts)}')
370             # Find Peak, Trough, UpTime, BTB of Peak and PPG height
    on PPG Signal
371             df_Output = pd.DataFrame({'Time':real_time})
372             df_Output.index = np.arange(len(real_time))
373             df_Output['ppg_filt'] = self.ppg_filt[0:len(real_time)]
374             df_Output.loc[Trs,'Tr'] = df_Output['ppg_filt'].iloc[Trs]
375             df_Output['SPs_new'] = df_Output['ppg_filt'].iloc[SPs_new
    ]
376             df_Output.loc[SPs_new,'UpTime'] = [i/self.fs for i in
    UpTime]
377             df_Output.loc[SPs_new,'BTB_PPG'] = [i/self.fs for i in
    BTB_ppg]
378
379             df_Output.loc[loc_PPG_h,'PPG_h'] = PPG_h
380             abp = self.df['ABP'].values
381             df_Output['ABP'] = abp[0:len(real_time)]
```

```
382             df_Output['DBP'] = self.df['ABP'].iloc[DBPs]
383             df_Output['SBP']= self.df['ABP'].iloc[SBPs]
384 
385             # Find R,P,T,Q,S on ECG Signal
386             df_Output['ecg_filt'] = self.ecg_filt[0:len(real_time)]
387             df_Output['R'] = df_Output['ecg_filt'].iloc[Rs]
388             df_Output.loc[Rs,'BTB_R']=[i/self.fs for i in BTB_R]
389             df_Output['P'] = df_Output['ecg_filt'].iloc[Ps]
390             df_Output['T'] = df_Output['ecg_filt'].iloc[Ts]
391             df_Output['Q'] = df_Output['ecg_filt'].iloc[Qs]
392             df_Output['S'] = df_Output['ecg_filt'].iloc[Ss]
393 
394             # Find PTT and HR
395             SP = SintecProj()
396             dataset = SP.find_PTT(self.ecg_filt,Rs,self.ppg_filt,
    SPs_new,self.patient)
397             hr = dataset['HR'].values
398             ptt = dataset['PTT'].values
399 
400             d = abs(len(dataset['HR'])-len(df_Output['ecg_filt']))
401             if d != 0:
402                 # print(d)
403                 for i in range(d):
404                     hr = np.append(hr, np.nan)
405                     ptt= np.append(ptt, np.nan)
406 
407             df_Output['HR'] = hr
408             df_Output['PTT'] = ptt
409             return df_Output
410 
411         def plot_feature(self,df_Output):
412             # plt.close('all')
413             fig, axs = plt.subplots(3, 1,sharex=True)
414             fig.set_size_inches(15,9)
415 
416             axs[0].plot(df_Output['ecg_filt'])
417             axs[0].scatter(x=df_Output.index, y=df_Output['T'], c='y'
    , s=40, label='T')
418             axs[0].scatter(x=df_Output.index, y=df_Output['P'], c='c'
    , s=40, label='P')
419             axs[0].scatter(x=df_Output.index, y=df_Output['R'], c='r'
    , s=40, label='R')
420             axs[0].scatter(x=df_Output.index, y=df_Output['Q'], c='k'
    , s=40, label='Q')
421             axs[0].scatter(x=df_Output.index, y=df_Output['S'], c='m'
    , s=40, label='S')
422             axs[0].set_ylabel('ECG[(mV)]')
423 
424             df_Output['ppg_filt'].plot(ax=axs[1])
```

```
425             axs[1].scatter(x=df_Output.index, y=df_Output['SPs_new'],
     c='y', s=40, label='SPs_new')
426             axs[1].scatter(x=df_Output.index, y=df_Output['Tr'], c='r
    ', s=40, label='Trough')
427             axs[1].set_ylabel('PPG[mV]')
428
429             df_Output['ABP'].plot(ax=axs[2])
430             axs[2].scatter(x=df_Output.index, y=df_Output['SBP'], c=
    'r', s=40, label='SBP')
431             axs[2].scatter(x=df_Output.index, y=df_Output['DBP'],c='y
    ',s=40, label='DBP')
432             axs[2].set_ylabel('ABP')
433
434             x_ticks = np.arange(0, len(self.ppg_filt)+1, 500)
435             for i in range(2):
436                 axs[i].set_xlabel('Time [s]')
437                 axs[i].set_xticks(x_ticks)
438                 axs[i].set_xticklabels((x_ticks/125).astype(int))
439
440                 axs[i].legend()
441                 axs[i].grid('both')
442             axs[0].set_title(f'Features selection of patient : {self.
    patient}')
443             plt.savefig(f'{self.plt_Feat_path}/{self.patient}_feat.
    png')
444             if self.showplot: plt.show()
445             plt.close()
446
447         def main(self):
448
449             df_Output = self.Build_DataFrame()
450             self.plot_feature(df_Output)
451             interp_output = df_Output.interpolate(method='polynomial'
    ,order=1)
452             self.Med_df_Out = interp_output.round(3)
453             self.Med_df_Out.to_csv(f'{self.regr_path}/{self.patient}.
    csv')
454
455         def removeoutliers(self, _data, _list):
456
457             min_samplesv = round(0.04 * len(_list))
458             tmp_data = _data[_list]
459             epsv = np.std(tmp_data)
460             # print(f'epsv = {epsv}')
461             Np=len(tmp_data)
462             _datareshape = np.reshape(tmp_data,(Np, 1))
463             clusters = DBSCAN(eps=epsv, min_samples=min_samplesv,
    metric='euclidean').fit(_datareshape)
```

```
464             numOutliers = np.argwhere(clusters.labels_ == 1|-1|2).
    flatten()
465             # print(numOutliers)
466             for i in numOutliers:
467                 if i == 0 or i==1 or i==2 or i==3:
468                     mean_value = np.mean(_data[_list[i+1:7]])
469                 elif i == len(_list)-1 or i == len(_list)-2 or i ==
    len(_list)-3 or i == len(_list)-4:
470                     mean_value = np.mean(_data[_list[i-6:i]])
471                 else:
472                     mean_value = (np.mean(_data[_list[i-3:i]]) + np.
    mean(_data[_list[i+1:i+4]]))/2 #(_data[_list[i-1]] + _data[_list[i
    +1]])/2
473 
474                 _data[_list[i]] = mean_value
475             return _data
476 
477     if __name__=='__main__':
478 
479         df_err = pd.DataFrame(columns=['patient', 'error'])
480 
481         for n,file in enumerate(os.listdir(patients_folder)[3400:]):
482 
483             if len(file.split('.')) > 1:
484                 if file.split('.')[1] == 'csv':
485                     patient = file.split('_')[0]
486                     print(f'Patient: {patient} - {n}\{len(os.listdir(
    patients_folder))}')
487                 try:
488                     PrePS = PreProcess_Sintec(file=file,pat_dir =
    patients_folder,result_dir=result_dir)   #3600490    (3602521
    shekam dard)
489                     PrePS.main()
490                 except Exception as e:
491                     print(f"{patient} didn't complete\n{e}")
492                     df_err.loc[len(df_err)] = {'patient':patient,'
    error':e}
493 
494         df_err.to_csv(f'{result_dir}/Feat_Error.csv')
495 
496         print(len(os.listdir(f'{result_dir}/Dataset')))
```

A.3 Traing and Testing the LSTM Model

```
1     # -*- coding: utf-8 -*-
```

```
    import pandas as pd
    import numpy as np
    import os, datetime, time
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.metrics import mean_absolute_error
    from sklearn.model_selection import GridSearchCV
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler
    from sklearn.inspection import permutation_importance
    # import tensorflow as tf
    import keras
    from sklearn.model_selection import KFold
    from keras.models import Sequential
    from keras.layers import Conv1D, LSTM, Dense, BatchNormalization, Dropout, Reshape
    from keras.layers.convolutional import MaxPooling1D
    from keras.wrappers.scikit_learn import KerasRegressor
    from sklearn.model_selection import train_test_split
    # from keras.callbacks import TensorBoard

    class lstm_sintec(object):
        '''
        LSTM NN Model for Prediction the Blood Presure
        '''
        def __init__(self, patient, showplot= False,root_dir =''):
            self.TRAIN_PERC = 0.75
            self.regr_path = f'{root_dir}/Dataset'
            self.check_dir(root_dir)
            self.plot_path = f'{root_dir}/Plots_ML'
            self.final_model = f'{root_dir}/Final_Model'
            self.log_dir = f'{root_dir}/logs/fit/'
            self.patient = patient
            self.df = pd.read_csv(f'{self.regr_path}/{patient}.csv').dropna()
            self.input_col = ['Tr', 'SPs_new', 'UpTime', 'BTB_PPG', 'PPG_h',
                              'R', 'BTB_R', 'P', 'T', 'Q', 'S', 'HR','PTT']
            output_col = ['DBP', 'SBP']
            self.X = self.df[self.input_col]
            self.y = self.df[output_col]
            self.time = self.df['Time'].values
            self.showplot = showplot
            plt.style.use('seaborn-darkgrid')
            self.figsize = (15,9)

        def check_dir(self,root_dir):
            if not os.path.exists(f'{root_dir}/Plots_ML'):
```

```
47                os.mkdir(f'{root_dir}/Plots_ML')
48            if not os.path.exists(f'{root_dir}/Final_Model'):
49                os.mkdir(f'{root_dir}/Final_Model')
50            if not os.path.exists(f'{root_dir}/logs/fit'):
51                os.mkdir(f'{root_dir}/logs')
52                os.mkdir(f'{root_dir}/logs/fit')
53
54        def data_prepare(self):
55            # Normaliziation
56            x = self.X.values
57            y = self.y.values
58            scale_x = StandardScaler()
59            X_scaled = scale_x.fit_transform(x)
60
61            scale_y = StandardScaler()
62            y_scaled = scale_y.fit_transform(y)
63
64            # print(f"Number of features before PCA: {X_scaled.shape
    [1]}")
65            # Apply PCA to the data
66            self.pca = PCA(n_components=0.95)
67            X_scaled = self.pca.fit_transform(X_scaled)
68
69            train_size = int(self.TRAIN_PERC*len(X_scaled))
70
71            x_test, y_test = X_scaled[train_size:], y_scaled[
    train_size:]
72            X_scaled, y_scaled = X_scaled[:train_size], y_scaled[:
    train_size]
73
74            # Divide the data into train, and test sets
75            x_train, x_val, y_train, y_val = train_test_split(
    X_scaled, y_scaled, test_size=0.15, random_state=0)
76
77            data = {
78                'Train':{'x':x_train,'y':y_train},
79                'Val':{'x':x_val,'y':y_val},
80                'test':{'x':x_test,'y':y_test},
81                'Dataset':{'x':X_scaled,'y':y_scaled},
82                'scaler':{'x':scale_x,'y':scale_y}
83            }
84            return data
85
86        def get_model(self, n_features, units1=128,units2=128,
    _learningRate=.01):
87
88            model = Sequential(name='model_LSTM')
89            model.add(Reshape((n_features, 1), input_shape=(
    n_features,)))
```

```
90              model.add(Conv1D(units1, kernel_size=2,
91                               activation='relu', input_shape=(n_features, 1),
    name='Conv1D_1'))
92              model.add(MaxPooling1D(pool_size=2, strides=2, name='
    MaxPooling1D_1'))
93              model.add(BatchNormalization())
94              # model.add(Conv1D(units2, kernel_size=2,
95              #                  activation='relu', name='Conv1D_2'))
96              # model.add(MaxPooling1D(pool_size=2, strides=2, name='
    MaxPooling1D_2'))
97              model.add(LSTM(units2, activation='tanh', name='LSTM'))
98              model.add(Dropout(0.5))
99              model.add(Dense(2, activation='linear', name='Dense'))  #
    scaled_sigmoid
100
101             opt = keras.optimizers.Adam(learning_rate =_learningRate)
102             model.compile(loss='mean_squared_error',
103                           optimizer=opt, metrics=['accuracy'])
104             return model
105
106         def history_plot(self, history):
107             # plot_history(history)
108             plt.figure()
109             plt.plot(history.history['loss'])
110             plt.plot(history.history['val_loss'])
111             plt.title('Model Loss')
112             plt.ylabel('Loss')
113             plt.xlabel('Epoch')
114             plt.legend(['Train', 'Validation'], loc='upper left')
115             plt.savefig(f'{self.plot_path}/{self.patient}_loss.png')
116             if self.showplot: plt.show()
117             plt.close()
118
119         def plot_pred(self,model,X_scaled,y_scaled, y_test,
    y_test_hat,data):
120
121             y_hat = model.predict(X_scaled)
122             y_hat = data['scaler']['y'].inverse_transform(y_hat)
123             y = data['scaler']['y'].inverse_transform(y_scaled)
124             # y_hat = y_hat * self.ssy + self.mmy
125             # y = y_scaled * self.ssy + self.mmy
126
127             y = np.concatenate((y,y_test), axis=0)
128             y_hat = np.concatenate((y_hat,y_test_hat), axis=0)
129
130             fig, axs = plt.subplots(2, 2)
131             fig.set_size_inches(self.figsize)
132             for i in range(2):
133                 if i==0:
```

```
                    _title = 'DBP'
                else:
                    _title = 'SBP'

                # plot regression plot
                sns.regplot(ax=axs[i,0],x=y[:,i],y=y_hat[:,i],
    scatter_kws={'s': 2})
                axs[i,0].set_xlabel("True Values[mmHg]")
                axs[i,0].set_ylabel("Predicted Values[mmHg]")
                axs[i,0].set_title(f"Regression Plot of True Values
    vs. Predicted Values({_title})")

                # Plot error histogram
                error = y [:,i]- y_hat[:,i]
                axs[i,1].hist(error, bins=20, rwidth=0.8)
                axs[i,1].set_xlabel("Error(mmHg)")
                axs[i,1].set_ylabel("Frequency")
                axs[i,1].set_title(f"Histogram of Error ({_title})")

            plt.savefig(f'{self.plot_path}/{self.patient}_error_plt.
    png')
            plt.tight_layout()
            if self.showplot: plt.show()
            plt.close()

            fig,axs = plt.subplots(2)
            fig.set_size_inches(self.figsize)
            for i in range(2):
                if i==0:
                    _title = 'DBP'
                else:
                    _title = 'SBP'
                        # Plot Y and Y_hat
                axs[i].plot(self.time, y_hat[:,i], c='r', label = '
    Y_predict')
                axs[i].plot(self.time, y[:,i], c='b', label = 'Y')
                axs[i].set_ylabel('DBP[mmHg]')
                axs[i].set_xlabel('Time[s]')
                axs[i].set_title(f'Real value of {_title} and Predict
     value of {_title}')
                axs[i].legend()
                # represent the test region
                axs[i].fill_betweenx([min(y[:,i]),max(y[:,i])], self.
    time[int(self.TRAIN_PERC*len(self.time))], self.time[-1], color='
    gray', alpha=0.5)
                axs[i].grid()
            plt.savefig(f'{self.plot_path}/{self.patient}_Pred.png')
            plt.tight_layout()
            if self.showplot: plt.show()
```

```
176             plt.close()
177
178         def train_model(self,units1,units2,_learningRate):
179
180             # log_dir = self.log_dir + datetime.datetime.now().
    strftime("%Y%m%d-%H%M%S")
181             # tensorboard_callback = TensorBoard(log_dir=log_dir,
    histogram_freq=1)
182
183             data = self.data_prepare()
184
185             xtrain = data['Train']['x']
186             ytrain = data['Train']['y']
187             _, n_features = xtrain.shape
188
189             # print(f"Number of features after PCA: {xtrain.shape
    [1]}")
190
191             # Make Model
192
193             model = self.get_model(n_features, units1, units2,
    _learningRate)
194
195             # tf.keras.utils.plot_model(model, to_file='conv1d.png',
    show_shapes=True)
196
197             # model.summary()
198
199             print('The Model is in training mode ...')
200             start = time.time()
201             history = model.fit(xtrain,
202                                 ytrain,
203                                 epochs=30,
204                                 validation_data=(data['Val']['x'],
    data['Val']['y']),
205                                 verbose=0),
206                                 # callbacks=[tensorboard_callback])
207             training_time = time.time() - start
208             print('Training complete.')
209             print(history)
210             breakpoint()
211
212             model.save(f'{self.final_model}/model_{self.patient}.h5')
213
214             self.history_plot(history)
215
216             feature_imp = self.feature_effect(model,data)
217
218             return history, model, feature_imp, training_time
```

```
219
220         def check_model(self,model):
221             MAE_dict ={'DBP':{},'SBP':{}}
222             data = self.data_prepare()
223             # make predictions
224             trainPredict = model.predict(data['Train']['x'])
225             # testPredict = model.predict(data['Test']['x'])
226             y_test_hat = model.predict(data['test']['x'])
227
228             X_scaled = data['Dataset']['x'] #np.reshape(data['Dataset
    ']['x'],(data['Dataset']['x'].shape[0],data['Dataset']['x'].shape
    [1],1))
229
230             trainPredict = data['scaler']['y'].inverse_transform(
    trainPredict)
231             ytrain = data['scaler']['y'].inverse_transform(data['
    Train']['y'])
232             y_test_hat = data['scaler']['y'].inverse_transform(
    y_test_hat)
233             y_test = data['scaler']['y'].inverse_transform(data['test
    ']['y'])
234
235             MAE_dict['DBP']['Train'] = round(mean_absolute_error(
    ytrain[:,0], trainPredict[:,0]))
236             print( 'Train Score [DBP] for %s : %.2f MAE' % (self.
    patient,MAE_dict['DBP']['Train']))
237             MAE_dict['DBP']['Test'] = round(mean_absolute_error(
    y_test[:,0], y_test_hat[:,0]))
238             print( 'Test Score [DBP] for %s : %.2f MAE' % (self.
    patient, MAE_dict['DBP']['Test']))
239
240             MAE_dict['SBP']['Train'] = round(mean_absolute_error(
    ytrain[:,1], trainPredict[:,1]))
241             print( 'Train Score [SBP]: %.2f MAE' % (MAE_dict['SBP']['
    Train']))
242             MAE_dict['SBP']['Test'] = round(mean_absolute_error(
    y_test[:,1], y_test_hat[:,1]))
243             print( 'Test Score [SBP]: %.2f MAE' % (MAE_dict['SBP']['
    Test']))
244             self.plot_pred(model, X_scaled, data['Dataset']['y'],
    y_test,y_test_hat,data)
245
246             return MAE_dict
247
248         def run_gridsearch(self,param_grid):
249
250             data =  self.data_prepare()
251             _, n_features= data['Train']['x'].shape
252             param_grid['n_features'] = [n_features]
```

```
253
254             model = KerasRegressor(build_fn=self.get_model, epochs
    =70, batch_size=5, verbose=0)
255
256             kfold = KFold(n_splits=5, shuffle=True)
257
258             grid = GridSearchCV(estimator=model, param_grid=
    param_grid,
259                                 cv=kfold, verbose=1,
260                                 return_train_score=True)
261
262             grid_result = grid.fit(data['Train']['x'], data['Train']['
    y'])
263             print("Best: %f using %s" % (grid_result.best_score_,
    grid_result.best_params_))
264             return grid_result
265
266         def total_err(self,error_dict):
267             data = error_dict
268             dbp_err =[]
269             sbp_err =[]
270             pat = []
271             for key,val in data.items():
272                 pat.append(key)
273                 dbp_err.append(val['DBP']['Test'])
274                 sbp_err.append(val['SBP']['Test'])
275
276             fig,axs = plt.subplots(1,2)
277             fig.set_size_inches(self.figsize)
278
279             output={axs[0]:[dbp_err,'DBP'], axs[1]:[sbp_err, 'SBP']}
280             for ax in axs:
281                 ax.hist(output[ax][0], bins=20, rwidth=0.8)
282                 ax.set_xlabel('Error(mmHg)')
283                 ax.set_ylabel('The number of patients')
284                 ax.set_title(f'Histogram of error {output[ax][1]}')
285                 ax.grid()
286             if self.showplot: plt.show()
287             plt.savefig(f'{self.plot_path}/Hist_all_MAE.png')
288
289         def feature_effect(self,model,data):
290
291             result = permutation_importance(model,data['Val']['x'],
    data['Val']['y'], scoring='neg_mean_squared_error')
292
293             # Reverse PCA transformation to get feature importance
    scores
294             importances = result.importances_mean
295             inv_importances = self.pca.inverse_transform(importances)
```

```
296
297             importances_std = result.importances_std
298             inv_importances_std = self.pca.inverse_transform(
    importances_std)
299
300             for index, i in enumerate(inv_importances):
301                 print(f"Features {self.input_col[index]} : {i:.3f}
    +/- {inv_importances_std[index]:.3f}")
302
303             return (inv_importances, inv_importances_std)
304
305
306     if __name__=='__main__':
307     if os.path.exists(f'{regr_path}/result_map/MAE_results.csv'):
308       df_MAE = pd.read_csv(f'{regr_path}/result_map/MAE_results.csv')
309     else:
310       df_MAE = pd.DataFrame(columns = ['Patient', 'MAE_DBP_Train','
    MAE_DBP_Test','MAE_SBP_Train','MAE_SBP_Test', 'MAE_MAP'], index =
    None)
311     if os.path.exists(f'{regr_path}/result_map/feat_imp.csv'):
312       df_feat_imp = pd.read_csv(f'{regr_path}/result_map/feat_imp.csv
    ')
313     else:
314       df_feat_imp = pd.DataFrame(columns=['Patient','PPG_Trough', '
    SPs', 'UpTime', 'BTB_PPG', 'PPG_height','R', 'BTB_R', 'P', 'T', 'Q
    ', 'S', 'HR','PTT'], index = None)
315     if os.path.exists(f'{regr_path}/result_map/ML_sys_Error.csv') :
316       df_sys_err = pd.read_csv(f'{regr_path}/result_map/ML_sys_Error.
    csv')
317     else:
318       df_sys_err = pd.DataFrame(columns=['patient', 'error'])
319     for n,file in enumerate(os.listdir('./Dataset - done')):
320         if len(file.split('.')) > 1:
321             if file.split('.')[1] == 'csv':
322                 patient = file.split('.')[0]
323                 path ='./Dataset - done'
324                 print(f'Patient: {patient} - {n}/{len(os.listdir(path
    ))}')
325                 try:
326                   ls = lstm_sintec(patient=patient,dataset_dir='
    Dataset - done')
327
328                   history, model, feature_imp = ls.train_model(units1
    =64, units2=128, _learningRate=.001)
329
330                   # print(len(np.insert(feature_imp[0],0,patient)))
331                   # print(df_feat_imp.shape())
332                   df_feat_imp.loc[len(df_feat_imp)] = np.insert(
    feature_imp[0],0,patient)
```

```
333
334                     MAE_dict = ls.check_model(model)
335
336                     df_MAE.loc[len(df_MAE)] = [patient, MAE_dict['DBP'
    ]['Train'], MAE_dict['DBP']['Test'], MAE_dict['SBP']['Train'],
    MAE_dict['SBP']['Test'],MAE_dict['MAP']['Test']]
337                 except Exception as e:
338                     print(f"{patient} didn't complete")
339                     df_sys_err.loc[len(df_sys_err)] = {'patient':
    patient,'error':e}
340                     print("An error occurred:", e)
341     print(f'Number of patients was completed: {len(df_MAE)}')
342     df_sys_err.to_csv(f'{regr_path}/result_map/ML_sys_Error.csv')
343     df_feat_imp.to_csv(f'{regr_path}/result_map/feat_imp.csv', index
    = None)
344     df_MAE.to_csv(f'{regr_path}/result_map/MAE_results.csv', index =
    None)
345
346     df_MAE_res = pd.read_csv(f'{regr_path}/result_map/MAE_results.csv
    ')
347     print(len(df_MAE_res))
348     df = pd.read_csv('./result/feat_imp.csv')
349     df = df.drop("Patient", axis=1)
350     sns.violinplot(data=df, orient='h')
351     plt.xlabel("MAE decrease")
352     plt.show()
353     sns.boxplot(data=df, orient='h')
354     plt.xlabel("MAE decrease")
355     plt.show()
```

A.4 Features correlation

```
1
2     input_col = ['Tr', 'SPs_new', 'UpTime', 'BTB_PPG', 'PPG_h',
3                  'R', 'BTB_R', 'P', 'T', 'Q', 'S', 'HR', 'PTT
    ']
4     n_features = len(input_col)
5     mean_corr = np.zeros((n_features,n_features))
6     num_corr = 0
7
8     if not os.path.exists('./Plots/corr_feat'):
9       os.mkdir('./Plots/corr_feat')
10    for n,file in enumerate(os.listdir(f'./Dataset')):
11        if len(file.split('.')) > 1:
12            if file.split('.')[1] == 'csv':
```

```
13                  patient = file.split('.')[0]
14                  path =f'{regr_path}/Dataset'
15                  df = pd.read_csv('./Dataset/{file}').dropna()
16                  df = df[input_col]
17                  col_name = {'Tr':'PPG_Trough', 'SPs_new':'SPs', '
    PPG_h':'PPG_height'}
18                  df = df.rename(columns=col_name)
19                  corr = df.corr()
20                  corr = corr.replace(np.nan, 0)
21                  mean_corr += corr
22                  num_corr += 1
23
24                  # sns.set(rc = {'figure.figsize':(7,6)})
25                  # sns.heatmap(df.corr(), annot = False, fmt='.2g',
    cmap= 'coolwarm')
26                  # plt.title(f'The correlation between features for
    patient: {patient}')
27                  # plt.savefig('./Plots/corr_feat/{patient}.png')
28                  # plt.show()
29      mean_corr_f = mean_corr/num_corr
30      sns.heatmap(mean_corr_f, annot = False, fmt='.2g',cmap= 'coolwarm
    ')
31      plt.title(f'The mean of correlation between features')
32      plt.savefig('./Plots/corr_feat/mean_correlation.png')
33      plt.show()
```

Bibliography

[1] *A Global Brief on Hypertension*. World Health Organization. Geneva, Switzerland, 2013. URL: http://apps.who.int/iris/%20bitstream/10665/79059/1/WHO_DCO_WHD_2013.2_eng.pdf?ua=1 (cit. on p. 1).

[2] Morton P. Printz and Rebecca L. Jaworski. «Hypertension; Overview». In: *Encyclopedia of Endocrine Diseases (Second Edition)*. Ed. by Ilpo Huhtaniemi and Luciano Martini. Second Edition. Oxford: Academic Press, 2018, pp. 369–380. ISBN: 978-0-12-812200-6. DOI: https://doi.org/10.1016/B978-0-12-801238-3.03801-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128012383038010 (cit. on p. 1).

[3] George Silvay and Jacob Michael Lurie. «Chapter 39 - Thoracic Aortic Aneurysm Resection». In: *Cohen's Comprehensive Thoracic Anesthesia*. Ed. by Edmond Cohen. Philadelphia: Elsevier, 2022, pp. 557–578. ISBN: 978-0-323-71301-6. DOI: https://doi.org/10.1016/B978-0-323-71301-6.00039-1. URL: https://www.sciencedirect.com/science/article/pii/B9780323713016000391 (cit. on p. 1).

[4] Rahul Kumar, P.K. Dubey, Afaqul Zafer, Ashok Kumar, and Sanjay Yadav. «Past, present and future of blood pressure measuring instruments and their calibration». In: *Measurement* 172 (2021), p. 108845. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2020.108845. URL: https://www.sciencedirect.com/science/article/pii/S0263224120313373 (cit. on pp. 2, 6).

[5] URL: http://perioperativeCPD.com (visited on 06/07/2023) (cit. on p. 2).

[6] Ethel M. Frese, Ann Fick, and Steven H. Sadowsky. «Blood Pressure Measurement Guidelines for Physical Therapists». In: *Cardiopulmonary Physical Therapy Journal* 22.2 (2011). ISSN: 1541-7891. URL: https://journals.lww.com/cptj/Fulltext/2011/22020/Blood_Pressure_Measurement_Guidelines_for_Physical.2.aspx (cit. on p. 3).

[7] Valeria Figini. «Development of a Cuff-less Blood Monitoring Device». Master's thesis. Politecnico di Torino, 2020 (cit. on pp. 3, 6, 7, 10, 12).

[8] Chaya Gopalan and Erik Kirk. «Chapter 8 - Blood pressure, hypertension, and exercise». In: *Biology of Cardiovascular and Metabolic Diseases*. Ed. by Chaya Gopalan and Erik Kirk. Academic Press, 2022, pp. 141–156. ISBN: 978-0-12-823421-1. DOI: https://doi.org/10.1016/B978-0-12-823421-1.00007-X. URL: https://www.sciencedirect.com/science/article/pii/B978012823421100007X (cit. on p. 5).

[9] Kazunori Uemura, Toru Kawada, and Masaru Sugimachi. «A Novel Minimally Occlusive Cuff Method Utilizing Ultrasound Vascular Imaging for Stress-Free Blood Pressure Measurement: A Proof-of-Concept Study». In: *IEEE Transactions on Biomedical Engineering* 66.4 (Apr. 2019), pp. 934–945. ISSN: 1558-2531. DOI: 10.1109/TBME.2018.2865556. URL: https://doi.org/10.1109/TBME.2018.2865556 (cit. on p. 5).

[10] Mohamed Elgendi, Richard Fletcher, Yun Liang, Nicole Howard, Nigel H Lovell, Derek Abbott, Kean Lim, and Rabab Ward. «The use of photoplethysmography for assessing hypertension». In: *NPJ digital medicine* 2 (June 2019), p. 60. DOI: 10.1038/s41746-019-0136-7 (cit. on pp. 5, 9).

[11] *Beper BP Measurements*. URL: https://www.beper.com/eng/products/wrist-blood-pressure-monitor (visited on 05/02/2023) (cit. on p. 6).

[12] Cindy L. Stanfield. *Principles of Human Physiology*. 2016 (cit. on p. 7).

[13] J. Eric Piña-Garza and Kaitlin C. James. «4 - Increased Intracranial Pressure». In: *Fenichel's Clinical Pediatric Neurology (Eighth Edition)*. Ed. by J. Eric Piña-Garza and Kaitlin C. James. Eighth Edition. Philadelphia: Elsevier, 2019, pp. 91–114. ISBN: 978-0-323-48528-9. DOI: https://doi.org/10.1016/B978-0-323-48528-9.00004-2. URL: https://www.sciencedirect.com/science/article/pii/B9780323485289000042 (cit. on p. 6).

[14] Jean-Pierre Barral and Alain Croibier. «2 - Circulatory physiology». In: *Visceral Vascular Manipulations*. Ed. by Jean-Pierre Barral and Alain Croibier. Oxford: Churchill Livingstone, 2011, pp. 27–45. ISBN: 978-0-7020-4351-2. DOI: https://doi.org/10.1016/B978-0-7020-4351-2.00002-8. URL: https://www.sciencedirect.com/science/article/pii/B9780702043512000028 (cit. on p. 7).

[15] Wilbert S. Aronow. «Treatment of hypertensive emergencies». In: *Annals of Translational Medicine* 5.Suppl 1 (May 2017), S5. DOI: 10.21037/atm.2017.03.34 (cit. on p. 7).

[16] Junyung Park, Hyeon Seok Seok, Sang-Su Kim, and Hangsik Shin. «Photoplethysmogram Analysis and Applications: An Integrative Review». In: *Frontiers in Physiology* 12 (2022). ISSN: 1664-042X. DOI: 10.3389/fphys.2021.808451. URL: https://www.frontiersin.org/articles/10.3389/fphys.2021.808451 (cit. on pp. 9–11).

[17] C. Lee, H. S. Shin, and M. Lee. «Relations between ac-dc components and optical path length in photoplethysmography». In: *Journal of Biomedical Optics* 16.7 (2011), p. 077012 (cit. on pp. 9, 10).

[18] N. Utami, A. W. Setiawan, H. Zakaria, T. R. Mengko, and R. Mengko. «Extracting blood flow parameters from Photoplethysmograph signals: A review». In: *The 3rd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*. IEEE. Bandung, Indonesia, 2013, pp. 403–407 (cit. on p. 9).

[19] Yasmeen Sattar and Lovneesh Chhabra. *Electrocardiogram*. [Internet]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK549803/. Treasure Island, FL, Apr. 2023 (cit. on p. 11).

[20] Selcan Kaplan Berkaya, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal, and M. Bilginer Gulmezoglu. «A survey on ECG analysis». In: *Biomedical Signal Processing and Control* 43 (2018), pp. 216–235. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2018.03.003. URL: https://www.sciencedirect.com/science/article/pii/S1746809418300636 (cit. on p. 11).

[21] Ahilan Appathurai, J. Jerusalin Carol, C. Raja, S.N. Kumar, Ashy V. Daniel, A. Jasmine Gnana Malar, A. Lenin Fred, and Sujatha Krishnamoorthy. «A study on ECG signal characterization and practical implementation of some ECG characterization techniques». In: *Measurement* 147 (2019), p. 106384. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2019.02.040. URL: https://www.sciencedirect.com/science/article/pii/S0263224119301599 (cit. on p. 11).

[22] Selim Dilmac and Mehmet Korurek. «ECG heart beat classification method based on modified ABC algorithm». In: *Applied Soft Computing* 36 (2015), pp. 641–655. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.07.010. URL: https://www.sciencedirect.com/science/article/pii/S1568494615004408 (cit. on pp. 11, 12).

[23] Venkata Anuhya Ardeti, Venkata Ratnam Kolluru, George Tom Varghese, and Rajesh Kumar Patjoshi. «An overview on state-of-the-art electrocardiogram signal processing methods: Traditional to AI-based approaches». In: *Expert Systems with Applications* 217 (2023), p. 119561. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.119561. URL: https://www.sciencedirect.com/science/article/pii/S0957417423000623 (cit. on p. 12).

[24] Masaki Kyoso and Akihiko Uchiyama. «Development of an ECG identification system». In: *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 4. IEEE. 2001, pp. 3721–3723 (cit. on pp. 12, 14).

[25] URL: https://www.adinstruments.com/support/knowledge-base/what-correct-electrode-placement-conventional-ecg-recording (visited on 05/26/2023) (cit. on pp. 12, 13).

[26] URL: https://en.wikipedia.org/wiki/ST_segment (visited on 05/27/2023) (cit. on p. 13).

[27] Xinwen Liu, Huan Wang, Zongjin Li, and Lang Qin. «Deep learning in ECG diagnosis: A review». In: *Knowledge-Based Systems* 227 (2021), p. 107187. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107187. URL: https://www.sciencedirect.com/science/article/pii/S0950705121004494 (cit. on pp. 13, 14).

[28] C Saritha, V Sukanya, and Y Narasimha Murthy. «ECG signal analysis using wavelet transforms». In: *Bulg. J. Phys* 35.1 (2008), pp. 68–77 (cit. on p. 14).

[29] Leif Sörnmo and Pablo Laguna. «Electrocardiogram (ECG) signal processing». In: *Wiley encyclopedia of biomedical engineering* (2006) (cit. on p. 14).

[30] A. Ebrahimzadeh, B. Shakiba, and A. Khazaee. «Detection of electrocardiogram signals using an efficient method». In: *Applied Soft Computing* 22 (2014), pp. 108–117. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2014.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1568494614002191 (cit. on p. 14).

[31] Chandan Kumar Jha and Maheshkumar H. Kolekar. «Cardiac arrhythmia classification using tunable Q-wavelet transform based features and support vector machine classifier». In: *Biomedical Signal Processing and Control* 59 (2020), p. 101875. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2020.101875. URL: https://www.sciencedirect.com/science/article/pii/S1746809420300318 (cit. on p. 14).

[32] Moody B, Moody G, Villarroel M, Clifford G D, and Silva I. «MIMIC-III Waveform Database (version 1.0). PhysioNet». In: (2020). URL: https://doi.org/10.13026/c2607m (cit. on p. 16).

[33] Alistair E.W. Johnson et al. «MIMIC-III, a freely accessible critical care database». In: *Scientific Data* 3.1 (May 2016), p. 160035. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.35. URL: https://doi.org/10.1038/sdata.2016.35 (cit. on p. 16).

[34] URL: https://physionet.org/content/mimic3wdb/1.0/ (visited on 03/15/2023) (cit. on p. 16).

[35] Guido VanRossum and Fred L Drake. *The python language reference*. Vol. 561. Python Software Foundation Amsterdam, Netherlands, 2010 (cit. on p. 17).

[36] Guido Van Rossum et al. «Python Programming Language.» In: *USENIX annual technical conference*. Vol. 41. 1. Santa Clara, CA. 2007, pp. 1–36 (cit. on p. 17).

[37] Fabian Pedregosa et al. «Scikit-learn: Machine learning in Python». In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830 (cit. on p. 17).

[38] Sebastian Raschka and Vahid Mirjalili. «Python machine learning: Machine learning and deep learning with python». In: *Scikit-Learn, and TensorFlow. Second edition ed* 3 (2017) (cit. on p. 17).

[39] S. Lee and J. Chang. «Oscillometric blood pressure estimation based on deep learning». In: *IEEE Transactions on Industrial Informatics* 13.2 (2017), pp. 461–472. DOI: 10.1109/TII.2016.2612640 (cit. on p. 18).

[40] D.E. Adkins. «Machine learning and electronic health records: a paradigm shift». In: *International Journal of Medical Informatics* 105 (2017), pp. 66–70. DOI: 10.1016/j.ijmedinf.2017.06.006 (cit. on p. 18).

[41] X. Fan, Q. Yao, Y. Cai, and et al. «Multiscaled fusion of deep convolutional neural networks for screening atrial fibrillation from single lead short ECG recordings». In: *IEEE Journal of Biomedical and Health Informatics* 22.6 (2018), pp. 1744–1753. DOI: 10.1109/JBHI.2018.2858789 (cit. on p. 18).

[42] Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. «Cuffless Blood Pressure Estimation Algorithms for Continuous Health-Care Monitoring». In: *IEEE Transactions on Biomedical Engineering* 64.4 (2017), pp. 859–869. DOI: 10.1109/TBME.2016.2580904 (cit. on pp. 19, 23).

[43] Zhenqi Li, Cong Yan, Wei Zhao, Jing Hu, Dongya Jia, Hongmei Wang, and Tianyuan You. «A Novel Method for Calibration-Based Cuff-Less Blood Pressure Estimation». In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019, pp. 4266–4269. DOI: 10.1109/EMBC.2019.8857373 (cit. on pp. 19, 23).

[44] Mohammad Kachuee, Mohammadreza M. Kiani, Hossein Mohammadzadeh, and et al. «Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time». In: *Proceedings - IEEE International Symposium on Circuits and Systems*. Vol. 2015-July. 2015, pp. 1006–1009. DOI: 10.1109/ISCAS.2015.7168806 (cit. on p. 19).

[45] Gašper Slapničar, Nejc Mlakar, and Mitja Luštrek. «Blood Pressure Estimation from Photoplethysmogram Using a Spectro-Temporal Deep Neural Network». In: *Sensors* 19.15 (2019). ISSN: 1424-8220. DOI: 10.3390/s19153420. URL: https://www.mdpi.com/1424-8220/19/15/3420 (cit. on pp. 19, 20, 23).

[46] Umit Senturk, Kemal Polat, and Ibrahim Yucedag. «A non-invasive continuous cuffless blood pressure estimation using dynamic Recurrent Neural Networks». In: *Applied Acoustics* 170 (2020), p. 107534. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2020.107534. URL: https://www.sciencedirect.com/science/article/pii/S0003682X20306381 (cit. on pp. 20, 23).

[47] Madhuri Panwar, Arvind Gautam, Dwaipayan Biswas, and Amit Acharyya. «PP-Net: A Deep Learning Framework for PPG-Based Blood Pressure and Heart Rate Estimation». In: *IEEE Sensors Journal* 20.17 (2020), pp. 10000–10011. DOI: 10.1109/JSEN.2020.2990864 (cit. on pp. 21, 23).

[48] Stephanie Baker, Wei Xiang, and Ian Atkinson. «A hybrid neural network for continuous and non-invasive estimation of blood pressure from raw electrocardiogram and photoplethysmogram waveforms». In: *Computer Methods and Programs in Biomedicine* 207 (2021), p. 106191. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2021.106191. URL: https://www.sciencedirect.com/science/article/pii/S0169260721002650 (cit. on pp. 21, 23).

[49] Chih-Ta Yen, Sheng-Nan Chang, and Cheng-Hong Liao. «Estimation of Beat-by-Beat Blood Pressure and Heart Rate From ECG and PPG Using a Fine-Tuned Deep CNN Model». In: *IEEE Access* 10 (2022), pp. 85459–85469. DOI: 10.1109/ACCESS.2022.3195857 (cit. on pp. 22, 23).

[50] Valeria Figini, Sofia Galici, Daniele Russo, Ilenia Centonze, Monica Visintin, and Guido Pagana. «Improving Cuff-Less Continuous Blood Pressure Estimation with Linear Regression Analysis». In: *Electronics* 11.9 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11091442. URL: https://www.mdpi.com/2079-9292/11/9/1442 (cit. on pp. 22, 23, 25).

[51] Pin-You Ko, Lien-Fu Lai, Tien-Hsiung Ku, and Yue-Der Lin. «A CNN Approach To Predicting Non-Invasive Blood Pressure of Surgical Patients». In: *2022 IEEE 5th International Conference on Knowledge Innovation and Invention (ICKII)*. 2022, pp. 9–13. DOI: 10.1109/ICKII55100.2022.9983601 (cit. on p. 23).

[52] Stephanie Baker, Wei Xiang, and Ian Atkinson. «A computationally efficient CNN-LSTM neural network for estimation of blood pressure from features of electrocardiogram and photoplethysmogram waveforms». In: *Knowledge-Based Systems* 250 (2022), p. 109151. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.109151. URL: https://www.sciencedirect.com/science/article/pii/S0950705122005731 (cit. on p. 24).

[53] Association for the Advancement of Medical Instrumentation. *The National Standard for Electronic or Automated Sphygmomanometers*. AAMI. Arlington, VA, 1987 (cit. on p. 37).