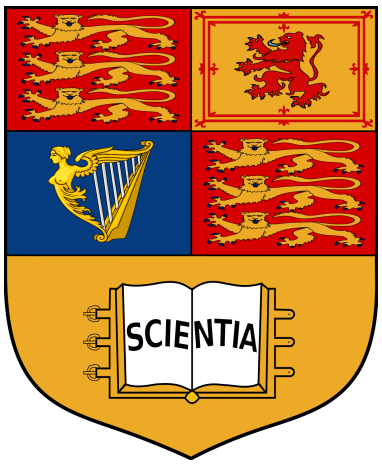
Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2022



Project Title:	**Model Development for Severity Classification of Dengue Patients using Photoplethysmography (PPG) Data**
Student:	**Vasileios Manginas**
CID:	**01542774**
Course:	**EEE4**
Project Supervisor:	**Prof. Pantelis Georgiou**
Second Marker:	**Prof. E. Rodriguez-Villegas**

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

Abstract

Dengue is a viral disease spread through the bites of infected mosquitoes, and currently places more than half the world's population at risk. During epidemic season, the countries most heavily affected by the disease witness extensive pressure on the healthcare system, which in turn significantly complicates patient management and early detection of disease progression. PPG (photoplethysmography) sensors present themselves as an ideal candidate for assisting in patient monitoring in the case of mass-scale diseases due to their inexpensive and promptly available nature.

This work investigates the effectiveness of utilising PPG data through signal processing and machine learning techniques in predicting disease severity for dengue patients. Through our experimentation with several configurations for the feature extraction and model development stages, we conclude that performance is maximised through the use of time-frequency representations in the form of the STFT as input data to CNN models. Using this setup, we are able to achieve 89.2% accuracy and weighted F1 scores in distinguishing between healthy and unhealthy subjects, and 78% in classifying between mild, moderate, and severe infection.

Acknowledgements

I would like to express my gratitude towards everyone who supported me in the completion of my final year project. First, my supervisor, Prof. Pantelis Georgiou, for providing me with the opportunity to contribute to a project characterised by novelty and fueled by the pursuit for positive impact. Secondly, my co-supervisor, Stefan Karolcik, on who I could always rely for assistance and advice, and whose guidance was decisive for the successful completion of the project. I would then like to thank John Daniels and my brother Nikos for their valuable technical knowledge and their continuous willingness to help, as well as my friend Claire for proof reading my dissertation. Lastly, I would like to express my deep gratitude and appreciation towards my friends, girlfriend, and family, whose support and love throughout the project's duration has been truly invaluable.

Contents

Chapter 1

Introduction

1.1 Motivation

Dengue is a a viral disease found primarily in countries with tropical and subtropical climates, and is transmitted through the bites of infected mosquitoes [1]. Although dengue outbreaks occur mostly in Latin American and Asian countries, around 4 billion people, more than half of the world's population, are at risk of viral infection, with an estimated number of 390 million cases annually [2]. While the majority of cases exhibit mild or even no symptoms, leading to an overall mortality rate of less than 1%, dengue patients occasionally progress to a more serious stage of the disease, severe dengue. One potentially lethal manifestation of severe dengue is Dengue Shock Syndrome (DSS). This dangerous complication, which will be referred to as "shock" in the remainder of this report, is associated with fatality rates ranging from $1 - 10\%$, and approaches 30% when untreated [3]. While there is no direct treatment for severe dengue or DSS, detecting deterioration early on lowers the fatality rate dramatically to 1% [1]. Despite this, in the areas affected by the disease, dengue epidemics can place extensive pressure on the healthcare system, especially during rainy seasons when transmission rates increase. This makes early detection of the disease much more challenging. Coupled with the fact that many of these areas include primarily lower-income countries, it is obvious that assistance in the form of patient management and prevention is both decisive and urgent.

Photoplethysmography (PPG) is an optical technique used to detect volumetric changes in blood in peripheral circulation, achieved by measuring and recording changes in the absorption of light through the tissue on measuring sites [4]. This process results in the PPG signal, a signal rich in information. Further, the PPG signal allows for estimation of vital sign parameters, such as a patient's heart rate, and is therefore suitable as a monitoring device. PPG sensors utilise simpler hardware than other alternatives, are non-invasive, cheap to manufacture and purchase, and thus promptly available [5].

Given the above, PPG sensors present themselves as an ideal candidate for patient monitoring in the case of massive-scale diseases, such as dengue, especially in lower-income countries. In this light, it is undeniable that utilising PPG data with the aim of aiding in the management of dengue outbreaks is a goal truly worth pursuing. It is this large potential for positive impact that serves as the main motivational force behind the entirety of this project.

1.2 Project Aims

In broad terms, the aim of this project is to investigate clinical questions for dengue patients through the use of PPG data, signal processing, and machine learning techniques. It is worth mentioning that the use of PPG data in the context of dengue is a particularly novel field of study, meaning that the nature of this project is highly exploratory. While this entails that results are not at all guaranteed, it also increases the value and significance of the investigation. The clinical questions chosen for study in this project are related primarily to the severity of the infection. In this setting, exploring such questions is equivalent to investigating whether we can establish, through the use of the aforementioned technical tools, a relationship between a patient's PPG signal, and the severity of their dengue infection.

If this process is successful, we would then be able, given a patient's PPG signal, to predict their disease severity in real time. If a severe prediction is provided, then this can be seen as a red flag, potentially helpful in cases where severity has perhaps been underestimated. In this case, the red flag would motivate a re-examination and potential subsequent treatment adjustment. As a more complicated, but also much more valuable application, by repeating this prediction over time, such as every one hour for example, one could theoretically be able to detect imminent patient deterioration. In turn, this would allow for better resource management and organisation in hospitals, truly bringing forth the potential of saving lives.

One of the primary drivers behind this exploration, and the reason why this work is novel in nature, is the existence of a new dataset. In Section 1.2.1, we present a brief overview of the new dataset and its contents in order to provide further context for the material we have to work with. After this, in Section 1.2.2 we will examine the technical objectives of this project in more detail by specifying the precise clinical questions we aim to investigate, as well as introduce the processing and analysis pipeline. Finally, in Section 1.3 we provide a brief summary of the structure of the report.

1.2.1 New Dataset

The new dataset was acquired in Hospital for Tropic Diseases (HTD) in Ho Chi Minh City, Vietnam in partnership with Oxford University Clinical Research Unit (OUCRU). Collected using a commercial wearable PPG device from SmartCare [6] featuring a finger probe, this dataset is larger and, in a sense, more complete than its predecessors, which have been used in preliminary work [7]. In this case, "completeness" refers to a larger number of patients, PPG recordings of longer durations, diversity in the patient cohorts, as well as the inclusion of a multitude of clinical data for each patient. A simplified representation of the dataset structure is shown in Figure 1.1 in the form of a directory tree [1].

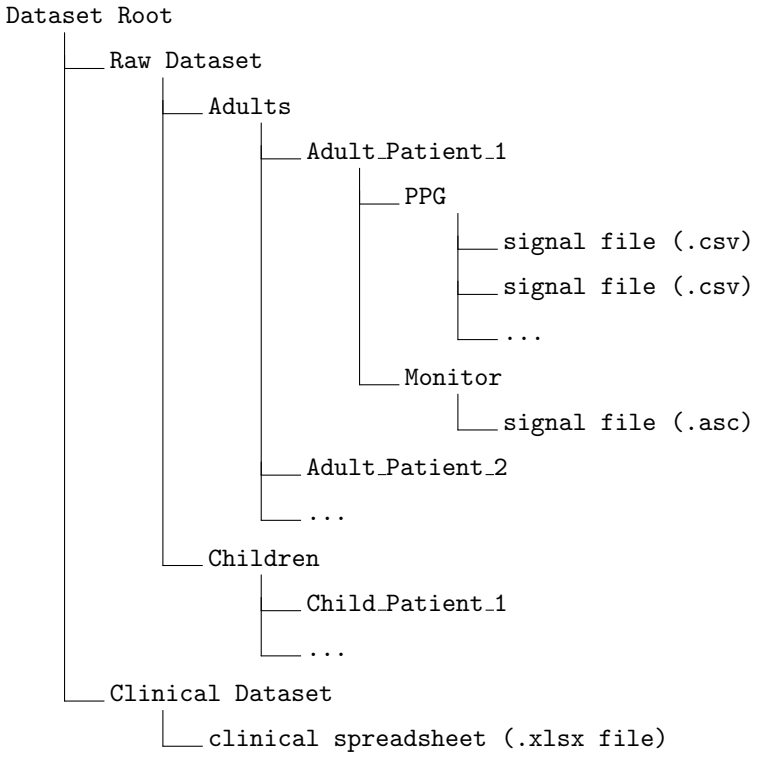


Figure 1.1: Simplified representation of the dataset structure in the form of a directory tree. The raw dataset contains the patients' PPG signals, whereas the clinical dataset includes vital parameters, as well as other clinical events, such as fluid administration and shock/reshock.

[1] We refer to the representation as simplified since the full depth (*i.e.,* from the dataset root to the furthest file) is only shown for one patient, as well as because we only show folders and files that are relevant to this project.

From Figure 1.1 we see that the dataset is split into two cohorts: the adult cohort (currently consisting of 135 patients), and the children cohort (consisting of 55 patients). Regardless of cohort, within each patient folder in the raw dataset there exist at least two folders: PPG, and Monitor. The PPG folder refers to the SmartCare device whereas the Monitor folder refers to a GE-brand patient monitor with synchronized ECG and PPG acquisition. Moreover, a patient can have more than one file in the case where there is more than one period of PPG recording. Finally, the clinical spreadsheet includes a multitude of data for each patient, such as personal information (their age, sex, etc.), vital signs parameters (heart rate, oxygen saturation, etc.), shock and reshock date and time, and fluid administration.

1.2.2 Technical Objectives

As mentioned, our study is focused around infection severity for dengue patients. To approach this task, we focus on two classification problems, which we aim to tackle sequentially.

1. **ICU-vs-Follow-up:** This task, referred to as "ICU-FU" in the remainder of this report, stems from the fact that a subset of patients from the children cohort attended follow-up recording sessions. Follow-up sessions refer to hospital visits that occur some time after the patient has been discharged. A typical example of this would be a follow-up visit one week after a patient discharge. The aim of these sessions is to record a patient's PPG in a healthy, post-infection state. Consequently, for a given patient, the dataset contains PPG recordings both from unhealthy (ICU), and healthy (follow-up) states. The essence of this classification task is therefore to investigate the binary separability between patients during and after their dengue infection.
2. **3-class severity classification:** This task, referred to as "severities" in the remainder of this report, can be seen as a continuation of the ICU-vs-Follow-up task. This is because in this case, rather than classifying between healthy and unhealthy subjects, we now aim to investigate the separability between 3 levels of disease severity. This question can be investigated due to the fact that the dataset contains explicit information on all adult patients regarding the severity of their infection. This is in the form of 3 classes, which we can name mild, moderate, and severe. While results for this multiclass classification task are expected to be harder to achieve, they would also be more valuable in a clinical setting.

Ultimately, our aim is to explore these two questions via a set of technical tools and present a collection of interesting and meaningful results. In order to conduct this investigation, however, we require a platform which is able to parse the large and occasionally inconsistent dataset into a set of results, achieved through a sequence of well-defined stages. This is accomplished through the design and implementation of a processing pipeline, the successful development of which constitutes one of the main technical objectives of this project. Furthermore, we identify three main desirable design characteristics, which serve as criteria for evaluating the quality of the pipeline implementation.

- **Configurability / Versatility:** As it is important that for each of the explored tasks a large range of testing arrangements is achievable, a highly configurable pipeline is not only a desirable characteristic, but also a prerequisite to the generation of meaningful and diverse results. Therefore, each stage of the pipeline must be designed to be versatile.
- **Efficiency:** Since the experimental setup of the project will demand a large number of trials and different testing conditions, an efficient implementation is crucial to the successful completion of the experimental process.
- **Ease of use:** Perhaps the most important objective of this project is that the work done here is useful (and usable) for future research of the topic. Since the pipeline is the main technical tool used for processing, analysis, and generation of results, it is of high importance that it is easy to use, customise, and build upon.

For this section, we present a simplified, high-level version of the pipeline, including only the core blocks of the project. Figure 1.2 depicts these main blocks, as well as a brief overview of each block. Again, technical details on the individual components are provided in Section 3.

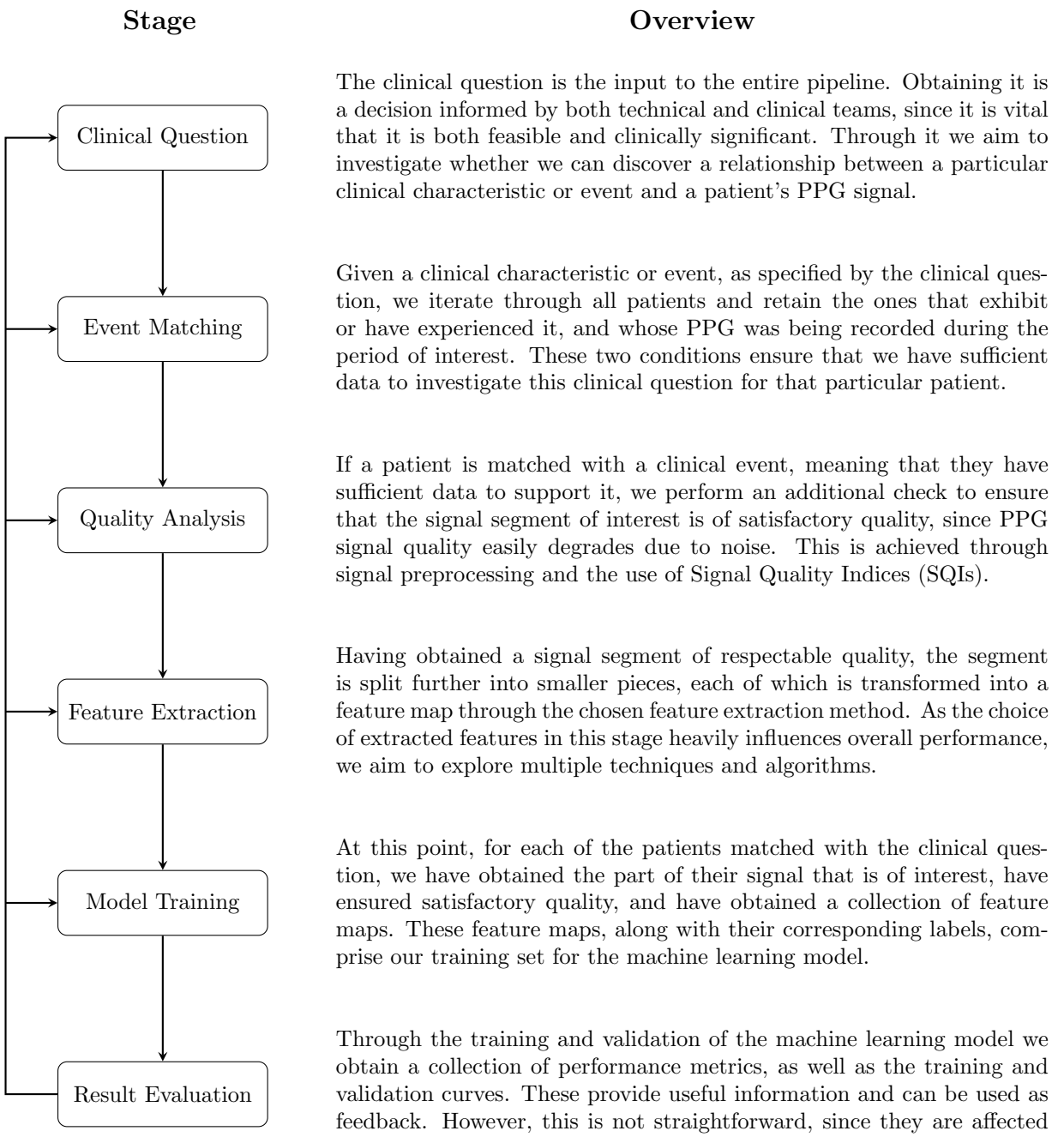


Figure 1.2: Implementation pipeline showing the main blocks of the project (left) and brief summary for each of these blocks (right).

1.3 Summary of Report Structure

This report begins with a background section focused on providing a comprehensive literature review on all aspects of this project. Subsequently, we delve into the technical work performed with regards to design and implementation of the pipeline. This includes technical details on system design, data flow, and algorithms used. We then introduce the experimental setup, outlining the different configurations we aim to test and the reasons for choosing them. The results of these experiments are then presented, evaluated, and discussed. Finally, we conclude this report by providing a summary and commenting on limitations and suggested future work.

Chapter 2

Background

2.1 Dengue Virus

Dengue is a viral infection caused by the Dengue virus (DENV), part of the Flaviviridae family [1]. The transmission of the virus primarily occurs through the bites of infected female mosquitoes of the species *Aedes aegypti* and, less frequently, *Aedes albopictus*. While dengue can be found in many countries around the world, bringing more than half the planet's population at risk, outbreaks are most common in Central and South America, Southeast Asia, the Pacific Islands, and the Caribbean [8]. Figure 2.1 shows a global map where areas are coloured according to the level of risk of a dengue outbreak.

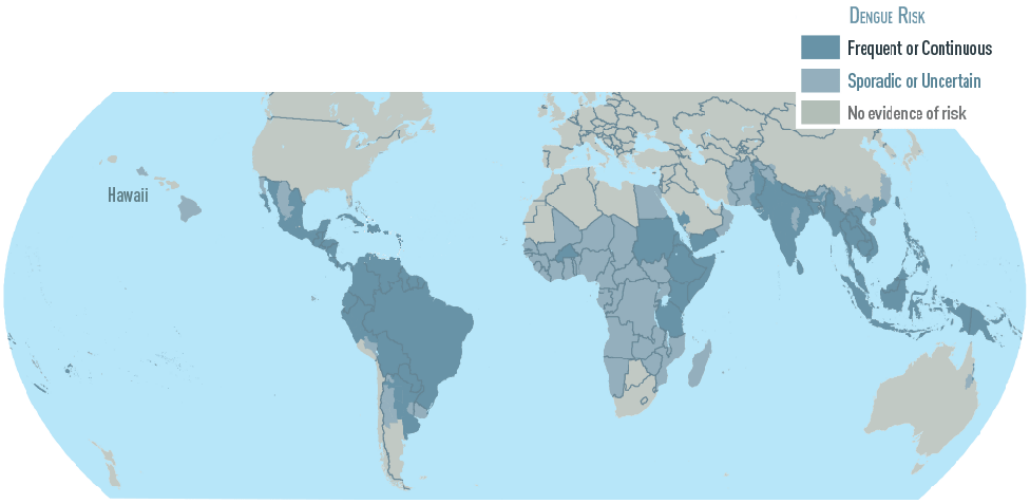


Figure 2.1: Dengue map, areas are coloured according to the risk level of a dengue outbreak (image from [9]).

Dengue is often divided into two categories: dengue (with and without warning signs), and severe dengue [1]. In general, most cases exhibit only mild symptoms or are completely asymptomatic. For dengue, these include high fever, as well as possible headaches, joint and muscle pain, rash, vomiting, swollen glands, abdominal pains, and nausea. In the case of severe dengue and the further complication of Dengue Shock Syndrome (DSS), the patients may experience severe bleeding, respiratory distress, organ impairment, as well as circulatory collapse or plasma leakage [1, 10].

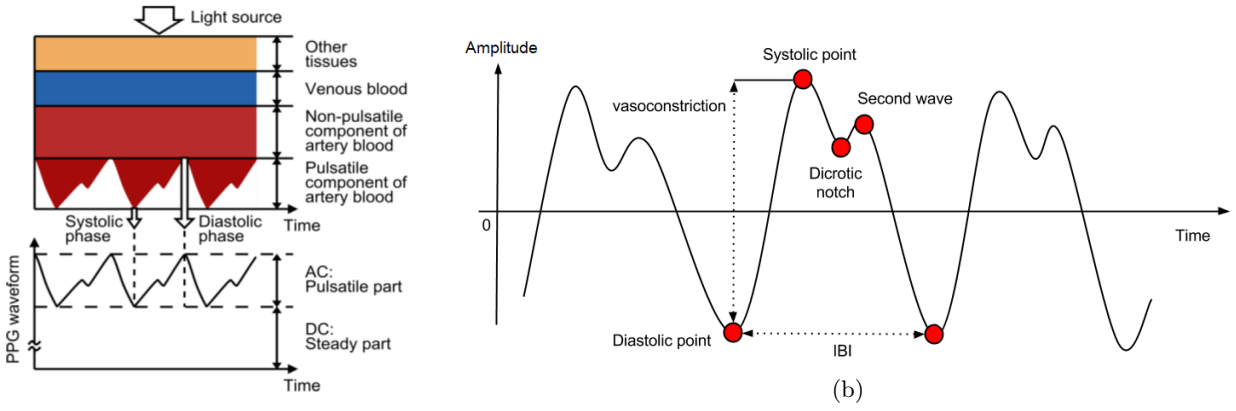
There are currently believed to be four distinct serotypes of the virus, DENV-1, DENV-2, DENV-3, and DENV-4, meaning that the same individual can be infected up to four times. However, it has been speculated that the differentiation according to serotypes may not paint the entire picture, and that instead, what is more significant are the antigenic differences between individual strains [11]. *Katzelnick et al.* (2015) found differences of similar order between strains of the same serotype and between ones of different serotypes, implying that "an individual infected with one type may not be protected against antigenically different viruses of the same type, and that in some cases the individual may be protected against some antigenically similar strains of a different type" [11].

Further complicating the situation, it has been observed that the second time an individual is infected, they exhibit more severe symptoms than the first time. *Halstead* (2002) proposed the concept of "antibody-dependent enhancement (ADE) of infection", suggesting that the antibodies developed from the first infection serotype actually exacerbate the spread of the disease the second time [12]. Later research concluded that ADE does in fact occur, but only at a precise range of antibody concentrations. More specifically, only intermediate levels of antibodies aggravated the disease, while high levels offered protection against severe disease and low levels had little effect: "Too much or too little—better than some" [13].

There is no directed treatment specifically against dengue [1]. Recommended practices for symptomatic dengue patients consist of fluid administration for hydration and paracetamol as analgesic and antipyretic [10]. For severe dengue, patients might require blood transfusions in cases of substantial bleeding or urgent administration of intravenous fluids for resuscitation in the case of plasma leakage. Prevention for dengue mainly comes in the form of infection avoidance (suggested practice is use of mosquito repellent). A vaccine also exists, but this has been shown to be unreliable for general administration. Although the vaccine provides protection for individuals who have contracted the virus prior to vaccination, it turns out to be harmful to people who have never been infected in the past. For those, not only does it show increased hospitalisations relative to an unvaccinated control group, but also heightens the risk of severe dengue progression [14, 15].

2.2 Photoplethysmography

Photoplethysmography (PPG) is a measuring technique used to detect changes in blood volume at a particular measuring spot [16]. This is achieved via the use of optical devices, namely a light source and a photodetector. When the light source is directed into skin, part of the light is absorbed by skin tissue as well as by the blood vessels within that tissue. If the wavelength of the light is within a specific range (more specifically around the boundary between visible and infrared), the amount absorbed by the skin tissue itself is constant and small in magnitude. This is turn means that the remainder of the light, which will be detected by the photodetector, can be used to estimate volumetric changes in blood content. Although the absorption, reflection, and scattering processes of light within tissue and blood vessels are not simple, in general, the larger the amount of blood within the vessels, the larger the light attenuation [4, 5, 16]. This process is depicted in Figure 2.2a along with a sample PPG waveform in Figure 2.2b.



(a) (b)

Figure 2.2: (a) Principle of operation for a PPG device (image from [17]) (b) A sample PPG waveform (image from [18]).

In modern PPG devices, the light source and photodetector can be placed in two configurations: on opposite sides of the measuring site tissue (transmission mode) or on the same side (reflection mode) [19]. Figure 2.3 depicts these two possible modes for PPG devices using finger probes. PPG sensors can also utilise different light sources. The most common choices are infrared light emitting diodes (IR-LEDs), red LEDs, and green LEDs, each having slightly different characteristics [4, 5, 17]. The device responsible for the data acquisition

of this project used a red and an infrared LED, resulting in two distinct signals [6]. It is important to note here that LEDs, as well as photodetectors, are very simple components, and it is because of this simplicity in hardware that PPG devices are considerably cheaper than other alternatives within the monitoring device market.

Figure 2.3: Transmission (left) and reflection (right) modes of a finger PPG sensor (image from [19]).

Regarding real world usability, the clinical utility of PPG devices has been a topic of research for the larger part of the past century. For a detailed review of the early history of PPG research we refer to two comprehensive review articles, since historical analysis is beyond the scope of this report: *Challoner et al.* (1974) [16], and *Allen* (2007) [4]. In recent decades, aided by technological advancements in the areas of optoelectronics, semiconductors, digital signal processing, and clinical instrumentation in general, PPG has gathered more and more interest as a clinical tool [4, 5]. In the past years, this has further been enforced by the large rise in popularity of wearable devices, both for clinical and for health and fitness purposes. Nowadays, research has used PPG to measure (with varying degrees of success) several vital signs, such as heart rate, blood pressure, oxygen saturation, respiratory rate, hematocrit and temperature, further increasing its potential as a monitoring device. Details of how these are achieved will be explored in Section 3.5. Nevertheless, the potential of PPG is still being investigated.

2.3 Processing and Analysis of PPG Signals

The raw dataset as presented in Section 1.2.1 cannot directly be used as the input to the machine learning algorithms of the pipeline. This is because it is relatively unstructured (signals for each patient are of different length, patients can have more than one PPG recording), and may contain parts of unusable quality (patients may not be wearing the sensor correctly or at all, noise may be too significant in some segments). The aim of the steps presented in Sections 2.3.1, 2.3.2, and 2.3.3, is to transform the unstructured and noisy raw dataset to a structured and cleaner representation of our data, which can then be used as input to our machine learning models. This process consists of preprocessing, quality analysis, and feature extraction, each of which will be analysed in detail in the next sections.

2.3.1 PPG Preprocessing

Preprocessing refers to a collection of algorithms, the main goal of which is to remove unwanted noise from the PPG signal. In our application, preprocessing can consist of filtering, normalising, and smoothing.

2.3.1.1 PPG Filtering

PPG signals are highly susceptible to noise interference. This can be attributed to several factors, such as the surrounding environment, respiration, and motion artefacts [20, 21]. Some of these forms of noise even overlap with the signal in the frequency domain, bringing forth the need for effective filtering to reject as many unwanted signal components as possible. It is important to note that the literature on signal denoising is vast, meaning that there exists a very large number of techniques and algorithms for the task. Examples include digital filters (FIR/IIR), machine learning methods, such as autoencoders, wavelet denoising, dynamic time warping, clustering, and heuristic algorithms [22]. Since advanced techniques for optimal denoising are beyond the scope of this project, we limit our background research and use to conventional filters. Effective filtering requires sensible choices both for the filtering frequency range, as well as for the filter type. Research exists on both aspects.

Regarding the former, most studies use a filter with lower cutoff frequency in the range of $0.1 - 1$Hz and an upper cutoff frequency of $10 - 20$Hz. However, this isn't always the case. For example, *Moraes et al.* (2018) claim that the PPG pulse range exists within the range of $0.5 - 4$Hz, implying a similar passband for the filter

[21]. Another study perhaps worth mentioning is by *Allen et al.* (2004), who investigated the cutoff frequency of the highpass filter, ultimately suggesting 0.15Hz [23]. *Waugh et al.* (2018) also used a passband of $0.15 - 20$Hz in their study [24].

Regarding filter type, *Liang et al.* (2018) performed a thorough investigation on short ($2.1s$) PPG signals, experimenting with 9 different filter types, each with 10 orders. Using the skewness SQI, which will be discussed in detail in Section 2.3.2, as a performance metric, they found that optimal filtering was achieved with a 4^{th} Chebyshev II filter, surpassing the previously considered "gold standard", the Butterworth filter. They also validated that "the lower the order, the better the filter performance in analyzing biomedical signals" [20].

2.3.1.2 PPG Normalising and Smoothing

The amplitude of the AC component of the PPG varies wildly from application to application, as it is dependent on a large number of factors, such as the measuring spot, the light source and photodetector, and the gain used internally by the sensor. We therefore require normalisation in order to standardise the dynamic range of our signal to $[-1, 1]$. Regarding smoothing, the literature is, once again, extensive. We can approach smoothing through different types of averaging, such as the moving average or the Savitzky-Golay algorithm, or via other methods, such as spline smoothing.

2.3.2 Signal Quality Indices (SQIs)

Signal quality indices (SQIs) are mathematical concepts or algorithms used to describe the quality of a signal. In this project, we utilise SQIs for PPG quality analysis, and more specifically to reject PPG signal segments that are of poor quality due to large amounts of noise. These signal segments would not be beneficial to the training of our machine learning models, and could even prove to be harmful since they provide inaccurate representations of PPG signals.

The comprehensive article by *Elgendi* (2016) investigated eight different SQIs, perfusion, kurtosis, skewness, relative power, non-stationarity, zero crossing, entropy, and the matching of systolic wave detectors, in order to find the optimal SQI for classifying PPG signal quality [25]. He found that the skewness SQI was best for assessing quality, outperforming the previously considered "gold standard", the perfusion index SQI. That said, the study was conducted using only healthy subjects. Furthermore, it was reported that the performance of the skewness SQI deteriorated as the window size on which it was calculated increased. We should keep this in mind, since our work will more than likely be using segments longer than $3 - 5s$. Finally, investigating the use of a combination of SQIs is also mentioned as future work [25].

Several different techniques and algorithms for PPG quality checking have also been found, although most of these lie beyond the scope of this project. One example is by *Orphanidou* (2018), who proposed an SQI which labelled a signal segment as "good" or "bad" based on whether a reliable heart rate could be derived [26]. Other more advanced techniques that we discovered in literature include dynamic time warping, fuzzy neural networks, deep convolutional neural networks, and random forest classifiers [27, 28, 29, 30].

2.3.3 Feature Extraction

Feature extraction refers to the process of creating a representation of a given signal by means of a particular method or algorithm. The aim of obtaining this representation rather than using the original signal is to reduce the amount of data to be processed, while simultaneously retaining the useful information from the signal, or even extracting information that was not already there. In the past, feature extraction for PPG signals has followed both blind and insightful approaches, as defined by *Elgendi et al.* (2018) [31]. An insightful approach refers to a system which extracts or localises features from the PPG signal or parts of it, whereas a blind approach rather uses the entire raw signal. We investigate both of these in the next sections.

2.3.3.1 Raw Signal, Derivatives, and Localised Features

The most straightforward choice for feature selection is to use the the entire raw signal after preprocessing. This is the easiest approach, but also the most "blind" one. An extension of this would be to use localised

features from the raw signal, such as the amplitude of specific points, or the distance between two of them. *Elgendi* (2012) carried out extensive analysis on these localised features, along with the corresponding clinical phenomena and causes linked to each feature [32].
Besides the raw signal itself, we can look at the first and second derivatives of the PPG signal, as these have been shown to hold important information [32]. *Takazawa et al.* (1998) introduced the first and second derivatives of the PPG signal [33]. *Elgendi* (2012) provided a thorough review as well as further analysis on the first and second derivative in [32], and also attempted to standardise terminology on these features in [31]. As in the case of the raw signal, we can extract specific local features from the derivative signals. Again, *Elgendi* (2012) extensively presented this in [32].

2.3.3.2 Vital Signs and Clinical Data

Using vital signs and clinical data as features is probably the most intuitive choice, at least from the clinical staff's perspective, since it is these clinical features that they themselves use for a diagnosis. We have found several examples where these vital signs were directly used as features in the ML algorithms [34, 35, 36]. We will explore the actual performance of these ML models in Section 2.4.2. We see two ways to obtain such features. The first way is to directly access them from the clinical dataset. The largest issue with this is that vital signs are not taken regularly in a standardised manner across all patients, and so it would be very hard to create a well-structured dataset of clinical features solely from the clinical spreadsheets. The second way, which is more involved but also more promising, is to estimate these vital signs from the PPG signal and its derivatives. This has the advantage that we can estimate these at any points where we have a PPG signal.

It is important to note that we are only interested in vital signs that have been shown to have some kind of correlation with dengue, as this would imply that there may be some benefit in using these vital signs as features. Below we have categorised some of the common clinical features that are related to dengue and have been investigated through PPG signals. We note that we do not present this as a thorough literature review of all the research on vital sign feature extraction from PPG, but rather as a collection of vital signs that we have found which have been shown to be correlated to dengue and which can be estimated through the PPG signal.

1. **Cardiovascular:** Dengue has been observed to have an effect on the cardiovascular system, although "the pathophysiology of cardiac disease in dengue infection is unclear" [37]. Despite bradycardia being the most common cardiovascular abnormality, others have also been observed [37, 38], meaning related clinical features might contain useful information.
   (a) **Heart rate:** Heart rate is the most basic and easiest vital sign to estimate since the PPG period is closely linked to a single heart beat. *Reiss et al.* (2019) provide a review of the classical methods to heart rate estimation, as well as their novel deep learning approach in [39].
   (b) **Blood pressure:** Cuff-less methods to estimate blood pressure (BP) using only the PPG signal have been proposed in recent years. One example is by *Addison* (2016), who introduced slope transit time (STT), a method which unlike its predecessor, pulse transit time (PTT), which required both ECG and PPG measurements, needs only the PPG signal [40] [41]. Motivation for BP estimation through PPG was further enforced by *Martínez et al.* (2018), who concluded that "PPG holds most informative features that exist in [arterial blood pressure] ABP" [42].
   (c) **Aging index (AGI) and arterial stiffness:** First investigated by *Takazawa et al.* (1998) through the amplitudes at several points of the second derivative [33], and was reviewed by *Elgendi* (2012) in [32]. An algorithm for this process was proposed by *Pilt et al.* (2013) in [43].

2. **Respiratory:** Dengue can also have respiratory and pulmonary manifestations [44, 45]. According to the WHO, one of the potential symptoms of severe dengue is respiratory distress, implying that respiration rate and oxygen saturation (SpO2) may offer useful information [1].
   (a) **Respiratory rate:** The PPG waveform is heavily influenced by respiration rate and so its detection has been established and is relatively straightforward [46, 47].
   (b) **Oxygen saturation (SpO2):** The most common usage of PPG signals is for pulse oximetry, the measurement of oxygen saturation. We refer to the excellent progress article by *Tamura* (2019) for further information on various aspects of pulse oximeters [48].

3. **Blood components and features:** *Chaloemwong et al.* (2018) investigated the differences in complete blood count (CBC) between patients with acute febrile illness as a result of dengue infection, and as a result of other causes, and found that "the dengue group [compared to the control group] had higher haemoglobin levels and a higher hematocrit as a result of the plasma leakage" [49]. *Ralapanawa et al.* (2018) also found a significant difference in haemoglobin value between dengue patients who progressed to severe dengue and patients who did not. These results imply that extraction of haemoglobin and hematocrit value might be of interest.

   (a) **Haemoglobin value:** *Kavsaoğlu et al.* (2015) investigated this by using several PPG features, feature selection algorithms, and machine learning algorithms [51]. In fact, they extracted 40 time-domain features, 21 of which were from the PPG signal, 8 from the first derivative, 7 from the second derivative, and 4 from the combination of the two derivatives.

   (b) **Hematocrit:** Hematocrit estimation through PPG is a challenging task. Although we found studies exploring this, the literature was visibly much more limited [52, 53].

2.3.3.3 Time-Frequency Representations

Time-frequency analysis is a feature extraction approach which is very commonly used for PPG data, as well as for biomedical signals in general. Time-frequency methods are used to create a representation which displays the signal in the time and frequency domain simultaneously. The motivation behind this lies in the fact that these two domains are closely linked for any function, suggesting that having both together might result in a powerful representation.

Traditionally, the algorithms used are the Short Time Fourier Transform (STFT) and the Continuous Wavelet Transform (CWT). For the STFT, the signal is split into windows, and the Fourier transform is applied on each one individually. This leads to a uniform time-frequency spectrum, the spectrogram [54]. The CWT does not explicitly window the function, but rather produces a time-frequency representation by convolving the signal with a new function, the wavelet. The wavelet function is scaled (stretched or compressed along the time axis) and then shifted (translated along the time axis). As the wavelet is scaled by different factors (changing its frequency range) and shifted along the signal, this creates a time-frequency spectrum, the scalogram. Due to this scaling and shifting approach, the scalogram offers varying time-frequency resolution, which is often seen as an advantage over the STFT's fixed time-frequency resolution [54]. This distinction is displayed in Figure 2.4.

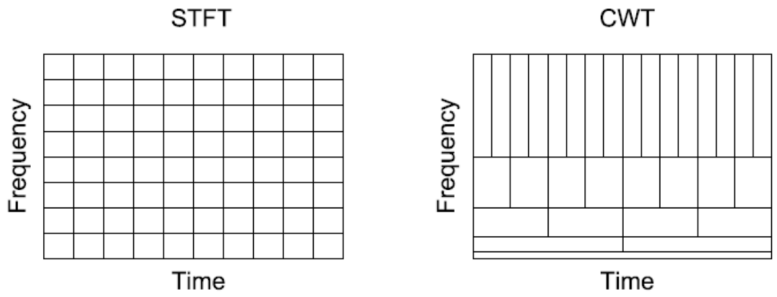


Figure 2.4: STFT spectrogram (fixed time-frequency resolution) and CWT scalogram (varying time-frequency resolution) (image from [55]).

That said, this technical difference has not prevented studies from utilising the STFT. For example, *Tjahjadi et al.* (2020) used the STFT for feature extraction for PPG signals, while *Allen et al.* (2021) used the CWT [56, 57]. These are only two of the numerous examples using time-frequency analysis as a feature extractor for PPG data. It is worth noting that a plethora of more advanced time-frequency analysis algorithms have been developed and studied in order to improve upon the time-frequency spectrum resolution of STFT and CWT. *Wang et al.* (2006) compares various of these algorithms, such as continuous wavelet transform (CWT), fixed-frequency complex demodulation (FFCDM), and variable-frequency complex demodulation (VFCDM), with regards to resolution [58]. Additionally, some of these have been compared with regards to PPG data [59]. Finally, we note that an improved time-frequency resolution does not necessarily imply better feature extraction, since other factors such as sparsity and dimensionality of feature space are also very important.

2.4 Machine Learning

Through its immense rise in popularity in the past decades, machine learning (ML) has established itself as an incredibly powerful tool, delivering high performance in tasks such as classification, regression, and pattern recognition across a vast range of applications. Despite being heavily affected by two of modern ML's most pressing issues, its interpretability and explainability, the biomedical and clinical sectors have been no exception, and have seen a plethora of ML applications.

In Section 2.4.1 we provide a brief overview of some of the ML algorithms that are relevant to this project. In order to further support our decisions, for each of the ML models presented in Section 2.4.1, we also note studies that have used these types of models in an application for dengue or PPG (Section 2.4.2). Finally, in Section 2.4.3 we provide a brief review on the evaluation metrics commonly used in ML.

2.4.1 Relevant ML Algorithms

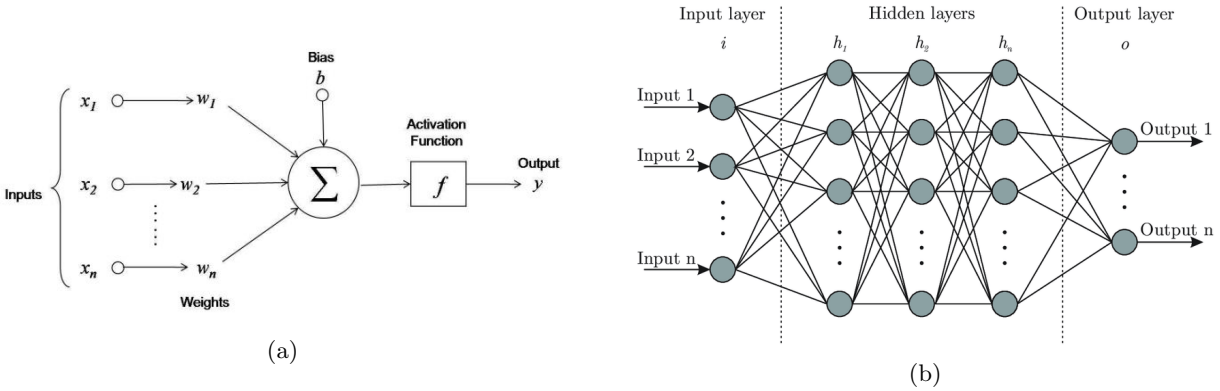
Given that our aim is to train using labelled segments of the PPG signals, we only consider supervised ML algorithms in the next sections. Supervised machine learning refers to the subset of ML algorithms which work and achieve learning through labelled data. Naturally, the labels in this project are dependent on the clinical question. For example, in the ICU-FU question, each segment would have to have a binary label indicating the origin of the recording (e.g. 1 - ICU, 0 - FU). In the case of the severity classification question the labels would not be binary (e.g. 0 - mild, 1 - moderate, 2 - severe).

2.4.1.1 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are a class of supervised machine learning algorithms loosely inspired after the operation of the biological brain. They are formed through the interconnection of a collection of individual units named artificial neurons. The structure of an artificial neuron and of an example artificial neural network are shown in Figures 2.5a and 2.5b respectively. The artificial neuron operates by multiplying each input with a weight, summing these products along with a bias, and finally passing that sum through an activation function. The output y is given by the equation below, where the second expression is derived by defining $x_0 = 1, w_0 = b$.

$$y = f\left(b + \sum_{j=1}^{n} w_j x_j\right) = f\left(\sum_{j=0}^{n} w_j x_j\right) \tag{2.1}$$

The ANN is constructed by placing consecutive columns of artificial neurons called layers. The first layer is called the input layer, followed by a number of "hidden" layers and finally an output layer. Every neuron of layer j is connected to all neurons of layers $j-1$ and $j+1$, meaning that its inputs are the weighted outputs of the previous layer's neurons, and its output after being weighted is one of the inputs of the next layer's neurons. This is the reason we refer to the layers of ANNs as "fully connected" or "dense".



(a) (b)

Figure 2.5: (a) Structure of an artificial neuron unit (image from [60]) (b) Example of the architecture of an artificial neural network (ANN) (image from [61]).

Learning in the ANN is achieved by changing the weights across the network in order to minimise a loss function. When an input is fed through the network, a process called a feedforward pass, the loss function is computed at the output layer. This refers to the error between the network's prediction for a particular input and the correct output for that input, the label. Based on this error, or more precisely the average of the error for several training examples, all the network parameters, consisting of the weights and biases, are adapted according to the optimiser and the backpropagation algorithm. These two algorithms combined are able to determine how much each of the network parameters should change in order to minimise the loss function. The process of feedforward, backpropagation, and weight update is repeated until the optimiser deems the current value of the loss function to be at or near a global, or most likely local, minimum.

There is a large number of parameters that can be changed in an ANN model, such as network architecture (number of layers, neurons per layer, activation function), optimiser and optimiser learning rate, batch size, and regularisation strategies, to name a few. Thus, finetuning is a task that must be approached in a structured way. We will provide further information on how this is attempted in our project in Section 3.6. Finally, it is worth noting that for this section, as well as for the ones that follow, we will not analyse in depth all the aspects of these complex systems, and rather aim for conciseness and clarity.

2.4.1.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are usually constructed as a two-stage system. The second stage of this pipeline is an ANN, that is, a series of fully connected layers leading to one or more output neurons. The number of output neurons as well as the activation function, just like in the pure ANN case, depend on the task the network is being trained for, such as binary classification, multi-class classification, or regression. The first stage of the system, which precedes the fully connected layers, is essentially a feature extractor, the main highlight of which is that it is designed to operate with 2-dimensional inputs. The core of this feature extraction stage is convolutional and pooling layers [62]. Figure 2.6 depicts a common and influential CNN architecture named VGG-16 [63]. We can see that the input image is passed through several convolutional and pooling layers before reaching the fully connected layers and, finally, the output.

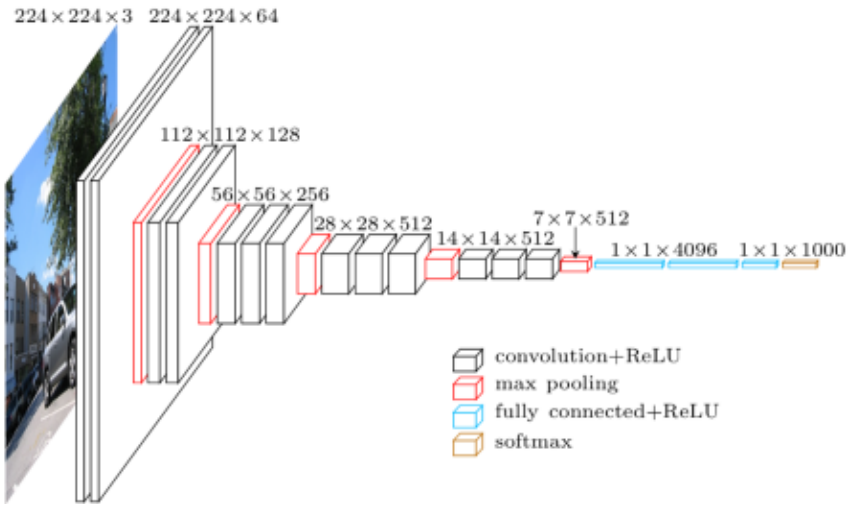


Figure 2.6: Convolutional and pooling layers followed by fully connected layers in the VGG-16 CNN (image from [64]).

Convolutional layers operate by sliding a filter over the input feature vector. In CNNs, both the filter and the feature vector are 2D images, mathematically denoted by matrices. The sliding process refers to performing an element-wise multiplication between the filter and a part of the feature vector and subsequently moving the same filter across the image and repeating. This process can be seen in Figure 2.7a. Note that the filter is (3x3) and its values are the smaller numbers in the bottom right corner of the darker cells. It is this localisation of features provided by the filter within the input that has allowed CNNs to achieve groundbreaking performance in image-related tasks. The result of this process is another feature map, the matrix on the right. This will

then be used as the input image to the next stage.

Pooling layers are used with the aim of downsampling the input representation, reducing its dimensionality and thus the network's computational cost. The most common types of pooling work by picking the maximum, average, or minimum element of a part of the image, leading to max, average, and min pooling. This process is shown for max pooling in Figure 2.7b, where the pooling size is (2x2), meaning that the algorithm picks the biggest number from each (2x2) block, reducing the number of features in the result.

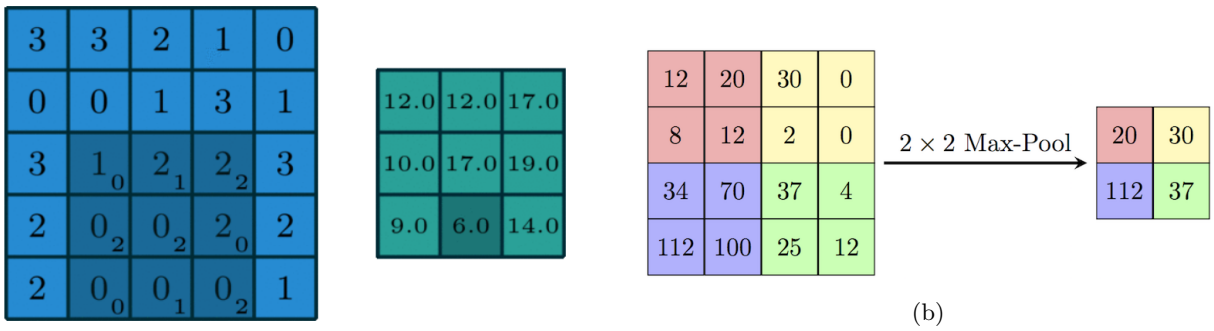


Figure 2.7: (a) Principle of operation of convolutional layer, filter is slided across the input image and element-wise multiplication between the filter and the input image occurs at each point (image from [62]) ; (b) Principle of operation of pooling layer, max pooling with pooling size (2x2) is shown (image from [65]).

Learning in CNNs occurs with the same exact way as in ANNs, through backpropagation and an optimiser. The weights that are trainable in this case are not numbers which multiply the outputs of the neurons, but rather they are the parameters of the filters, since it is through these filter weights and the elementwise multiplication that the features are localised and extracted. Note that pooling layers, contrary to convolutional layers, do not have trainable parameters. Regarding tuning, CNNs contain all changeable parameters of ANNs along with many more, and so are even harder to tune. As deep learning and computer vision have now been the topic of research for long enough, besides successful architectures and good practices, third-party frameworks that work as wrappers for model-building libraries have been developed, aiding us in the design of our CNN networks. More information on this is provided in Section 3.6.

2.4.2 Relevant Work for Dengue and PPG

Machine learning algorithms and techniques have already been applied extensively for both dengue and for PPG signals. Regarding PPG, ML models have been developed both to detect or predict a variety of diseases, as well as to estimate vital signs from the PPG signal. For dengue, ML models have been created in order to explore several aspects of the disease, such as management, prediction, and prevention. Very seldom, however, have ML models that combine PPG and dengue been explored. For us, this is both a positive and a negative fact, since it entails that our work is innovative and to a degree unexplored, but also that there does not exist a body of literature which can be used as reference or to provide confidence.

Table 2.1 shows a summary of some of the machine learning-related works that we found. As we previously mentioned the literature on both PPG and dengue separately is extensive, and so we only show some of the studies that piqued our curiosity. In the table, we include whether the application is PPG, dengue, or both, the type of ML model used, the inputs, the broad aim of the study, and a reference. For conciseness within the table, we use abbreviations for the different ML models. These can be found in Appendix A.

We also comment further on one study in particular [66]. This study refers to an FDA-approved medical device which uses feature extraction and ML algorithms with the aim of detecting haemorrhage. Unfortunately, however, the system, which is named Compensatory Reserve Index (CRI), is owned by Flashback Technologies and thus, even though their study appears to provide meaningful results, no information on internal algorithms is given. Finally, its proprietary nature makes it expensive, meaning that mass population application is unlikely.

Topic	ML model	Inputs	Aim	Reference
Both	ANN, CNN, LSTM	Time-Frequency	Investigate the relationship between a PPG signal and dengue severity	(2021) [7]
Both	-	-	Use CRI to detect shock and track the progress made through fluid resuscitation	(2016) [66]
PPG	CNN	Time-Frequency	Outperform classical HR estimation methods through a novel deep learning approach	(2019) [39]
PPG	CNN	Time-Frequency	Detect peripheral arterial disease by finetuning a pretrained AlexNet CNN	(2021) [57]
PPG	CNN, LSTM, variations	Time-Frequency	Detect and delineate PPG beats under noise-corrupted signals	(2021) [67]
Dengue	ANN, SVM, RF, XGBoost	Clinical and personal data	Predict whether a dengue patient would go into shock during their hospitalisation period	(2022) [68]
Dengue	DT, RF	Clinical and personal data	Predict dengue fever	(2020) [34]
Dengue	MLR, MnLR	Clinical data	Predict dengue case fatality	(2021) [35] [36]

Table 2.1: Different types of ML models along with their inputs and aims applied in the topics of PPG, dengue, or both.

2.4.3 ML Evaluation Metrics

This section refers to the contents of the "Result Evaluation" block of the high-level pipeline of Section 1.2.2. Naturally, choosing a metric first and foremost depends on the task, that is, a classification and a regression model employ fundamentally different metrics. Here we focus on the former as our models explore solely classification tasks. Finally, we note that the statistical literature on metrics is extensive and so we will focus on the most commonly used machine learning tools for evaluation.

ML models often report model performance by quoting a single number: accuracy. However, in most cases this does not provide sufficient information for how the model is in fact performing. This is particularly important in medical applications, such as in biomedical and clinical settings. Using the concepts of true positive, true negative, false positive, and false negative, it is possible to define metrics which are able to provide more information. We introduce these through the confusion matrix as shown in Figure 2.8. After this, Table 2.2 presents the concepts and metrics from the confusion matrix along with a brief description.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Figure 2.8: Confusion matrix along with metrics that can be computed from matrix entries (image from [69]).

It is worth noting that these metrics are computed for each of the classes, meaning that to produce a single metric for the dataset, one must average. The choices most commonly used are "macro-averaging", which computes the arithmetic mean across classes, or "weighted-averaging", which takes into account the proportion of each label within the dataset. Finally, we note that is much more insightful to calculate these metrics on the validation set, rather than on the training set, since the former accounts for overfitting, whereas the latter does not.

Concept	Expression	Description
True Positives (TP)	a	The number of examples that were correctly identified by the model as positive.
False Positives (FP)	b	The number of examples that were incorrectly identified by the model as positive.
True Negatives (TN)	d	The number of examples that were correctly identified by the model as negative.
False Negatives (FN)	c	The number of examples that were incorrectly identified by the model as negative.
Accuracy	$\frac{(a+d)}{(a+b+c+d)}$	The proportion of total predictions that was correctly identified.
Positive Predictive Value (Precision)	$\frac{a}{(a+b)}$	The proportion of actual positive examples from all the examples identified as positive.
Negative Predictive Value	$\frac{d}{(c+d)}$	The proportion of actual positive examples from all the examples identified as positive.
Sensitivity (Recall)	$\frac{a}{(a+c)}$	The proportion of actual positive examples that was identified correctly.
Specificity	$\frac{d}{(b+d)}$	The proportion of actual negative examples that was identified correctly.

Table 2.2: Name (left), mathematical expression (middle), and brief description (right) of the concepts and metrics derived from the confusion matrix.

Where we want to maximise precision and recall simultaneously, we introduce the F1 score, defined as the harmonic mean of precision and recall. The choice of harmonic mean over arithmetic mean can be justified as the harmonic mean punishes extreme values. Thus, if a model has either very poor precision or very poor recall, the resulting F1 score is low. Further, if we want to assign more importance to recall than to precision we can do so by slightly altering the F1 score equation through the parameter β, resulting in the F-beta score.

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.2}$$

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{2.3}$$

The last metric we introduce is the area under the receiver operator characteristic curve (AUC-ROC). The ROC curve is a graphical representation of the performance of a classification model for different values of classification thresholds. The classification thresholds refer to values which we define in order to map the probabilities of logistic classification to a nominal value, that is, one of the classes [70]. The ROC curve depicts performance by plotting sensitivity against (1-specificity) for different values of the classification threshold. The AUC simply measures the area underneath the ROC curve in the $[0, 1]$ range. Thus, the AUC-ROC metric attempts to provide "an aggregate measure of performance across all possible classification thresholds" [70].

Although we could argue that some metrics have a particular advantage over others, all of the metrics mentioned above have their own drawbacks. In fact, the most common conclusion found in literature regarding the choice of metric for model evaluation is that there is no single metric that is able to encompass the entirety of a model's performance or which can capture all the desirable properties of a model. For this reason, a collection of several metrics are used to report about a model's performance [71].

Chapter 3

Design and Implementation

In order to be able to investigate the clinical questions of interest, we develop a processing and analysis pipeline. The pipeline serves as a tool for accessing and utilising the raw dataset with the aim of running experiments. This section is dedicated to providing a technical description of the design and implementation choices made in the development of this tool. In Section 3.1, we provide a high-level overview through a system flowchart as well as highlight key aspects of the operation of the system. Then, in Sections 3.2-3.6, we analyse the technical implementation of each of the pipeline stages separately.

3.1 System Design

As explained in Section 1.2.2, one of the main aims of the project is that the entire system is designed and implemented with the aim of being modular. More specifically, we pursue modularity in being able to ask different clinical questions, as well as in the ability to easily experiment with different preprocessing, quality checking, feature extraction, and machine learning methods. For these stages we base our design on the literature presented in Chapter 2. That said, we do not consider implementing "as much as possible" from what is presented in Chapter 2 a project objective. In other words, the aim of modularity is not to increase the quantity of implemented material, but rather to enable effective experimentation and exploration of the problem. Given this, it is intuitive as a design choice to implement stages as separate modules, each of which is parameterised in order to achieve different configurations.

3.1.1 System Flowchart

Figure 3.1 depicts a complete system flowchart consisting of the main controller (green module), the different pipeline stages (blue modules), and a summary of the processes which occur for each of these stages. Additionally, we show the stages of the pipeline where the raw dataset is accessed.

The main controller module sets global parameters dictating a particular experimentation configuration and subsequently executes the stages sequentially. The execution flow goes from top to bottom as indicated by the arrow on the left, while the arrows between the main controller and the pipeline blocks specify the input and output data structures for each of the blocks. More details on the data structures for each block are found in the corresponding sections below, as well as in the Wiki section of the GitHub repository[1]. In the subsequent sections, we expand on the processes of each block as shown in the system flowchart and discuss the methods and algorithms used for their implementation.

[1]It is worth noting that it is important that the inputs and outputs to each block are well-defined and documented, since this allows for easier interfacing in the case of future use. For example, if someone wishes to use the pipeline but wants to use their own feature extraction algorithms, then they are able to do so by altering the input and output data structures to match those required by the main controller module.

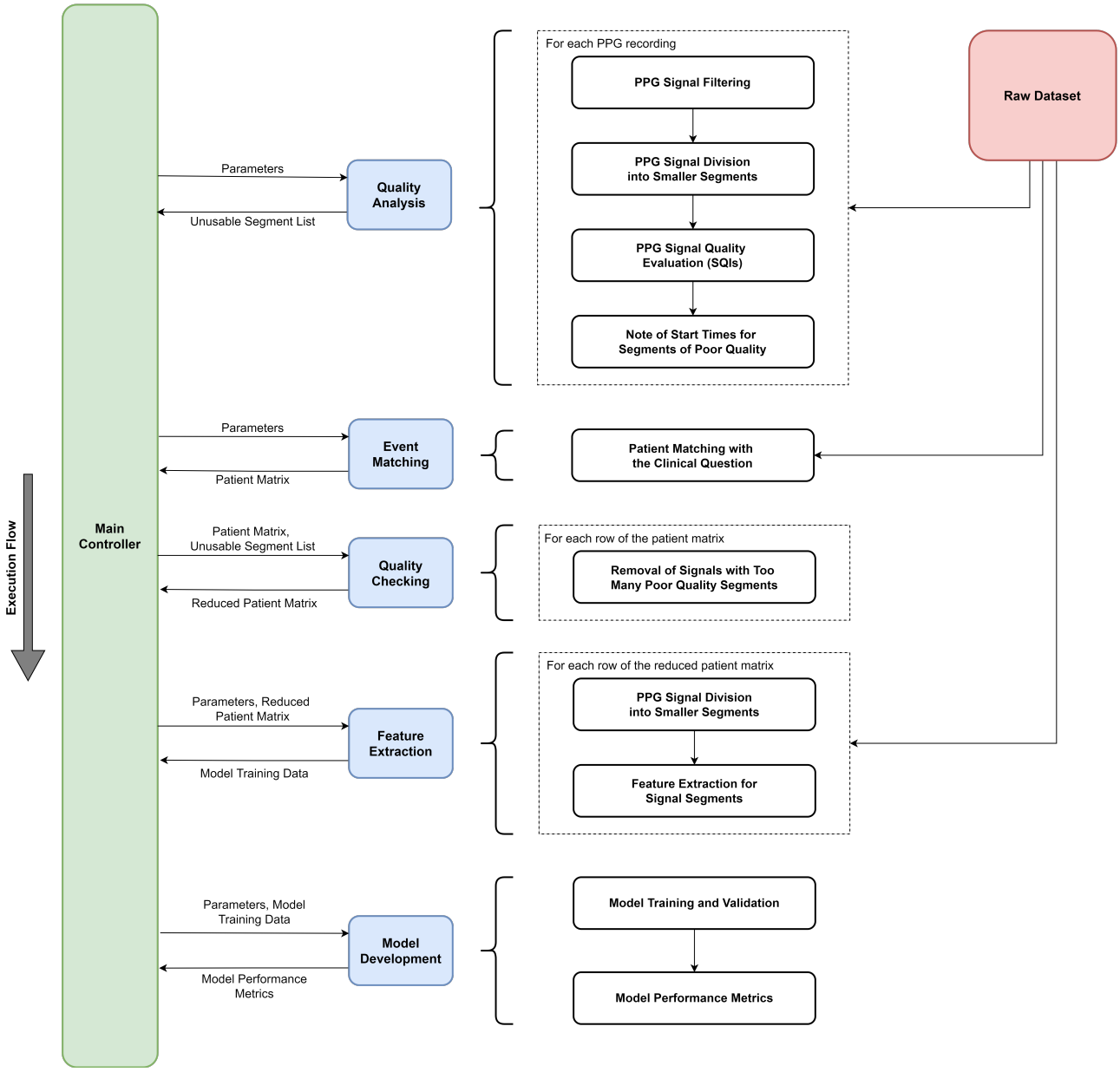


Figure 3.1: The system flowchart is depicted. The main controller (green module) sets global parameters specifying a particular experimental configuration and subsequently executes the pipeline stages (blue modules) sequentially. Also shown is a summary of processes or each stage and access points of the dataset.

3.1.2 Use of Checkpoints

In order to design the pipeline in an efficient manner, as is specified by the pipeline evaluation criteria of Section 1.2.2, we make use of checkpoints. Checkpoints refer to the process of saving the result (*i.e.* the output data structure) of each stage to a file after the execution of that stage. The implication of this is that if a particular configuration of a stage has been executed at some time in the past, then the result is loaded automatically through the corresponding pickle[2]. For example, if we want to run an experiment for the ICU-FU question and event matching for this task has been done in the past, then the result of the event matching process (i.e. the patient matrix) already exists, and we just have to load it. If a process has not been ran in the past, then it is executed, and its result is saved in a pickle file upon completion. Consequently, the file can be loaded in the future, avoiding the repetition of work and allowing for efficient experimentation.

[2]The saved files are create with the Python *pickle* library, and so we refer to them as pickles.

3.2 Quality Analysis

For quality analysis we have adopted a perhaps unexpected approach. Instead of performing quality checking on each individual signal segment during feature extraction, we carry out a process on the entire dataset independently as the first stage of the pipeline, as shown in Figure 3.1. This process includes undertaking quality analysis for all patients throughout a cohort (either adult or children cohort) and noting parts that are of poor quality. We chose this approach for the sole reason that quality analysis is dependent only on one factor: the length of the signal segments. Note that we are thinking in terms of signal segments as it is these smaller parts of the signals that we will later be performing feature extraction on. As we expect that a large number of experiments will have to be run using the same signal segment length, performing quality checking as part of the feature extraction stage each time would be unnecessary repetition of the same process. Thus, we avoid this by carrying out quality analysis only once as a separate pipeline block.

For each patient, or, more specifically, for each PPG recording file since patients often have multiple recordings, quality analysis consists of three steps: signal preprocessing and segmentation, evaluation of SQIs, and a binary decision on usability. Sections 3.2.1 and 3.2.2 provide the technical details about the first two stages, after which Section 3.2.3 presents the full quality analysis algorithm.

3.2.1 Signal Preprocessing

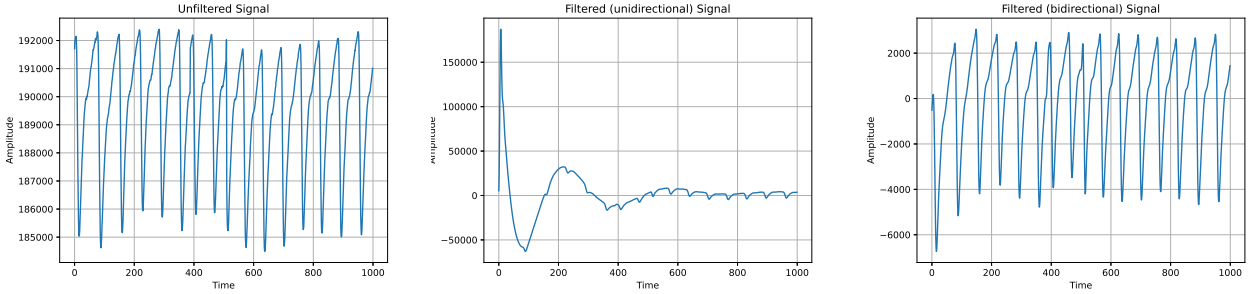
The aim of a preprocessing stage is the removal of the maximum amount of unwanted signal components, and consists of filtering and normalising. For filtering we have created a function *filter_signal*. The input parameters to this function are the signal, a dictionary of filter parameters as shown in the code listing of Figure 3.2, and a Boolean for optionally plotting the filter frequency response. In reality, the function is a wrapper for several functions of the popular Python signal processing library *scipy*, and was written for modularity and ease of use. As shown in the figure, we are currently using a 4^{th} order Chebyshev II filter with a $0.15 - 20$Hz passband, as suggested by *Liang et al.* (2018) in [20].

```
filter_params = {
    "type" : "cheby",
    "order" : 4,
    "sample_rate_Hz" : 100,
    "low_cutoff_Hz" : 0.15,
    "high_cutoff_Hz" : 20
}
```

Figure 3.2: Filter parameter dictionary.

The function creates the filter using these parameters and subsequently filters the signal using cascaded second-order sections. We note that the implementation utilises a bidirectional (zero-phase) filter. This is because unidirectional digital filters possess two main drawbacks. First, they naturally introduce large transients, which, for the filter shown above, last around 5-10 seconds, during which the signal is essentially rendered unusable. These do not occur in the case of bidirectional filters. More importantly, however, unidirectional filters can introduce phase distortion which may prove problematic when using frequency-based approaches in the feature extraction stage later on, while bidirectional filters do not. The results of filtering with both of these types of filters are shown in Figure 3.3, where the massive transients cause by the unidirectional filters are clear.

After filtering the entire patient signal. we split it into smaller segments, which are then normalised between $[-1, 1]$. The reason we do not normalise the entire signal at once is because of the existence of large spikes throughout the dataset, which would diminish the amplitude of normal signal segments. Finally, it is worth noting that an additional technique potentially useful in the preprocessing stage is signal smoothing. However, the SmartCare data which we use is already subject to some basic internal preprocessing and so smoothing and averaging were deemed to not be necessary.



(a) (b) (c)

Figure 3.3: Comparison of (a) unfiltered signal (b) filtered signal using a unidirectional filter and (c) filtered signal using a bidirectional filter.

3.2.2 Signal Quality Indices

Signal quality indices are utilised as a decision-making tool regarding whether a piece of signal is of sufficiently high quality to be used as training data for the model. Given the fact that quality analysis exploration is not one of the project objectives, the sole aim of this stage was to obtain a method which can safely reject poor quality signals. Therefore, even though our literature review suggests utilising skewness, the optimal SQI for PPG signals as discussed by *Elgendi* (2016) in [25], it was decided that it is preferred to start with a simpler approach and increase complexity incrementally, should the simple approach prove unreliable.

Thus, as a starting point, we implemented only two basic SQIs: zero-crossing rate, and MSQ. It is worth noting that the exact functions for the calculation of these metrics are copied from the Imperial-developed software library *vital_sqi*. The zero-crossing rate Z_{SQI} represents the rate at which the signal changes sign, and is defined as shown in Equation 3.1.

$$\mathrm{Z_{SQI}} = \frac{1}{N}\sum_{n=1}^{N} I\{y < 0\} \tag{3.1}$$

The MSQ SQI as defined by *Elgendi* (2016) in [25] tracks the agreement between two peak detectors when applied on the PPG signal. This is given by Equation 3.2, where S_{PD1}, S_{PD2} represent the set of peaks generated by two distinct peak detector algorithms. The current implementation uses adaptive thresholds as PD1 and Billauer's method as PD2.

$$\mathrm{M_{SQI}} = \left(S_{\mathrm{PD1}} \cap S_{\mathrm{PD2}}\right) / S_{\mathrm{PD1}} \tag{3.2}$$

The evaluation of these SQIs is performed on each segment, rather than the entire signal. For each of the SQIs evaluated we provide a range of "allowed" values. The zero-crossing rate is directly related to the frequency of the signal. Assuming a clean PPG signal, we would expect 2 zero crosses for a single period. Therefore, assuming a (relatively broad) heart rate range between $45 - 120$ bpm, we would expect zero crossing to lie between $0.015 - 0.04$, given that the sampling frequency is 100Hz. For MSQ, any value different from 1 entails that there is a disagreement between the two peak detectors, meaning that peaks are not well-defined. However, different peak detector algorithms are derived through very diverse implementation methods, and are therefore not completely reliable. For this reason, we relax the condition to a minimum MSQ value of 0.9.

For a given segment, if any of the evaluated SQIs are outside the allowed range, the segment is considered of poor quality and is marked as unusable. Figure 3.4 depicts the evaluation of these two SQIs for four signal segments of varying quality, along with a binary decision on whether the segment is usable. As can be seen, despite their simplicity, the evaluated SQIs are able in all cases to quantify the quality. Both SQIs are in the allowed range for the excellent and good segments, while in the poor segment one of the SQIs is not, and so the segment is rejected. Finally, in the completely unusable segment case, both SQIs are outside the valid range.

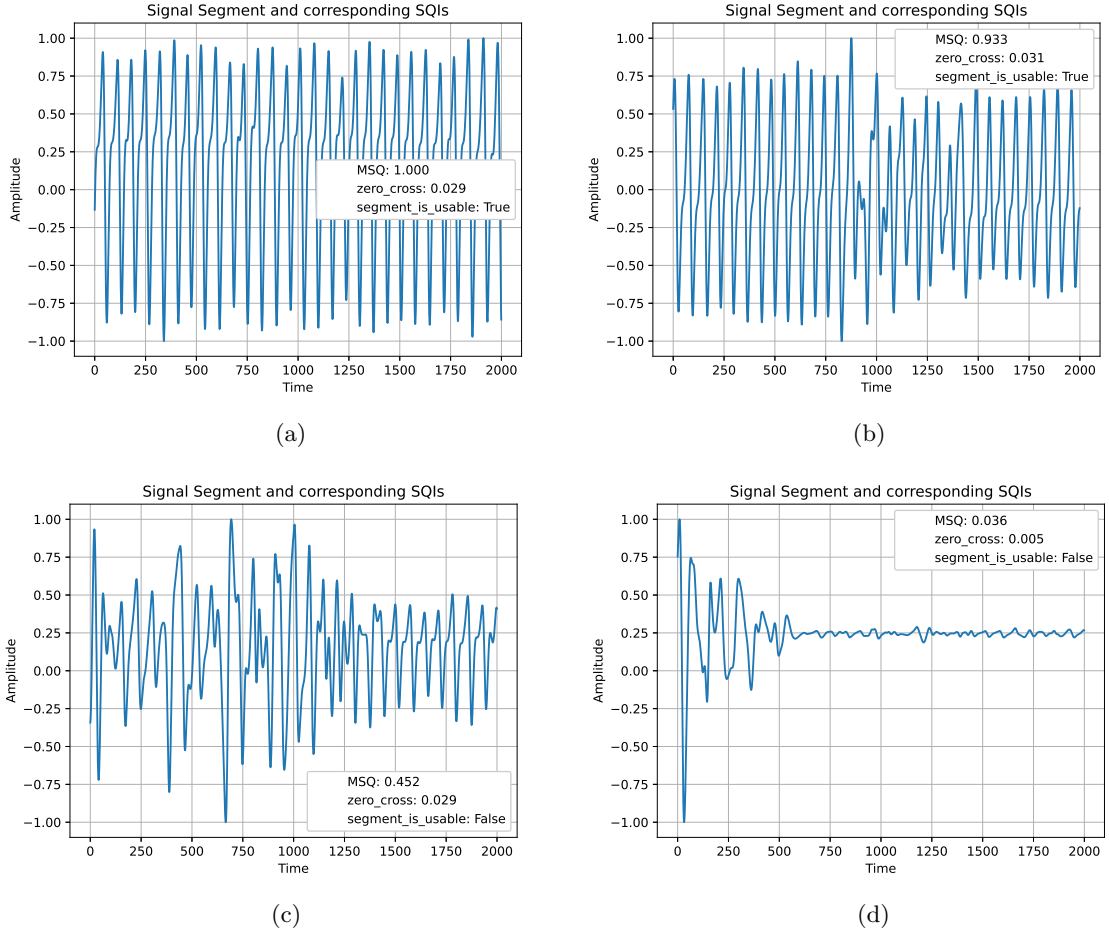


Figure 3.4: Evaluation of zero-crossing rate and MSQ SQIs on a (a) excellent segment, (b) good segment, (c) poor segment, (d) completely unusable segment.

3.2.3 Quality Analysis Algorithm

In this section we briefly outline the entire quality analysis algorithm by using the methods explained in the sections above. The function developed for quality analysis was named *unusable_segment_list*. For a given value of signal segment length, the output of this process for each patient file is a collection of starting times of unusable segments.

For all PPG recording files within the cohort:

1. Read the signal from the patient file.
2. Filter the entire signal using the aforementioned *filter_signal* function.
3. Split the filtered signal into segments. For each segment the function *segment_is_usable* is called:
   (a) Preprocess the segment. Since the entire signal is already filtered, this only performs normalisation.
   (b) Compute SQIs (*i.e.* MSQ and zero-crossing rate) for the segment.
   (c) If any of the calculated SQIs is outside the allowed range then the segment is deemed unusable and the *segment_is_usable* function returns *False*, otherwise returns *True*.
   (d) If the *segment_is_usable* function returns *False* then the start time of this segment is added to the output list, named *unusable_segment_start_times*.

3.3 Event Matching

The role of the event matching stage is to select the patients that will be used for forming the model dataset given a particular clinical question. It is also required to identify the correct parts of these patients' recordings to be used, as well as to label them accordingly since we operate within a supervised learning context. Given this, the output of this stage for a single patient is a 3-tuple of the format (*(patientID, file_name), (start_datetime, end_datetime), label*). Here *start_datetime* and *end_datetime* refer to the beginning and end of the time period of interest. Thus, the overall output of this stage is a list of 3-tuples, and is referred to as the "patient matrix" in the remainder of this project.

Unfortunately, the finalised clinical questions were specified relatively late in the duration of the project. Because of this, a significant amount of work was allocated to developing a general event matching framework based on a particular type of clinical event (such as shock, reshock, or fluid administration), an algorithm which remained unused in the end. That said, and even though this is not used to generate results, we consider it important to analyse since it may be useful for future work. Therefore, in Section 3.3.1 we analyse our approach regarding a general event matching algorithm for particular clinical events, and move on to event matching for the two final clinical questions, ICU-vs-Follow-up and severity classification, in Sections 3.3.2 and 3.3.3 respectively.

3.3.1 General Event Matching

3.3.1.1 The Mismatch Problem

Given the contents of the clinical dataset as mentioned in Section 1.2.1, we should, theoretically, be able to ask clinical questions related to any subset of this dataset. For example, given that the clinical dataset includes details on fluid administration, such as which patient was administered fluids, the type and amount of fluid, and the date and time of administration, we should be able to investigate the effect of fluid administration on a patient's PPG. This, unfortunately, is not the case.

For example, in the case presented in Figure 3.5, SC (orange) and GE (blue) refer to the two monitoring devices that extract a PPG signal from the patient. We can see that the period during which fluids were being administered (grey) overlaps with the period of PPG recording. This would therefore allow us to investigate the effect of fluid administration further. However, looking at shock and reshock events, these do not occur within the period of recording, and therefore we would not be able to use this patient if we wanted to investigate the effect of shock or reshock on the PPG.

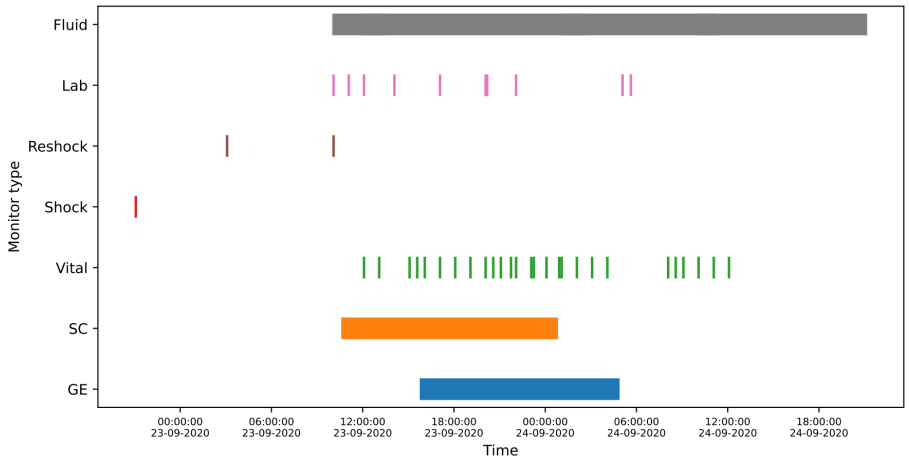


Figure 3.5: Summary of data for one patient. This includes the time intervals where their PPG signal was extracted (SC and GE graphs), as well as any clinical events. We observe that Shock and Reshock events do not occur during the PPG recording periods.

3.3.1.2 General Event Matching Algorithm

This motivates the development of an algorithm which automatically performs checks of this kind in order to produce a list of valid patients for a given question. We express the functional requirements of this algorithm as follows. Given a clinical question, we should be able to iterate through all patients and determine the subset of them who are able to support this question with sufficient data from their PPG recordings. Furthermore, we should note the relevant time interval for each of those patients. By relevant time interval we mean that if a patient's PPG recording overlaps with the clinical event under consideration, we should extract a time interval of interest; for example from some time before the event until some time after the event. Naturally, this is dependent on the clinical event itself. If, for example, we are looking at shock we might focus on three distinct intervals: pre-shock, during-shock, and post-shock. If, however, we are investigating the effect of fluid administration, we might only consider time intervals for pre-administration and post-administration. The lengths of these time intervals are also dependent on the clinical question.

Part of the solution had already been developed prior to the beginning of this project[3] and is able to iterate through the entire dataset and output information for each patient. This is in the form of a Python dictionary in which the key is the patient ID and the value is a collection of information for that patient. The format of the information for one patient is shown in the code listing of Figure 3.6. Note that "dataframe" refers to the main data structure of the Python data analysis library *pandas*. The fields listed underneath are columns of that dataframe.

```
{patientID: # Different files from GE monitor
            [(start_datetime, end_datetime, duration), ...]

            # Different files from SC monitor
            [(start_datetime, end_datetime, duration), ...]

            # Vital signs dataframe
            Time, Temperature, Pulse, LB1, SystolicBP, DiastolicBP, RespiratoryRate, SpO2

            # Shock dataframe
            Time, Pulse, SystolicBP, DiastolicBP, RespiratoryRate, Hematocrit, Temperature

            # Reshock dataframe
            Time, Pulse, LB1, SystolicBP, DiastolicBP, RespiratoryRate, Hematocrit, Temperature

            # Lab tests dataframe
            Time, WBC, HCT, PLAT, UREA, CREAT, AST, ALT

            # Fluid administration dataframe
            StartDatetime, EndDatetime, FluidVolume
}
```

Figure 3.6: Matching dictionary for one patient. The key is the patient ID and the value is a collection of information about that patient.

The remainder of the algorithm, depicted in Figure 3.7 as a flowchart, iterates through all the patients, that is, all the keys of the dictionary, and verifies whether the patient can be matched with the question. This is achieved by sequentially examining the different reasons for which a patient cannot be matched:

- The dataframe corresponding to the clinical event under investigation is empty.
- The time of occurrence / time period of the clinical event does not exist within the time span of one of the files of the SC monitor.
- There is insufficient data before and after the time of occurrence / time period.

[3]Algorithm developed by Stefan Karolcik.

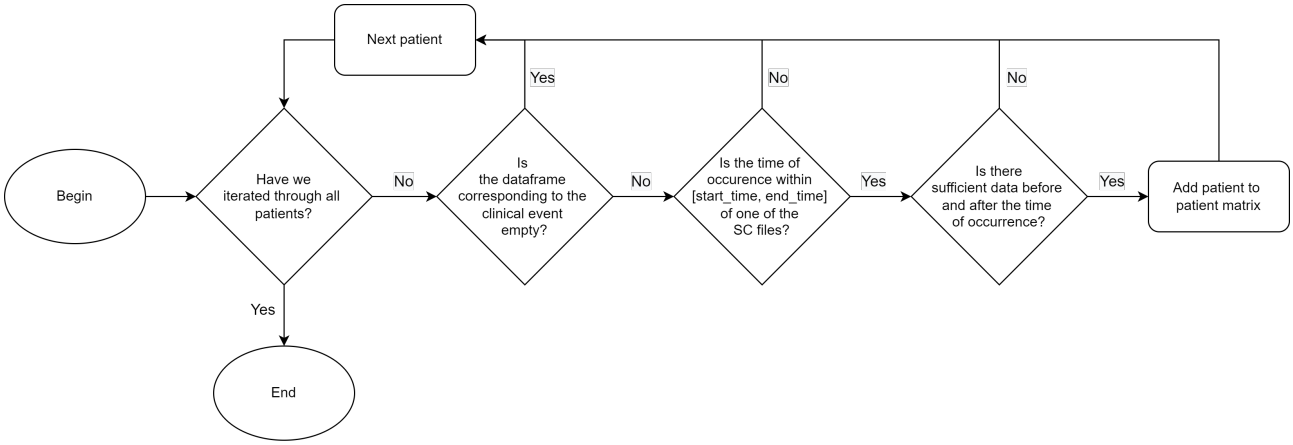


Figure 3.7: Matching algorithm flowchart, each patient has to pass several staged checks (shown here as conditionals) in order to be added to the list of matched patients.

3.3.2 Event Matching for ICU-vs-Follow-up

Event matching for the ICU-FU question is relatively simpler than general event matching, primarily because the list of patients for the question was given from the beginning. Thus, the task was reduced to obtaining the time interval of interest for each of the classes (ICU, FU) for all patients within the given list. Our approach for obtaining these time intervals was based on the fact that follow-up recordings are always much shorter than ICU recordings. For example, the code listing of Figure 3.8 shows the four SC recordings for patient 2211, where the format is *(file_name, start_datetime, end_datetime, duration_hours)*. We observe that the duration of the follow-up session, which is 6 days after discharge, is many times smaller than the ICU recordings, lasting less than 9 minutes.

```
('20200806T115336.285+0700.csv', '2020-08-06␣11:53:36', '2020-08-06␣17:53:48', '6.003')
('20200806T175459.064+0700.csv', '2020-08-06␣17:54:59', '2020-08-07␣07:58:33', '14.06')
('20200807T081656.406+0800.csv', '2020-08-07␣08:16:56', '2020-08-07␣14:11:54', '5.916')
('20200813T094524.441+0800.csv', '2020-08-13␣09:45:24', '2020-08-13␣09:54:10', '0.146')
```

Figure 3.8: Example of list of SC recordings for patient in the ICU-FU question. Note the difference in the duration (last entry) for the final (follow-up) recording against the remaining (ICU) recordings.

The implication of this is that if we were to use the entire signal for both the ICU and FU classes, then this would lead to a massively imbalanced dataset dominated by the ICU class. In this case, we decide to aim for a balanced dataset, containing both classes in equal amounts. Consequently, the length of the time interval is determined by the duration of the follow-up recording. Therefore, for a given patient, we take the entire follow-up recording from start to finish and subsequently take an equally long signal from their ICU recording.

3.3.3 Event Matching for Severity Classification

For the severity classification question, the list of patients was also given in advance. This is because patient severity is documented within the dataset via a naming convention. Namely, patient IDs are generated using the name '01NVa-003-XXXX' where the number XXXX is selected according to the following convention:

- 2151-2199 - Adult mild patient in ward (mild class)
- 2000-2099 - Adult patient admitted to ED with some complications (moderate class)
- 2100-2149 - Adult ICU patient (severe class)

It should be noted that if a patient had a mild infection in the beginning but then at any point developed severe symptoms they are classified as severe, and so mild patients can be assumed to be mild throughout the entire duration of hospitalisation. Finally, since signal length constraints similar to the ICU-FU question do not apply

for this question, we use the entirety of the signals for the three classes. While in this case the dataset is not perfectly balanced, it is not dominated by one of the classes as would be the case in the ICU-FU question, and so we do not explicitly enforce a balanced dataset.

3.4 Quality Checking

At this stage of the pipeline we have obtained the patient matrix and the unusable segment lists, as provided by the outputs of the event matching and quality analysis blocks respectively. For a particular patient, the patient matrix is equivalent to possessing a labelled time interval of interest. Rather than quality checking the segment defined by that time interval by splitting it into segments and computing the SQIs, we only need to ensure that it contains sufficiently few unusable parts. This is achieved by looking in the list of unusable segments for that patient. More specifically, we calculate the percentage of the signal that is unusable, as shown in Figure 3.9.

In this case, the time interval of interest for this patient is comprised by 9 segments, according to the length of the unusable segments. Since 3 out of the 9 segments are of unacceptable quality, the unusable percentage for this signal is 33%. However, the current state of our implementation defines the maximum allowed unusable percentage as 10%. In these cases, where the percentage for a patient is higher than the maximum allowed percentage, then that patient is removed from the patient matrix. This process is repeated for all rows of the patient matrix. After having potentially removed patients who were unfit from a quality perspective, we refer to the result of this stage as the reduced patient matrix.

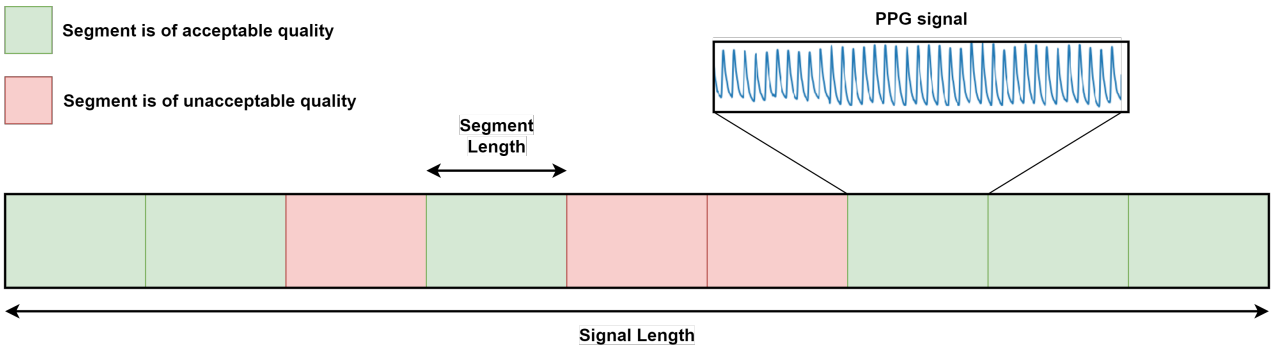


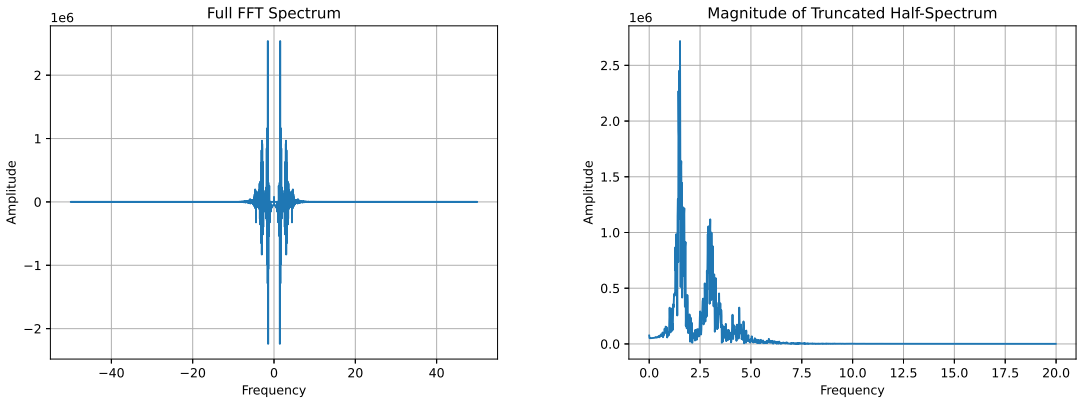
Figure 3.9: Example of a signal from the patient matrix in order to outline the operation of the quality checking process. In this case, 3/9 segments are of unacceptable quality, equivalent to a percentage of 33%. Since this is higher than the maximum allowed 10%, the patient is removed from the patient matrix.

3.5 Feature Extraction

The aim of the feature extraction stage as a pipeline block is to transform one row from the patient matrix (*i.e.* a labelled time interval) into a collection of (*feature_map, label*) tuples. These structures can then be used in a supervised learning setting for training, validating, and testing a machine learning model. Obtaining a collection of labelled feature maps for each patient in the patient matrix is achieved by segmenting the signal defined by the [*start_datetime, end_datetime*] time interval, and subsequently applying a FE algorithm on each segment. Once again, one of the main aims of this stage is modularity, so as to enable easy comparisons between the different FE approaches through their corresponding results.

As a baseline feature extraction method we simply use the raw, filtered PPG signal itself. In this case, for each of the segments the feature map will be an array-like structure holding the signal values. As mentioned in Section 2.3.3.1, however, although using the raw signal is the simplest approach, it is also the most uninformative. Given that this is the initial exploration of this dataset, especially with respect to investigating severity, there is merit in testing general feature extraction methods. More specifically, we want the FE stage to be able to achieve both time-domain, frequency domain, and time-frequency representations.

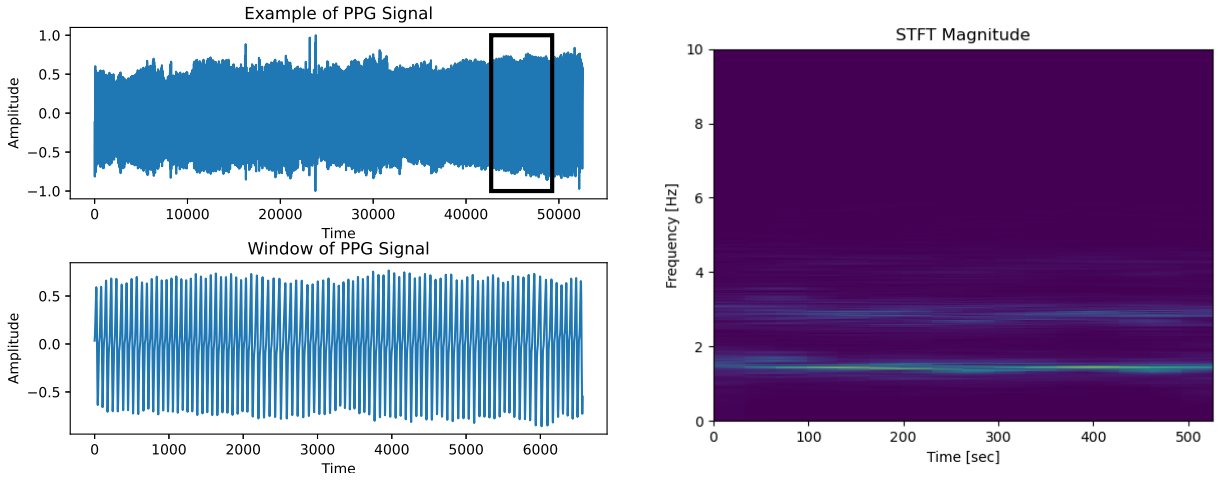
With regards to frequency-domain representations, the FE stage includes an implementation for the Fast Fourier Transform (FFT). Since the PPG signals are real-valued the FT produces a symmetric frequency spectrum, an example of which is shown in Figure 3.10a. Therefore, we use the function *rfft*, which provides the one-sided (*i.e.* half-spectrum) FFT. Given the sampling frequency of 100Hz, the spectrum is defined for $[0-50]$Hz. However, since we use a filter with a cutoff frequency at 20Hz, the $[20-50]$Hz range is likely to be negligible in amplitude and therefore unlikely to contain valuable learnable features. Therefore, the feature map which we use as model data is the magnitude of the spectrum in the range $[0-20]$Hz, as shown in Figure 3.10b.



(a) (b)

Figure 3.10: (a) Full FFT spectrum on PPG signal segment and (b) magnitude of truncated half-spectrum.

Regarding time-frequency representations we utilise a *scipy* STFT implementation as a starting point for our analysis. This process splits the signal, computing the Fourier transform for each of the windows individually. In Figure 3.11a we show the signal on which we apply the STFT, along with a magnified plot of the annotated window for which the FFT will be computed. In this case, the selected window size will lead to a signal split into 8 windows. Thus, the corresponding STFT spectrogram depicted in Figure 3.11b is composed by 8 stacked FFTs. For easier visualisation the view in the plot has been limited to the $[0-10]$Hz range.



(a) (b)

Figure 3.11: (a) Example of PPG signal (top) and magnified window (bottom) on which the FFT is going to be applied during STFT. (b) Corresponding STFT for the entire signal.

As mentioned in Section 2.3.3.2, obtaining vital signs and clinical features from the PPG signal could potentially be a highly beneficial approach, hence our literature review on the topic. One of the reasons for this is that they are relatively more intuitive, especially for clinical staff who are unlikely to come from a technical background.

Additionally, clinical features may in general provide more explainability for the models, a characteristic which is highly desirable and currently one of the most pressing issues in the machine learning area. Unfortunately, however, the lack of available algorithms for extracting such features which would work "out of the box" entailed that we would have to allocate a prohibitively large amount of work for their implementation, especially since we would have to obtain several of them to provide sufficient features for the models. Since this would in turn likely severely limit the amount of time for experimentation, it was decided to stick to the remaining FE methods and instead increase the amount of experimentation.

3.6 Model Development

Our approach regarding the model development stage consists of using the closest study we found as a starting point; this is *Hadjimarkou* 's MSc thesis (2021) [7]. Using only STFT as feature extraction, he explored several configurations of ANNs, CNNs, and LSTM RNNs, and subsequently presented the highest performing one for each of the investigated experiments. However, given that we are working with a new dataset and a more extensive set of feature extraction methods, while the reported results are a solid starting point, we also aim to undertake an exploratory approach regarding these models.

Furthermore, we believe that the novelty of the project motivates the implementation of a set of baseline models besides those presented in [7][4]. This refers to experimentation using a collection of commonly-applied and well-studied algorithms within the machine learning domain. By performing a wider exploration of the model search space, this approach aims to provide a more complete analysis of the questions under investigation. While we believe that an in depth study of a single model type is likely more effective in terms of optimising performance, we consider a wider, albeit not as deep exploration to be more beneficial to the value of this work. More specifically, we believe that the collection of baseline results gathered through this approach is more valuable to future research than an in depth investigation of a single model type.

The set of baseline models consists of decision trees, random forests, and support vector machines (SVMs). For the use of these baseline models as well as for multi-layer perceptron (MLP) ANNs, we utilise several implementations of the Python library *scikit-learn*. For the use of CNNs we utilise the high level API *keras* with *tensorflow* as backend.

With regards to CNNs specifically, we are aware of how difficult it is to properly optimise these systems manually, especially with regards to understanding performance and tuning the large number of hyperparameters. For this reason, we employ a third-party framework named *comet.ml*. We use this in order to visualise our experiments by tracking specific metrics, as well as for hyperparameter optimisation. Concerning the latter, *comet.ml* provides an "Optimizer" tool, which is able to perform different types of exploration of the hyperparameter search space. These include random search, grid search, and, most importantly, a Bayesian optimisation algorithm which implements the adaptive Parzen-Rosenblatt estimator [72]. Therefore, we aim to utilise the Bayesian optimiser for hyperparameter tuning. It is worth noting that the *comet.ml* platform can be used for exploring and optimising any of the implemented models, but was used only for CNNs due to their notoriety for difficult manual tuning.

[4]This refers to ANNs and CNNs. Different types of RNNs are not relevant for us since we do not perform any experiments which use the order or the sequentiality of the data.

Chapter 4

Experimental Setup

The aim of this chapter is to provide an outline of the set of experiments we aim to explore using the processing and analysis pipeline described in Chapter 3. In Section 4.1, we define the baseline models which will be used for experimentation, specify the type of results we obtain, and finally comment on the way through which we obtain those results. Then, Sections 4.2 - 4.4 present each of the experiments, highlighting their aims and structure. The results for each of these will be presented in Chapter 5. Finally, we note that the experimental setup is identical for both clinical questions (ICU-FU and severities).

4.1 Definition of Baseline Models

As aforementioned in Section 3.6, the novel nature of the project motivates the use of baseline models for achieving a wider exploration of the model search space. The collection of baseline models consists of decision tree, random forest, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP) classifiers.

To begin, we use out-of-the-box configurations for all of the models. This refers to the use of default values for all model hyperparameters, as provided by the implementation library *scipy*[1]. Subsequently, in order to further broaden the exploration, for each of the model types we arbitrarily choose one of the main model hyperparameters and experiment with a limited set of values. The aim of this process is not to optimise the reported metrics, since, as mentioned in Section 3.6, this is not the primary goal of our experimentation. Rather, we do so to examine whether hyperparameter alterations strongly affect model performance. If our intuition is validated and a correlation is in fact observed, we can conclude that the reported results are an underestimation of the true capabilities of the underlying models since they can be further optimised. Consequently, we will be able to suggest that performing hyperparameter tuning as part of future research is very likely to yield higher model performance, especially for model types which exhibit strong correlation.

The resulting set of baseline models which will be used for experimentation is presented in Table 4.1, and consists of 8 distinct configurations. For each of the configurations we note the model type, the name of the hyperparameter that is altered, the value for that hyperparameter, as well as an abbreviation for that model configuration for ease of reference in the next sections[2].

Finally, we comment on the format of the results and the way through which they are obtained. In all of the experiments, we track a set of five metrics: accuracy, macro-precision, macro-recall, macro-F1, and weighted-F1[3]. Additionally, in order to obtain a more realistic estimate of performance we employ K-fold cross validation throughout all the experiments. This refers to the process of running K training iterations, each of which uses a different portion of the dataset for validation.

[1]The default values for each of the model types can easily be found in the library's online documentation.

[2]It is worth mentioning that the only exception to using *scipy*'s default values is the choice of architecture for all three MLP baseline models. The default setting consists of a single layer of 100 neurons, which, through experimentation, was concluded to be insufficient for learning. Thus, we opt for a fixed architecture of four hidden layers of the specified number of neurons. For example, MLP(100) is comprised of an input layer, 4 hidden layers of 100 neurons each, and an output layer.

[3]The difference between macro and weighted averaging is explained in Section 2.4.3.

Model Type	Hyperparameter Name	Hyperparameter Value	Abbreviation
Decision Tree	-	-	DT
Random Forest	Number of Estimators	10	RF(10)
Random Forest	Number of Estimators	100	RF(100)
SVM	Kernel Function	Radial Basis Function	SVM(RBF)
SVM	Kernel Function	Polynomial	SVM(poly)
MLP	Number of Neurons	10	MLP(10)
MLP	Number of Neurons	100	MLP(100)
MLP	Number of Neurons	500	MLP(500)

Table 4.1: Collection of baseline models used for experimentation; Total of 8 models derived from 4 model types and through the alteration of one main hyperparameter.

The choices for the number of folds most commonly found in applications consist of 5-fold, and 10-fold cross validation [73]. The decision regarding the number of folds is made with the aim of ensuring that both the training and validation sets are drawn from the same distribution. Additionally, both sets must contain a sufficient number of samples in order to be representative of that distribution. As some of our experimental configurations include a very limited number of samples, a choice of 10 folds would likely result in validation sets inadequate size to be representative of the underlying distribution. Therefore, we always use 5-fold cross validation. Consequently, all of the metrics we report are an average over the 5 folds. It is also worth mentioning that all experiments employ *stratified* K-fold cross validation, meaning that each of the folds aims to preserve the percentage of samples from each of the classes. Finally, in the experiments conducted with the baseline models we perform 5 trials of the experiment (each of which includes 5-fold cross validation) and average the reported metrics over these 5 trials in order to give a more realistic estimate of performance.

4.2 Baseline Models on Time-Domain Inputs

The experimentation begins with the simplest possible approach, that is, using the filtered PPG signal as input. In this case, the sole variable to choose is the length of the signal, for which we base our decision on similar literature. More specifically, we take into account the claims given by *Malik* (1996) in a study on Heart Rate Variability (HRV) [74]. The study mentions:

The recording should last for at least 10 times the wavelength of the lower frequency bound of the investigated component, and, in order to ensure the stability of the signal, should not be substantially extended.

In our case, the lower bound of the frequency range we want to investigate is 0.15Hz, hence our filter choices in Section 3.2.1. This leads to a wavelength of $1/0.15 = 6.\overline{6}$ seconds, and a corresponding signal length of $66.\overline{6}$ seconds, equal to 10 wavelengths and just over a minute long. Therefore, in this case the PPG signals will simply be segmented into slices of this length. We also aim to explore fractions and multiples of this "optimal" signal length in order to view the effect on model performance. Consequently, this section will experiment with time-domain inputs of different sizes and report model performance for the specified baseline models.

4.3 Baseline Models on Frequency-Domain Inputs

For the next set of experiments, we investigate the potential merit of using frequency-domain representations as model input data. This is achieved by segmenting the PPG signals and applying the FFT on the smaller segments. If we use the suggested signal length of 10 wavelengths as mentioned above, we obtain slices of 6666 samples (since the sampling rate is 100Hz). This leads to a symmetric FFT of 6666 values, and a corresponding half-spectrum of 3334 values. As mentioned in Section 3.5, however, we are concerned only with the $[0 - 20]$Hz range, and so we slice the array to 2/5 of the original, resulting in an array of 1333 values.

We then wish to explore the effect of varying frequency resolution on model performance. Variable frequency resolution is achieved by defining the number of frequency bins in which the $[0-20]$Hz range is segmented. So, for example, if we dictate we want 4 bins, we average the 1333-value-spectrum every 333 values, obtaining 4 values for the $[0-5]$, $[5-10]$, $[10-15]$, and $[15-20]$Hz ranges respectively. The exploration is carried out by repeating the experiment for different values for the number of bins More specifically, the values explored are [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]. As the actual FFT is 1333 values, the highest we can go in powers of two is 1024 bins. In summary, for each of the baseline models we evaluate model performance using the FFT for 9 distinct values of frequency resolution.

4.4 CNN Models on Time-Frequency-Domain Inputs

In an attempt to potentially encapsulate more temporal information than is contained in the FFT, we experiment with a time-frequency representation in the form of the STFT. Since all of the baseline models accept 1D inputs, they are unable to truly exploit the two-dimensional nature of the STFT. CNNs, on the other hand, are engineered for manipulating 2D inputs, and so for this experiment we use only these convolutional models.

In the context of the STFT, the suggested signal length of 10 wavelengths refers to the size of the sliding window, and not to the size of the entire STFT. Therefore, since an STFT is obtained through applying the FFT on consecutive signal windows, we will need to use consecutive $\sim$66 second segments. Unfortunately, this poses some complications regarding the ICU-FU question due to the short duration of the follow-up recordings. The details of our workaround for this issue were deemed to be too low-level for report content and so has been placed in Appendix Section C.1. The result of the process was a calculation of a window size of $\sim$9.86 wavelengths, rather than the suggested 10, a deviation which we consider negligible.

However, even under these assumptions, and using the calculated window size and corresponding STFT size, we obtain a total of 79 STFTs as a training set for the ICU-FU question, a clearly unimpressive number. To mitigate for this lack of data, we explore dividing the calculated sizes for the window size by 2, 4, and 8 in order to generate more training samples. It should be noted that the severity classification task does not suffer from this problem, as there is a lot more data. Finally, it is worth mentioning that the *scipy* implementation of the STFT which we use already returns one-sided (*i.e.* half-spectrum) FFTs for real data. Therefore, with regards to preprocessing, we aim to truncate the frequency range from $[0-50]$Hz to $[0-20]$Hz and use the "optimal" frequency resolution we have found through our analysis in the previous section.

To summarise, the aim of this set of experiments is to investigate whether using 2D STFT inputs in combination with CNNs provides any advantage relative to the use of FFTs and baseline models. Our hypothesis is that CNNs using STFT inputs will lead to a small performance boost, which may in turn justify the increase in complexity in both the feature extraction and model development stages. Finally, the tradeoff between number of training samples and "suboptimal" (<10 wavelength) signal lengths will also be explored for the ICU-FU task.

Finally, we consider it very important to reiterate that the aim of this set of experiments is not solely to obtain the maximum possible performance across all of the model types tested. While tracking metrics for all experiments and making comparisons is clearly part of the experimentation process, the main goal of this investigation is to perform a relatively wide exploration of the search space of possible configurations, and subsequently note areas that seem to be correlated with improved performance. Moreover, it should be mentioned that the value of any given model cannot be judged solely by a collection of performance metrics, since other factors, such as computational overhead, time and memory efficiency, and interpretability are also decisive in the application of these models in the real world.

Chapter 5

Results and Discussion

This chapter contains the results for the experiments outlined in Chapter 4, and is therefore structured similarly. It is important to note that we run almost identical experiments for both classification tasks (*i.e.* ICU-FU and severities). Consequently, for each of the experiments presented in the next sections we include a subsection for both tasks, each of which contains the respective results for that section's experiment.

5.1 Baseline Models on Time-Domain Inputs

5.1.1 ICU-vs-Follow-up

We begin our analysis by presenting the performance metrics achieved by the different baseline models when using a time-domain input of duration equal to 10 wavelengths of the lowest frequency of interest, as suggested by *Malik* (1996) in [74]. Results throughout this section are presented in tabular form as shown in Table 5.1, which includes values for the set of chosen metrics for all 8 baseline models. Additionally, for each of the metrics we highlight the model configuration that achieved the highest and lowest scores, shown in green and red respectively. Finally, note that throughout this chapter the baseline models are referred to using their corresponding abbreviations as defined in Section 4.1.

	DT	**RF (10)**	**RF (100)**	**SVM (RBF)**	**SVM (poly)**	**MLP (10)**	**MLP (100)**	**MPL (500)**
Accuracy	0.614	0.654	0.730	0.675	0.594	0.593	0.620	0.628
Macro Precision	0.611	0.655	0.732	0.673	0.637	0.585	0.616	0.627
Macro Recall	0.609	0.636	0.721	0.663	0.553	0.589	0.612	0.624
Macro F1	0.608	0.632	0.722	0.664	0.495	0.582	0.612	0.622
Weighted F1	0.613	0.642	0.727	0.670	0.518	0.587	0.618	0.626

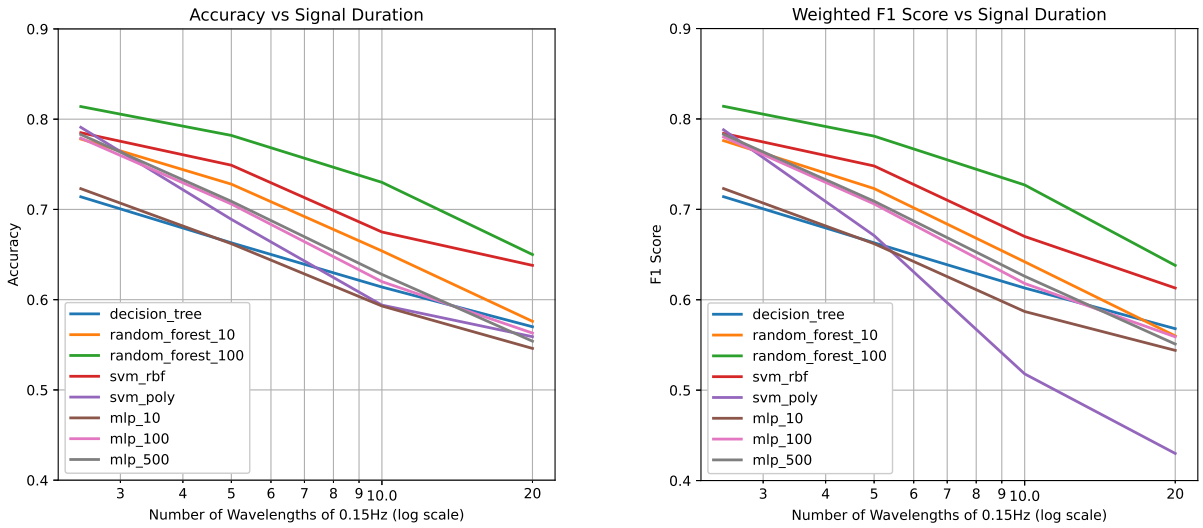
Table 5.1: Baseline model performance on time-domain inputs of duration equal to 10 wavelengths of the lowest frequency of interest (0.15Hz).

First and foremost, we observe that even when using the entire filtered PPG signal as input data, almost all models are able to achieve learning to some extent, meaning that they are able to achieve scores better than random guessing. The SVM with polynomial kernel and the smallest MLP achieve the lowest scores, while the random forest with 100 estimators achieves the highest score, with 73% accuracy and 72.7% weighted F1 score.

Moreover, we discover that hyperparameter alterations within the same model type result in significant discrepancies in performance. The RF(100) model shows an average improvement of approximately 8% over its smallest counterpart, the RF(10) model, whereas the SVM with RBF kernel achieves metrics 11% higher on average over the SVM with polynomial kernel. The differences between MLP models are not as pronounced,

which leads us to believe that, when using time-domain inputs, further tuning of the number of neurons per layer given a fixed number of layers is unlikely to yield significantly higher performance.

Next, we experiment with PPG signals of length equal to a multiple or a fraction of the "optimal" 10 wavelength duration. More specifically, we use input signals of duration equal to 20 wavelengths (twice as long), 5 wavelengths (half the length), and 2.5 wavelengths (quarter of the length). For reasons of conciseness and clarity we do not show all of the corresponding tables for these 3 configurations here, but rather place them in the Appendix Section B. Instead, we use the accuracy and weighted F1 score metrics in order to enable comparisons between the different durations. The results are shown in Figure 5.1, which plots the accuracy and weighted F1 score for all the baseline models as a function of signal duration.



(a) (b)

Figure 5.1: (a) Accuracy and (b) weighted F1 score for baseline models as a function of signal duration. Signal duration quantified through the number of wavelengths of 0.15Hz frequency (plotted in log scale).

First, we discuss the effect of signal duration on model performance. Contrary to our expectations, we see that performance improves as the duration of the signal is reduced. Equivalently, we discover no sense of optimality in the 10-wavelength duration, and do not observe that using relatively shorter signals as input leads to a degradation in performance. A possible explanation of this is that during feature extraction (*i.e.* simple signal segmentation for this section), dictating a smaller duration naturally leads to a larger training set. For example, when using signals of duration 5 wavelengths rather than 10 wavelengths, the feature extraction stage naturally produces double the training samples. Table 5.2 below depicts the input signal length and the corresponding number of training samples for each of the different tested durations, where the duration is again given as the number of wavelengths of the lowest frequency of interest.

	20	**10**	**5**	**2.5**
Input Signal Length	13333	6666	3333	1666
# Training Samples	338	694	1405	2829

Table 5.2: Input signal length and number of training samples for the different durations tested (duration given as the number of wavelengths of the lowest frequency of interest (0.15Hz)).

Therefore, we can hypothesise that the increase in performance observed in the shorter durations may be caused by the larger corresponding dataset. In other words, when comparing 5 wavelengths with 10 wavelengths, we gain more due to the existence of more data than we lose due to "suboptimal" ($<$ 10-wavelength-long) signal length. We investigate this further by re-running the experiment for signals of duration equal to 5 wavelengths,

but with the same number of training samples as the 10-wavelength case. Thus, following Table 5.2 we truncate the dataset at 694 samples, before which we of course shuffle in order to preserve the existing label balance and to include data from all patients.

	DT	RF (10)	RF (100)	SVM (RBF)	SVM (poly)	MLP (10)	MLP (100)	MPL (500)
Accuracy	0.633	0.677	0.753	0.719	0.650	0.642	0.666	0.665
Macro Precision	0.630	0.681	0.754	0.720	0.677	0.638	0.664	0.662
Macro Recall	0.630	0.664	0.746	0.707	0.618	0.636	0.661	0.661
Macro F1	0.629	0.662	0.747	0.708	0.601	0.635	0.661	0.660
Weighted F1	0.633	0.669	0.751	0.715	0.617	0.640	0.665	0.664

Table 5.3: Baseline model performance on time-domain inputs of duration equal to 5 wavelengths of the lowest frequency of interest (0.15Hz) and with a truncated dataset in order to contain the same number of training samples as the 10-wavelength case.

Surprisingly, we observe that the reduced 5-wavelength dataset still performs better than the 10-wavelength case even when both contain the same number of training samples. We identify two potential conclusions that can be drawn from this result:

1. Signal durations shorter than 10 wavelengths are simply beneficial to the baseline models tested within this experimental framework. Consequently, the claims mentioned in [74] do not apply for PPG signals in the context of classifying between healthy and unhealthy dengue patients. In this case, it is reasonable to expect that we can further increase performance by using even shorter signal durations. That said, performing an in depth investigation of this hypothesis is beyond the aims of this set of experiments.

2. The claims in [74] are in fact correct, but we are considering a potentially inaccurate value for the lowest frequency of interest. More specifically, we assume that there is useful information in frequencies down to 0.15Hz, and calculate the durations according to wavelengths of this frequency. However, if useful information were to hypothetically exist only down to 0.3Hz, then the durations would be calculated in a different way (*e.g.* 5 wavelengths of 0.15Hz is 10 wavelengths of 0.3Hz).

We argue that, without further investigation on which frequencies are comparatively more useful to the models, it is unreasonable and futile to settle on one of the two points.

Having discussed the effect of signal duration, we now comment on the performance difference between the baseline models. For all of the experiments and regardless of input signal duration, the random forest model with 100 estimators consistently outperforms all others, achieving a peak accuracy and weighted F1 score of 81.4% in the 2.5-wavelength configuration. Regarding the weakest models, for larger durations (10 and 20 wavelengths), the SVM(poly) and MLP(10) models achieve the lowest scores, whereas for lower durations (5 and 2.5 wavelengths), weakest performance is seen in the DT and MLP(10) models.

5.1.2 Severity Classification

We now conduct a similar set of experiments using time-domain inputs for the severity classification question. It is important to note in advance that the dataset for the severity classification task is many times larger than the one for the ICU-FU task. More specifically, for a signal duration of 10 wavelengths the ICU-FU dataset contained 694 samples, while the severities dataset contains 73380 samples, and is thus more than 100 times larger. We are interested in investigating whether the difference in the size of the training set alters the observations made in the previous section. It should be noted that due to the much larger dataset and consequently significantly slower training times, all metrics were collected using one trial of 5-fold cross validation, rather than the average of 5 trials as in the previous section.

Unfortunately, the increased dataset size also introduced complications regarding the use of SVMs. This is explained through the online documentation of the *scipy* library: "The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples" [75]. Thus, given 73380 training samples for the initial 10-wavelength experiment, the time complexity of SVM model training proves problematic. In practice, we stopped execution after a large number of hours, at which point no sign of progress was visible. Our approach concerning this issue consists of truncating the dataset to 10000 samples. Although this is likely to affect the reported performance to some extent, we consider that there is still merit in including the SVM(RBF) and SVM(poly) models. The resulting performance metrics for a signal duration of 10 wavelengths are shown in Table 5.4.

	DT	**RF (10)**	**RF (100)**	**SVM (RBF)**	**SVM (poly)**	**MLP (10)**	**MLP (100)**	**MPL (500)**
Accuracy	0.431	0.486	0.548	0.498	0.440	0.478	0.442	0.461
Macro Precision	0.421	0.479	0.586	0.534	0.501	0.479	0.430	0.449
Macro Recall	0.421	0.454	0.503	0.434	0.359	0.430	0.428	0.444
Macro F1	0.421	0.455	0.504	0.404	0.269	0.419	0.428	0.445
Weighted F1	0.431	0.472	0.523	0.430	0.435	0.444	0.440	0.457

Table 5.4: Baseline model performance on time-domain inputs of duration equal to 10 wavelengths of the lowest frequency of interest (0.15Hz).

Once again, we see that all models are able to learn to some extent, although the scores are lower than the ICU-FU case. While this observation could be considered natural due to the increased number of classes, it also implies that the existence of a large training set does not guarantee the ability of baseline models to effectively separate between the classes when using time-domain inputs. That said, the effect of the larger dataset will be re-examined in the next section with the frequency-domain representation.

Given the results of the previous section, which suggested that 10 wavelengths of a frequency of 0.15Hz may not represent an optimal signal duration, we are interested in investigating whether this effect persists in the case of a different classification task and with a larger dataset. Due to the time limitations imposed by the increased testing duration for this task, we only recreate the truncated 5-wavelength experiment to test our hypothesis.

	DT	**RF (10)**	**RF (100)**	**SVM (RBF)**	**SVM (poly)**	**MLP (10)**	**MLP (100)**	**MPL (500)**
Accuracy	0.434	0.494	0.553	0.502	0.468	0.494	0.461	0.476
Macro Precision	0.426	0.490	0.585	0.548	0.501	0.515	0.451	0.467
Macro Recall	0.426	0.466	0.512	0.445	0.386	0.443	0.447	0.461
Macro F1	0.426	0.468	0.515	0.424	0.335	0.434	0.448	0.463
Weighted F1	0.435	0.483	0.532	0.452	0.378	0.458	0.459	0.473

Table 5.5: Baseline model performance on time-domain inputs of duration equal to 5 wavelengths of the lowest frequency of interest (0.15Hz) and with a truncated dataset in order to contain the same number of training samples as the 10-wavelength case.

As shown in Table 5.5, the observations remain identical to the previous section, with almost all models seeing a boost in performance. The RF(100) model achieves the highest scores, while the SVM(poly) and DT models achieve the lowest scores. Therefore, we are able to conclude that the observations of the previous section were not driven by the limitations of the dataset size, but are rather caused by inherent characteristics of the data.

5.2 Baseline Models on Frequency-Domain Inputs

The experiments of this section explore the use of frequency-domain inputs for the baseline models, and compare the results with those achieved by the time-domain representation. In order to generate the frequency-domain dataset we use 10-wavelength-long signals. Primarily, this is because we wish to be able to compare the two representations on the same signal segments. Additionally, as aforementioned, without further experimentation we are unable to know for sure which of the two conclusions drawn in Section 5.1.1 is more accurate (*i.e.* whether the 10-wavelength claim in [74] is actually applicable in the context of our investigation). Therefore, we use 10-wavelength-long segments assuming a lower frequency bound of 0.15Hz throughout this section.

Given the chosen signal duration, for each segment we obtain the one-sided FFT, and subsequently truncate the frequency range to $[0 - 20]$Hz. Moreover, as mentioned in Section 4.3, we are also interested in exploring the effect of varying frequency resolution. More specifically, for each of the baseline models we evaluate model performance using the FFT for 9 distinct values of frequency resolution, which we quantify through the number of frequency bins into which the $[0 - 20]$Hz range is segmented.

5.2.1 ICU-vs-Follow-up

We begin our analysis with investigation of the frequency representation for the ICU-FU question. Table 5.6 depicts an example of the format of the obtained results. The metrics shown in this case are the results for the RF(100) model, which, once again, yielded the highest performance.

	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
Accuracy	0.692	0.697	0.849	0.863	0.875	0.882	0.882	0.858	0.835
Macro Precision	0.689	0.696	0.848	0.862	0.875	0.881	0.882	0.859	0.837
Macro Recall	0.685	0.692	0.847	0.861	0.874	0.881	0.880	0.856	0.831
Macro F1	0.685	0.692	0.847	0.861	0.874	0.880	0.880	0.856	0.832
Weighted F1	0.690	0.696	0.849	0.862	0.875	0.882	0.882	0.858	0.835

Table 5.6: Performance metrics for the RF(100) model on frequency-domain inputs of varying frequency resolution. Frequency resolution quantified through the logarithm of the number of frequency bins.

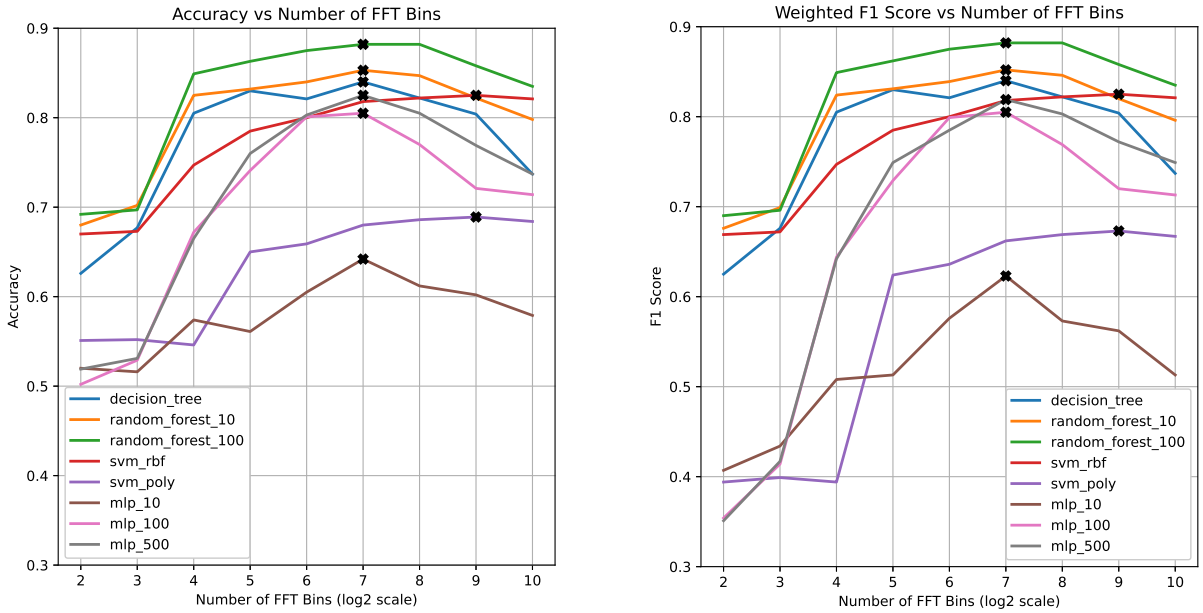
As before, we do not present the results for the remaining 7 models in tabular form, and rather rely on the accuracy and weighted F1 scores to enable comparisons. The results of this process are depicted in Figure 5.2, which plots the accuracy and weighted F1 score for all the baseline models as a function of frequency resolution.

First and foremost, it is evident that the frequency-domain representation is able to achieve better performance than the time-domain representation. All models, with the exception of SVM(poly) and MLP(10), were able to surpass the maximum performance achieved in the entirety of the previous section. It is worth mentioning here that the frequency representation is able to outperform the time representation while at the same time being much more memory efficient. In the case of 10-wavelength segments, the time-domain representation contains 6666 values, while the frequency-domain representation uses only as many values as the number of frequency bins. This is especially important for embedded computing scenarios, which is likely to be the case for a device used for mass-scale screening within a clinical setting.

Secondly, we observe an interesting result regarding performance as a function of frequency resolution. More specifically, we discover a concept of peak resolution, that is, higher resolution does not imply improved performance. We investigate this further by placing a marker at the frequency resolution at which maximum performance is achieved by each model. Our observations are as follows:

- Frequency resolution lower than 8 bins leads to large performance drops, with SVM(poly) and all three MLP models achieving very low (essentially random) accuracy scores. During testing these models often predicted only one class with these frequency resolutions, leading to undefined precision scores, which are automatically set to zero.

- As frequency resolution is increased, all models witness a boost in performance. In the case of the SVM(RBF) and SVM(poly) models, both accuracy and weighted F1 metrics increase and reach a plateau as resolution approaches 1024 bins. In the case of all other baseline models, both metrics reach a maximum value at 128 bins, indicating a type of convexity.



(a) (b)

Figure 5.2: (a) Accuracy and (b) weighted F1 score for baseline models as a function of frequency resolution. Frequency resolution quantified through the logarithm of the number of frequency bins.

5.2.2 Severity Classification

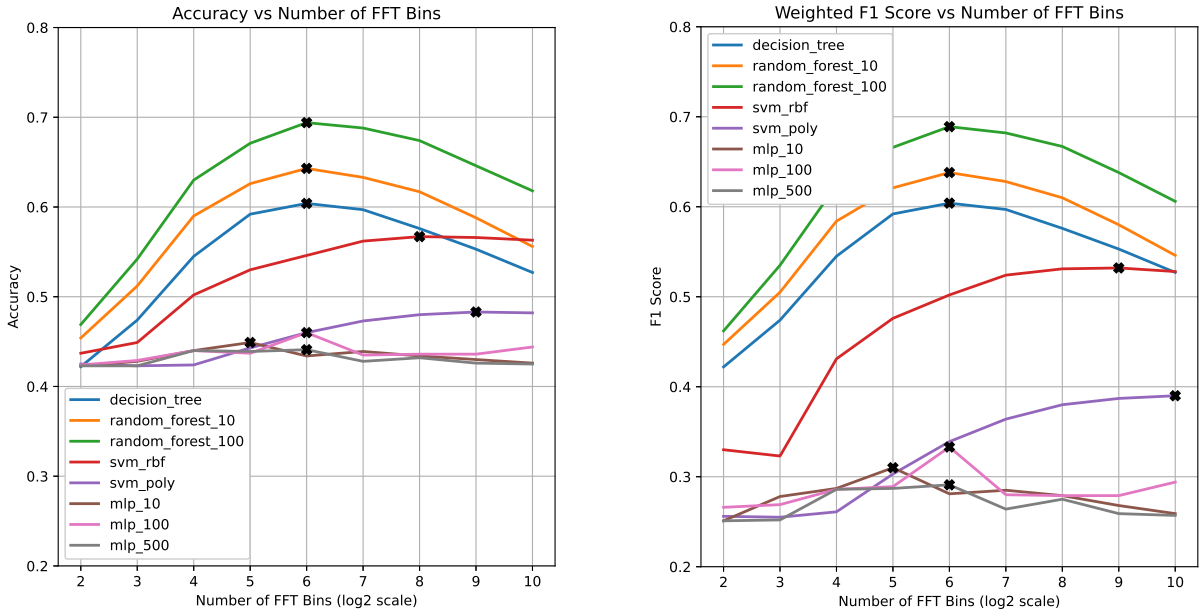
In this section we repeat the experiments using frequency-domain inputs for the severities classification task. Again, we only show the full set of metrics for the best-performing model in Table 5.7, and place the full collection of results in the Appendix. Subsequently, we perform the same investigation as before by plotting the accuracy and weighted F1 metrics as a function of frequency resolution in Figure 5.3.

	4	8	16	32	64	128	256	512	1024
Accuracy	0.469	0.542	0.630	0.671	0.694	0.688	0.674	0.646	0.618
Macro Precision	0.454	0.541	0.636	0.685	0.709	0.704	0.692	0.667	0.641
Macro Recall	0.446	0.519	0.606	0.646	0.669	0.662	0.646	0.616	0.584
Macro F1 Score	0.446	0.524	0.613	0.656	0.680	0.672	0.656	0.625	0.592
Weighted F1 Score	0.462	0.535	0.624	0.666	0.689	0.682	0.667	0.638	0.606

Table 5.7: Performance metrics for the RF(100) model on frequency-domain inputs of varying frequency resolution. Frequency resolution quantified through the logarithm of the number of frequency bins.

Through the results depicted in both Table 5.7 and Figure 5.3 we can deduce that the application of frequency-domain inputs for the severities task leads to similar observations to those made for the ICU-FU task. Consequently, this allows us to validate the conclusions drawn in the previous section.

Once again, the use of the frequency-domain representation as model data is able to outperform the time-domain representation by a significant margin. More specifically, the RF(100) model using frequency-domain inputs achieves accuracy and weighted F1 scores approximately 15% higher than those achieved by the same model using time-domain inputs on the reduced 5-wavelength dataset. Since the maximum metrics for this model occur at a frequency resolution of 64 bins, the large improvement in performance is accompanied by a 50-fold reduction in memory usage, given that the time-domain representation requires 3333 values for the 5-wavelength duration. Thus, the frequency representation is, once again, shown to be superior.



(a) (b)

Figure 5.3: (a) Accuracy and (b) weighted F1 score for baseline models as a function of frequency resolution. Frequency resolution quantified through the logarithm of the number of frequency bins.

Furthermore, we observe that the decision tree and random forest models again reach clear maxima for a given value of number of bins, exhibiting even more pronounced convexity. The SVM models achieve lower performance, and behave very similarly to the ICU-FU task, where both metrics increase and subsequently plateau as resolution increases. The MLP models, however, are consistently observed to perform poorly independently of frequency resolution. Upon further investigation, we discover that performance on the training set for the MLP models is equally low, and that therefore the three MLP models are underfitting. Consequently, we can conclude that despite the large number of training samples for the severity classification question, the models are not able to encapsulate the complexity of the task.

5.3 CNN Models on Time-Frequency-Domain Inputs

In this section we examine whether the combination of STFTs and CNNs is able to improve upon the results achieved by the baseline models when utilising a frequency-domain representation. Regarding the input data to the CNN models, the STFTs are preprocessed by truncating the frequency range of the contained FFTs to $[0 - 20]$Hz and averaging to the highest-performing frequency resolution found in the frequency representation section. Although it is not guaranteed that the optimal resolution found for the baseline models is also the optimal resolution for the CNNs, we consider it a solid starting point. Consequently, all input STFTs used to train the CNNs have a frequency resolution of 128 bins.

With regards to choosing the CNN model architecture we use the optimisation tool provided by the *comet.ml* platform. We begin by defining a search space within which we want to find the best model. For our use case we

wish for the search space to be relatively wide, so as to be able to encapsulate different architectures and sizes for the network. This is achieved by defining the search space through the parameters and the corresponding ranges depicted in Table 5.8.

Parameter Name	**Range of Values**	**Range Type**
Number of Convolutional Layers	[2, 3, 4, 5, 6]	Discrete
Kernel Size	[(3, 3), (4, 4), (5, 5)]	Discrete
Number of Dense Layers	[1, 2, 3]	Discrete
Number of Nodes per Dense Layer	[4, 8, 16, 32]	Discrete
Learning Rate	[5e-5, 5e-3]	Continuous

Table 5.8: Model search space, as defined by a number of parameters and a corresponding range of values for each, to be explored through the Bayesian optimisation scheme of the *comet.ml* platform.

Our choice for the number of filters of the convolutional layers is inspired by several well-established CNN architectures, such as VGG16, which increase the number of filters as the network deepens [63]. Therefore, our approach is to double the filters of each consecutive layer, while maintaining the same number for the last two layers. For example, if we choose 8 filters for the 1st layer, a 4-layer network would have 16 filters in the 2nd layer, and 32 filters in the 3rd and 4th layers. A max pooling layer is also placed after each convolutional layer to assist with feature extraction within the CNN, as well as to reduce the computational overhead. Finally, the bayesian optimisation scheme used for model selection requires information about the metric we wish to minimise or maximise as well as the number of iterations for which the algorithm will run. In our case, we chose 20 iterations for the maximisation of the average weighted F1 score over 5 folds of cross validation.

5.3.1 ICU-vs-Follow-up

In the case of the ICU-FU task, performing the STFT on signal segments of 10-wavelength duration leads to a very limiting training set consisting of 79 STFTs. Therefore, we experiment with manually dividing the length of the STFT in order to extend the size of the dataset. The set of tested configurations consists of the full STFT, and subsequent divisions of the window size of the STFT by 2, 4, and 8. The corresponding number of training samples as well as the STFT shape for each of these configurations are shown in Table 5.9. Since the STFT is comprised of a fixed number of windows, manually dividing the window size by two effectively doubles the number of training samples. The shape of the STFT itself changes since the frequency resolution is proportional to the length of the input signal[1].

	1	**1/2**	**1/4**	**1/8**
# Training Samples	79	168	349	707
STFT Shape	(3286, 9)	(1643, 9)	(822, 9)	(411, 9)

Table 5.9: Training set size and STFT shape for different sizes of the STFT input. Size represented by the number by which the full STFT length, as computed from a 10-wavelength long signal, is multiplied (*e.g.* 1/4 represents a quarter of the full STFT length).

The optimisation scheme is then run for each of the configurations of Table 5.9, producing the results shown below. Table 5.10 presents the performance metrics achieved by the best model found by the optimiser for each of the configurations, while Table 5.11 notes the set of parameters defining each of these models. We also note that the optimiser was run two times, using both 4 and 8 as the number of filters for the first convolutional

[1]As explained in Appendix Section C for the ICU-FU question we dictate that we want 8 STFT windows. The reason the STFT shapes shown here contain 9 windows is because the implementation of the *scipy* library uses an additional window when zero-padding the signal. In turn, the first and last windows are of half the specified window length, leading to an overall 8-window duration.

layer. Table 5.11 presents the highest performing model across both of these configurations and notes the corresponding value.

	1	**1/2**	**1/4**	**1/8**
Accuracy	0.760	0.791	0.827	0.892
Macro Precision	0.776	0.798	0.833	0.891
Macro Recall	0.744	0.783	0.823	0.891
Macro F1	0.744	0.785	0.824	0.891
Weighted F1	0.751	0.789	0.826	0.892

Table 5.10: Performance metrics for the best CNN mode found by the optimiser for different sizes of the STFT. Size represented by the number by which the STFT window length is multiplied (*e.g.* 1/4 represents a quarter of the normal, 10-wavelength-long, window length).

	1	**1/2**	**1/4**	**1/8**
Number of Convolutional Layers	2	5	6	5
Kernel Size	(2, 2)	(3, 3)	(3, 3)	(3, 3)
Number of Dense Layers	2	2	1	1
Number of Nodes per Dense Layer	16	4	32	32
Learning Rate	$2e-3$	$8e-4$	$2e-3$	$8e-4$
Number of Filters of 1st Conv. Layer	4	8	8	8

Table 5.11: Model parameters for each of the models shown in Table 5.10.

To begin, we observe that using the full STFT as input to the CNN models leads to performance significantly lower than what can be achieved with the frequency representation. This can easily be attributed to the fact that the training set is very limited, making learning a highly challenging task for a complex model such as the CNN. As we manually reduce the window size by consecutive powers of two, and thereby inversely increase the size of the dataset, we observe a consistent increase in performance. More specifically, we note an average increase of 10% across the reported metrics between the full STFT and the 1/8 configuration. Ultimately, this configuration is able to achieve performance marginally higher than the highest performing baseline model using the frequency-domain representation.

It is interesting to note that since the full STFT length consists of 8 windows of ∼10-wavelength duration, when manually divided by 8, the STFT length is equivalent to that of a single segment. Consequently, this configuration produces an STFT for the same exact segments for which an FFT was computed in the previous section. Since similar performance was achieved in both these scenarios, we can speculate that given a particular segment, using the STFT as input to a CNN is equally effective in utilising the information within that segment as computing the FFT and using an RF(100) model with the appropriate number of frequency bins. Taking into consideration the increase in both design complexity as well as computational overhead of the STFT-CNN setup relative to the FFT-RF arrangement, we can argue that in the context of this clinical question and given the limitations of the dataset size, the implementation and use of STFTs and CNNs is not worthwhile.

Finally, we consider it useful for future work to mention that the *comet.ml* framework is capable of much more than simply returning the best model found. This stems from the fact that the platform tracks all experiments which are run with the optimiser, and is therefore able to provide further insights regarding the model search space. One example which we found particularly useful is the parallel coordinates chart, which provides a comprehensive summary of the effect of different hyperparameters on model performance. An example of this is shown in Figure 5.4, where the hyperparameters shown are the number of convolutional and dense layers,

the kernel size, and the number of nodes per dense layer, while the metric is the average cross validation weighted F1 score. This representation is especially useful in the case where an iterative hyperparameter search is performed. The aim of this process, which was not undertaken within this project due to time limitations, is to define consecutive search spaces in order to achieve convergence to a near-optimal configuration.

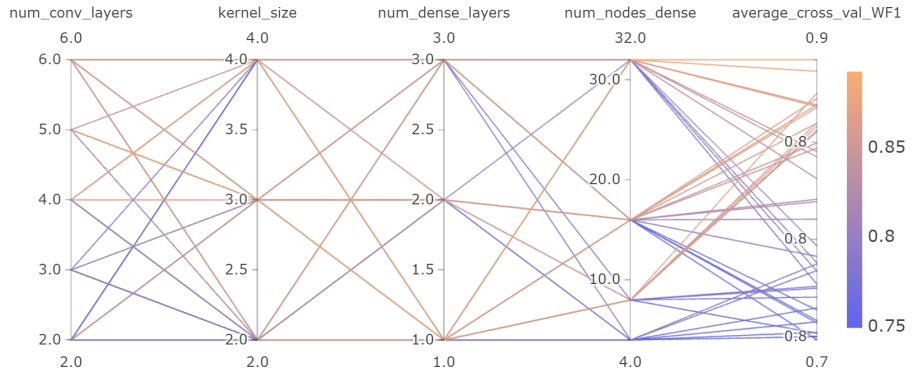


Figure 5.4: Example of parallel coordinates chart produced by *comet.ml* across the experiments completed by the optimiser. This depicts the effect of different values of the hyperparameters defining the search space on a metric of model performance.

5.3.2 Severity Classification

The same experimental process is now applied to the severity classification task. The main difference between the tasks is, once again, the size of the dataset. More specifically, when using the full STFT, as defined by 8 windows of 10-wavelength duration, the dataset in the ICU-FU case consists of 79 samples whereas the severities dataset includes 9094 samples. This is of particular interest in this section as CNN models are known to achieve high performance when presented with larger datasets. Thus, we begin this task using solely the full STFT without performing any manual division of the window length as in the previous section.

The experimental setup consists of running the same optimisation scheme for the full STFT configuration, experimenting with 4, 8, and 16 as the number of filters of the first convolutional layer. The columns labelled "1" of Tables 5.12 and 5.13 depict the metrics achieved by the best performing model as well as the optimal model parameters found by the algorithm over the total 60 iterations.

	1	**1/8**	**1 (32 windows)**
Accuracy	0.689	0.780	0.615
Macro Precision	0.709	0.785	0.629
Macro Recall	0.665	0.771	0.591
Macro F1	0.676	0.776	0.599
Weighted F1	0.684	0.780	0.609
# Training Samples	9094	75443	2266

Table 5.12: Performance metrics for the best CNN mode found by the optimiser for different sizes of the STFT. Size represented by the number by which the STFT window length is multiplied (*e.g.* 1/4 represents a quarter of the normal, 10-wavelength-long, window length).

	1	**1/8**	**1 (32 windows)**
Number of Convolutional Layers	5	5	5
Kernel Size	(4, 4)	(4, 4)	(4, 4)
Number of Dense Layers	3	1	2
Number of Nodes per Dense Layer	32	4	4
Learning Rate	$5e-3$	$6e-4$	$7e-4$
Number of Filters of 1st Conv. Layer	4	8	4

Table 5.13: Model parameters for each of the models shown in Table 5.12.

Perhaps surprisingly, we are able to make similar observations with regards to model performance to those in the ICU-FU task. Namely, we observe that the best CNN model found by the optimiser again matches, but is unable to notably surpass, the performance of the highest performing model in the frequency-domain case, the RF(100) model. We identify two potential explanations for this behaviour:

1. There is a lack of temporal variation in the PPG signal within the span of the STFT, that it, within 8 windows of ∼66 seconds. Equivalently, this STFT duration is insufficient for providing the model with useful temporal features beyond those found in the FFT. Thus, since the advantage of STFTs is due to the ability of encapsulating more temporal variation, the lack thereof within the PPG signal leads to similar performance between the STFT and FFT approaches.

2. Despite the increased size of the dataset for this task, the number of training samples is still insufficient for the CNN models to exhibit their true capabilities.

Given the above, we set up two additional experiments in order to further investigate both points.
Regarding the first explanation, our approach consists of using the same window length as the full STFT (*i.e.* 10 wavelengths), but constructing the STFT through a larger number of windows. More specifically, the STFT will now consist of 32 10-wavelength-long windows, lasting for ∼35 minutes, rather than the previously considered ∼9 minute duration. This experiment will allow us to investigate whether applying the STFT on longer signals is able to encapsulate temporal variation within the PPG signals which is useful to the models.

Regarding the second point, we repeat the approach of the previous section by manually dividing the STFT window length by 8. Consequently, the length of the signals is equivalent to that in the FFT case and the dataset is now ∼8 times larger. The results for these two experiments are depicted in Tables 5.12 and 5.13 under the column names "1 (32 windows)" and "1/8" respectively.

We observe that using longer durations, such as the ∼35-minute-long segments explored in this case, does not provide any advantage in reported metrics, but rather results in a degradation in performance. We argue that the reduction in dataset size brought forth by the use of longer segments incurs a cost which greatly outweighs the potential benefits of encapsulating more temporal variation. On the other hand, the large increase in training samples caused by manual division of the window length leads to a significant boost in performance. More specifically, we note an average 9% increase across the set of reported metrics compared to the maximum performance achieved using the frequency representation. Seeing as this is the only occurrence of a significant improvement through the use of STFTs as CNN inputs, we speculate that it is only when provided with very extensive datasets that CNNs are truly able to exploit the feature space provided by the STFT.

Chapter 6

Project Evaluation, Conclusions, and Future Work

6.1 Project Evaluation

In this section we discuss the extent to which the project objectives have been fulfilled. The section is divided into two parts which evaluate the pipeline implementation and the experimentation respectively.

6.1.1 Evaluation of Pipeline Implementation

The processing and analysis pipeline was implemented successfully, fulfilling the evaluation criteria presented in the technical objectives of Section 1.2.2. The pipeline is able to successfully parse the dataset, selecting the patients who can be used given a particular clinical question, preprocessing their PPG signals and performing quality analysis, and creating a training set for the machine learning models by means of feature extraction techniques. The preprocessing and quality analysis stages have also been shown to be able to effectively remove unwanted signal components, and subsequently reject the parts of the signals which were of poor quality.

Regarding the pipeline as a whole, the large degree of configurability in the design combined with the use of checkpoints encouraged efficient experimentation, and was decisive in allowing us to complete the wide range of experiments presented in this work. Furthermore, we believe that the proposed system design, as shown in Section 3.1.1 and explained throughout Chapter 3, enables easy exploration of the various aspects of the pipeline in the case of future work. This consists of, but is not limited to, investigation of the following:

- Preprocessing strategies, consisting of filtering, normalising, and smoothening techniques.
- Quality analysis approaches through the implementation of additional SQIs and exploration of their allowed ranges.
- Feature extraction algorithms.
- Machine learning model types and testing of different configurations.

The codebase as well as more implementation details regarding the pipeline can be found in the GitHub repository.

6.1.2 Evaluation of Experimentation

The primary objective of this project involved sufficient investigation of two severity-related clinical questions for dengue patients. More specifically, this work aimed to perform a set of exploratory experiments through the use of the processing and analysis pipeline as applied onto a novel PPG signal dataset. We believe that the experimental process laid out in Chapter 4 and subsequently extended and discussed in Chapter 5 is of both sufficiently high quantity and quality.

We have implemented three central data representations, consisting of time-domain, frequency-domain, and time-frequency domain approaches. Subsequently, these were used as inputs for a collection of machine learning models. Moreover, we implemented a set of baseline models consisting of a variety of different schemes in order to allow for wider exploration. Using the different feature extraction and model configurations, we have been methodical in running, documenting, and discussing a collection of interesting experiments which explore various aspects of the problem. We have also made comparisons between the different configurations, noting the parts of our experimentation which were correlated with higher performance. Furthermore, this has allowed us to provide empirical suggestions for future work, as will be expanded upon in the next section. Ultimately, we understand that our project is not meant to be able to single-handedly solve the underlying problems, but rather to serve as a step forward towards a product capable of real-world positive impact. Consequently, given that we have presented a wide range of experiments in the novel field of PPG applications for dengue, we have evaluated the experimentation process of this project to be successful.

6.2 Conclusions and Future Work

In this section we summarise the most important aspects of the work carried out in this project and provide suggestions for future work for each of these aspects.

We begin by discussing data representations. Throughout our experiments, the frequency-domain representation is shown to be superior to the time-domain representation in both investigated questions. In terms of performance, using time-domain inputs achieved a maximum weighted F1 score of 81.4% for the ICU-FU question and 53.2% for the severity classification task, whereas the respective metrics using frequency-domain inputs were 88.2% and 68.9%. Averaging over all five reported metrics, the frequency representation led to an improvement of +8% for the ICU-FU task and +15% for the severities task. The frequency representation was also found to be preferable with regards to memory efficiency, allowing for 10- to 100-fold less memory usage for both tasks. The time-frequency representation was in almost all cases found to yield comparable performance to the frequency approach. This, however, can be attributed to the limited data size, since the CNN models that were used require an extensive dataset. In the configuration where this was provided, we were able to observe a +9% boost in metrics over the frequency representation in the case of the severities task.

We now summarise our findings with regards to model exploration. Throughout our experiments, we find unanimously that alterations in hyperparameters of the baseline models are strongly correlated to the final reported metrics. Therefore, we can conclude that all reported results are an underestimation of the true capabilities of the underlying models, and that, consequently, extensive hyperparameter tuning is very likely to yield improved model performance. For this reason, this is recommended as future work. Clearly, we believe that the order in which the models should be tuned is based on their relative performance as shown in this work. For example, given that throughout our experimentation the random forest classifiers consistently outperformed the remaining baseline models, we suggest that hyperparameter tuning is performed on this model type first. It is worth noting that we highly recommend the utilisation of the *comet.ml* platform for this type of optimisation, since our experience suggests that there is merit in its use as a tuning tool.

Regarding the use of convolutional models, we observe that in most experiments using STFTs as input data to CNNs does not provide an advantage over the use of baseline models with frequency-domain inputs. The use of convolutional models was only observed to provide a clear improvement when using the largest dataset size tested within this work. Thus, we can speculate that the use of CNNs is motivated only when the dataset is extensive, including several tens of thousands of samples. Equivalently, we argue that convolutional models require large amount of data in order to be able to exploit the richer feature space provided by the STFT.

We also comment on the interpretability of our results. Unfortunately, most of the models tested within this work are used as black boxes which are unable to provide any sort of explainable or even interpretable results without the implementation of additional techniques. We recognise this as a general drawback of the current state of the machine learning domain which is of particular importance for clinical applications. We strongly believe that future work should be at least partly focused on taking a step forward towards a more explainable, or at least interpretable, system. The interepretability techniques attempted but not completed within this project due to timing limitations consist of LIME classification as well as saliency maps for convolutional

models [76, 77]. As an additional note, we believe that the successful implementation of clinical features could also serve as an important step towards increased interpretability and explainability.

Finally, we comment on the way in which we report results. Across all of our experiments we have compared performance mainly through the use of the accuracy and weighted F1 metrics. However, given that the ultimate aim of this project is in the form of a clinical assistant able to flag unhealthy or severe subjects, we speculate that it is perhaps of greater importance to maximise recall for this application. Equivalently, the system must attempt to minimise false negatives, since the cost of not flagging an unhealthy patient is significantly lower than that of flagging a healthy patient. Lastly, we encourage future work to utilise confidence intervals instead of point estimates for reporting performance, since this would allow for more assurance in the fact that the metrics reported are a more realistic estimate of real-world performance.

Appendix A

Abbreviations

Concept	Abbreviation
DSS	Dengue Shock Syndrome
PPG	Photoplethysmography
SC	SmartCare monitoring device
GE	General Electric monitoring device
FE	Feature Extraction
FFT	Fast Fourier Transform
STFT	Short Time Fourier Transform
CWT	Continuous Wavelet Transform
DT	Decision Tree
RF	Ranfom Forest
SVM	Support Vector Machine
RVM	Relevance Vector Machine
XGBoost	Extreme Gradient Boosting
MLR	Multiple Linear Regression
MnLR	Multinomial Logistic Regression
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
AUC-ROC	Area Under the Receiver Operator Characteristic Curve

Appendix B

Full Results

B.1 Baseline Model Performance using Variable Duration Time-Domain Inputs for the ICU-FU task

	DT	RF (10)	RF (100)	SVM (RBF)	SVM (poly)	MLP (10)	MLP (100)	MPL (500)
Accuracy	0.570	0.576	0.650	0.638	0.559	0.546	0.563	0.554
Macro Precision	0.569	0.569	0.651	0.653	0.541	0.545	0.557	0.554
Macro Recall	0.567	0.557	0.632	0.612	0.510	0.544	0.555	0.553
Macro F1	0.564	0.548	0.628	0.600	0.396	0.540	0.553	0.547
Weighted F1	0.568	0.560	0.638	0.613	0.430	0.544	0.559	0.551

Table B.1: Baseline model performance on time-domain inputs of duration equal to 20 wavelengths of the lowest frequency of interest (0.15Hz).

	DT	RF (10)	RF (100)	SVM (RBF)	SVM (poly)	MLP (10)	MLP (100)	MPL (500)
Accuracy	0.663	0.728	0.782	0.749	0.689	0.662	0.706	0.709
Macro Precision	0.660	0.732	0.781	0.747	0.712	0.659	0.704	0.708
Macro Recall	0.659	0.716	0.777	0.744	0.666	0.658	0.702	0.706
Macro F1	0.659	0.717	0.778	0.745	0.660	0.658	0.702	0.705
Weighted F1	0.663	0.723	0.781	0.748	0.671	0.662	0.706	0.709

Table B.2: Baseline model performance on time-domain inputs of duration equal to 5 wavelengths of the lowest frequency of interest (0.15Hz).

	DT	RF (10)	RF (100)	SVM (RBF)	SVM (poly)	MLP (10)	MLP (100)	MPL (500)
Accuracy	0.714	0.778	0.814	0.785	0.791	0.723	0.779	0.783
Macro Precision	0.711	0.779	0.813	0.784	0.798	0.720	0.778	0.781
Macro Recall	0.710	0.770	0.812	0.780	0.781	0.720	0.779	0.782
Macro F1	0.710	0.772	0.812	0.781	0.784	0.720	0.778	0.781
Weighted F1	0.714	0.776	0.814	0.784	0.788	0.723	0.780	0.783

Table B.3: Baseline model performance on time-domain inputs of duration equal to 2.5 wavelengths of the lowest frequency of interest (0.15Hz).

B.2 Baseline Model Performance using Variable Frequency Resolution FFT for the ICU-FU task

DT	4	8	16	32	64	128	256	512	1024
Accuracy	0.626	0.677	0.805	0.83	0.821	0.84	0.822	0.804	0.737
Macro Precision	0.624	0.674	0.804	0.83	0.82	0.839	0.822	0.804	0.736
Macro Recall	0.622	0.673	0.804	0.828	0.82	0.838	0.819	0.802	0.735
Macro F1	0.621	0.673	0.803	0.828	0.819	0.838	0.819	0.802	0.734
Weighted F1	0.625	0.676	0.805	0.83	0.821	0.84	0.822	0.804	0.737

RF(10)	4	8	16	32	64	128	256	512	1024
Accuracy	0.68	0.702	0.825	0.832	0.84	0.853	0.847	0.822	0.798
Macro Precision	0.678	0.701	0.826	0.836	0.845	0.856	0.85	0.828	0.804
Macro Recall	0.67	0.692	0.82	0.827	0.834	0.847	0.841	0.813	0.789
Macro F1	0.67	0.693	0.822	0.829	0.836	0.85	0.843	0.816	0.792
Weighted F1	0.676	0.699	0.824	0.831	0.839	0.852	0.846	0.82	0.796

RF(100)	4	8	16	32	64	128	256	512	1024
Accuracy	0.692	0.697	0.849	0.863	0.875	0.882	0.882	0.858	0.835
Macro Precision	0.689	0.696	0.848	0.862	0.875	0.881	0.882	0.859	0.837
Macro Recall	0.685	0.692	0.847	0.861	0.874	0.881	0.88	0.856	0.831
Macro F1	0.685	0.692	0.847	0.861	0.874	0.88	0.88	0.856	0.832
Weighted F1	0.69	0.696	0.849	0.862	0.875	0.882	0.882	0.858	0.835

SVM(RBF)	4	8	16	32	64	128	256	512	1024
Accuracy	0.67	0.673	0.747	0.785	0.8	0.818	0.822	0.825	0.821
Macro Precision	0.683	0.685	0.756	0.792	0.802	0.822	0.826	0.829	0.827
Macro Recall	0.68	0.683	0.755	0.792	0.804	0.823	0.828	0.831	0.828
Macro F1	0.67	0.672	0.747	0.784	0.799	0.817	0.822	0.824	0.821
Weighted F1	0.669	0.672	0.747	0.785	0.8	0.818	0.822	0.825	0.821

SVM(poly)	4	8	16	32	64	128	256	512	1024
Accuracy	0.551	0.552	0.546	0.65	0.659	0.68	0.686	0.689	0.684
Macro Precision	0.296	0.437	0.315	0.744	0.755	0.761	0.767	0.769	0.766
Macro Recall	0.498	0.5	0.494	0.679	0.689	0.707	0.713	0.715	0.711
Macro F1	0.356	0.362	0.357	0.632	0.643	0.668	0.676	0.679	0.674
Weighted F1	0.394	0.399	0.394	0.624	0.636	0.662	0.669	0.673	0.667

MLP(10)	4	8	16	32	64	128	256	512	1024
Accuracy	0.52	0.516	0.574	0.561	0.605	0.642	0.612	0.602	0.579
Macro Precision	0.363	0.426	0.487	0.548	0.582	0.643	0.612	0.601	0.579
Macro Recall	0.498	0.498	0.556	0.547	0.593	0.63	0.591	0.59	0.558
Macro F1	0.391	0.42	0.497	0.504	0.567	0.615	0.559	0.552	0.497
Weighted F1	0.407	0.434	0.508	0.513	0.576	0.623	0.573	0.562	0.513

MLP(100)	4	8	16	32	64	128	256	512	1024
Accuracy	0.502	0.529	0.672	0.741	0.801	0.805	0.77	0.721	0.714
Macro Precision	0.308	0.443	0.722	0.776	0.809	0.806	0.772	0.719	0.715
Macro Recall	0.497	0.516	0.659	0.735	0.792	0.802	0.765	0.717	0.712
Macro F1	0.348	0.403	0.636	0.725	0.795	0.802	0.766	0.717	0.711
Weighted F1	0.354	0.414	0.644	0.729	0.799	0.805	0.769	0.72	0.713

MLP(500)	4	8	16	32	64	128	256	512	1024
Accuracy	0.513	0.539	0.669	0.758	0.793	0.821	0.804	0.773	0.751
Macro Precision	0.277	0.357	0.713	0.785	0.815	0.829	0.807	0.773	0.752
Macro Recall	0.498	0.505	0.66	0.747	0.785	0.817	0.799	0.768	0.744
Macro F1	0.339	0.392	0.635	0.743	0.781	0.816	0.8	0.769	0.745
Weighted F1	0.351	0.417	0.642	0.749	0.785	0.819	0.803	0.772	0.749

B.3 Baseline Model Performance using Variable Frequency Resolution FFT for the severities task

DT	4	8	16	32	64	128	256	512	1024
Accuracy	0.422	0.474	0.545	0.592	0.604	0.597	0.576	0.553	0.527
Macro Precision	0.41	0.465	0.536	0.584	0.596	0.589	0.567	0.543	0.516
Macro Recall	0.41	0.465	0.536	0.583	0.596	0.589	0.566	0.543	0.516
Macro F1 Score	0.41	0.465	0.536	0.583	0.596	0.589	0.567	0.543	0.516
Weighted F1 Score	0.422	0.474	0.545	0.592	0.604	0.597	0.576	0.553	0.527

RF(10)	4	8	16	32	64	128	256	512	1024
Accuracy	0.454	0.512	0.59	0.626	0.643	0.633	0.617	0.588	0.556
Macro Precision	0.439	0.506	0.591	0.631	0.648	0.638	0.621	0.591	0.556
Macro Recall	0.43	0.489	0.567	0.602	0.62	0.609	0.591	0.561	0.528
Macro F1 Score	0.43	0.492	0.572	0.609	0.627	0.616	0.598	0.567	0.532
Weighted F1 Score	0.447	0.505	0.584	0.621	0.638	0.628	0.61	0.58	0.546

RF(100)	4	8	16	32	64	128	256	512	1024
Accuracy	0.469	0.542	0.63	0.671	0.694	0.688	0.674	0.646	0.618
Macro Precision	0.454	0.541	0.636	0.685	0.709	0.704	0.692	0.667	0.641
Macro Recall	0.446	0.519	0.606	0.646	0.669	0.662	0.646	0.616	0.584
Macro F1 Score	0.446	0.524	0.613	0.656	0.68	0.672	0.656	0.625	0.592
Weighted F1 Score	0.462	0.535	0.624	0.666	0.689	0.682	0.667	0.638	0.606

SVM(RBF)	4	8	16	32	64	128	256	512	1024
Accuracy	0.437	0.449	0.502	0.53	0.546	0.562	0.567	0.566	0.563
Macro Precision	0.293	0.42	0.619	0.629	0.634	0.644	0.645	0.642	0.635
Macro Recall	0.373	0.373	0.437	0.47	0.49	0.509	0.515	0.516	0.513
Macro F1 Score	0.297	0.285	0.402	0.451	0.48	0.505	0.513	0.513	0.51
Weighted F1 Score	0.33	0.323	0.431	0.476	0.502	0.524	0.531	0.532	0.528

SVM(poly)	4	8	16	32	64	128	256	512	1024
Accuracy	0.425	0.423	0.424	0.443	0.46	0.473	0.48	0.483	0.482
Macro Precision	0.301	0.291	0.436	0.614	0.631	0.637	0.638	0.629	0.611
Macro Recall	0.336	0.335	0.337	0.36	0.38	0.396	0.405	0.409	0.411
Macro F1 Score	0.203	0.203	0.21	0.259	0.298	0.327	0.345	0.353	0.357
Weighted F1 Score	0.256	0.255	0.261	0.303	0.339	0.364	0.38	0.387	0.39

MLP(10)	4	8	16	32	64	128	256	512	1024
Accuracy	0.423	0.428	0.44	0.449	0.434	0.439	0.434	0.43	0.426
Macro Precision	0.174	0.378	0.49	0.41	0.426	0.633	0.454	0.559	0.524
Macro Recall	0.333	0.349	0.352	0.363	0.348	0.352	0.347	0.341	0.337
Macro F1 Score	0.198	0.232	0.236	0.261	0.231	0.234	0.229	0.215	0.206
Weighted F1 Score	0.251	0.278	0.287	0.31	0.281	0.285	0.279	0.268	0.259

MLP(100)	4	8	16	32	64	128	256	512	1024
Accuracy	0.424	0.429	0.44	0.437	0.46	0.435	0.436	0.436	0.444
Macro Precision	0.17	0.403	0.467	0.4	0.41	0.432	0.474	0.631	0.722
Macro Recall	0.34	0.343	0.353	0.351	0.378	0.347	0.348	0.348	0.356
Macro F1 Score	0.217	0.219	0.237	0.238	0.286	0.229	0.228	0.227	0.244
Weighted F1 Score	0.266	0.269	0.286	0.289	0.333	0.28	0.279	0.279	0.294

MLP(500)	4	8	16	32	64	128	256	512	1024
Accuracy	0.423	0.423	0.44	0.439	0.441	0.428	0.432	0.426	0.425
Macro Precision	0.151	0.408	0.256	0.439	0.508	0.388	0.372	0.275	0.54
Macro Recall	0.333	0.333	0.352	0.352	0.353	0.339	0.345	0.337	0.336
Macro F1 Score	0.198	0.198	0.235	0.236	0.24	0.212	0.224	0.206	0.204
Weighted F1 Score	0.251	0.252	0.286	0.287	0.291	0.264	0.275	0.259	0.257

Appendix C

Low-Level Explanations

C.1 Complications in STFT Application for ICU-FU

Using an STFT window size of ∼66 seconds leads to some complications in the ICU-FU case. Our thought process for a workaround is as follows:

1. The shortest follow-up recording is ∼8.8 minutes.
2. Using an STFT window size of ∼66 seconds we could fit ∼7.97 windows. Clearly, this would get rounded down to 7, which seems unfortunate since we would effectively waste 1/8 of this follow-up recording, as well as from the corresponding, equally long ICU recording.
3. Therefore, we essentially dictate that we want to be able to fit 8 windows for that particular recording and obtain the corresponding window size. This comes out to ∼65.7 seconds or ∼9.86 wavelengths, which we consider a negligibly small deviation from the suggested 10.
4. Consequently, the entire STFT length would be equal to $STFT\_size = window\_size * number\_of\_windows = 65.7 * 8 = 525.6$ seconds.
5. After this, each patient with a longer recording will fit as many of these fixed-size STFTs within their recording. So for example, if the recording duration is L and L / $STFT\_size = 3.3$, we would obtain 3 STFTs for this recording.

Appendix D

Ethics

The majority of ethical, legal, and safety concerns arise from the collection of data from patients. The new dataset was acquired in accordance with the IRB approved clinical study that was conducted at the Hospital for Tropical Diseases (HTD) in Ho Chi Minh City, Vietnam. The study was part of a flagship Wellcome Trust funded project named VITAL. All participants have consented to the study.

The only potential ethical concern we can identify regarding the project itself stems from the use of machine learning models. Models are known to carry over the biases within the input data distribution, such as race, sex, and age, to their outputs. Given a particular dataset, the most we can do as practitioners is to remain aware of this fact, performing thorough and critical evaluation of the obtained results, and avoiding the overestimation of their applicability.

Bibliography

[1] W. H. O. (WHO). *Dengue and Severe Dengue*. URL: https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue. (last accessed: 28.01.2022).

[2] W. M. Program. *Dengue*. URL: https://www.worldmosquitoprogram.org/en/learn/mosquito-borne-diseases/dengue. (last accessed: 28.01.2022).

[3] L. A. Beltz. *Zika and Other Neglected and Emerging Flaviviruses-E-Book: The Continuing Threat to Human Health*. Elsevier Health Sciences, 2021.

[4] J. Allen. "Photoplethysmography and its application in clinical physiological measurement". In: *Physiological measurement* 28.3 (2007), R1.

[5] D. Castaneda *et al.* "A review on wearable photoplethysmography sensors and their potential future applications in health care". In: *International journal of biosensors & bioelectronics* 4.4 (2018), p. 195.

[6] SmartCare. *An open pulse oximeter to support healthcare research*. URL: http://devices.smartcareanalytics.co.uk/. (last accessed: 31.01.2022).

[7] P. A. Hadjimarkou. "Model Development for Understanding the Relationship Between Photoplethysmography (PPG) and Physiological Parameters of Dengue Patients". In: (2021).

[8] C. for Disease Control *et al.* *Areas with Risk of Dengue*. URL: https://www.cdc.gov/dengue/areaswithrisk/index.html#:~:text=The\%20disease\%20is\%20common\%20in,humid\%20climates\%20and\%20Aedes\%20mosquitoes.. (last accessed: 30.01.2022).

[9] C. for Disease Control *et al.* *Dengue Around the World*. URL: https://www.cdc.gov/dengue/areaswithrisk/around-the-world.html. (last accessed: 30.01.2022).

[10] Amboss. *Dengue*. URL: https://next.amboss.com/us/article/350SPg?q=dengue#Z84d1d83616dbfa563e6b5afcd13 (last accessed: 30.01.2022).

[11] L. C. Katzelnick *et al.* "Dengue viruses cluster antigenically but not as discrete serotypes". In: *Science* 349.6254 (2015), pp. 1338–1343.

[12] S. B. Halstead. "Dengue hemorrhagic fever: two infections and antibody dependent enhancement, a brief history and personal memoir". In: *Revista cubana de medicina tropical* 54.3 (2002), pp. 171–179.

[13] L. C. Katzelnick *et al.* "Antibody-dependent enhancement of severe dengue disease in humans". In: *Science* 358.6365 (2017), pp. 929–932.

[14] W. H. O. (WHO). "Dengue vaccine: WHO position paper, September 2018 - Recommendations". In: *Vaccine* 37.35 (Aug. 2019), pp. 4848–4849.

[15] S. Sridhar *et al.* "Effect of dengue serostatus on dengue vaccine safety and efficacy". In: *New England Journal of Medicine* 379.4 (2018), pp. 327–340.

[16] A. Challoner *et al.* "A photoelectric plethysmograph for the measurement of cutaneous blood flow". In: *Physics in Medicine & Biology* 19.3 (1974), p. 317.

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

[18] Empatica. *E4 data - BVP expected signal*. URL: https://support.empatica.com/hc/en-us/articles/360029719792-E4-data-BVP-expected-signal. (last accessed: 31.01.2022).

[19] J. Přibil *et al.* "Comparative Measurement of the PPG Signal on Different Human Body Positions by Sensors Working in Reflection and Transmission Modes". In: *Engineering Proceedings*. Vol. 2. 1. Multidisciplinary Digital Publishing Institute. 2020, p. 69.

[20] Y. Liang *et al.* "An optimal filter for short photoplethysmogram signals". In: *Scientific data* 5.1 (2018), pp. 1–12.

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

[22] F. Rundo *et al.* "An advanced bio-inspired photoplethysmography (PPG) and ECG pattern recognition system for medical assessment". In: *Sensors* 18.2 (2018), p. 405.

[23] J Allen *et al.* "Effects of filtering on multisite photoplethysmography pulse waveform characteristics". In: *Computers in Cardiology, 2004.* IEEE. 2004, pp. 485–488.

[24] W. Waugh *et al.* "Novel signal noise reduction method through cluster analysis, applied to photoplethysmography". In: *Computational and mathematical methods in medicine* 2018 (2018).

[25] M. Elgendi. "Optimal signal quality index for photoplethysmogram signals". In: *Bioengineering* 3.4 (2016), p. 21.

[26] C. Orphanidou. "Quality Assessment for the Photoplethysmogram (PPG)". In: *Signal Quality Assessment in Physiological Monitoring.* Springer, 2018, pp. 41–63.

[27] Q. Li *et al.* "Dynamic time warping and machine learning for signal quality assessment of pulsatile signals". In: *Physiological measurement* 33.9 (2012), p. 1491.

[28] S.-H. Liu *et al.* "Classification of photoplethysmographic signal quality with fuzzy neural network for improvement of stroke volume measurement". In: *Applied Sciences* 10.4 (2020), p. 1476.

[29] S.-H. Liu *et al.* "Classification of photoplethysmographic signal quality with deep convolution neural networks for accurate measurement of cardiac stroke volume". In: *Applied Sciences* 10.13 (2020), p. 4612.

[30] N. Pradhan *et al.* "Evaluation of the signal quality of wrist-based photoplethysmography". In: *Physiological Measurement* 40.6 (2019), p. 065008.

[31] M. Elgendi *et al.* "Toward generating more diagnostic features from photoplethysmogram waveforms". In: *Diseases* 6.1 (2018), p. 20.

[32] M. Elgendi. "On the analysis of fingertip photoplethysmogram signals". In: *Current cardiology reviews* 8.1 (2012), pp. 14–25.

[33] K. Takazawa *et al.* "Assessment of vasoactive agents and vascular aging by the second derivative of photoplethysmogram waveform". In: *Hypertension* 32.2 (1998), pp. 365–370.

[34] D. Sarma *et al.* "Dengue Prediction using Machine Learning Algorithms". In: *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC).* IEEE. 2020, pp. 1–6.

[35] A. K. Chattopadhyay *et al.* "VIRDOCD: A VIRtual DOCtor to predict dengue fatality". In: *Expert Systems* (2021), e12796.

[36] S. Chattopadhyay *et al.* "Predicting Case Fatality of Dengue Epidemic: Statistical Machine Learning Towards a Virtual Doctor". In: *Journal of Nanotechnology in Diagnosis and Treatment* 7 (2021), pp. 10–24.

[37] A. Parchani *et al.* "Electrocardiographic Changes in Dengue Fever: A Review of Literature". In: *International Journal of General Medicine* 14 (2021), p. 5607.

[38] A. Lateef *et al.* "Dengue and relative bradycardia". In: *Emerging infectious diseases* 13.4 (2007), p. 650.

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

[40] P. S. Addison. "Slope transit time (STT): A pulse transit time proxy requiring only a single signal fiducial point". In: *IEEE Transactions on Biomedical Engineering* 63.11 (2016), pp. 2441–2444.

[41] S Yang *et al.* "Cuff-less blood pressure measurement using fingertip photoplethysmogram signals and physiological characteristics". In: *Optics in Health Care and Biomedical Optics VIII.* Vol. 10820. International Society for Optics and Photonics. 2018, p. 1082036.

[42] G. Martínez *et al.* "Can photoplethysmography replace arterial blood pressure in the assessment of blood pressure?" In: *Journal of clinical medicine* 7.10 (2018), p. 316.

[43] K. Pilt *et al.* "New photoplethysmographic signal analysis algorithm for arterial stiffness estimation". In: *The scientific world journal* 2013 (2013).

[44] R. R. de Almeida *et al.* "Dengue hemorrhagic fever: a state-of-the-art review focused in pulmonary involvement". In: *Lung* 195.4 (2017), pp. 389–395.

[45] E. Marchiori *et al.* "Pulmonary manifestations of dengue". In: *Jornal Brasileiro de Pneumologia* 46 (2020).

[46] L. Nilsson *et al.* "Respiration can be monitored by photoplethysmography with high sensitivity and specificity regardless of anaesthesia and ventilatory mode". In: *Acta anaesthesiologica scandinavica* 49.8 (2005), pp. 1157–1162.

[47] D. Jarchi *et al.* "Validation of instantaneous respiratory rate using reflectance PPG from different body positions". In: *Sensors* 18.11 (2018), p. 3705.

[48] T. Tamura. "Current progress of photoplethysmography and SPO2 for health monitoring". In: *Biomedical engineering letters* 9.1 (2019), pp. 21–36.

[49] J. Chaloemwong *et al.* "Useful clinical features and hematological parameters for the diagnosis of dengue infection in patients with acute febrile illness: a retrospective study". In: *BMC hematology* 18.1 (2018), pp. 1–10.

[50] U. Ralapanawa *et al.* "Value of peripheral blood count for dengue severity prediction". In: *BMC research notes* 11.1 (2018), pp. 1–6.

[51] A. R. Kavsaoğlu *et al.* "Non-invasive prediction of hemoglobin level using machine learning techniques with the PPG signal's characteristics features". In: *Applied Soft Computing* 37 (2015), pp. 983–991.

[52] G. Yoon *et al.* "Multiple diagnosis based on photoplethysmography: Hematocrit, SpO2, pulse, and respiration". In: *Optics in Health Care and Biomedical optics: Diagnostics and Treatment*. Vol. 4916. SPIE. 2002, pp. 185–188.

[53] M. Azarnoosh *et al.* "Increasing the Accuracy of Blood Hematocrit Measurement by Triplicate Wavelength Photoplethysmography Method". In: *Jorjani Biomedicine Journal* 6.4 (2018), pp. 19–28.

[54] K. Wirsing. "Time Frequency Analysis of Wavelet and Fourier Transform". In: *Wavelet Theory*. IntechOpen London, UK, 2020.

[55] A. Jansen. "Modal Identification of Lightweight Pedestrian Bridges based on Time-Frequency Analysis". PhD thesis. Mar. 2016. DOI: 10.13140/RG.2.2.17116.54407.

[56] H. Tjahjadi *et al.* "Noninvasive classification of blood pressure based on photoplethysmography signals using bidirectional long short-term memory and time-frequency analysis". In: *IEEE Access* 8 (2020), pp. 20735–20748.

[57] J. Allen *et al.* "Deep learning-based photoplethysmography classification for peripheral arterial disease detection: a proof-of-concept study". In: *Physiological Measurement* 42.5 (2021), p. 054002.

[58] H. Wang *et al.* "A high resolution approach to estimating time-frequency spectra and their amplitudes". In: *Annals of biomedical engineering* 34.2 (2006), pp. 326–338.

[59] K. H. Chon *et al.* "Estimation of respiratory rate from photoplethysmogram data using time–frequency spectral estimation". In: *IEEE Transactions on Biomedical Engineering* 56.8 (2009), pp. 2054–2063.

[60] Q. Zhao. *Neuron Weights*. URL: https://qichaozhao.github.io/potato-lemon-2/. (last accessed: 06.02.2022).

[61] F. Bre *et al.* "Prediction of wind pressure coefficients on building surfaces using artificial neural networks". In: *Energy and Buildings* 158 (2018), pp. 1429–1441.

[62] A. L. Barroso *et al. Match-Lab Imperial: Deep Learning Course*. URL: https://github.com/MatchLab-Imperial/deep-learning-course. (last accessed: 06.02.2022).

[63] K. Simonyan *et al.* "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[64] R. Thakur. *Step by step VGG16 implementation in Keras for beginners*. URL: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c. (last accessed: 07.02.2022).

[65] C. S. Wiki. *Max-pooling / Pooling*. URL: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. (last accessed: 07.02.2022).

[66] S. L. Moulton *et al.* "State-of-the-art monitoring in treatment of dengue shock syndrome: a case series". In: *Journal of medical case reports* 10.1 (2016), pp. 1–7.

[67] F. Esgalhado *et al.* "The Application of Deep Learning Algorithms for PPG Signal Processing and Classification". In: *Computers* 10.12 (2021), p. 158.

[68] D. K. Ming *et al.* "Applied machine learning for the risk-stratification and clinical decision support of hospitalised patients with dengue in Vietnam". In: *PLOS Digital Health* 1.1 (2022), e0000005.

[69] T. Srivastava. *11 Important Model Evaluation Metrics for Machine Learning Everyone should know*. URL: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/. (last accessed: 07.02.2022).

[70] Google. *Machine Learning Crash Course - Classification: ROC Curve and AUC*. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. (last accessed: 07.02.2022).

[71] S. Hicks *et al.* "On evaluation metrics for medical applications of artificial intelligence". In: *medRxiv* (2021).

[72] Comet.ml. *Introduction to Optimizer*. URL: https://www.comet.ml/docs/python-sdk/introduction-optimizer/. (last accessed: 12.06.2022).

[73] M. Kuhn *et al. Applied predictive modeling*. Vol. 26. Springer, 2013.

[74] M. Malik. "Heart rate variability: Standards of measurement, physiological interpretation, and clinical use: Task force of the European Society of Cardiology and the North American Society for Pacing and Electrophysiology". In: *Annals of Noninvasive Electrocardiology* 1.2 (1996), pp. 151–181.

[75] scikit learn. *sklearn.svm.SVC*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. (last accessed: 17.06.2022).

[76] M. T. Ribeiro *et al.* "" Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[77] J. Adebayo *et al.* "Sanity checks for saliency maps". In: *Advances in neural information processing systems* 31 (2018).