Imperial College London
Department of Electrical and Electronic Engineering

Model Development for Understanding the Relationship Between Photoplethysmography (PPG) and Physiological Parameters of Dengue Patients

Philippos Arkis Hadjimarkou

Supervised by Dr. Bernard A. Hernandez Perez

Submitted in part fulfilment of the requirements for the degree of
Master of Science in Applied Machine Learning at
Imperial College London, September 2021

Abstract

Dengue is a mosquito-born virus that can cause a range of complications to those affected; ranging from skin rashes to even death. With no effective treatment currently available and vaccines that can have serious side effects, the virus is a major risk to human lives. Detecting and predicting a severe case of Dengue can be the turning point in the Dengue epidemic management, increasing survival rates of patients experiencing Dengue shock and improving healthcare facility processes. Using wearable photoplethysmography devices, which are cost effective and user-friendly, patients' vital parameters can be continuously monitored and analysed, allowing for the recognition of patterns that may lead to the development of Dengue detection and prediction algorithms. This study will focus on the utilisation of photoplethysmography signals for the exploration, design and implementation of supervised machine learning models for the classification of Dengue patients' records, based on clinically recorded physiological parameters. Signal and data processing techniques will be utilised, leading up to the definition of evaluative experiments and model development. From design and implementation to testing and evaluation, we will be presenting our findings in the following exploratory report, examining a recently formed PPG dataset.

Statement of Originality

This is to certify that the presented project, submitted for the degree of Master of Science in Applied Machine Learning, presents the design, implementation and evaluation of the study I conducted under the supervision of Dr. Hernandez Perez, Bernard A. and portrays work I have conducted diligently, by myself. No third-parties have been involved in the design, implementation and write-up of this study, excluding the ongoing guidance from my supervisor and co-supervisors.

Acknowledgements

I would like to express my gratitude and appreciation to everyone who supported me on an academic and personal level throughout the completion of my project. First, my supervisor Dr. Bernard A. Hernandez Perez and co-supervisor Stefan Karolcik for the tremendous amount of guidance they provided me and for all the troubleshooting we went through together. Their help was invaluable for the completion of this project. I would then like to thank Professor Pantelis Georgiou and Dr. Damien Keng Yen Ming, as co-supervisors of the project, enabling me to carry out this cardinal and novel study. Finally, I want to express my gratitude towards my Girlfriend, Friends and Family, for their constant support in the past year.

‘There are three kinds of lies: lies, damned lies, and statistics.’

-Sir Charles Dilke, 2nd Baronet

Contents

List of Tables

List of Figures

List of Abbreviations

ADC:	Analogue to Digital Converter
ANN:	Artificial Neural Network
CNN:	Convolutional Neural Network
CNS:	Central Nervous System
CWT:	Continuous Wavelet Transform
DSS:	Dengue Shock Syndrome
DSTFT:	Discrete Short-Time Fourier Transform
DWT:	Discrete Wavelet Transform
FE:	Feature Extraction
HTD:	Hospital for Tropical Diseases
IR:	Infrared
KFCV:	K-Fold Cross Validation
LSTM:	Long Short-term Memory
ML:	Machine Learning
MLP:	Multi-layer Perceptron
NN:	Neural Network
OUCRU:	Oxford University Clinical Research Unit
PCA:	Principal Component Analysis
PSD:	Power Spectral Density
RNN:	Recurrent Neural Network
SQI:	Signal Quality Index
STFT:	Short-time Fourier transform

Chapter 1

Introduction

1.1 Motivation for Study

Dengue is a virus transmitted to humans through bites of infected female mosquitoes and can cause a range of complications, including death, to those affected. While cases are more common in areas with hot and humid climates, it is estimated that mosquitoes carrying the virus exist in over 100 countries, affecting over 3 billion people [40]. Annually, there are approximately 400 million infections recorded, with a quarter of those people falling sick and around 22 thousand of them dying from severe infection [14]. Due to the nature of the virus, which has 4 serotypes, a human can get infected up to 4 times, deeming immunity a difficult task. Even though a vaccine is available, it is only safe to administer it to people who have already contracted the virus once, else it could result in the development of severe dengue. Up to date, no targeted treatment is available but fortunately, an early clinical evaluation and detection of severe dengue can cause the rate of fatalities to drop significantly [40][14]. Due to that, the motivation behind the study arises from the need to control infections and most importantly devise a way to monitor and manage epidemics in ways that we can provide clinical decision support; improving care processes.

Photoplethysmography (PPG) uses optical sensors, e.g. photo diodes, and pulses of light emitted from a light source, e.g. a light-emitting diode (LED), to detect and monitor vital sign parameters such as heart rate and blood pressure [50]. This is done through the detection of blood volume alterations, measuring and recording changes in the absorption of light through the tissue on measuring sites [2]. Some of these sites include, but are not limited to, the fingertips and the wrist. Waveforms that are recorded can then be analysed locally on wearable devices, or on the cloud, to extract useful features that can be used in a range of clinical applications. The fact that PPG is an inexpensive means of recording vital sign parameters continuously and non-invasively, makes it a paramount component in the management of the various conditions. In conjunction with advancements in computing, this can enable large-scale monitoring of humans infected with the dengue virus, even in low-income countries with few resources. It is vital that medical centres and hospitals know when a patient is likely to develop severe symptoms of dengue in order to proactively accommodate for the patient's needs and treatment plans, but also manage the logistics associated with a person requiring medical attention [32].

Lastly, it should be noted that it is the first time that a dataset of the form we are to explore has been collected and as a result, it is particularly interesting to evaluate the relationship between PPG signals and dengue patients' physiological parameters. Since this is the first study of this kind, its novelty is incontrovertible.

1.2 Project Objectives

In the bigger picture, the objective of this project is to explore the relationship between dengue patients' physiological parameters and PPG signals. Particularly, the focus falls onto severe dengue cases, as establishing a link between those and PPG, can possibly transform the future of epidemic management and reduce the toll of human lives. To achieve that, signal analysis and machine learning (ML) techniques will be employed. These will be used to facilitate the creation of classification and regression experiments that will test PPG's capacity in characterising patients' physiological parameters.

The first step in the process involves the exploration of the dataset that will be used in the experiments. As it is newly formed, a lot of data structuring and debugging is required. Once a clear structure is defined and raw data is processed for usability in later processing steps, we can continue with matching clinical events to recorded PPG signal instances. PPG signal denoising will follow, ensuring that no important signal components are removed. We can then proceed in calculating signal quality indexes so that we can knowingly select the best part of each signal to be used for the extraction of features. In the extraction stage, we are aiming to process PPG signal data to enhance data representation and boost performance in the ML algorithms that follow. Lastly, a set of experiments will be defined, which will help us in assessing the link between PPG data and dengue patients' physiological parameters, focusing on severity. For example, can we detect a shock if we are only given just a PPG signal segment of a patient? In each experiment, a defined set of supervised ML algorithms will be employed, along with some extra processing steps, to assess PPG's eligibility for use in clinical decision support for dengue. Information on the ML modelling performance on the specific dataset will also be extracted and compared, using standard metrics. Informed decisions and precise execution of all steps in the process is of utmost importance to reach this project's milestones. Successful exploration of the aforementioned steps will flag this study as successful. Limitations and obstacles are expected throughout the process, given that the dataset is new and raw. It is important that our findings are useful for future research in the domain.

Agile practices were employed for the project's management, adjusted to fit the purpose. Weekly supervisor meetings and progress updates were carried out and work has been consistent. For the specific timeline followed, please refer to the Gantt chart in Appendix B. Code listings and example documentation links are available in Appendix A.

Chapter 2

Background Theory and Related Work

2.1 Dengue Virus

Dengue virus belongs to the Flaviviridae family and affects humans on a global scale, with some serious implications in both human health and healthcare management [34]. Dengue epidemics can severely disrupt the normal function of healthcare systems around the globe, especially during rainy seasons [40]. In many cases, symptoms of dengue can be mild, with patients experiencing headaches, muscle and joint pains, as well as moderate to high fever. However, in some cases more severe reactions to the infection are recorded, with patients developing Dengue Hemorrhagic Fever (DHF) or Dengue Shock Syndrome (DSS), which can be fatal [14]. DHF is usually associated with continuous vomiting, bleeding in the gastrointestinal tract and mouth, hematuria, as well as water build up in the lungs and the abdominal cavities [36][21][18]. DSS is an even more severe form of the infection, with patients experiencing dangerously low blood pressure, fast heart rate and circulatory disruptions, amongst other serious symptoms which result in a fatality rate of 2.5% [36][21]. Through literature, there are multiple sets of symptoms that can define severe dengue. Oxford University clinical Research Unit (OUCRU) in Vietnam's Hospital for Tropical Diseases (HTD), in partnership with Imperial College London, have created documentation accompanying their dengue dataset collections [21]. For this study, the severe dengue features that will be incorporated in our experiments will be sourced from this documentation, as we will be using their dataset. In Table 2.1 a summary of these Features is presented, as sourced from the dataset's documentation [21].

Outcome	**Features**
Severe Dengue Shock	Shock i.e. Drop in Pulse Pressure etc.
Severe Dengue Leakage	Ascites Overload Pulmonary Oedema Respiratory Distress Ventilation Required Diuretics
Sever Dengue Bleeding	Gastrointestinal Bleeding * Hematuria* Severe Bleeding
Severe Dengue Organ Impairment	Central Nervous System (CNS) Abnormality * Liver Abnormality * Kidney Abnormalilty *

Table 2.1: Overview of Features Defining Severe Dengue. Asterisks suggest that the Presented Feature is a Compound of Other Physiological Features. (Table Sourced From [21])

An up-to-date clinical review of the advances in dengue research by *Herencia, J. S. S.*, indicates that no antiviral treatment is yet available for dengue and symptoms can only be managed through already known practices, such as admission of fluids and "nasal continuous positive airway pressure" [20][46]. This comes to complement past studies, emphasising the importance of accurate diagnosis in managing epidemics. *Wiwanitkit, V.* in particular, highlights that the key to proper patient care stems from a timely diagnosis [65].

2.2 Photoplethysmography

As presented in Chapter 1, PPG devices are predominantly inexpensive, easy to use and effective in vital sign monitoring and recording. In Figure 2.1, a simplified schematic on how the PPG works is presented, along with a sample waveform output showing systolic and diastolic peaks and the dicrotic notch [10]. Using pulse analysis, e.g. peak and trough detection algorithms, PPG can be used to measure heart rate, blood pressure, oxygen saturation in the blood, respiratory rate, and temperature. More advanced processing can be used for the measurement of other vitals, such as blood pressure and hematocrit levels. As presented in a recent study

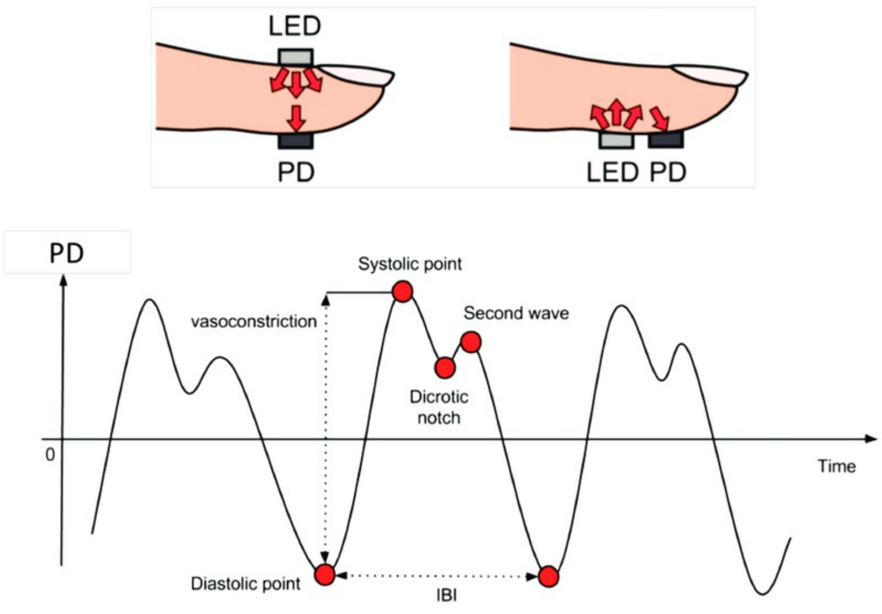


Figure 2.1: Different Finger Setups of the PPG Light Emitting Diodes and Photodetectors, along with a Sample PPG Waveform.(Image Sourced from [10])

by *Castaneda et al.*, PPG has the potential to aid in the diagnosis of a range of cardiovascular diseases, such as arteriosclerosis [9]. In the same report, it is emphasised that advances in wireless communication and wearable technologies will push the scientific community into researching more applications for PPG in clinical decision support, aiding in early diagnosis of various illnesses and conditions [43][9].

Different types of light sources can be used for PPG and depending on the application, a combination of them can be employed [9]. The most common light sources include red LEDs, green LEDs and infrared (IR) LEDs [2][9]. The different wavelengths can enable measurement at different depths of the tissue as *Lindberg L.G. et al.* suggested in 1991 [28]. More recent studies suggest that the use of green LEDs results in deep tissue penetration, as well as more accurate measurements [9]. However, IR LEDs result in the deepest penetration and return more energy on the photodetector, deeming them ideal for taking measurements on denser body areas i.e. the muscle [9][61]. Unfortunately, IR may also transmit more noise [61]. The PPG signal, also called Pleth, is essentially the detected photodetector signal, through some filtering [33]. The Pleth signal is made up of both AC and DC components arising from changes in blood volume in the arteries, light absorption in the tissue, respiratory activity, temperature and other functions of the nervous system [9]. Even though the DC component holds useful information about physiological parameters, AC component frequencies between the range of $0.5 - 4\ Hz$ are usually of most concern during analysis [33]. However, no study has been found that defined the frequency bands present in a PPG signal and what each characterised physiologically.

2.3 PPG Signal Processing

Signal processing applications are vital in improving the quality of signals, rejecting redundant components and extracting useful information that can be used as is, or in further analysis. In our study, signal processing will be used with PPG to enable the successful completion of the experiments by removing noise, describing the signal quality of patient records and extracting features that can be used in characterising the signal in more depth. In short, signal processing is the link between the raw, clinical and PPG data and our ML algorithms.

2.3.1 Filtering of PPG Signals

Filtering PPG signals is an important first step in removing unwanted signal components. Through it, we can remove extrinsic and intrinsic noise. The former relates more to noise from the surrounding environment such as ambient light interference, whereas the latter refers to noise such as motion artefacts. An excellent research article published in 2018 by *Liang. Y. et al.*, provides a thorough review of 9 different filter types with 10 orders each, as applied onto PPG signals [27]. Using the skewness quality index for their evaluation, the report stated that Chebyshev II filter type of order 4, outperformed other filters. Along with the Chebyshev II, the Butteworth filter also performed exceptionally well and it can be considered as the "Gold standard" [27]. For both of the aforementioned filters, low orders are adequate in achieving good performance [27]. In general, the Butterworth filter is more widely adopted for PPG filtering and is suggested in multiple other papers, such as that of *Nilsson. L. et al.* [37]. Other more complex methods of noise reduction, i.e. using cluster analysis, do exist in research, but their application was outside the scope of this project [62].

For completeness, it should be noted that the Butterworth filter, introduced in 1930 by *Butterw–orth S.*, is an infinite impulse response filter, which offers a fast and effective roll-off at the cut-off frequency depending on the filter's order, it has a non-linear phase response and a monotonic frequency response [8].

As discussed earlier, no specific PPG frequency ranges are explicitly defined for use in clinical applications. While in some cases, $0.5 - 4\ Hz$ is suggested to hold enough information for analysis, *Waugh W. et al.* express that the $0.15 - 20\ Hz$ range can be useful in analysis [33][62]. Motion artefact noise overlaps with the PPG signal in all of those ranges and therefore, the frequency filtering process can be abstract and application dependent. Rejection of low frequencies ($< 1\ Hz$) will have a larger effect on the DC component of the PPG [62].

2.3.2 Signal Quality Indexes

Signal quality indexes (SQIs) are derived from the mathematical analysis of raw and filtered PPG signals. They can allow for the rejection of segments, or the whole signal, if the resulting indexes indicate a pour quality signal. This process can be used to complement prior filtering steps and give us a better insight on the quality of signals. Interesting research exists, trying to establish threshold boundaries for various calculated SQIs. For example, *Elgendi M.* presented a clear process on annotating and assessing PPG signals by calculating indexes of skewness, perfusion, kurtosis, entropy, zero crossing rate, signal-to-noise ratio, matching of systolic peaks using two different peak detectors and relative power [12]. While the article was completed successfully, highlighting skewness as one of the best SQIs, limitations do exists which can affect our interpretation of the results and how we apply the knowledge to our project. More specifically, since *Elgendi M.*'s report was only examining healthy subjects, SQIs for diseases subjects might vary significantly "due to the nature of the arrhythmic signals and the associated morphology changes" in unhealthy PPG datasets [12]. As the scope of this project does encapsulate the evaluation of SQI methods, their implementation will be carried out to aid other signal processing stages.

Software libraries were also created for the exploration of SQIs and can be applied onto PPG.

For example, the $vital - sqi$ library, available for Python, provides tools for ECG and PPG signal quality indexing and aids in the creation of quality check pipelines [22]. While parts of this package can be used for our study, it can also be used as a reference for our SQI implementations.

2.3.3 PPG Feature Extraction

Feature extraction (FE) is an important step in PPG pre-processing, allowing us to enhance the descriptive profile of our dataset. While there is not a single approach to this, previous studies give us some insight on approaches that worked on a variety of applications. *Sarma D. et al.* converted physiological clinical variables into numerical values i.e. high fever was labelled as 2, medium fever was labelled as 1 and no fever was labelled as 0 [49]. Vectors consisting of multiple of these clinical variables may be formed, describing instances or periods of time. However, such approach may be computationally very demanding and practically inefficient, if continuous data is required. *Gotlibovych I. et al.* experimented with raw PPG data, directly inputted into classification algorithms to detect ”Atrial Fibrillation” [17]. This proved to be an effective method, which outperformed earlier studies using specially engineered features [17]. Other studies may choose to use a combination of features, forming complex multi-variate feature vectors. Of course, this can result in an increasingly complex task, which not only can prohibit the use in real time scenarios but would also require more complex algorithms during modelling.

2.3.3.1 Time-Frequency Analysis of PPG Signals

A FE approach which can be computationally inexpensive to implement, depending on the algorithm, is the extraction of time-frequency characteristics of a PPG signal. Studies by $Wijshoff$ *R. W. C. G. R. et al.*, *Allen J. et al.*, *Tjahjadi H. et al.*, are only a few of the example papers using time-frequency analysis for PPG FE [63][3][58]. No specific methodology in performing the time-frequency analysis is prevalent but Continuous Wavelet Transforms

(CWT) and Short-time Fourier Transforms (STFT) seem to be the preferred methods of transforming a signal from the time domain into the time-frequency domain. Wavelet transforms work by shifting and scaling a so-called "Mother" wavelet over a signal at different frequencies. This process is carried out by the CWT and can be visualised using a scalogram [64]. STFT works by splitting the signal into smaller windows over time and uses the Fourier transform to calculate their frequency profile. These windows can then be used to construct the spectrogram, displaying frequency magnitudes over time. The main difference between CWT and STFT, is that the latter has a fixed time-frequency resolution in all frequencies, whereas the wavelet transform has a varying resolution due to it's shift and scale approach. Figure 2.2 shows this relationship. To implement these time-frequency analyses, the Discrete STFT (DSTFT) and Discrete Wavelet Transforms (DWT) are used [60]. For completeness, Equations 2.1 and 2.2 express STFT and CWT mathematically [64][52].

$$X_l(\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n - lH)e^{-j\omega n} \tag{2.1}$$

where:

X_l	$=$ DSTFT of Windowed Data
$x(n)$	$=$ Input Signal
$w(n)$	$=$ Window Function e.g. Hamming of Length L
H	$=$ Hop Size i.e. Frame Size Divided by the Overlap Factor

$$cwt(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} x(t)\psi\left(\frac{t-\tau}{s}\right) dt \tag{2.2}$$

where:

τ	$=$ Translation Factor
s	$=$ Scale
$\psi(t)$	$=$ Mother Wavelet Function
$\frac{t-\tau}{s}$	$=$ Scale Factor

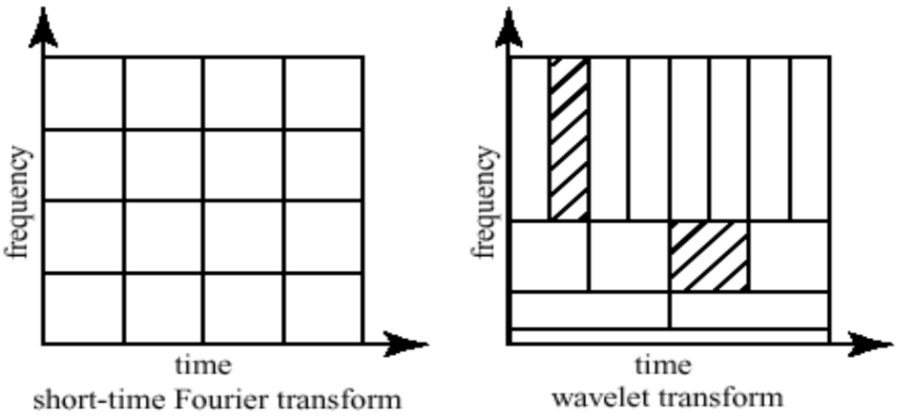


Figure 2.2: Time-frequency representation STFT Spectrogram (left) and Wavelet Transform Scalogram (right). (Image Sourced from [41])

Our study is exploring a novel topic in dengue clinical decision support using PPG and hence, there is no past reference as to which FE method is most effective. Therefore, while wavelet transforms can be more descriptive, they may result in a more complex feature space with higher dimensionality. Thus, STFT can be a preferred starting point.

2.4 Machine Learning and Modelling

2.4.1 ML for PPG and Dengue

Machine Learning is used extensively in many PPG applications, taking advantage of the variety of powerful algorithms, both supervised and unsupervised, to carry out tasks of pattern recognition, classification and regression. As it is the first time our study's problem statement has been looked upon, no existing results are available for direct comparisons to be made. Moreover, a successfully implemented data processing and modelling pipeline in a similar project does not guarantee success in ours. Therefore, while we can use our knowledge and research to make informed decisions, our project is highly exploratory.

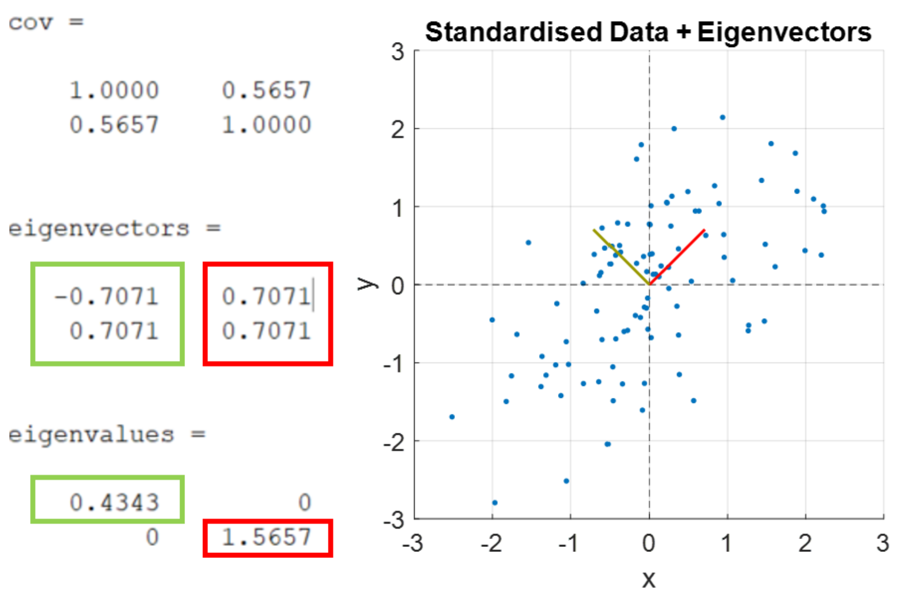
A large number of studies have implemented ML algorithms using PPG derived feature vectors, to model a range of clinical decision processes. Studies by *Tadesse G. A. et al.*, *Yu C. et al.*, *Reiss A. et al.*, *Allen J. et al.*, *Gotlibovych I. et al.* and *Shobitha S. et al.* have implemented both simplistic and complex ML algorithms for the evaluation, detection and prediction of physiological parameters in a clinical environment; either associated with disease or calculated for better diagnostic insights i.e. blood glucose prediction [57][66][47][3][51]. In *Tadesse G. A. et al.*'s study, transfer learning models, utilising deep Convolutional Neural Networks (CNN), improved PPG performance in detecting Tetanus infection severity by "at least 12%" compared to support vector machine algorithms [57]. *Allen J. et al.* research also employs transfer learning methods, using the AlexNet CNN architecture, allowing for arterial disease detection with accuracies of 88.9% [47]. *Gotlibovych I. et al.* have implemented CNN networks in tandem with Long Short-term Memory (LSTM) networks to detect atrial fibrillation using raw PPG signals, resulting in very high area under ROC curve of 0.9999 [3]. *Reiss A. et al.* concluded that CNNs were able to outperform other studies in heart rate estimation, using time-frequency representations of PPG data [47]. Finally, support vector machine and relevance vector machine algorithms were implemented by *Yu C. et al.* and *Shobitha S. et al.* respectively [66][51]. In the former, heart rate identification was explored with a resulting misclassification rate of just 8%. In the latter, blood glucose level predictions were evaluated, with relevance vector machines outperforming the decision tree baseline model with a difference of 0.196 in the kappa score. In many of the explored studies, deep learning approaches outperformed shallow learning algorithms; without implying that the shallow algorithms did not perform well enough. However, as CNNs proved to be powerful enough when handling PPG data, it suggests that they are worth exploring.

Interestingly, a complex algorithm already approved by the FDA exists, utilising machine learning to calculate the Compensatory Reserve Index, using PPG [35]. The Compensatory Reserved Index has been particularly successful in detecting haemorrhage, which can be a symptom of severe dengue and thus may prove to be an important tool in the dengue epidemic management. However, this tool is only available from "Flashback Technologies". Due to this monopoly, it can be expensive to purchase and apply onto the mass population. After all, one of the incentives of using PPG is its low cost and prompt availability.

Research also exists on ML modelling in the dengue clinical decision support domain in general. While many of these studies are not utilising PPG, the approaches followed can definitely be useful to look at when deciding on the methodology to apply in our study; but cannot really be used as reference for our ML models. This is because we are focusing on exploring PPG data. Nevertheless, two studies exist which piqued our intellectual curiosity. One is by *Nguyen T. H. et al.*, using Artificial Neural Networks (ANNs) and other ML models, to associate symptoms of ascites and narrow pulse pressure to predict a ”recurrent shock” in dengue patients[24]. While the results are relatively poor (”Area Under Curve of 0.73”), the methodology followed can be useful. The second study, by *Lam P. K. et al.* aimed to predict DSS in a dataset of children, using data from blood tests [26]. The study utilised a relatively simple logistic regression algorithm, achieving accuracies of 81%. Explicit comparisons cannot be made with the two aforementioned studies, as one focuses on recurrent DSS instances and the other study explores DSS on a dataset of children; whereas we look for a more general link between PPG and severe dengue parameters, focusing on adult data. However, we can use the results of these two studies as a reference point for when we are trying to evaluate and establish a relationship between PPG and severe dengue.

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of data and can be implemented for FE, visualisation and compression purposes. Used commonly in ML to reduce the complexity of feature vectors inputted to a model, PCA can result in better model performance. It does that by projecting data, in our case a feature vector, into its principal components. Principal components are chosen so that the projected data has a maximised variance. These principal components are derived from the covariance matrix of the data, extracted using the singular value decomposition. Once the covariance matrix is found, its eigenvectors and eigenvalues are calculated. Figure 2.3, sourced from [31], illustrates a summary of the procedure for a 2-dimensional example. The eigenvector circled in red, corresponding to the eigenvalue with the largest magnitude, is the first principle component. The one circled in green corresponds to the second principal component. On the left plot in the figure, you can see the original data along with the plotted eigenvectors, where red corresponds to the first principal component, and green corresponds to the second principal component. By a simple matrix multiplication illustrated on Equation 2.3, you can see how we can use the first principle component to transform the observed 2-dimensional data onto 1-dimension [31]. This will cause less sparsity between the features of a dataset, which can allow for ML algorithms to perform better, with less computational resources and complexity.



```
cov =

    1.0000    0.5657
    0.5657    1.0000


eigenvectors =

   -0.7071    0.7071
    0.7071    0.7071


eigenvalues =

    0.4343         0
         0    1.5657
```

Figure 2.3: Example of Principal Component Analysis on 2D Data. Red - First Principal Component, Green - Second Principal Component. (Image Sourced from [31])

$$\tilde{\mathbf{x}}_n = \mathbf{u}_1^\top \mathbf{x}_n \tag{2.3}$$

where:

$\tilde{\mathbf{x}}_n$ = Transformed Dataset of 1 Dimension

$\mathbf{u}_1^\top$ = Transpose of 1st Principal Component

$\mathbf{x}_n$ = Original Dataset of 2 Dimensions

2.4.3 Artificial Neural Networks

Multi-layer perceptron (MLP) models, most commonly referred to as ANNs, are derived from the single perceptron model and belong to the category of supervised ML algorithms. An

MLP architecture is illustrated in Figure 2.4, where each circle represents a "neuron" i.e. Threshold Logic Unit, and each column of circles represent a layer [5]. Each "neuron", holds an activation function, where the input is mapped to an output, through a defined function, i.e. sigmoid. An MLP consists of an input layer, a number of hidden layers (for non-linearity) and an output layer. These are fully connected using weighted connections, whose values are adjusted during training. For classification, the $SoftMax$ function is commonly included in the output "neurons", returning output class probabilities. Learning is made possible with the use of the backpropagation algorithm, a loss function (e.g. categorical cross-entropy), and an optimizer (e.g. Stochastic Gradient Descent). At first, the inputs are fed through the MLP, usually with randomly initialised weights. Then, the loss function is responsible for calculating the loss between the predicted output and the actual labels of classes. The backpropagation algorithm is then employed, going back through the network to calculate the contribution of each layer's connection to the loss value. The optimizer then calculates the loss gradient and uses it to adjust the layer connection weights. The feed-forward, backpropagation and gradient calculation steps are repeated, aiming for a gradient that is close or equal to zero; suggesting that the model has converged to an optimum, or a near-optimum. The number of repetitions is specified by the number of epochs, and the number of training inputs used in learning is specified by the batch size. [15][4]

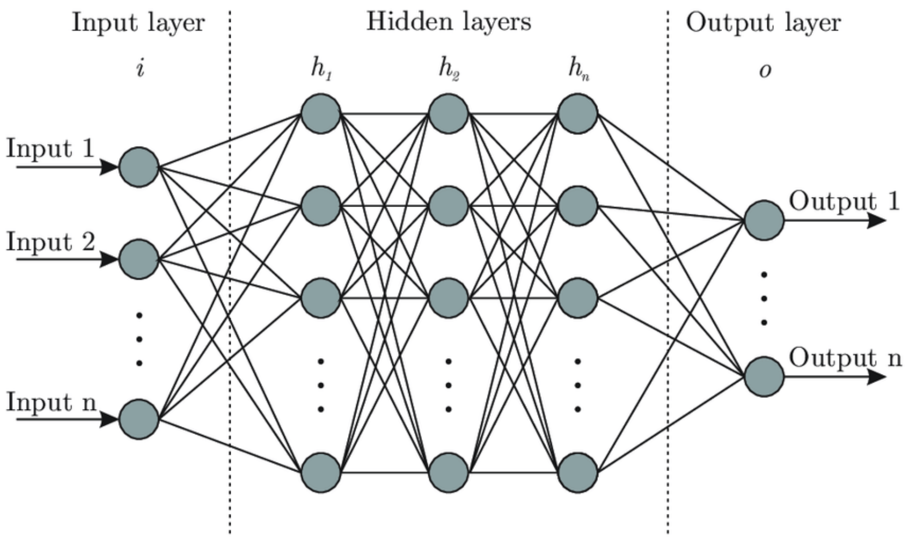


Figure 2.4: ANN Architecture Overview. (Image Sourced from [5])

In many applications, ANNs can be really complex structures to optimize, with multiple hyperparameters to be adjusted in order to achieve optimum convergence. In addition, there are also factors, often revolving around input data and model complexity, which can cause adverse effects on model performance. Therefore, not only good theoretical knowledge is important, but also a good implementation strategy, allowing for substantial model exploration. [15][4]

2.4.4 Convolutional Neural Networks

Convolutional Neural Networks have a fully connected MLP as part of them, which behaves as described in the previous section. This comes after the FE stage, which is what makes CNNs really sought after for pattern recognition and image analysis tasks. Contrary to MLPs where 2-dimensional feature vectors are flattened at the input, CNNs can also process 2-dimensional inputs. These go through the FE stage and are pooled, or flatten, before inputted into the MLP stage. The FE stage is commonly comprised of convolutional and pooling layers.

Convolutional layers work by sliding a kernel i.e. filter above an input feature vector e.g. 2D image. This process is similar to the mathematical convolution which is depicted in Equation 2.4, showing two functions $f(t)$ and $g(t)$, "reversed and shifted" on top of each other [53]. When a filter kernel is convoluted with an input image, a feature map is the result. The height, size and depth of this feature map is dependent on the number of filters, size of the filters, strides (i.e. shift parameter) and padding (zero padded or not). [15][44][4]

After a convolutional layer, a pooling layer is commonly added which acts as a down-sampling mechanism on the features i.e. extracted edges from the convolution activity. In the case of max pooling, kernels are shifted on top of the convolutional layer output and features captured within the kernel are averaged. This process is clearly illustrated in Figure 2.5 [44]. Pooling not only results in better generalisation, but also in lower complexity.[15][44][4]

It is important to note that convolutional layers hold trainable parameters i.e. weights, whereas pooling layers do not.

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.4}$$

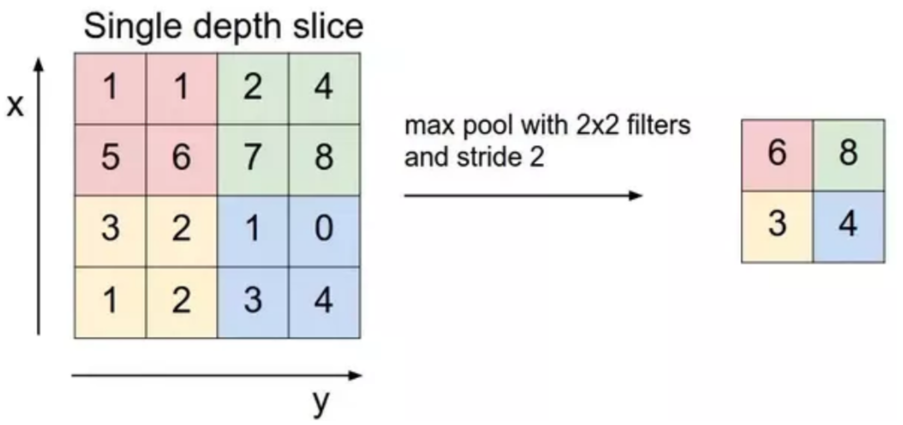


Figure 2.5: Example of Pooling Operation in a CNN. (Image Sourced from [44])

Figure 2.6 shows an example CNN process that presents the operation of a simple CNN. From the example, it can be seen that the feature maps have a smaller height and width as they progress through the FE stage. However, they have are progressively much deeper. [15][4]

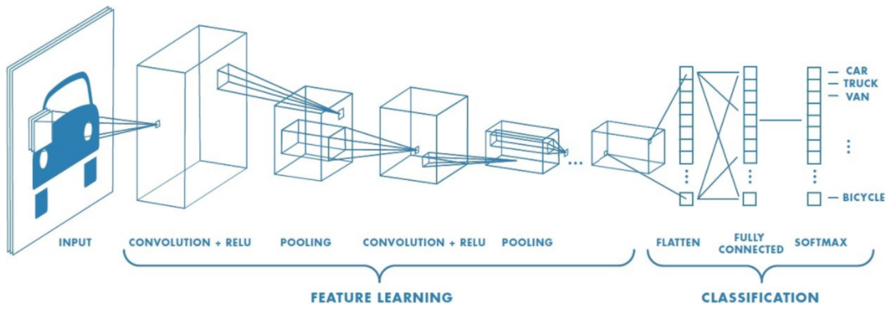


Figure 2.6: Example CNN Process Illustrating FE and Fully Connected Network Classification Stage. (Image Sourced from [44])

CNNs are even more complex to build and train than ANNs, as there are more hyperparameters to define and contain relatively more trainable parameters within the network. Finally, it should be noted that transfer learning applications are common with CNNs, where numerous famous architectures, e.g. AlexNet, are available for a range of applications. [15][4]

2.4.5 Long Short-term Memory Networks

For time-series classification and regression applications, Recurrent Neural Networks (RNNs) are commonly known for their ability to "remember" data sequences. Using this memory, an RNN can easily be employed in applications were we want the output to depend on previous data inputs. Therefore, RNNs are most commonly applied onto forecasting scenarios. Theoretically, RNNs can ”remember” indefinite sequences of data but that is far from the truth in practical applications. Fortunately, in 1997 *Hochreiter* & *Schmidhuber* first introduced LSTM networks [23]. This type of RNN is capable of remembering longer sequences, due to their more complex cell-type architectures, illustrated in Figure 2.7. In contrast to simple RNNs which only have 1 neural network within their cells, LSTMs contain four layers of neural networks. On the aforementioned figure, these neural networks are presented by the yellow rectangles, also illustrating the activations used. The pink features within the cell correspond to pointwise operators. [15][39]

In clinical research, LSTMs are often applied onto disease prediction processes. For example, *Saleh A. Y. et al.* employed LSTMs for the prediction of dengue outbreaks based on climate parameters. In our case, LSTMs can be utilised for predicting severe dengue, using PPG signals as an input.

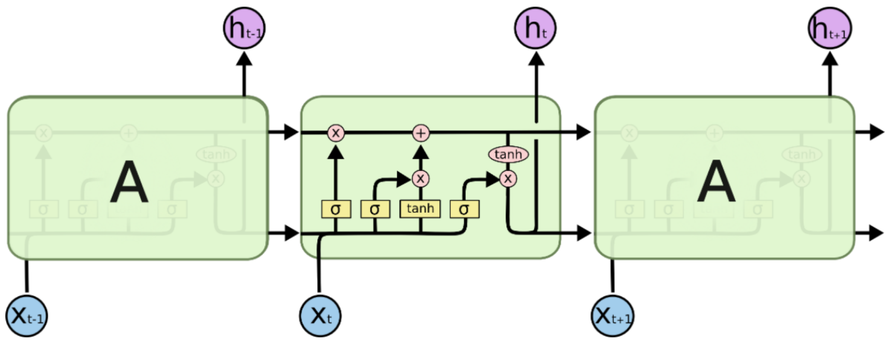


Figure 2.7: Example LSTM Cell Architecture (3 Cells). (Image Sourced from [39])

Similarly to ANNs, LSTM models can be implemented sequentially. Parameters such as activation, kernel initialisation and the number of units (i.e. output dimensions of the LSTM layer) can be selected. The input to a LSTM network should be structured in the following way (*Sample Size*, *Window Size*, *Feature Vector Size*), where the window size defines the number of prior inputs the network will use when making a prediction. This is true for a many-to-one architecture, where the output of the LSTM is based on a given number (> 1) of previous elements (feature vectors). Multiple LSTM layers can be stacked together, but the output is usually a perceptron layer (dense layer) where classification or regression take place. For the latter, no activation is required in the output layer, whereas for the former, the $SoftMax$ function can be applied. Common gradient descent optimizers, such as Adam, are required for learning and the number of epochs, as well as the batch size can be specified; similarly to the ANN training procedure. Noting that for regression, a common loss function is the mean squared error as opposed to classification's categorical cross-entropy. [15][4][48]

Training and testing methodologies for an LSTM network are similar to those utilised in ANNs and CNNs. Again, both theoretical and empirical knowledge should be used to implement and evaluate a model successfully.

Chapter 3

Design and Implementation

3.1 Introduction to Design

The design and approach to this study was very much based on the data made available by Imperial College London and its quality, as well as the abundance of clinical events occurring within each individual signal's time span. Therefore, the technical approach and experimental design of this project was ever-evolving as we progressed through it. This was important for the effective exploitation of the available data and resources. As the project is highly exploratory, assumptions were made in multiple instances, affecting model designs and resulting conclusions. These will be portrayed in further detail in the Chapters that follow. No matter the level of uncertainty that is encapsulating this study, a non-exhaustive pipeline of technical work was set from the very early stages and as we progressed through the project, this pipeline was updated and improved. This allowed for the review of each step in the process and a structured evaluation of the practices, models and the data itself. The final pipeline is presented in Figure 3.1 and can be used as a reference for the Chapters that follow and for an overview of the technical design procedure. A more detailed breakdown of the different models and experiments that were carried out will be presented in the last sections of this chapter, along with implementation details.

Dengue patients who experienced severe symptoms, as defined in Chapter 2, were the focal point of this study. Identifying a point of severe infection, either at the point of admission or during a clinical stay, would be of great benefit to healthcare professionals in diagnosing a patient; enabling them to better manage such incidents, work proactively in providing the necessary treatments and enabling an overall better resource organisation and allocation within a healthcare facility. As the end goal of this study was to evaluate whether or not PPG signals can be linked to the physiological characteristics of dengue patients, we are going to use severe dengue indicators as they evolve through time, to guide us in the evaluation of this relationship. With the help of ML, and deep learning as an extension, PPG signals will be analysed, processed and explored, aiming to established a link between them and severe dengue symptoms.

3.2 Design and Implementation Pipeline

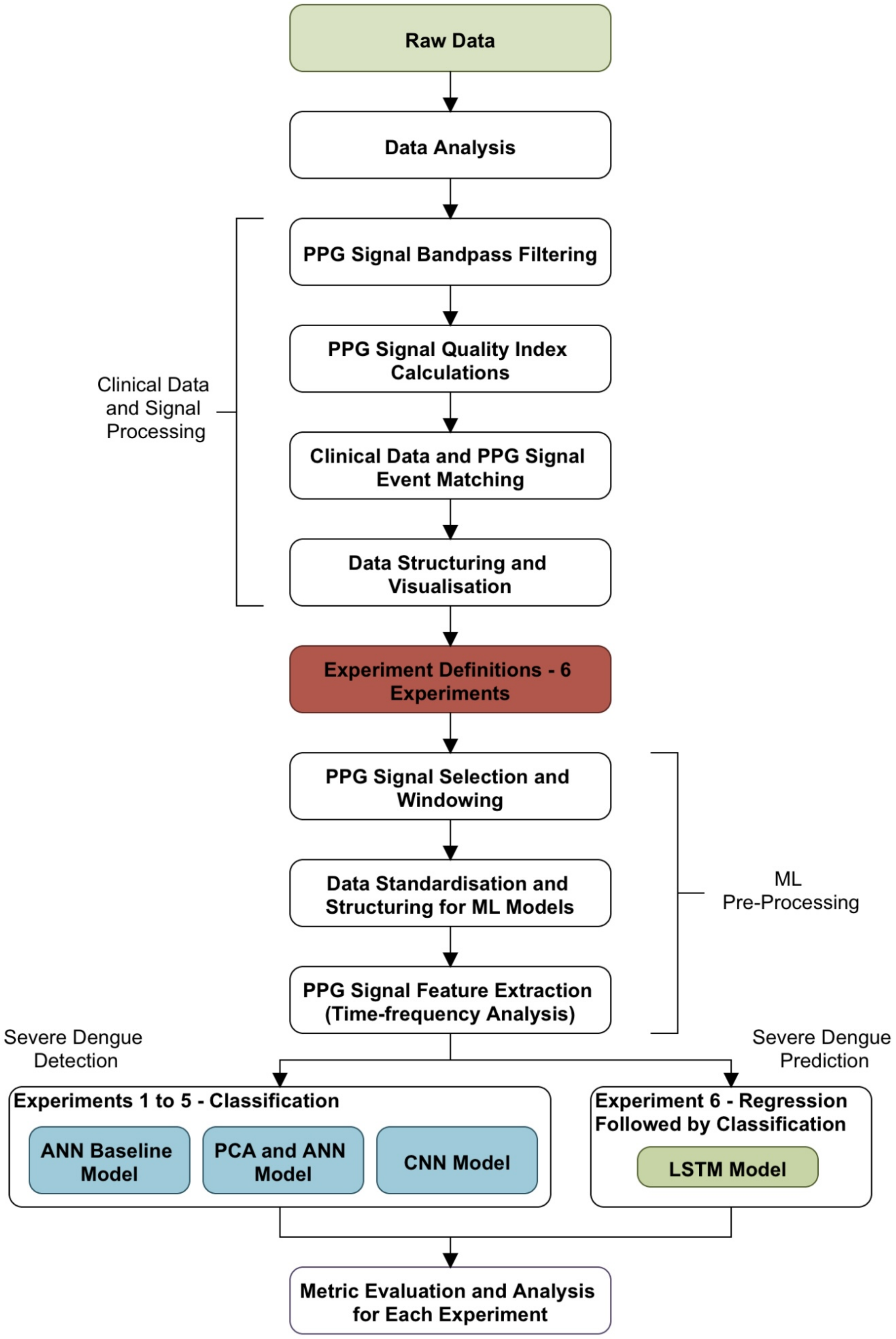

Raw Data
Data Analysis
PPG Signal Bandpass Filtering
PPG Signal Quality Index Calculations
Clinical Data and PPG Signal Event Matching
Data Structuring and Visualisation
Clinical Data and Signal Processing
Experiment Definitions - 6 Experiments
PPG Signal Selection and Windowing
Data Standardisation and Structuring for ML Models
PPG Signal Feature Extraction (Time-frequency Analysis)
ML Pre-Processing
Severe Dengue Detection
Severe Dengue Prediction
Experiments 1 to 5 - Classification
ANN Baseline Model
PCA and ANN Model
CNN Model
Experiment 6 - Regression Followed by Classification
LSTM Model
Metric Evaluation and Analysis for Each Experiment

Figure 3.1: The Project's Design and Technical Implementation Pipeline.

3.3 The Clinical PPG Dataset

Multiple hours of dengue patient data from study ”01nva”, recorded in 2020 in Vietnam’s HTD by OUCRU, were supplied by Imperial College London [21]. For our project, raw PPG data from 11 Adult patients, each of different length, was made available as separate records. This data included a timestamp in milliseconds, a counter, pulse readings in beats per minute, oxygen saturation levels and status, the plethysmography signal, the PPG battery percentage, the red LED’s analogue to digital converted (ADC) signal, the infrared LED’s analogue to digital converted signal (IR ADC) and lastly, the perfusion index. The sampling frequency used for all PPG records was at 100 Hz, resulting in readings to the closest 10th of a millisecond. To complement the PPG records, a single file with the concurrent raw clinical records for each patient was also supplied by the University, including details on patients’ demographic profiles, vital sign parameters and a range of clinical events that the OUCRU study monitored and captured. The latter file was a product of a multi-sheet spreadsheet that was reformatted by the University into a stacked comma separated value file that includes the study number, the complete date and time to the nearest second, and a column for the event name accompanied by a Boolean results column to indicate the occurrence of the event.

As this is a recently formed dataset, there is great opportunity of exploration and analysis. However, that also comes with a high level of uncertainty that requires constant adaptation of design and intense processing.

3.4 Processing the Raw PPG and Clinical Data

In order to prepare the raw data for our ML models that follow, lots of work had to be completed on structuring the dataset and extracting the main parameters of interest from the supplied raw data files. As this is a brand new dataset, a large portion of the project was spent in troubleshooting faulty inputs and missing information in the raw PPG and clinical files. Even though this process was time consuming, it was a cardinal point for our ML algorithms to be effective. The main tool used for processing the data was Python 3 and predominantly the

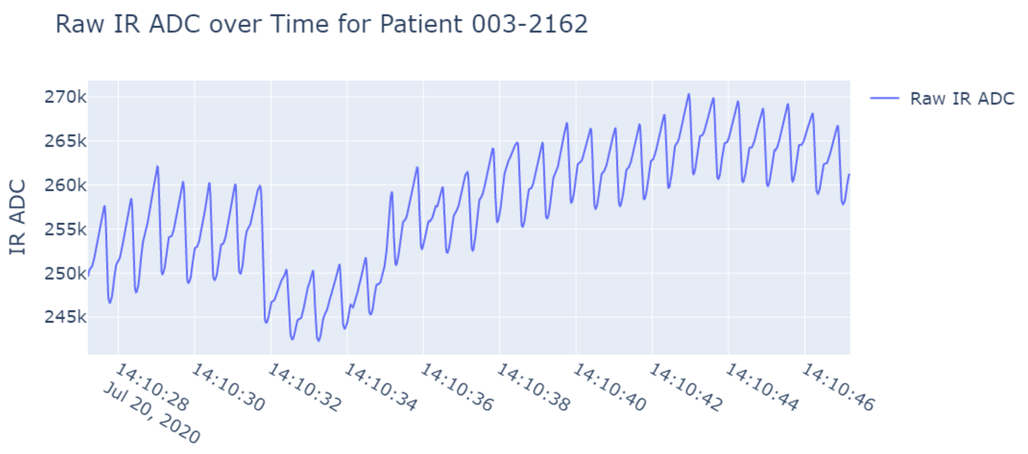
Pandas Library [59] [30]. Using the specified library, data manipulation is computationally more efficient and the resulting code listings are more readable. In addition, utilising libraries such as Pandas allows for better scaling and adjustments which makes the code highly reusable for future work.

The first step of the analysis required the meticulous examination of data in order to better understand the raw structure of the data set. Therefore, we began by plotting patients' raw PPG signals to get an initial glimpse of the waveforms. Both the raw Pleth and IR ADC signals were selected and used throughout the study due to them possessing different features. The latter is considered more noisy than the former, but at the same time it carries a wider spectrum of information compared to the cleaner Pleth signal [28]. It is important to note that throughout the study, and particularly during ML modelling, only one of the two signals were used at a time. Nevertheless, both were evaluated. First, the absolute start time of each PPG record was extracted from the filenames of the raw signal data supplied by Imperial College London; using a script in Python. This was an important step as we were aiming to match clinical data events to the raw signal data. Unfortunately, for patient 2001 the start time of the PPG was not recorded in their filename and as a result the data file was rejected; leaving us with 10 patients. Having collected the start date and time for each raw signal record, the Plotly library was used for plotting the signals with sampling frequency of 100 Hz [25]. Example plots for both IR ADC and Pleth signals can be seen in Figure 3.2.

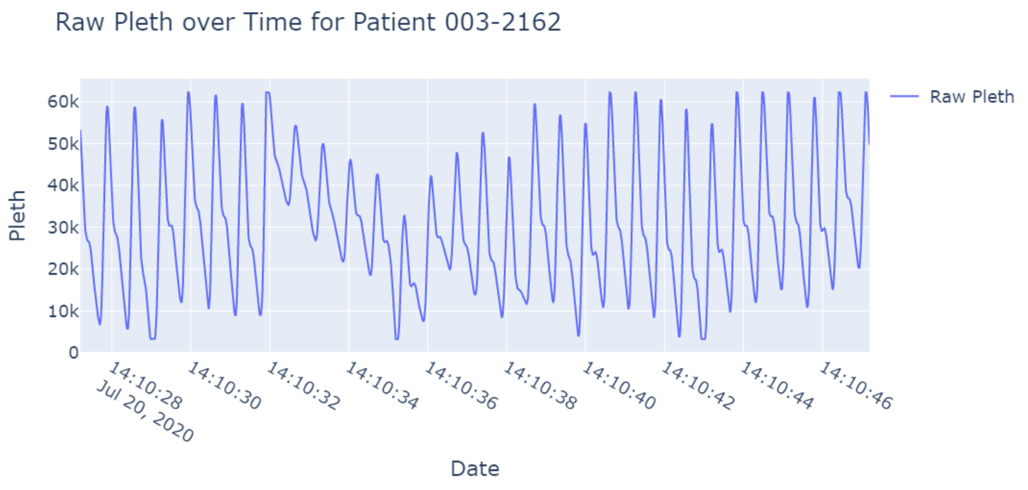
During the initial plotting procedure, it was discovered that only a few seconds of PPG signal data were recorded for patient 003-2026 and as a result, the file was removed from the dataset, leaving us with a total of 9 patient files.

3.4.1 Filtering

Since PPG signals can be highly corrupted with high-frequency noise, baseline wander and other motion artefacts, a band-pass Butterworth filter was implemented to improve signal quality [62]. A $1st$ order high pass filter with a cutoff frequency of 1 Hz and a $4th$ order low-pass filter with cutoff frequency of 20 Hz were implemented. The type and order of the filter, as



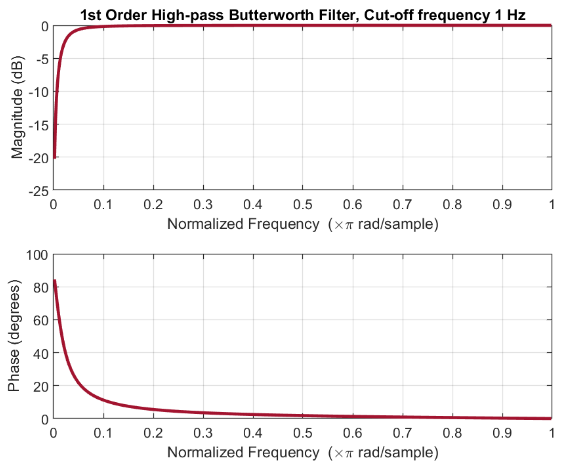
(a) IR ADC Signal



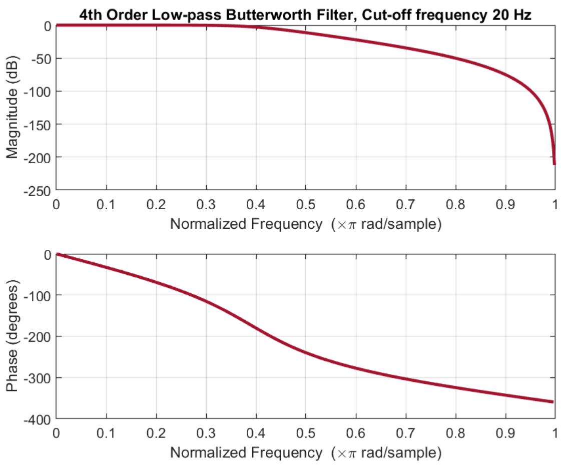
(b) Pleth Signal

Figure 3.2: Example of Raw PPG Waveforms for Patient 003-2162.

well as the cutoff frequencies, were chosen accordingly, as suggested by *Liang et al.* [27]. Even though we could apply a lower cutoff frequency for the low-pass filter at around $5 - 10$ Hz, 20 Hz were chosen to enable further testing in later modelling stages, as higher frequencies may encapsulate physiological effects of dengue that are still unknown. Nevertheless, including a higher frequency spectrum comes with the trade-off of having more noisy signals, especially in the case of IR ADC signals. The frequency response for the Butterworth filter can be described mathematically using Equation 3.1 [55]. The filter frequency response for both the high-pass and low-pass filters can be seen in Figure 3.3, while in Figure 3.4, an example of the resulting filtered PPG signals is portrayed. Please note that the resulting filtered PPG signals have zero mean and the signal's DC component was rejected.



(a) High-pass Butterworth Filter



(b) Low-pass Butterworth Filter

Figure 3.3: Band-pass Butterworth Filter Frequency Response - Magnitude and Phase Plots. First Order High-pass filter with 1 Hz Cut-off Frequency (a) and Fourth Order Low-pass filter with 20 Hz Cut-off Frequency (b).

The Butterworth filter was implemented using the SciPy Library in Python which would allow for an efficient application of the filter onto the signals within the constructed pipeline [60].

$$H_{(j\omega)} = \frac{1}{\sqrt{1 + \varepsilon^2 \left(\frac{\omega}{\omega_p}\right)^{2n}}} \tag{3.1}$$

where:

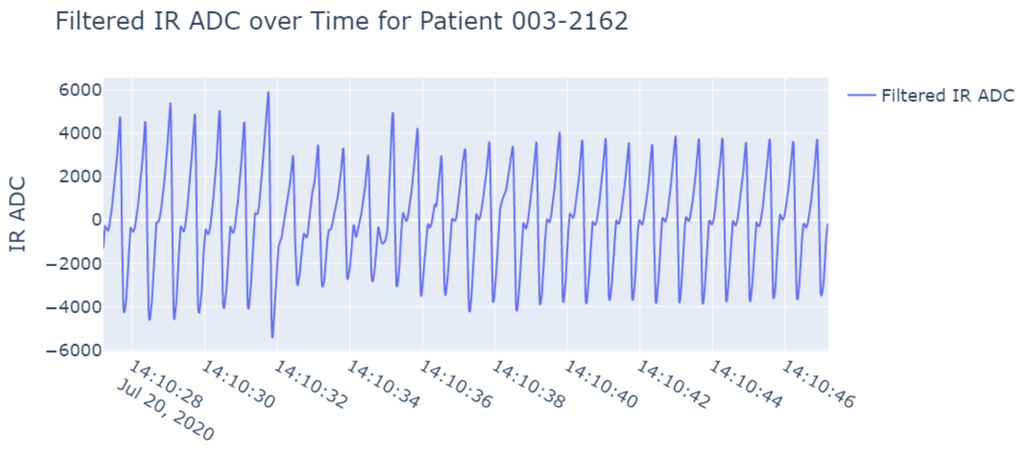
n = Order of the Filter

$\omega = 2\pi f$ (radian frequency)

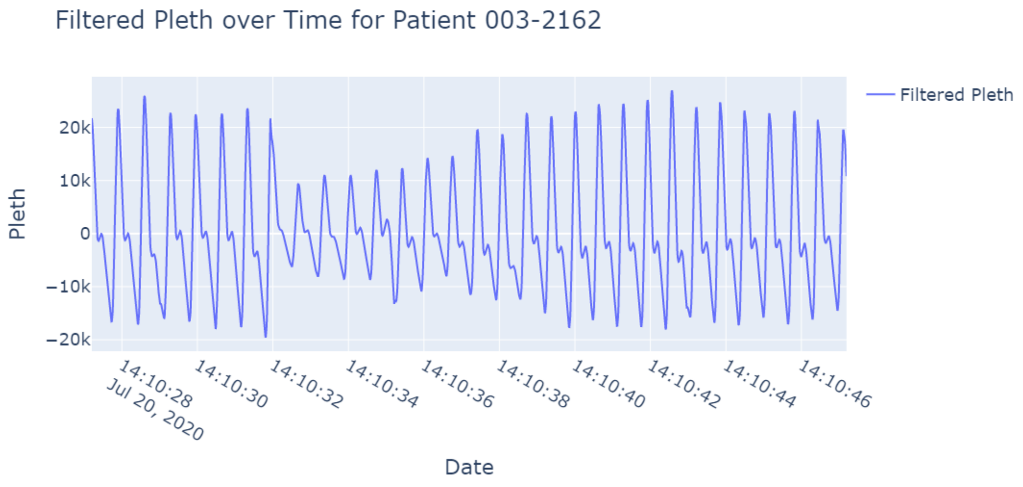
ε = Pass-band Maximum Gain

3.4.2 SQI Calculations

Signal Quality Indexes can be used to assess the quality of a signal, allowing us to pinpoint signal sections that would be more appropriate for use during FE and modelling. In order to calculate the SQIs, both raw and filtered signals of each patient had to be grouped in windows. Each window would then be passed through our pipeline in Python, where an SQI would be calculated and appended into the dataframe. For our application, a 30-second window length



(a) IR ADC Signal



(b) Pleth Signal

Figure 3.4: Example of Filtered PPG Waveforms for Patient 003-2162 using Butterworth Bandpass Filter.

was considered appropriate. The calculated SQIs are listed below in Equations 3.2-3.9 and their input signal requirements are listed in Table 3.1 [12]. The correlogram SQI is not presented mathematically as it portrays the auto-correlation of the PPG signal with itself over given time lags and was implemented using the Vital SQI Library [12][45]. All other SQIS were implemented using the SciPy and NumPy libraries, from scratch [19]. [12]

Skewness:

$$\tilde{\mu}_3 = \frac{\sum_i^N (x_i - \bar{x})^3}{(N-1) * \sigma^3} \tag{3.2}$$

where:

x = Input Signal - Raw PPG

$\bar{x}$ = Mean of Input Signal

N = Signal Length (No. of Samples)

σ = Standard Deviation

Kurtosis:

$$\phi = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{(x_i - \bar{x})^4}{\sigma^4}} \tag{3.3}$$

where:

x = Input Signal - Raw PPG

$\bar{x}$ = Mean of Input Signal

N = Signal Length (No. of Samples)

σ = Standard Deviation

Entropy:

$$S = -\sum_{n=1}^{N} {x_i}^2 \log_e \left({x_i}^2\right) \tag{3.4}$$

where:

x = Input Signal - Raw PPG

N = Signal Length (No. of Samples)

Zero-crossing rate:

$$ZCR = \frac{1}{N}\sum_{i=1}^{N} I\{y_i - y_{i-1} < 0\} \tag{3.5}$$

where:

y = Input Signal - Filtered PPG

N = Signal Length (No. of Samples)

I = Indicator Function - 0 if Argument is False, 1 if True

Mean-crossing rate:

$$MCR = \frac{1}{N}\sum_{i=1}^{N} I\{x_i - x_{i-1} < \bar{x}\} \tag{3.6}$$

where:

x = Input Signal - Raw PPG

$\bar{x}$ = Mean of Input Signal

N = Signal Length (No. of Samples)

I = Indicator Function - 0 if Argument is False, 1 if True

Signal-to-noise Ratio:

$$SNR = \frac{\bar{x}}{s} \tag{3.7}$$

where:

$\bar{x}$ = Mean of Input Signal - Raw PPG

s = Signal Sample Standard Deviation

Perfusion:

$$Perfusion = \left[\frac{(y_{\text{max}} - y_{\text{min}})}{|\bar{x}|}\right] \times 100 \tag{3.8}$$

where:

y = Input Signal - Filtered PPG

$\bar{x}$ = Mean of Input Signal - Raw PPG

Comparing Peak Detection Algorithms (MSQ):

$$MSQ = \frac{(S_{\text{Scipy}} \cap S_{\text{Billauer}})}{S_{\text{Scipy}}} \tag{3.9}$$

where:

S_{Scipy} = Peaks Detected by the Scipy Detecor

S_{Billauer} = Peaks Detected by Billauer Detector

Due to the lack of data, if we were to reject signal segments or a whole signal based on SQIs, the rejection thresholds would have to be very lenient and that would not be efficient or greatly

SQIs	**Requirements**
Kurtosis	Applied onto the Raw Signal
Skewness	Applied onto the Raw Signal
Signal-to-Noise Ratio	Applied onto the Raw Signal
Entropy	Applied onto the Raw Signal
Mean Crossing Rate	Applied onto the Raw Signal
Correlogram	Applied onto the Filtered Signal
MSQ	Applied onto the Filtered Signal using two differrent Peak-Detectors. primary detector = Default_Scipy secondary detector = Billauer_Method
Zero Crossing Rate	Applied onto the Filtered Signal
Perfusion	Applied onto both Raw and Filtered Signals

Table 3.1: SQI Signal Input Requirements Table.

beneficial. Therefore, the best approach was to compare candidate signal segments between each other before deciding if they're appropriate to use in our experiments. While the calculated SQIs can be a great tool for pinpointing sufficient quality signal segments for our ML applications, SQIs can also be used as ML input features, enabling classification and regression applications. However, that is mostly valid with the statistical SQIs and not the whole population of them. For example, calculating the MSQ value, comparing two peak-detectors' performance, does not carry essential signal information other than the level of noise that a signal might have. On the other hand, the perfusion index which relates to the absorption of light through the finger, carries significantly more information that can be linked to physiological parameters i.e. a dengue patient might have abnormal blood flow rates. Nevertheless, for the current study, SQIs' main purpose was to enable us to better understand our dataset and make informed decisions on deciding which signals to use further down the project.

3.4.3 Event Detection and Matching with Filtered Signals

After filtering the signals, splitting it into 30-second windows and calculating SQIs, the next and most important step was to match events from the raw clinical data to specific windows in time. To do this, an algorithm was designed by exploiting the Pandas Library, fetching a series of events from the raw clinical data, along with their incident times, and matching those to patient PPG records, using the start and end times of the 30-second signal windows.

The events of interest were those related to severe dengue, focusing on DSS. A lookup list was created using the following clinical event names: 'event_shock', 'reshock24', 'diagnosis_admission' (always indicating shock), 'shock_admission', 'ascites', 'respiratory_distress', 'ventilation_cannula', 'ventilation_mechanical', 'ventilation_ncpap', 'bleeding_severe', 'cns_abnormal', 'liver_mild', 'pleural_effusion' and 'skidney'. As seen in Section 2, these parameters are either indicating DSS or are closely associated with it. A query was then constructed that searched for events within the raw clinical data, matching it with patient PPG signals, if date and time were within the record's time range. Of course, the query also made sure to check for the correct patient IDs. The clinical event 'shock_admission' was positively marked when patients were admitted with dengue shock and was the only event that was not matched to an exact time window, as it describes the entirety of the patient's signal. By doing this, we also need to assume that the PPG records can still capture the effects of shock onto a patient, even if there is a slight delay between admission and the start of PPG signal recording. This assumption is valid due to the nature of the disease, namely, not having a dedicated treatment to alleviate physiological symptoms timely [46].

3.4.4 Final Data Structure

The processing steps presented in the three previous sections were applied for every patient. As this procedure was automated within our Python script, more data can be easily introduced in future work. In the very end of processing the raw data, two files were outputted in comma separated value format, converted directly from the Pandas dataframes that were used throughout processing.

The first output file contains the windowed signal date and time, followed by the SQIs and clinical event matches for every patient, stacked. In addition to the aforementioned features, we have also added a column with the duration of each patient's record, two columns with the counter start and counter end fetched from the original signal data of each patient, a column with the study number i.e. patient ID, a column indicating the file number for patients with more than one PPG record and finally, a "keep" column that can be toggled within Python to aid with choosing one or multiple signal windows. Having all patient data into a single file allows for easier and faster access for further processing. An in-depth illustration of the structure of this file is shown in Figure 3.5 along with the data types for each column variable.

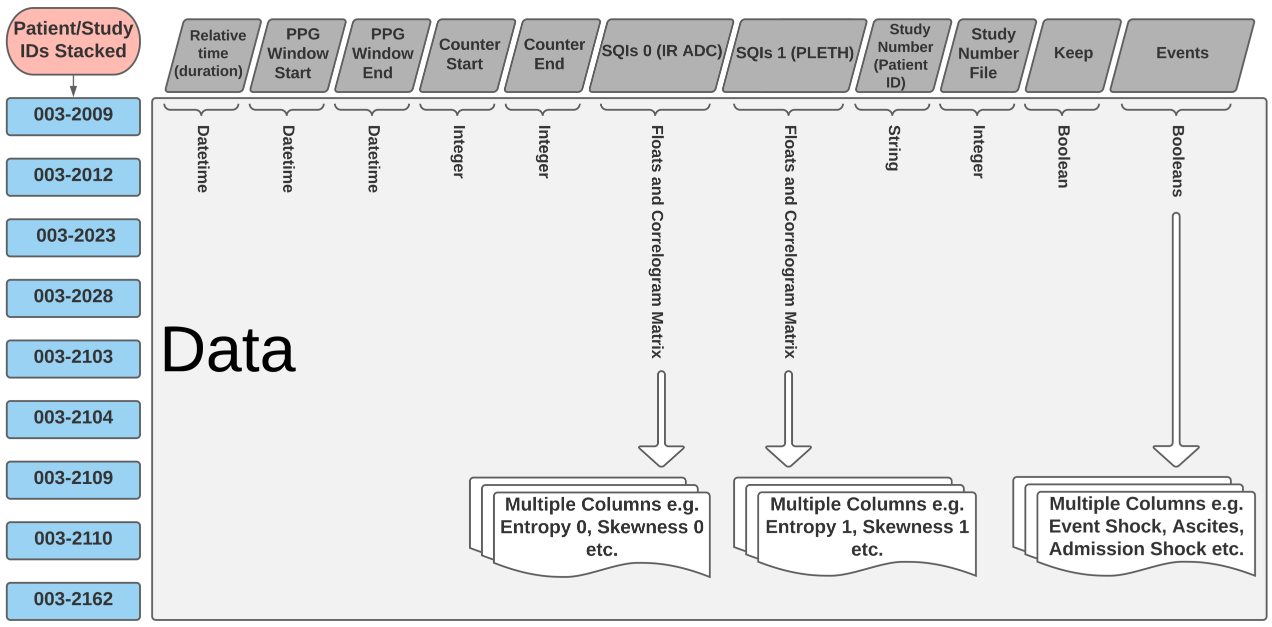


Figure 3.5: Data Processing Output 1: Windowed SQIs matched with Clinical Data - Dataframe Structure.

The second output file contains the full length of, raw and filtered IR ADC and Pleth signals for all patients, stacked. In the same file we have also included the duration of each record, the original counter from the raw data for each file, the study number matching the signals, a column indicating the file number - in case a patient had more than one PPG record - and finally, the PPG datetime. The structure for the second output file can be seen in Figure 3.6. In the same Figure, the data types of each column are also portrayed.

The ”PPG Datetime” in the second output file will be the linking variable with the first output file. The ”PPG Window Start” and ”PPG Window End” can then be used to select the portion of a signal that is required, extracting it from the first file. Using this methodology, it is also very easy to select between Pleth and IR ADC signals when creating the models.

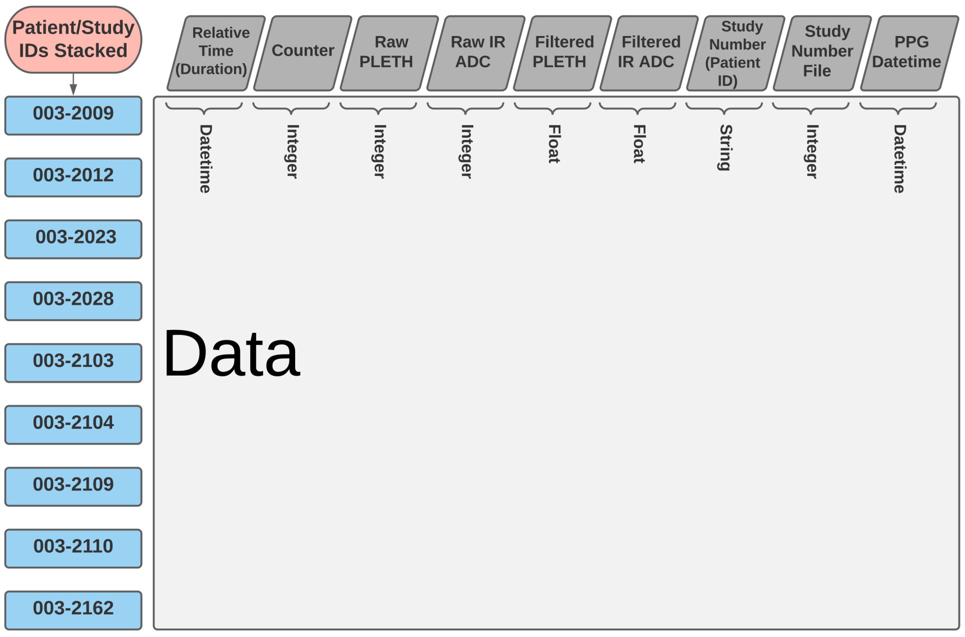
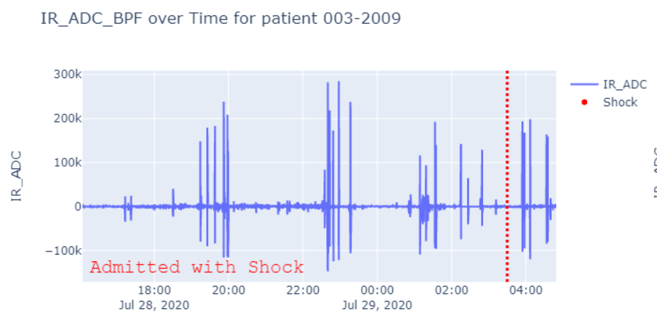


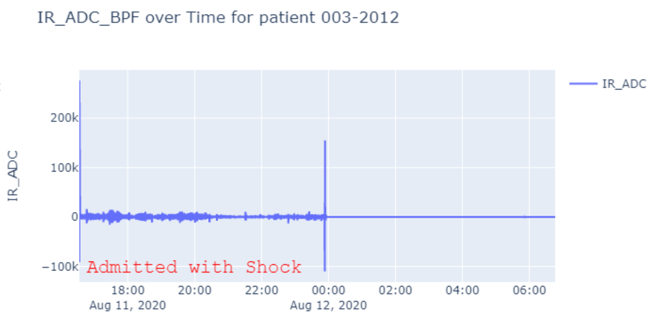
Figure 3.6: Data Processing Output 2: Raw and Filtered Signals - Dataframe Structure.

3.4.5 Visualising Shock Events in Time

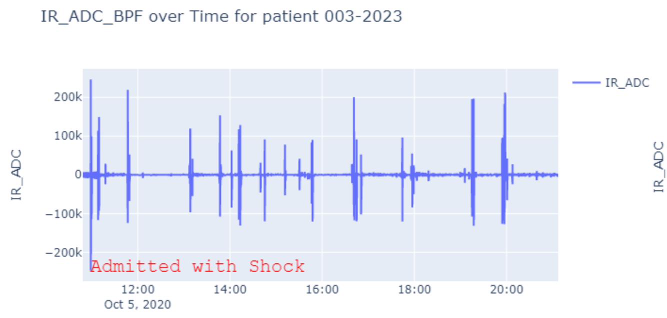
As a final step in the process, it is important that we plot the signals along with the registered shock events. This is paramount in understanding the data further, aiding towards a more informed design of further experiments and ML models. We can plot this data using our Pipeline's output files and the Plotly library for Python. Figure 3.7 illustrates the Filtered IR ADC signal for each of the 9 patients, along with the time and date of the shock event and an indication of shock on admission. Only shock events are illustrated on the figures, simply because no other severe dengue events were present in the clinical records. The same plot for Pleth signals can be found in Appendix Figure C.



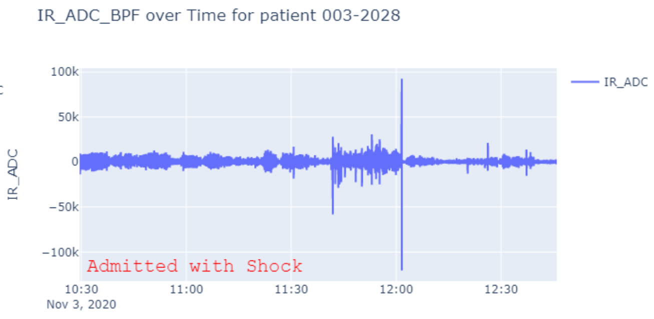
(a)



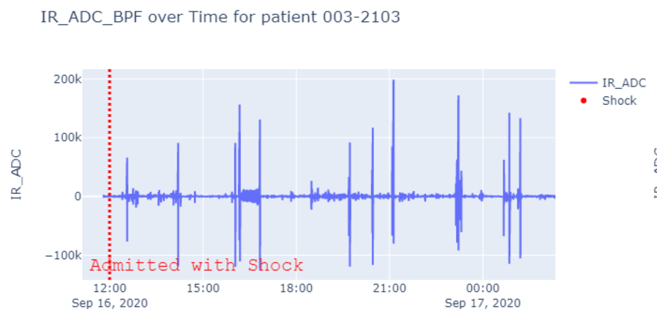
(b)



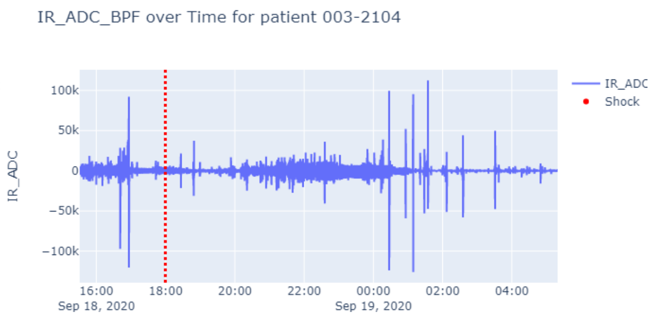
(c)



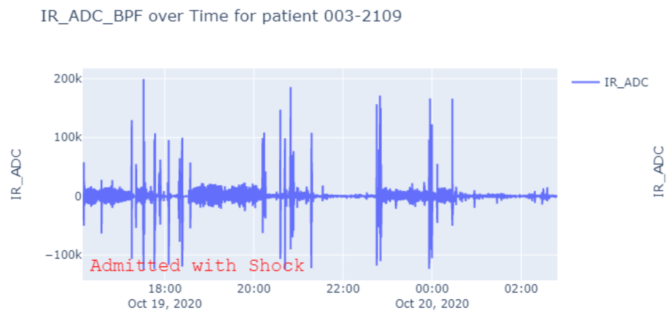
(d)



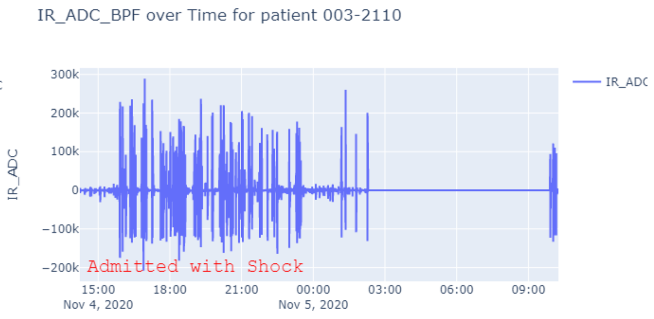
(e)



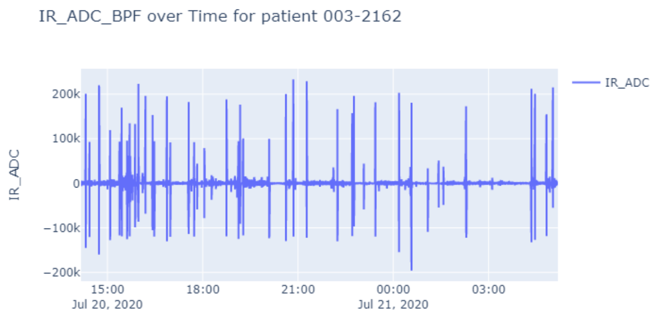
(f)



(g)



(h)



(i)

Figure 3.7: Visualising the Patients' IR ADC Signals along with Corresponding Shock Events Captured Within the PPG Record.

3.5 Experiments to Determine the Relationship Between PPG and Physiological Parameters

After examining the available data, an initial set of experiments had to be set up in order to explore the relationship between PPG data and physiological parameters of dengue patients. It is obvious that there are not enough records, and even less shock events present in those records, to apply complex models that look out for complex relationships within our data. Nevertheless, aiming to fully exploit what is available, we were able to define six different experiment scenarios that would aid in realising the true relationship between PPG signals and the effect of dengue on a patient. These six experiments are illustrated in Table 3.2.

Experiment No.	**Experiment Description**	**Patient ID/IDs used for the Main Experiment**
1	Binary classification of the signals of a dengue patient admitted with shock vs. the signals of a dengue patient who was admitted without shock.	003-2162 and 003-2028
2	For a dengue patient admitted without shock - Binary classification of PPG signals pre-shock and PPG signals post-shock (Given that a shock event was registered within the PPG record).	003-2104
3	For a dengue patient admitted with shock - Binary classification of PPG signals before a shock event and PPG signals post-shock (Given that a shock event was registered within the PPG record).	003-2009
4	Multiclass classification using the classes of Experiment 1 and 2 combined.	003-2162, 003-2028 and 003-2104
5	Multiclass classification using the classes of Experiment 1, 2 and 3 combined.	003-2162, 003-2028, 003-2104 and 003-2009
6	Explore the predictive capacity of a model given a PPG signal of a patient pre and post-shock, using LSTM regression followed by binary classification on the regression's output. (Patient 003-2009 was also admitted with shock)	003-2009

Table 3.2: Defining the Study's Fundamental Experiments.

While we have only defined six main experiments, through them we can investigate multiple hypotheses. For example, we can examine how well a model generalises onto signals of patients that were not involved during the model training and testing. Such tests will enable an even more comprehensive evaluation to be carried out, pinpointing important and useful data trends.

In Table 3.2, the patient ID/IDs used for the experiments are also presented. These were decided based on signal SQIs and availability of data. For example, for Experiment 1, we required a patient signal from a patient that was admitted with no shock and did not develop shock during the PPG record's length. The only patient available with these features is patient 003-2162, as seen in Figure 3.7i. However, this specific patient experienced shock approximately 3 to 4 hours after the end of the PPG record, which is why it is not visible onto the aforementioned figure. Even though this can interfere with the validity of our results, choosing a signal segment early in the recording can still allow us to carry out the experiment successfully. This may not be ideal but it is the closest we could get to a good example of no shock on admission. It is important to note, that attention was also paid onto the signal SQIs when selecting the segments of data to be used in modelling. Highly deteriorated signal portions were avoided completely or were chosen in a way that they represented only a very small percentage of the chosen sample. While we always look for a clean signal, in real time PPG applications that is not always available, especially when you need data to train a model on. Therefore, the best option is to use noise to our advantage and train ML models with some of it present. This will force the model to learn a wider distribution of features per class, resulting in better generalisation. Of course, that is not the case with extreme noise and a dataset with only a few training data points, which is something that we always need to keep in mind.

Some signals, such as that of patient 003-2012 (Figure 3.7b), were avoided entirely. In this specific example, the waveform after the $00:00$ mark was completely flat, which raises some questions on the general signal quality and conditions of recording. Even though the signal before the $00:00$ mark does not seem problematic, patient 003-2023's signal was considered as a better replacement due to it representing the same trend, but with a signal that looks better visually and has SQIs to support that.

The final patient and segment selections will be fully presented in the Results section.

3.6 ML Model Design and Development

In order to carry out the experiments presented in the previous section, various ML algorithms were constructed and explored. The first step in all of the ML models, was to extract the necessary features. Then, depending on the model, certain pre-processing steps had to be completed prior to building and testing the algorithms. Pre-defined metrics are then used to evaluate the model outputs. As part of exploring the relationship between PPG and physiological parameters, all of the ML models that were constructed in the duration of this project were also evaluated on their architectures, allowing us to draw further conclusions on the input PPG data.

3.6.1 ML Model Types Overview

A high-level overview of the ML models constructed for this study is presented in Table 3.3, along with the FE steps carried out. For each of the classification experiments (No. 1-5) presented in Table 3.2, model types 1-3 were built, tested and optimised. Comparing these three models between them, for each experiment, will result in a better comprehension of the data itself and allow for a deeper level of analysis. In the case of Experiment No. 6, a single LSTM regression model was built and evaluated, using a pre-trained classifier model, from earlier experiments. This ML model is shown in Table 3.3 as ML model type 4. Please note, that between FE and ML model fitting, certain pre-processing steps are required which are not presented in Table 3.3. However, those will be clearly explained in the subsections that follow.

3.6.2 ML Pre-processing

Utilising the Pandas library and the output files presented in Section 3.4.4, we were able to extract the exact PPG signal segments to be used as an input to our ML pre-processing stage. To do that, the study number (patient ID) was selected, and the start and end times of each signal segment were chosen. Since we did not have a lot of patients, the process of extracting

ML Model Type No.	**Model Pipeline Description**
1	FE using STFT and classification using an ANN.
2	FE using STFT followed by PCA for dimensionality reduction and classification using an ANN.
3	FE using STFT followed by further FE using a CNN and classification using CNN's MLP.
4	FE using STFT followed by either, no further processing **or** PCA for dimensionality reduction (both scenarios will be tested). Regression utilising an LSTM network will follow. Then, the regression's output will be classified using a pre-trained ANN binary classifier.

Table 3.3: Defining the Four Main Types of ML Models Used in Evaluating the Study's Experiments.

and querying shock event times and signals, as well as reviewing SQIs to make sure that the chosen signal segments were of adequate quality, were done manually. Even though this process was time consuming, it was essential for the quality of our models.

3.6.2.1 Signal Segment Selection and Windowing

As with the rest of the Python implementations, for this part of the experiment the Pandas, Numpy and SciPy libraries were utilised.

At the very start of each experiment, we had to extract the necessary signal segments to be inputted into our pre-processing pipeline and subsequently, through a constructed ML algorithm. For Experiments 1,2 and 4 from Table 3.2, we selected 2 hour signal segments for each class that had to be classified. Taking Experiment 1 as an example, a 2 hour signal segment was selected from patient 003-2162 and another 2 hour signal segment was selected from patient 003-2028. The length selection was based on the signal quality and availability, meaning that if we had exceeded a length of 2 hours, the signal selections would have started deteriorating or we would not have enough data to select from, due to the original signal length. As mentioned earlier, the 2 hour segments were selected diligently by comparing SQIs in different parts within the signal and by visually examining the waveforms. For Experiment 3, the segments were reduced to only 77 minutes of data, due to patient 003-2009's post-shock signal being too short. Unfortunately, this was the only patient exhibiting both shock on admission and a clinical shock within the PPG record that was long enough. While patient 003-2103 exhibited similar clinical features, the pre-shock signal length was minimal, deeming it unusable for this classification process. The aforementioned characteristics can be easily observed in Figure 3.7, presented earlier. It is important to note that while SQIs of the selected segment were not excellent, using less data was less desirable because of the way ML algorithms work; more data results in better training and in some cases better generalisation [15]. Since Experiment 5 is a classification of all of the classes presented in Experiments 1, 2 and 3, we decided to select 77 minute segments for all classes. That was done in order to balance the dataset and compare model training and testing performance fairly. Finally, for Experiment 6, as we are aiming to explore time-series prediction through regression, the whole length of patient 003-2009's data was used, as is. The exact times of the selected segments, for each experiment, will be presented in the next chapter of this study.

After selecting a signal segment for a patient in an experiment, we had to specify which signal of that patient to use. Both filtered IR ADC and filtered Pleth signals were tested out for each and every experiment, separately. This enables us to compare results and pinpoint our observations.

Following segment and type selection, we then proceeded with splitting the signal segments into smaller windows. For experiments with 2 hour signal segments representing each class, the segment was split into 100 windows, each representing 7200 signal data points i.e. 1 minute and 12 seconds. When 77-minute segments were chosen, i.e. in Experiments 3 and 5, we split the data into 77 windows, each corresponding to 6000 signal data points i.e. 1 minute. This was done in order to increase the training data that is available, but through that, we could also evaluate whether or not the window length was an important parameter in ML model performance. Once all the classes' signals were split into windows, those were concatenated into a single array, in order.

As a final step in this process, an array with labels was created, corresponding to each window, depending on the class it belonged to. For the case of Experiments 1,2 and 3, binary labels were created, whereas for Experiments 4 and 5, we formed 4 and 6 labels respectively. Even though Experiment 6 involved regression, labels had to be formed for the classification stage of the experiment. For all experiments, labels were concatenated in order into a single array.

3.6.2.2 Standardization and Structuring Data for Training and Testing

As a result of the previous section's processing, for each experiment, an array of signal segment windows was created together with an array of labels. Taking Experiment 1 as an example, we had an array of 200 windows in total, 100 from Class 0 and 100 from Class 1, each containing 7200 data points. This array was accompanied by another array of 200 labels, in this case 100 labels of 0 and 100 labels of 1.

In all ML models, the data has to be split into a train and a test set. For our case, since the main models we explored were neural networks, a validation set was also defined in a later stage. For now, Scikit-Learn's *train_test_split* function was used, with the $stratified$ and $shuffled$ attributes set, to shuffle the data and split it into 80% training and 20% test sets. The stratified option made sure that the classes were equally represented in both the train and test sets. This split was also applied to the array of labels. After exploring some models, in order to verify the results for the best performing ones, K-Fold Cross Validation (KFCV) was applied onto the data instead. In that case, instead of the *train_test_split* function, we used Scikit-Learn's $StratifiedKFold$ function to split and shuffle the data into train and test folds. Summary of the KFCV procedure is presented in Figure 3.8. Essentially, the data is split randomly into K folds, where K is decided by the user. In our case, also shown on the aforementioned figure, K was equal to 5. This resulted in an 80% / 20% split of the dataset, into training and test respectively. As K is equal to 5, each model that was put to the test, was run 5 times. In each fold, the model metrics (presented at the end of the chapter) were stored and once all folds were completed, they were averaged. KFCV ensures that results are less probabilistic and not dependent on data splits, which in turn ensures a less biased evaluation. Again, to make sure that the classes were equally represented in the training and test sets, we employed a stratified split. Please note that the split functions were used for all stateless classification experiments and not for the LSTM network model, which is stateful (Experiment 6). [6] [29]

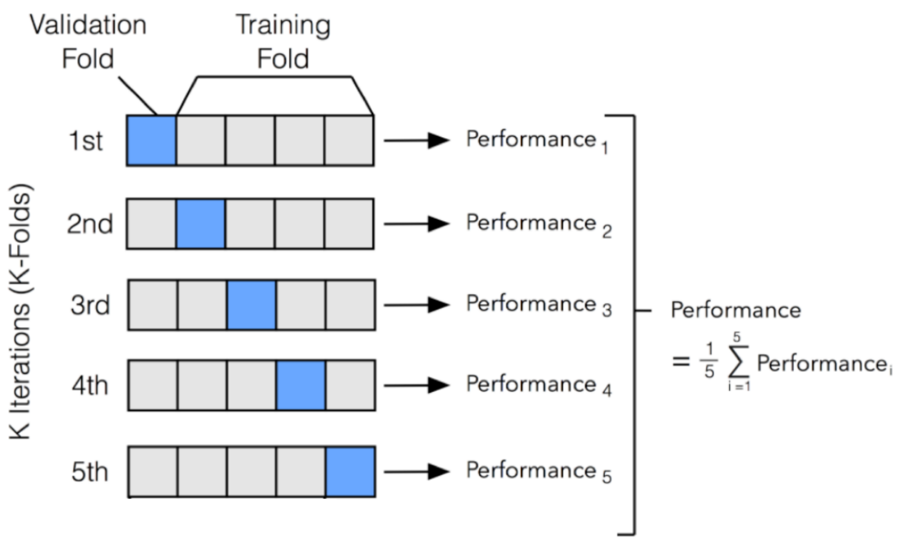


Figure 3.8: K-Fold Cross Validation Approach - Five Fold Case. (Source of Graphic: [29])

After the data is split into training and test sets, we apply standardization as per Equation 3.10. To do that, we first compute the training set's mean and standard deviation. Using these values for μ and σ, we apply the standardization process onto both, train and test sets. We only use the training set's scale parameters to avoid data leakage [7]. The z-score approach was chosen as PPG signals can be described as Gaussian distributions [16]. The standardization process is important so that we can accommodate for bias e.g. in hardware. This process will also result in better scaling of the feature vectors outputted from the FE steps, allowing them to be better represented into the ML algorithms.

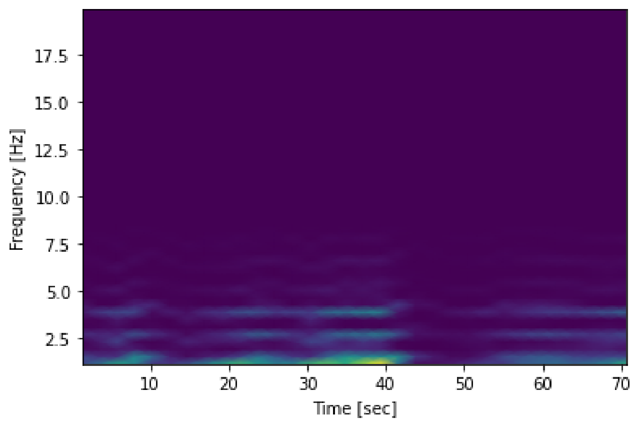
$$z = \frac{X - \mu}{\sigma} \tag{3.10}$$

where:

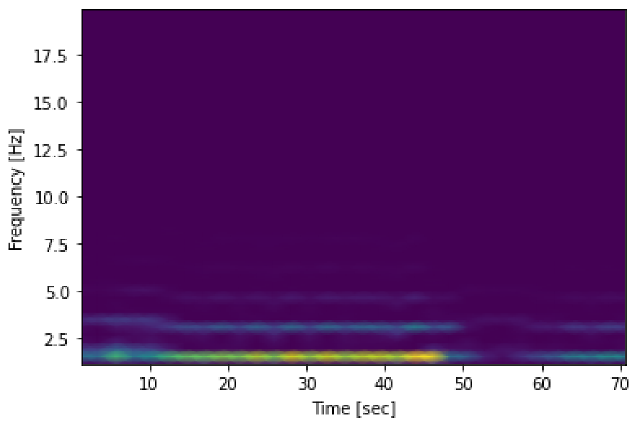
X = data point

μ = population mean

σ = population standard deviation



(a) $1 - 20\ Hz$ Frequency Range Example 1



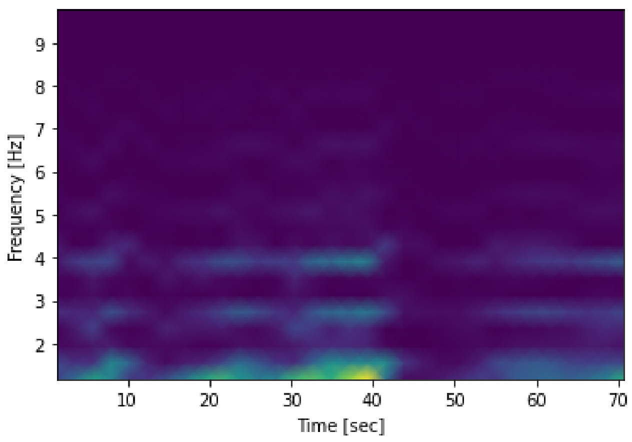
(b) $1 - 20\ Hz$ Frequency Range Example 2

Figure 3.9: Example PSD Spectrograms using STFT.

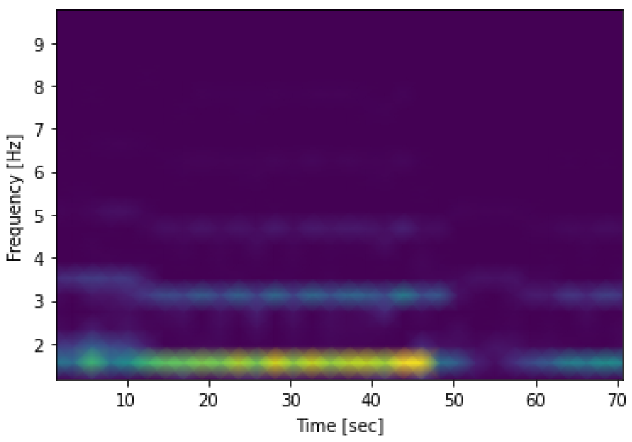
3.6.2.3 Feature Extraction using STFT

While some models might take as an input the raw data, FE can be highly beneficial in isolating important data characteristics. Even though we could use some of the pre-calculated SQIs to form the feature vectors of each window presented in the training and test set, spectral analysis was chosen instead. As presented in Section 2, past research supports the use of time-frequency analysis and the STFT is a good starting point.

The STFT is applied onto each window present in the training and test data. By doing this, we can extract the frequency components of the windowed signal as they are changing in time. The result is a 2-dimensional (2-D) matrix of power spectral density (PSD) amplitudes (V^2/Hz), where each data point corresponds to a time and a frequency. The matrix dimensions depend on the input signal length and the frequency range we wish to extract. For example, a signal window of 7200 data points in the original frequency range of 1 to 20 Hz (original frequency range after filtering), results in a PSD matrix of $49 - by - 32$ amplitude values, where 32 corresponds to the time bins and whereas 49 corresponds to the frequency bins. Example spectrogram plots of this configuration can be seen in Figure 3.9. The STFT was applied onto our windows using SciPy's *spectrogram* function, by iterating through training and test windows, specifying the sampling frequency and the desired output mode i.e. PSD amplitudes [60].



(a) $1 - 10\ Hz$ Frequency Range Example 1



(b) $1 - 10\ Hz$ Frequency Range Example 2

Figure 3.10: Example of PSD Spectrograms using STFT with a Narrower Frequency Selection.

In spectrograms 3.9a and 3.9b from Figure 3.9, it is obvious that at higher frequencies, the average PSD amplitude is significantly lower that at lower frequencies. In that case we can explore a lower range of frequencies to include in our matrix i.e. $1 - 10\ Hz$, which is seen on spectrograms 3.10a and 3.10b. This also causes the dimensionality of the each window's PSD matrix to decrease (i.e. $23 - by - 32$ for a 7200 data-point window) which reduces the amount of computational resources required during modelling and can lead to less model overfitting. We have implemented multiple frequency ranges during model implementations, allowing for comparisons to be made and results will be presented in the next Chapter.

3.6.3 ML Model Design and Implementation

In the following section we will look at our approach in designing the ML models presented in Table 3.3. These models were all implemented using TensorFlow's high-level API, Keras, and other TensorFlow utilities [1]. More specifically, the *Sequential* class within Keras was used for constructing our ML algorithms. All models were built from scratch.

3.6.3.1 Baseline Model

The first model that was explored for all binary and multi-class classification experiments (3.2) was the baseline model; as presented in Table 3.3. Even though baseline models are usually

Hyperparameter Name	Values Tested
No. of Hidden Layers	1, 2, 3, 4, 5, 6
No. of Neurons	32, 64, 128, 256 in Different Configurations
Bias and Kernel regularizers	None, l1 or l2 - 0.01, 0.001, 0.001
Kernel Initialiser	None or glorot_uniform
Batch Normalisation	With or Without (and Position)
Data Augmentation	None, Gaussian Noise, Rotating Features
No. of Dropout Layers	None, 1, 2, 3, 4 - Also dependent on the number of fully connected layers
Dropout Layers	None, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
Optimizers	Adam, Adagrad, RMSprop
Optimizer Learning Rates	1e-2, 1e-3, 1e-4, 1e-5, 2e-3, 2e-4, 3e-3, 3e-4, 4e-3, 4e-4
Activations	Sigmoid, Relu, Tanh
Epochs	50-1000, in increments of 50
Batch Size	16-112, in increments of 16
Validation Size	0.1 or 0.2

Table 3.4: ANN Hyperparameter Grid-search (ML Model Types 1 and 2).

oversimplified ML algorithms, we chose to create one that is simple enough but at the same time optimised, so that we could have a better comparison scale with other models. Essentially, our baseline could be considered as a very simplified version of the models that follow.

A feed-forward MLP was a sensible choice for this purpose. The model inputs are the flattened 2-D PSD amplitude matrices, and their corresponding labels; either binary or multi-class, depending on the experiment. In order to test the model's hyperparameters and see what works best for each experiment, a grid search methodology was employed. The parameters and values that were investigated are presented in Table 3.4. After testing, the best performing hyperparameters were selected and manual fine-tuning was performed to optimise the model further. In general, multi-class classification experiments required models with slightly more complex structures than binary classification problems, but the same grid search parameters were considered adequate.

The loss function for the baseline models was chosen to be *categorical_crossentropy* with a $SoftMax$ activation in the output layer. For binary classification, there were 2 output neurons, whereas for the multi-class classification, the number of output neurons matched the number of classes. That is because, categorical encoding was used, converting the labels into a binary matrix representation.

The cross-entropy loss function for the binary and multi-class classifications is presented in equation 3.11.

$$H(y,p) = -\sum_{i=1}^{k} y_i \log(p_i) \tag{3.11}$$

where:

$y =$ expected output

$p =$ predicted output

$k =$ number of classes

For each classification experiment (3.2), the same optimization procedure was repeated, establishing a baseline performance for each of the experiments.

3.6.3.2 PCA Model

For the PCA Model, a MLP was implemented again, using the exact same designing and training procedure seen in Section 3.6.3.1; including the grid-search approach. However, the difference of this model is the application of PCA onto each of the spectrogram matrices, before their input into the ML model. Through PCA we are aiming to reduce computational resources required in training, by reducing the dimensionality of input feature vectors. In addition, PCA can help ML models converge faster and better, by reducing complexity. The main investigation point for this model was the number of PCA components to use and evaluating the performance of the model compared to the baseline. To find the optimum number of PCA components, a grid-search approach was used, testing within the range of $20 - "number\ of\ feature\ vectors"$, in 20 component increments. For example, if our training set consisted of 200 2-D PSD matrices, the range was equal to 20-200. This test was carried out for numerous MLP architectures for each experiment and performance results were averaged; for each experiment independently. Multiple MLP architectures were tested, some more complex than others, as the number of input data points greatly influences model convergence e.g. a complex model with small input feature vectors will most likely result in the model overfitting.

In order to perform PCA, we added a step into our pipeline, and used SciKit Learn's decomposition library to apply the *PCA* transform onto our spectrogram matrices. This function applies singular value decomposition onto our data in order to project it onto the lower dimensions. In order to do that, the 2-D PSD amplitude matrices had to be flattened first. We then fit the PCA transformation onto the training set, defining the number of components. Lastly, we apply the PCA transformation onto both the training and test sets.

Again, this model was investigated for each classification experiment presented in 3.2 and results were recorded.

3.6.3.3 CNN Model

The next model that was implemented for the evaluation of experiments in Table 3.2 is a CNN. The inputs to the CNN are the 2-D PSD Amplitude matrices. This time, the features are not flattened as they are processed as whole spectrogram 'images' where each distinct point in time and frequency corresponds to an 'image' pixel. The first stage in a CNN is the FE stage, where convolutional and pooling layers are used to extract further information from the 2-D spectrograms. The second part of a CNN is a fully connected network (i.e. MLP), where extracted features are fed in after flattening, for classification.

For CNN models, different hyperparameters were tested for both constituent parts of the network. The grid-search hyperparameters tested can be seen in Table 3.5. Please note that this procedure is not exhaustive as there are millions of possible hyperparameter combinations that can be applied within a CNN. Nevertheless, all tested CNN models were fine-tuned and optimised to the best of our resources.

As described in Section 3.6.3.1, the *SoftMax* activation was used in the output, while *categorical_crossentropy* was used for the model's loss function.

While the grid-search was performed, using our theoretical knowledge on CNNs, we made sure that our actions and selections were coherent. For example, in literature, Adam seems to be the preferred optimizer choice. Another example could be the number of filters and kernel sizes

Feature Extraction Stage Hyperparameter Name	**Values Tested**
Data Augmentation	None, Gaussian Noise, Feature Rotation
No. of Convolutional Layers	1, 2, 3, 4, 5, 6
No. of filters	32, 64, 128, 256, 512
Kernel Size	(2,2), (3,3), (4,4)
Convolutional Layer Padding	Zero Padded or No Padding
Kernel Initialisater	None or glorot_uniform
Batch Normalisation	With or Without (and Position)
Activations	Relu, Sigmoid, Tanh
No. of Dropout Layers	None, 1, 2, 3 - Also dependent on the number of CNN Layers
Dropout Layer Rates	None, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
Max Pooling Layers	Matched the Number of Convolutional Layers
Pool Size	(2,2), (3,3), (4,4), (2,3), (3,4)
Padding when Pooling	Zero Padded or No Padding
Strides when Pooling	(1,1), (2,2), (3,3)
Fully Connected Layer Hyperparameter Name	**Values Tested**
No. of Hidden Layers	1, 2, 3, 4, 5
No. of Neurons	32, 64, 128, 256 in Different Configurations
No. of Dropout Layers	None, 1, 2 - Also dependent on the number of fully connected layers
Dropout Layer Rates	None, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
Activations	Relu, Sigmoid, Tanh
Optimizers	Adam, Adagrad, RMSprop
Bias Regularizer	None, l2 - 0.01, 0.001, 0.0001
Optimizer Learning Rates	1e-2, 1e-3, 1e-4, 1e-5, 2e-3, 2e-4, 3e-3, 3e-4, 4e-3, 4e-4
Epochs	50-1000, usually in increments of 50
Batch Size	16-112, in increments of 16
Validation Size	0.1 or 0.2

Table 3.5: CNN Hyperparameter Grid Search (ML Model Type 3).

that increase as we go deeper within a CNN, to represent different levels of information present in the input feature vectors [15]. However, architecture designs depend on the type of data and its distribution, as that influences the function the model is aiming to approximate. Therefore, empirical evaluation was the main decisive point in our model designs and selections, with theory allowing us to fine-tune and steer the experiment.

3.6.3.4 LSTM Model for Dengue Shock Prediction

The final model that was tested is an LSTM Model, carrying out Experiment 6 from Table 3.2.

Unfortunately, LSTMs require a lot of data for their successful training, due to the complexity of the network and the function they are trying to approximate [13]. This was a major restriction for our experiment. Not only did we not have enough data, the only patient experiencing a shock event enough time after the start of a PPG recording, is patient 003-2009. As you can see from the signal plot in Figure 3.7a, the post-shock data only lasts 77 minutes. Therefore, not enough training data was available for predicting a shock. As a result, the main goal of implementing this experiment and model, was to identify whether or not we could establish a relationship between consecutive spectrogram windows. Essentially, we will examine if PPG signal segment windows carry enough information for them to be used in tandem and allow for the prediction of the next time-step. This would allow us to understand PPG signals further and possibly influence future research, with more data.

Due to their architecture, LSTMs are very hard to optimise and therefore, for this study, we will be using Keras' $LSTM$ class as a black box for the most part, only altering the number of units per layer, the activation, initialisation and bias regularisation. For the whole LSTM model a grid-search implementation was employed, with Table 3.6 showing what was tested [39]. For this experiment and model, a regression procedure was initially performed by the LSTM network. Because of that, a dense layer without activation was added at the end of the sequential model, with units equal to the number of features in a single feature vector. The output was stored and then inputted into a pre-trained classification model that was chosen from earlier experiments. As both full-sized spectrograms and spectrograms with PCA applied on them were tested as inputs to the LSTM, two different models were tested. Both followed the same structure, but used different classifiers for the classification of the LSTM output. For the model with the plain spectrograms as an input, the best performing baseline model (ML model type 1) from Experiment 3 was chosen. For the spectrograms with PCA applied to them, the best performing PCA model (ML model type 2) from Experiment 3 was chosen as the classifier. In either scenarios, if the classifier could classify the LSTM output with success, i.e. with similar performance observed in the training of the chosen classifier, then a relationship could be established.

The PCA model was put to test in order to reduce the data dimensionality and in turn reduce the complexity of the function the model will try to approximate. As we only have very little data, this can be of benefit and can result in better model performance.

Hyperparameter	**Value Tested**
No. of LSTM Layers	1,2,3
No. of LSTM Units	32-300 in increaments of 32
LSTM Lookback Window Length	1-28 in increaments of 1
Number of Dropouts	None, 1, 2 - Before Each Additional LSTM Layer
Kernel Initializer	None, Glorot Uniform, Random Normal
Bias Regulirizers	None, l2 - 0.1, 0.001, 0.0001
Dropout Rate	0.1-0.6, increments of 0.1
Batch Size	16-100 in increaments of 16
Activations	Relu, Sigmoid, Tanh
Optimizers	Adam, Adagrad, RMSprop
Epochs	100-1000 in increaments of 100

Table 3.6: LSTM Hyperparameter Grid-search (ML Model Type 4).

3.6.3.5 Model Evaluation

All implemented models were evaluated using statistical metrics, applied to the ML model outputs. Since we are using supervised learning and our models perform either binary or multi-class classification (including model type 6 whose regression output was tested using a classifier) we first need to construct our confusion matrices. The true positive, true negative, false positive and false negative instances, between predicted and observed data labels, are recorded and usually displayed onto a table. An example confusion matrix for the binary classification case can be seen on Table 3.7. For the multi-class case, the table would simply have more dimensions with the main diagonal showing all the valid matches. [56][15]

	Actual Classes	
Predictions	*Negatives*	*Positives*
Negatives	True Negatives	False Negatives
Positives	False Positives	True Positives

Table 3.7: Example Confusion Matrix for Binary Classification.

There are multiple metrics used throughout literature to evaluate model performance. First, we calculate accuracy, which is defined in Equation 3.12. This shows how the model performs on test data and can give insights on how well models generalise. True predictions refer to the sum of True Positive Values and True Negative Values whereas the False predictions refer to the False Positive and False Negative values. [56]

$$Accuracy = \frac{True\ Prediction}{True\ Predictions + False\ Predictions} \tag{3.12}$$

Even though accuracy can be a good representation of model performance, in a clinical setting we are also interested in metrics which might provide a better link to the clinical requirements. For example, in detecting severe dengue, it might be more important to evaluate model performance based on how well it can detect Positive classes i.e. classes characterising severe dengue. That is because it is far more important to detect a shock signal, than to falsely classify a non-shock signal. The precision metric, can capture this information and it is calculated as seen in 3.13. [56]

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3.13}$$

Recall, also referred to as sensitivity and true positive rate, is another important metric for the evaluation of the results, with link to the clinical features. It essentially detects the percentage of observed positive events, that were actually detected. In our case, that is the number of severe dengue cases that were detected by our classifier, out of all severe cases observed. The calculation procedure is shown in 3.14. [56]

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.14}$$

The F1 score is a mixture of both recall and precision, calculated as the harmonic mean between of the two. This is shown in Equation 3.15. While it is a good metric for evaluation, it is not widely used in scientific literature. [56]

$$F1\ Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{3.15}$$

The confusion matrix was implemented using the $confusion\_matrix$ function and metrics of accuracy, precision, recall and F1 score were calculated using the $classification\_report$ function. This function also outputs macro averages, which are particularly useful for unbalanced classes e.g. classes presented in Experiment 6. Macro averages portray the performance of the classifier on all classes, by calculating metric scores per class and then averaging them. [42]

Chapter 4

Results

4.1 SQIs and Feature Extraction

4.1.1 Final Segment Selections for the Experiments

As presented in previous Sections, using the calculated SQIs and the plotted Data, patient signal segments were selected for each experiment. Table 4.1 portrays the results of this selection along with the defined classes (Numbers) for each experiment. For example, Class 1 in Experiment 1 represents the signal of patient 003-2028, which represents a shock on admission signal and the 03/11/2020 $10:30:00 - 12:30:00$ segment was used in ML modelling. This segment was then split into further segments which were then converted into PSD spectrograms, each representing Class 1.

4.1.2 Feature Extraction

Multiple frequency ranges were compared during implementation, to determine the ideal frequency range for our spectrogram FE process. The results from different frequency ranges were collected using a sample ANN model (similar to Model 1 from 3.3), allowing us to empirically compare the spectrogram frequency ranges. Figure 4.1 illustrates the results.

Experiment Number	**Patient ID and Signal Description**	**Signal Segments Used (Datetime)**	**ML Model Class**
1	003-2162 - No Admission Shock	20/07/2020 14:30:00-16:30:00	Class 0
	003-2028 - Admission Shock	03/11/2020 10:30:00-12:30:00	Class 1
2	003-2104 - Pre clinical Shock	18/09/2020 15:45:00-17:45:00	Class 0
	003-2104 - Post clinical Shock	18/09/2020 18:15:00-20:25:00	Class 1
3	003-2009 - Pre clinical Shock (Admitted with Shock)	29/07/2020 02:16:00-03:28:00	Class 0
	003-2009 - Post clinical Shock (Admitted with Shock)	29/07/2020 03:31:00-04:43:00	Class 1
4	003-2104 - Pre clinical Shock	18/09/2020 15:45:00-17:45:00	Class 0
	003-2104 - Post clinical Shock	18/09/2020 18:15:00-20:25:00	Class 1
	003-2162 - No Admission Shock	20/07/2020 14:30:00-16:30:00	Class 2
	003-2028 - Admission Shock	03/11/2020 10:30:00-12:30:00	Class 3
5	003-2104 - Pre clinical Shock	18/09/2020 15:45:00-17:45:00	Class 0
	003-2104 - Post clinical Shock	18/09/2020 18:15:00-20:25:00	Class 1
	003-2162 - No Admission Shock	20/07/2020 14:30:00-16:30:00	Class 2
	003-2028 - Admission Shock	03/11/2020 10:30:00-12:30:00	Class 3
	003-2009 - Pre clinical Shock (Admitted with Shock)	29/07/2020 02:16:00-03:28:00	Class 4
	003-2009 - Post clinical Shock (Admitted with Shock)	29/07/2020 03:31:00-04:43:00	Class 5
6	003-2009 Shock Signal of Patient Admitted with Shock and Experiencing a Clinical Shock	28/07/2020 16:04:00 - 29/07/2020 04:48:20	Regression

Table 4.1: Selected Signal Segments for All Experiments, Based on SQIs and Plotted Data Visualisations.

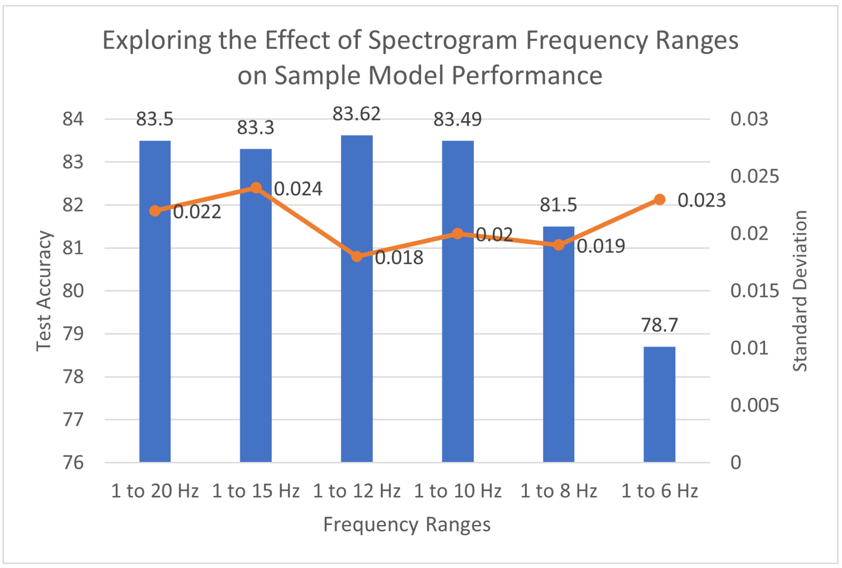


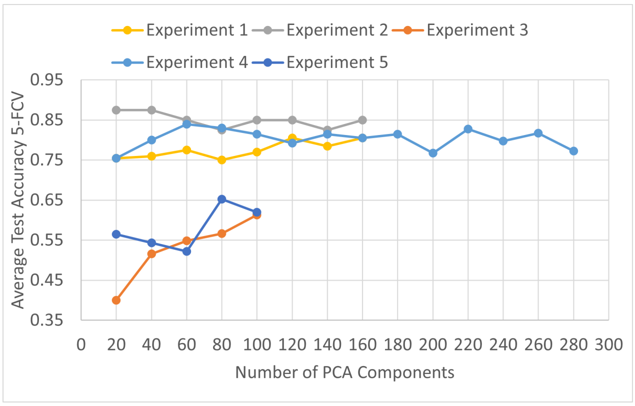
Figure 4.1: Frequency Range Exploration in Spectrogram FE - Inspired from Elbow Methodology to find the Optimum Spectrogram Frequency Range for our Feature Vector Representations.

It can be seen that the major turning point in test accuracy, in Figure 4.1, is from the $1-10$ Hz range to the $1-8$ Hz range, with an approximate difference of 2%. Therefore, computational resources can be minimised by selecting the $1-10$ Hz frequency range, without compromising models' performance.

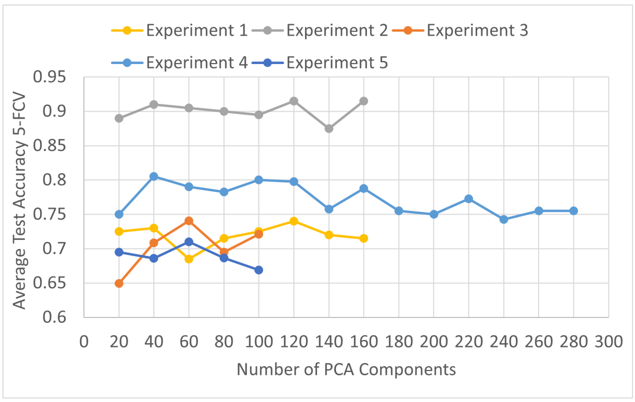
4.1.3 Dimensionality Reduction with PCA

To decide the final number of principal components to be used in each experiment's "PCA model", empirical data was collected, comparing the model performance in relation to the number of principal components used in the transformation of each input spectrogram window. This procedure was carried out on both IR ADC signals and Pleth signals. The results of this investigation are presented in Figures 4.2a and 4.2b below. Please note that using less than 20 components proved inadequate for all experiments.

From the figures, there is no universal number of components that outperforms others, and therefore for each experiment the number of components yielding the highest accuracy in the test set was chosen. Generally, the number of components that led to the best performance were in between the range of 40 and 120 principal components.



(a) IR ADC Signals



(b) Pleth Signals

Figure 4.2: Effect of the Number of Principal Components on Sample Model Performance for Each Experiment. The Maximum Number of Principal Components Varies due to the Size of the Training Sets.

4.2 Experiment Results

In the following sections, we will be presenting the results from each experiment that will help us in drawing conclusions on the relationship between PPG and physiological parameters. Moreover, through these results we can evaluate the data with regards to its overall quality and information that it carries. Lastly, we are presenting the models with the highest performance in each experiment, along with their training characteristics and designs.

4.2.1 Experiment 1

In Table 4.2 the binary classification results on Experiment 1 are portrayed. For simplicity, yet representative outlook, only the best performing model from each model type is presented on the Table. Only ML model types 1-3 from Table 3.3 are included, as the ML model of type 4 is used for regression i.e Experiment 6 only. In the same table, in each row representing a metric, the highest value is highlighted in red, while the lowest value is highlighted in yellow. This allows for the comparisons between different models. In Figure 4.3, key metrics are illustrated graphically allowing us to pinpoint that the best performing classifier for this experiment is the CNN, using IR ADC signals to form the input data.

Experiment 1

Metrics	**Baseline using IR ADC**	**PCA using IR ADC**	**CNN using IR ADC**	**Baseline using Pleth**	**PCA using Pleth**	**CNN using Pleth**
5 Fold Cross Validation Test Accuracies	0.875	0.725	0.975	0.875	0.750	0.975
	0.900	0.750	0.950	0.875	0.825	0.875
	0.800	0.850	0.900	0.950	0.750	0.875
	0.925	0.900	0.860	0.800	0.725	0.925
	0.800	0.875	0.860	0.925	0.800	0.875
Min Test Accuracy	0.800	0.725	0.860	0.800	0.725	0.875
Max Test Accuracy	0.925	0.900	0.975	0.950	0.825	0.975
Test Accuracy Average	0.860	0.820	0.909	0.885	0.770	0.905
Test Accuracy St. Dev.	0.051	0.070	0.047	0.051	0.037	0.040
Recall Average	0.860	0.815	0.909	0.885	0.77	0.901
Precision Average	0.869	0.820	0.912	0.892	0.774	0.904
F1 Score Average	0.864	0.817	0.910	0.888	0.772	0.902

Table 4.2: Experiment 1 Numeric Model Results. The Best Performing Models for Each Defined Model Type (1-3) are Presented, for IR ADC-based Spectrogram Inputs and Pleth-based Spectrogram Inputs.

The worst performing model for this experiment is the PCA model, using Pleth as an input signal. The CNN model outperform PCA by almost 14% in the KFCV test accuracy average.

It is interesting to note that the CNN model outperformed all of the other models, with the second best performing model being the CNN using Pleth signals as an input to the model. The best performing model also registered the highest scores on precision, recall and F1 score metrics, while the standard deviation on the test accuracy arising from the KFCV, was the 3rd lowest out of the 6 different approaches. Overall, the chosen CNN model successfully classified signals of patients admitted with and without dengue shock, with high sensitivity and precision of 90.9% and 91.2% respectively. Interestingly, CNNs using both Pleth and IR ADC signals as inputs, scored the highest maximum accuracy recorded.

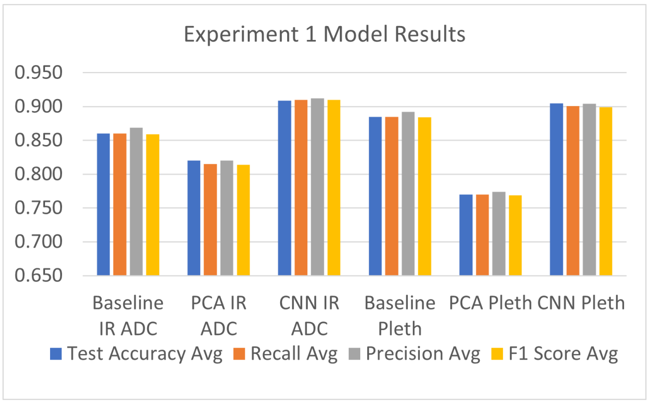


Figure 4.3: Visualising Key Experiment 1 Results.

On Figure 4.5a, the CNN model's architecture is presented, while in Figure 4.4 we can visualise it more intuitively by looking at how the data propagates through the convolutional, pooling and fully connected layers. However, in the latter figure, features such as dropout and batch normalisation are not included in the illustration. For the training of the model, the Adam optimiser with learning rate of $3e-4$ was found to be the best performing. The network was trained for 200 epochs with a batch size of 16, while the validation set was defined as 20% of the training set. The resulting training and validation curves can be observed in Figure 4.5b. The curves show effective learning with validation and training components moving closely to each other and no overfitting is observed. The curves seem to converge to an optimum at the end of training. Overfitting was visible in longer epochs and in cases were no regularisation, i.e. Dropout layers, was introduced.

The best performing model of this experiment was also tested on other patient data, foreign to the training and test sets. This was not part of the design procedure but due to the resulting performance of the models, a further test was completed. Three patients who were admitted with shock were chosen and selected signals were passed through the saved classifier. For each patient, two tests were ran. One was using signals of 1 hour length extracted only 10 minutes after the start of the PPG record and the second one was using the same length of signals, but this time extracted 3 hours after the PPG record's start. The results are illustrated in Table 4.3. From that, it is obvious that the classifier performed well on the foreign data, but it is also obvious that signals that were extracted only 10 minutes after the PPG record's start, were classified far more accurately than signals extracted 3 hours after the PPG record's start.

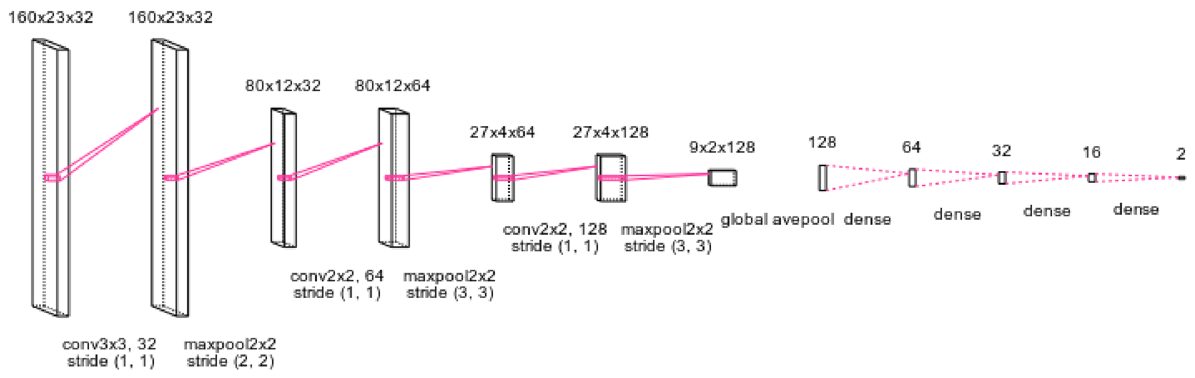
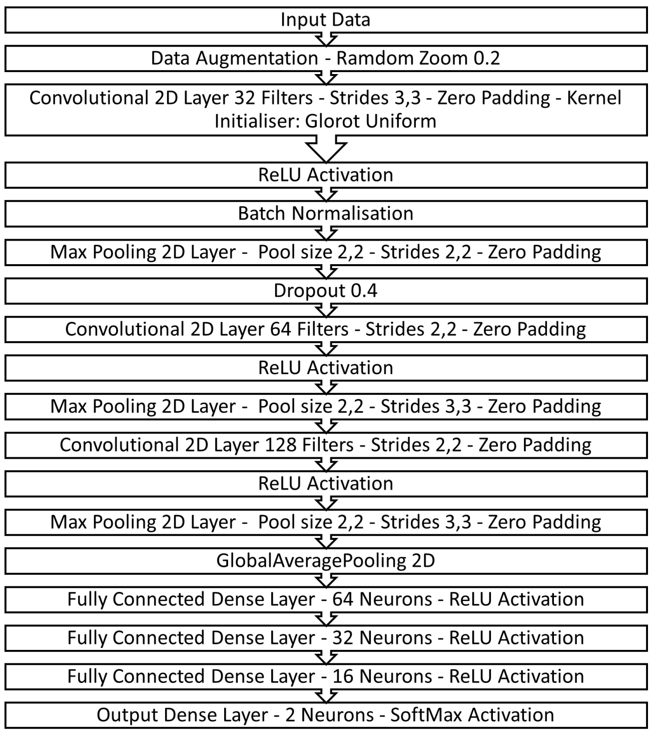


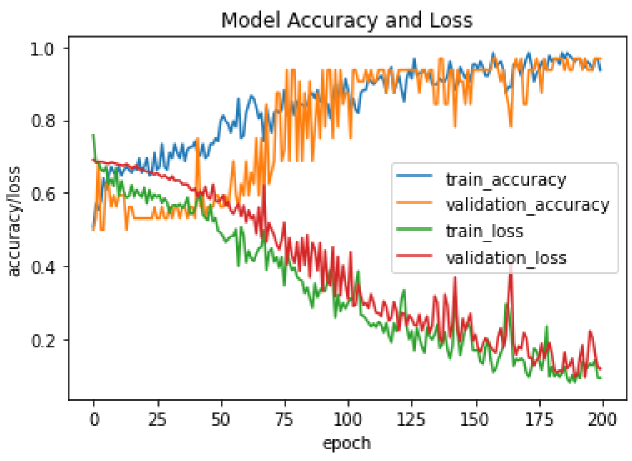
Figure 4.4: Visualising Experiment 1's Best Performing CNN Architecture.

Patient ID	**Time After PPG Record Start**	**Duration of Signal Tested**	**Accuracy of Best Performing Model**
003-2109	10 minutes	1 hour split into 50 Spectrograms	0.89
	3 hours	1 hour split into 50 Spectrograms	0.68
003-2110	10 minues	1 hour split into 50 Spectrograms	0.85
	3 hours	1 hour split into 50 Spectrograms	0.73
003-2012	10 minutes	1 hour split into 50 Spectrograms	0.86
	3 hours	1 hour split into 50 Spectrograms	0.76

Table 4.3: Experiment 1's Best Classifier, Tested on Other Patient Data.


Input Data
Data Augmentation - Ramdom Zoom 0.2
Convolutional 2D Layer 32 Filters - Strides 3,3 - Zero Padding - Kernel Initialiser: Glorot Uniform
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 64 Filters - Strides 2,2 - Zero Padding
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 3,3 - Zero Padding
Convolutional 2D Layer 128 Filters - Strides 2,2 - Zero Padding
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 3,3 - Zero Padding
GlobalAveragePooling 2D
Fully Connected Dense Layer - 64 Neurons - ReLU Activation
Fully Connected Dense Layer - 32 Neurons - ReLU Activation
Fully Connected Dense Layer - 16 Neurons - ReLU Activation
Output Dense Layer - 2 Neurons - SoftMax Activation

(a)



(b)

Figure 4.5: Experiment 1: (a) Best Performing Model's Architecture - CNN using IR ADC and (b) Example Training and Validation Curves.

4.2.2 Experiment 2

In Table 4.4 the binary classification results on Experiment 2 are illustrated, presenting the best performing models from each defined ML model type (1-3 from Table 3.3). Again, the highest value in each metric is highlighted in red, while the lowest value is highlighted in yellow. In Figure 4.6, the graphical representation of this data enables us to visualise and compare the performance of different models. In the aforementioned table and figure, it might not be clearly evident which model performs the best as both the CNN model using IR ADC signals and the PCA model using Pleth signals perform equally well on the test set accuracy average and the recall average. However, the CNN's higher F1 score, due to the higher precision of the model, and the fact that it has the lowest standard deviation in the test accuracy out of all models, suggests that the CNN model using IR ADC signals, performs the best out of all models. The worst performing model for this binary classification is the Baseline model using IR ADC signals, which lacks behind the best performing model by 4% in the test accuracy average and by approximately the same amount in recall, precision and F1 score. Interestingly, but not importantly, the PCA model using Pleth resulted in the highest maximum test accuracy of 97.5%, along with the CNN model using Pleth signals as an input.

The overall model performance is exceptional in detecting pre-shock and post-shock windows, with very high precision and sensitivity scores and a low standard deviation.

Experiment 2

Metrics	**Baseline using IR ADC**	**PCA using IR ADC**	**CNN using IR ADC**	**Baseline using Pleth**	**PCA using Pleth**	**CNN using Pleth**
5 Fold Cross Validation Test Accuracies	0.925	0.950	0.900	0.925	0.875	0.875
	0.900	0.900	0.900	0.900	0.900	0.900
	0.850	0.850	0.900	0.925	0.975	0.875
	0.825	0.900	0.925	0.850	0.925	0.975
	0.875	0.925	0.950	0.925	0.900	0.925
Min Test Accuracy	0.825	0.850	0.900	0.850	0.875	0.875
Max Test Accuracy	0.925	0.950	0.950	0.925	0.975	0.975
Test Accuracy Average	0.875	0.905	0.915	0.905	0.915	0.910
Test Accuracy St. Dev.	0.035	0.033	0.020	0.029	0.034	0.037
Recall Average	0.875	0.905	0.915	0.905	0.915	0.911
Precision Average	0.882	0.909	0.921	0.908	0.920	0.912
F1 Score Average	0.878	0.907	0.918	0.906	0.917	0.911

Table 4.4: Experiment 2 Numeric Model Results. The Best Performing Models for Each Defined Model Type (1-3) are Presented, for IR ADC-based Spectrogram Inputs and Pleth-based Spectrogram Inputs.

In Figure 4.7a the structure of the best performing CNN model using IR ADC signals can be seen. This model was trained for 150 epochs using Adam as the optimizer, with a learning rate of $3e-3$, batch size of 32 and a validation size of 20% of the training data. Example training and validation curves can be seen in Figure 4.7b, where adequate learning can be observed. The validation curves follow the movement of the training curves closely. Some overshooting can be observed in the validation set cures but this is expected given the limited data. Nevertheless, no overfitting is observed and the learning seems to be converging adequately to an optimum.

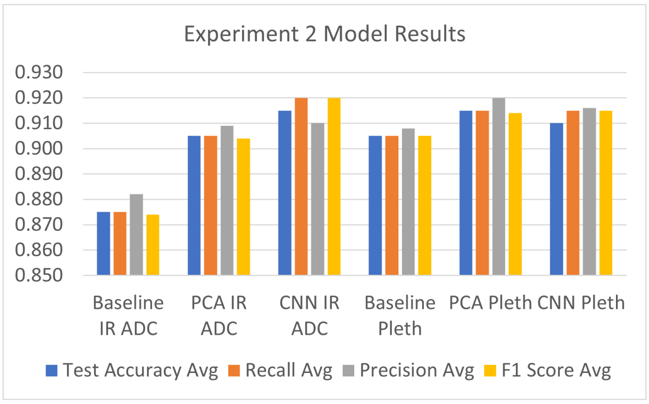
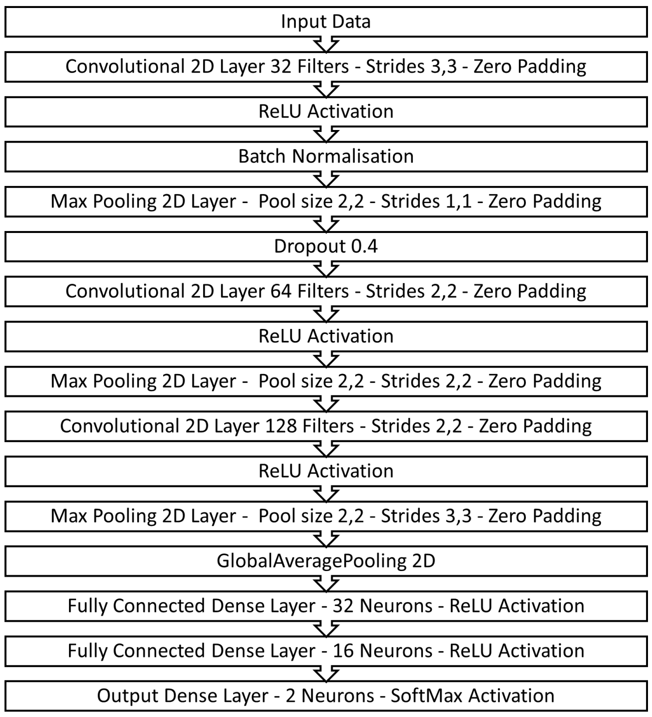
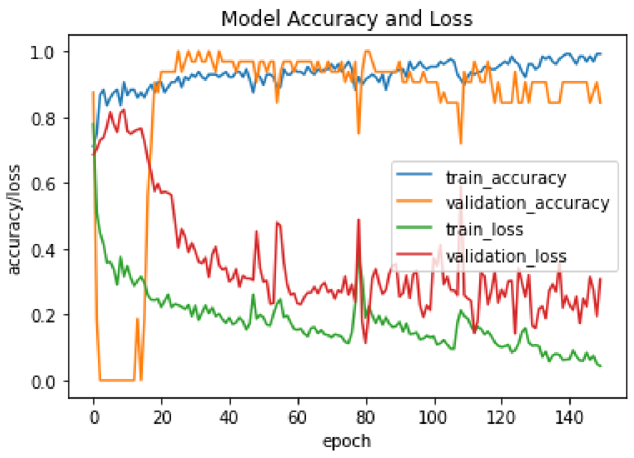


Figure 4.6: Visualising Key Experiment 2 Results.


Input Data
Convolutional 2D Layer 32 Filters - Strides 3,3 - Zero Padding
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 1,1 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 64 Filters - Strides 2,2 - Zero Padding
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Convolutional 2D Layer 128 Filters - Strides 2,2 - Zero Padding
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 3,3 - Zero Padding
GlobalAveragePooling 2D
Fully Connected Dense Layer - 32 Neurons - ReLU Activation
Fully Connected Dense Layer - 16 Neurons - ReLU Activation
Output Dense Layer - 2 Neurons - SoftMax Activation

(a)



(b)

Figure 4.7: Experiment 2: (a) Best Performing Model's Architecture - CNN using IR ADC and (b) Example Training and Validation Curves.

4.2.3 Experiment 3

For the third experiment, the overall model performance for this binary classification was much poorer; yet successful. Results can be seen in Table 4.5 and those can be visualised better in Figure 4.8, in similar fashion to the results presented for Experiment 1 and 2. It is evident that again, CNN using IR ADC signals performed the best, with test accuracies as high as 96.8%. The CNN's average test accuracy is at 83.7%, outperforming that of the second best model's, by 3.8%. The same pattern can be seen on recall, precision and F1 score metrics, outperforming the rest. The standard deviation of this model was the 4th lowest out of the 6 models presented on the results table. The worst performing model is again the Baseline model using IR ADC signals as an input. This model scored 21.9% points worse than the best performing CNN model and also registered the highest standard deviation in KFCV test accuracies out of all models. At 84.2% F1 Score Average, this model was able to classify pre-clinical shock signals of a patient admitted with shock and post-clinical shock signals of the same patient.

The CNN architecture for the best performing model can be seen in Figure 4.9a, while sample validation and training curves can be seen in Figure 4.9b. The model was trained at 700 epochs using the Adam optimizer with a learning rate of $3e-4$, batch size of 23 and a validation set consisting of 20% of the training data. The training and validation curves seem to be converging adequately and no continuous overfitting is observed. However, the validation curve seems to overshoot at many points and looks more noisy, even after aggressive regularisation is added to the model.

Experiment 3

Metrics	**Baseline using IR ADC**	**PCA using IR ADC**	**CNN using IR ADC**	**Baseline using Pleth**	**PCA using Pleth**	**CNN using Pleth**
5 Fold Cross Validation Test Accuracies	0.548	0.742	0.806	0.645	0.806	0.774
	0.581	0.774	0.839	0.806	0.806	0.645
	0.548	0.613	0.839	0.677	0.839	0.871
	0.613	0.742	0.968	0.710	0.710	0.839
	0.800	0.700	0.733	0.833	0.733	0.867
Min Test Accuracy	0.548	0.613	0.733	0.645	0.710	0.645
Max Test Accuracy	0.800	0.774	0.968	0.833	0.839	0.871
Test Accuracy Average	0.618	0.714	0.837	0.734	0.779	0.799
Test Accuracy St. Dev.	0.094	0.056	0.076	0.073	0.049	0.084
Recall Average	0.617	0.713	0.837	0.735	0.780	0.798
Precision Average	0.632	0.723	0.847	0.746	0.798	0.821
F1 Score Average	0.624	0.718	0.842	0.740	0.789	0.809

Table 4.5: Experiment 3 Numeric Model Results. The Best Performing Models for Each Defined Model Type (1-3) are Presented, for IR ADC-based Spectrogram Inputs and Pleth-based Spectrogram Inputs.

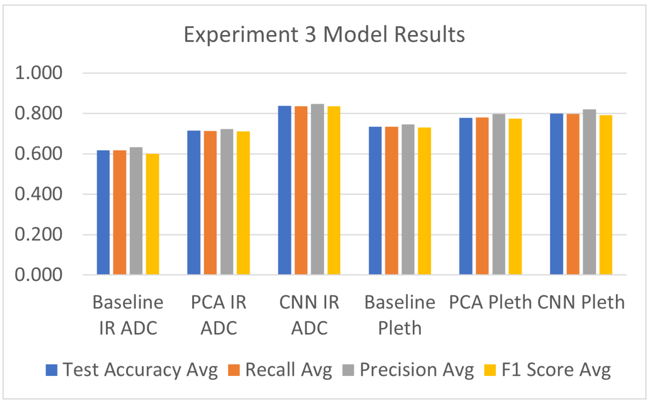
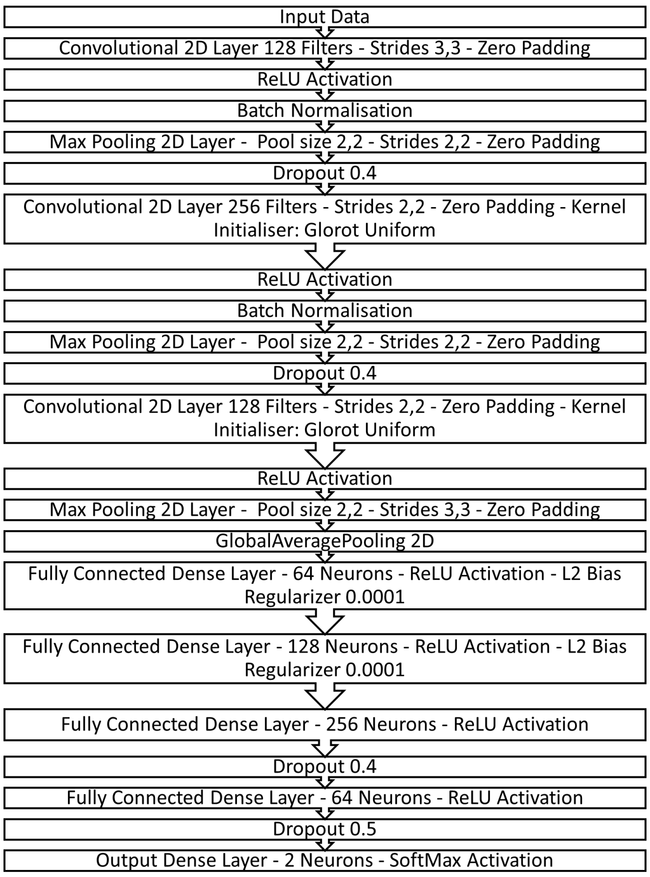
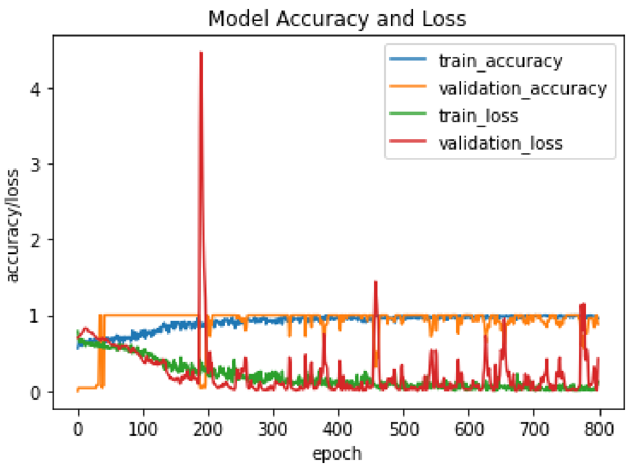


Figure 4.8: Visualising Key Experiment 3 Results.


Input Data
Convolutional 2D Layer 128 Filters - Strides 3,3 - Zero Padding
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 256 Filters - Strides 2,2 - Zero Padding - Kernel Initialiser: Glorot Uniform
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 128 Filters - Strides 2,2 - Zero Padding - Kernel Initialiser: Glorot Uniform
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 3,3 - Zero Padding
GlobalAveragePooling 2D
Fully Connected Dense Layer - 64 Neurons - ReLU Activation - L2 Bias Regularizer 0.0001
Fully Connected Dense Layer - 128 Neurons - ReLU Activation - L2 Bias Regularizer 0.0001
Fully Connected Dense Layer - 256 Neurons - ReLU Activation
Dropout 0.4
Fully Connected Dense Layer - 64 Neurons - ReLU Activation
Dropout 0.5
Output Dense Layer - 2 Neurons - SoftMax Activation

(a)



(b)

Figure 4.9: Experiment 3: (a) Best Performing Model's Architecture - CNN using IR ADC and (b) Example Training and Validation Curves.

4.2.4 Experiment 4

Experiment 4 is a multi-class classification problem, onto the combination of classes presented in Experiments 1 and 2. The results are portrayed in the same way as in previous sections, in Table 4.6 and graphically on Figure 4.10. From those, it is evident that the Baseline model using Pleth signal inputs, outperformed the previously dominant model which was a CNN using IR ADC signals. Specifically the Baseline model, scored 88% average test accuracy, surpassing the second best model by 1.2% points. Even though the results of all models were surprisingly close, the Baseline Model using Pleth outperformed all the rest on recall, precision and F1 score averages. The worst performing model in this case is the PCA model using Pleth signals, with an average test accuracy of 83.1% and F1 Score of 82.3%. While the overall classification performance was lower than that of individual performance presented in Experiments 1 and 2, the results are very promising, with high sensitivity and precision.

Experiment 4

Metrics	**Baseline using IR ADC**	**PCA using IR ADC**	**CNN using IR ADC**	**Baseline using Pleth**	**PCA using Pleth**	**CNN using Pleth**
5 Fold Cross Validation Test Accuracies	0.900	0.837	0.801	0.837	0.825	0.913
	0.800	0.813	0.938	0.925	0.837	0.838
	0.887	0.875	0.850	0.825	0.855	0.875
	0.887	0.813	0.900	0.900	0.825	0.887
	0.875	0.887	0.850	0.913	0.813	0.825
Min Test Accuracy	0.800	0.813	0.801	0.825	0.813	0.825
Max Test Accuracy	0.900	0.887	0.938	0.925	0.855	0.913
Test Accuracy Average	0.870	0.845	0.868	0.880	0.831	0.867
Test Accuracy St. Dev.	0.036	0.031	0.047	0.041	0.014	0.032
Recall Average	0.870	0.845	0.870	0.880	0.815	0.867
Precision Average	0.879	0.854	0.865	0.895	0.832	0.865
F1 Score Average	0.874	0.849	0.867	0.887	0.823	0.867

Table 4.6: Experiment 4 Numeric Model Results. The Best Performing Models for Each Defined Model Type (1-3) are Presented, for IR ADC-based Spectrogram Inputs and Pleth-based Spectrogram Inputs.

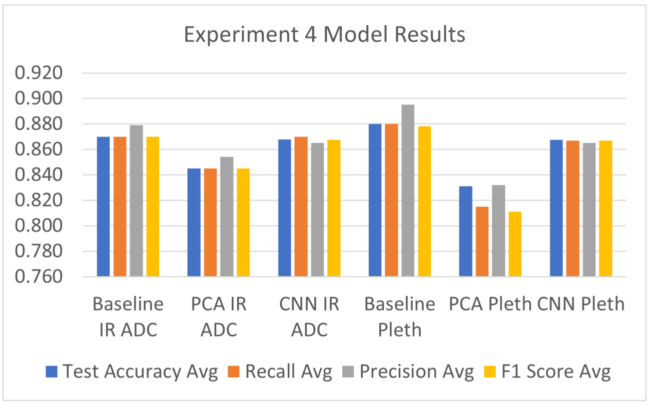
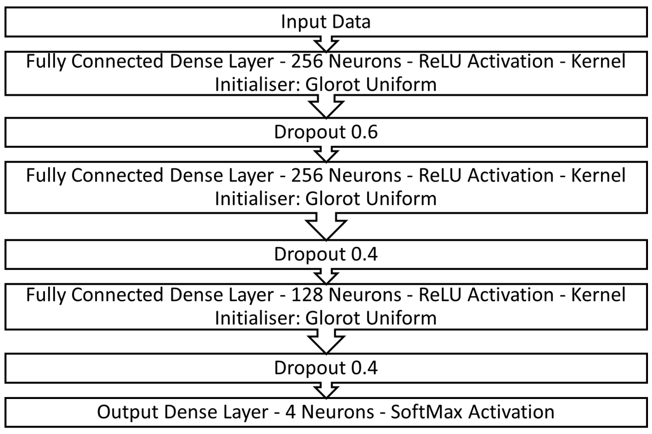
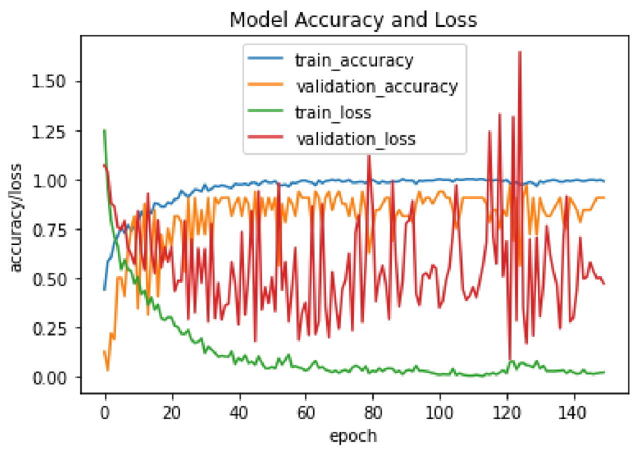


Figure 4.10: Visualising Key Experiment 4 Results.

The structure of the best baseline NN model for this experiment is much simpler than previously seen CNN structures. This is presented in Figure 4.11a. The model was trained with 150 Epochs, a learning rate of $1e-3$ using the Adam optimizer and a validation set consisting of 10% of the training data. The smaller validation set proved to be yielding better test results in this model. The validation curves follow the movement of the training curves implying that the model is learning, but the validation loss seems to be converging less effectively than the validation accuracy. Moreover, it is very noisy even though no clear overfitting trend is observed. This can be attributed to a non-representative validation set due to its size, causing it to fluctuate more than expected.


Input Data
Fully Connected Dense Layer - 256 Neurons - ReLU Activation - Kernel Initialiser: Glorot Uniform
Dropout 0.6
Fully Connected Dense Layer - 256 Neurons - ReLU Activation - Kernel Initialiser: Glorot Uniform
Dropout 0.4
Fully Connected Dense Layer - 128 Neurons - ReLU Activation - Kernel Initialiser: Glorot Uniform
Dropout 0.4
Output Dense Layer - 4 Neurons - SoftMax Activation

(a)



(b)

Figure 4.11: Experiment 4: (a) Best Performing Model's Architecture - Baseline using Pleth and (b) Example Training and Validation Curves.

4.2.5 Experiment 5

The results for this multi-class classification experiment are laid out in Table 4.7 and presented graphically on Figure 4.12. Not surprisingly, the best performing model is once again the CNN with IR ADC signals as an input. The average test accuracy of this model is at 76.8%, only 0.2% higher than the second best performing model which is a Baseline model using IR ADC signals. However, the best performing model has a much higher average F1 score and that is why it stood out from the rest. The model that scored the lowest average accuracy on the test set is the PCA model with Pleth signals as an input. However, the lowest F1 score was registered by the CNN model using Pleth signals. The best performing model managed to classify the 6 classes adequately yet, Figure 4.13 shows two sample confusion matrices, outlining perfectly the character of the majority of the tests that were carried out. From those it is evident that most misclassifications occurred on Class 5, that is window segments corresponding to a post clinical shock signal of a patient that was already admitted with dengue shock. These windows were falsely classified as either post-clinical shock signals of a patient that was admitted with no shock or the pre-clinical shock signals of a patient that was admitted with shock. Interestingly, this pattern was observed in all model types and most tests that were ran for this experiment.

Experiment 5

Metrics	**Baseline using IR ADC**	**PCA using IR ADC**	**CNN using IR ADC**	**Baseline using Pleth**	**PCA using Pleth**	**CNN using Pleth**
5 Fold Cross Validation Test Accuracies	0.720	0.710	0.785	0.688	0.742	0.753
	0.796	0.839	0.731	0.753	0.699	0.742
	0.761	0.783	0.739	0.728	0.728	0.783
	0.750	0.674	0.794	0.772	0.728	0.750
	0.804	0.772	0.794	0.804	0.707	0.783
Min Test Accuracy	0.720	0.674	0.731	0.688	0.699	0.742
Max Test Accuracy	0.804	0.839	0.794	0.804	0.742	0.783
Test Accuracy Average	0.766	0.755	0.768	0.749	0.721	0.762
Test Accuracy St. Dev.	0.031	0.058	0.028	0.039	0.016	0.017
Recall Average	0.745	0.755	0.769	0.749	0.722	0.762
Precision Average	0.763	0.772	0.781	0.747	0.725	0.664
F1 Score Average	0.754	0.763	0.775	0.748	0.723	0.710

Table 4.7: Experiment 5 Numeric Model Results. The Best Performing Models for Each Defined Model Type (1-3) are Presented, for IR ADC-based Spectrogram Inputs and Pleth-based Spectrogram Inputs.

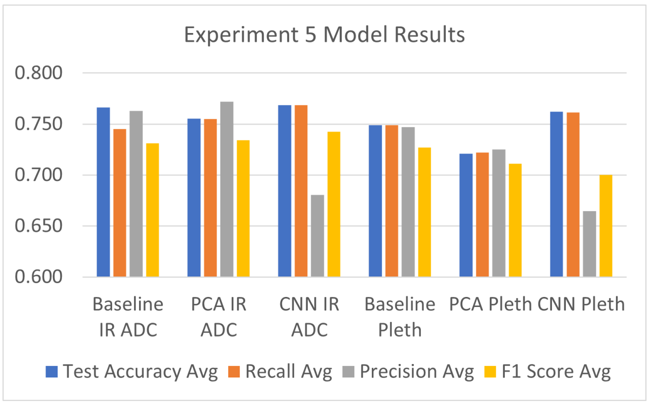
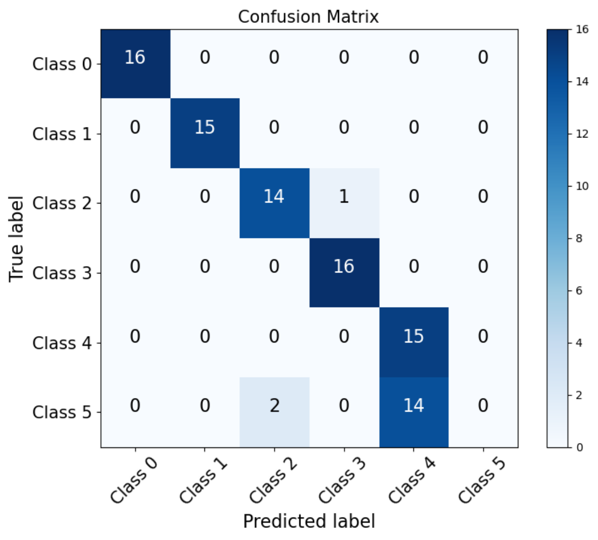
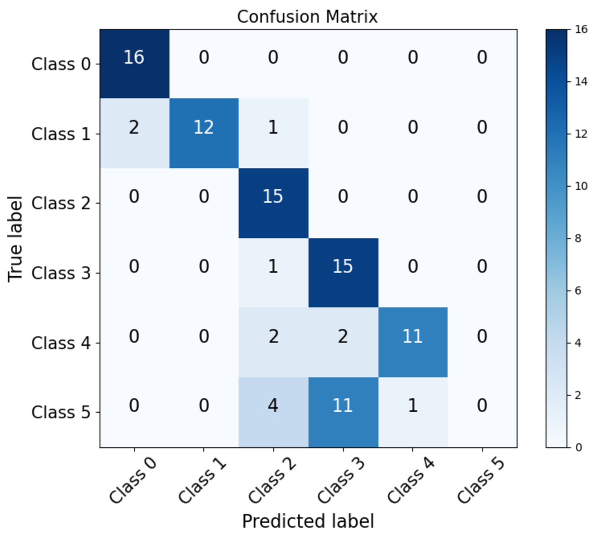


Figure 4.12: Visualising Key Experiment 5 Results.



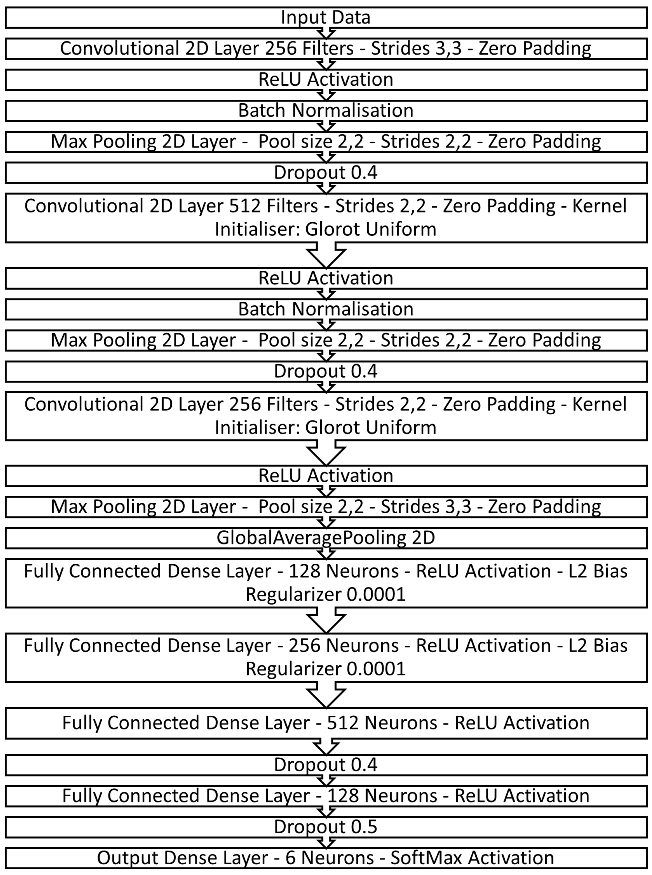
(a)



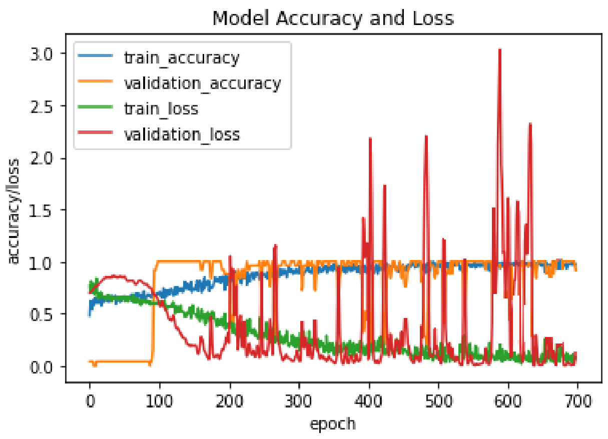
(b)

Figure 4.13: Sample Confusion Matrices from Experiment 5 Portraying Important Classification Characteristics. Confusion Matrices were Obtained in Different Folds of the Best Performing Model. For Class Descriptions please refer to Table 4.1.

The best performing model's CNN architecture is presented in Figure 4.14a. This model was trained on 700 epochs, with the Adam optimizer and a learning rate of $3e-4$. A batch-size of 23 was used and the validation set was 20% of the training data. An example of the training curves are presented in Figure 4.14b. Even though aggressive regularisation was added throughout the model architecture as seen in Figure 4.14a, the validation loss curve seems to be overshooting at multiple points. This is a similar behaviour to that observed in Experiment 3 training curves, but the magnitude of the overshoot is much higher in this case. This can be attributed to an unrepresentative validation set, due to the small dataset used. Nevertheless, the model seems to be converging effectively, with the overall movement in the validation set matching that of the training set. The result is a high train and validation accuracy and low train and validation loss.


Input Data
Convolutional 2D Layer 256 Filters - Strides 3,3 - Zero Padding
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 512 Filters - Strides 2,2 - Zero Padding - Kernel Initialiser: Glorot Uniform
ReLU Activation
Batch Normalisation
Max Pooling 2D Layer - Pool size 2,2 - Strides 2,2 - Zero Padding
Dropout 0.4
Convolutional 2D Layer 256 Filters - Strides 2,2 - Zero Padding - Kernel Initialiser: Glorot Uniform
ReLU Activation
Max Pooling 2D Layer - Pool size 2,2 - Strides 3,3 - Zero Padding
GlobalAveragePooling 2D
Fully Connected Dense Layer - 128 Neurons - ReLU Activation - L2 Bias Regularizer 0.0001
Fully Connected Dense Layer - 256 Neurons - ReLU Activation - L2 Bias Regularizer 0.0001
Fully Connected Dense Layer - 512 Neurons - ReLU Activation
Dropout 0.4
Fully Connected Dense Layer - 128 Neurons - ReLU Activation
Dropout 0.5
Output Dense Layer - 6 Neurons - SoftMax Activation

(a)



(b)

Figure 4.14: Experiment 5: (a) Best Performing Model's Architecture - CNN using IR ADC and (b) Example Training and Validation Curves.

4.2.6 Experiment 6

For Experiment 6, a regression LSTM model was built and tested. The output of this regression model was then inputted into a pre-trained classifier and classification results on the regression output were recorded. These are presented in Figure 4.8. Two different models were used as classifiers, suggesting that our LSTM models received two different inputs. As described in earlier sections, one was the raw spectrogram windows while the other was the PCA transformed spectrogram windows. This is distinguished in the aforementioned figure. Furthermore, for each of the classification models, both IR ADC and Pleth signals were tested and results were recorded. The training set consisted of total of 761 spectrogram windows, where only 77 of them belonged to the post clinical shock class. It is important to note that the Recall, Precision and F1 Score averages included in the table, refer to the macro-average scores, accounting for the imbalanced dataset. The macro average was chosen, as performance in both classes is equally important. However, the test accuracy averages portray the micro-average and that is why the score is significantly higher.

No values are highlighted in this Table as the macro-average scores presented, all lie between 45% and 55%. In a binary classifier, this performance is considered poor [15]. However, accounting for data imbalances, if we only look at the class that is mostly represented in the training data, that is the pre-clinical shock windows of a patient admitted with shock, we can see that the classifier's performance is good with the best model scoring an average accuracy of 99%. When combining the two classes, this accuracy falls to 89%, as seen on the results table, due to the misclassifications of the second class.

The architecture of a sample LSTM network used for extracting the results is seen in Figure 4.15. The network was optimised using average test accuracies, even though they are not representative of the whole data. That was done for the sake of experimentation and due to the lack of data. The network was trained on a batch size of 80, a window length of 16 and an Adam optimizer with a learning rate of 0.001. The model was trained for a total of 300 epochs.

Experiment 6	***PCA of 80 Components as LSTM Input, using PCA Classification Model as Classifier***		***Full-Length Spectrograms as LSTM Input using Baseline Classification Model as Classifier***	
	IR ADC	**Pleth**	**IR ADC**	**Pleth**
5 Fold Cross Validation Test Accuracies	0.870	0.830	0.770	0.890
	0.840	0.870	0.850	0.780
	0.750	0.860	0.770	0.880
	0.710	0.890	0.840	0.830
	0.850	0.870	0.830	0.830
Min Test Accuracy	0.710	0.830	0.770	0.780
Max Test Accuracy	0.870	0.890	0.850	0.890
Test Accuracy Average	0.804	0.864	0.812	0.842
Test Accuracy St. Dev.	0.062	0.020	0.035	0.040
Macro-Average Recall Average	0.500	0.550	0.510	0.530
Macro-Average Precision Average	0.450	0.520	0.490	0.520
Macro-Average F1 Score Average	0.474	0.535	0.500	0.525

Table 4.8: LSTM Classification Model Results (Averages of Macro-Averaged classes). On the Left Column the Results from the PCA-based Implementation are Portrayed using 80 Components to Construct the Input Feature Vectors. On the Right Column, Classification Results are Portrayed for the Full-length Spectrogram Feature Vectors. The Results Portray the Classification Performance on the Regression Output from the LSTM Model.

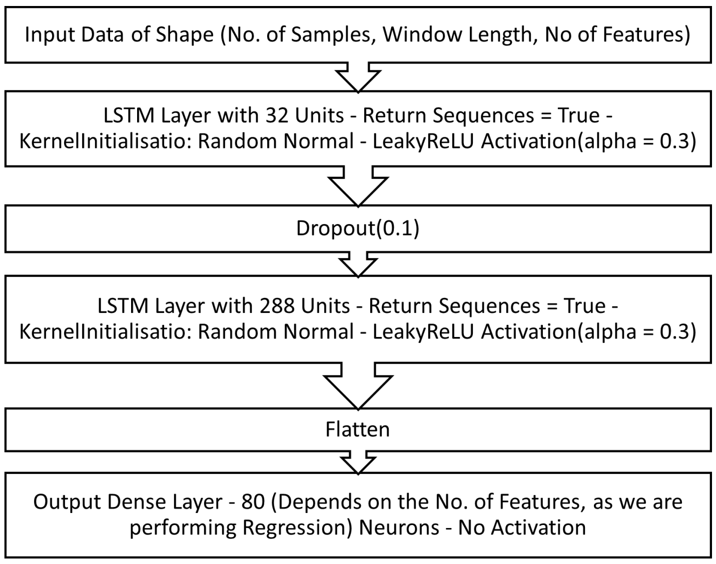

Input Data of Shape (No. of Samples, Window Length, No of Features)
LSTM Layer with 32 Units - Return Sequences = True - KernelInitialisatio: Random Normal - LeakyReLU Activation(alpha = 0.3)
Dropout(0.1)
LSTM Layer with 288 Units - Return Sequences = True - KernelInitialisatio: Random Normal - LeakyReLU Activation(alpha = 0.3)
Flatten
Output Dense Layer - 80 (Depends on the No. of Features, as we are performing Regression) Neurons - No Activation

Figure 4.15: Sample LSTM Model Architecture.

4.3 IR ADC vs Pleth

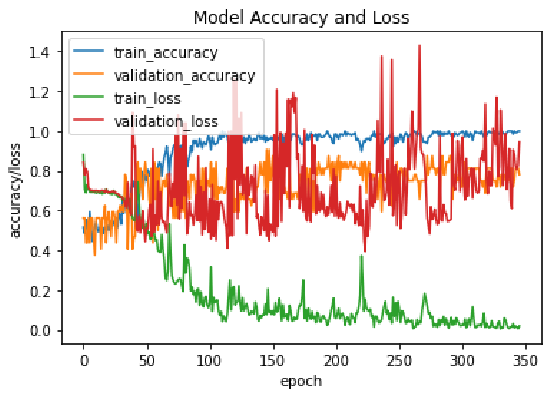
In 4 out 5 classification experiments, CNNs with IR ADC signals as an input were chosen as the best models. However, if we take the top 3 models for each experiment for all 5 classification experiments, then 8 out of the 15 best performing models were using Pleth signals as an input. Another interesting statistic is that when comparing Baseline models,in 4 out 5 Experiments, using the Pleth signals resulted in a better performance than when using IR ADC signals. When comparing PCA models though, in 3 out of 5 experiments, IR ADC outperformed Pleth signals. For CNN models, IR ADC signals outperformed Pleth signals in 5 out of 5 Experiments. Finally, in multi-class classification experiments, the performance of models using Pleth signals decreased by a proportionately higher amount compared to the performance of IR ADC models.

4.4 ML Model Overview

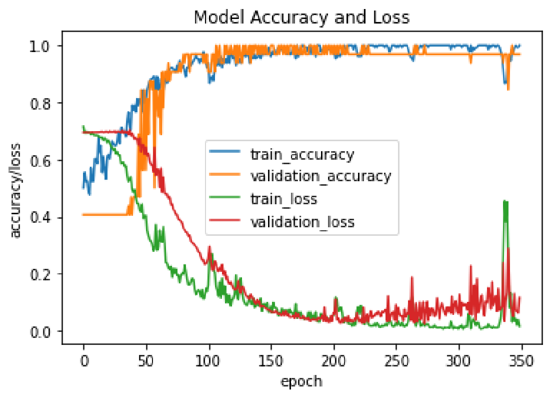
The ML models were designed and built successfully, with the grid-search methodology employed in optimizing them proving to be effective.

In general, Dropout and Regularisation layers proved to be highly needed in order to reduce overfitting in the data and enabled the models to perform adequately well in the test sets, consistently. These model features were used in all model types and designs. In Figure 4.16, the effect of Dropout and L2 Bias Regularisation are illustrated, where the curves on the right appear to be following the training curves more closely [15]. Specifically, this relationship is most important for the validation loss curve. Another important feature observed in all models, was kernel initialisation. This allowed for learning to take place more effectively, often requiring less epochs. Lastly, features such as Data Augmentation were found to be corrupting our inputs with too much noise, leading to poor performance overall. There were some rare exceptions, such as in Experiment 1, where a random rotation of 0.2 resulted in improved performance of the CNN model.

The baseline model performed surprisingly well on the raw spectrograms of both Pleth and IR ADC overall, and was even chosen as the best performing model for Experiment 4. Baseline architectures were simpler compared to those of CNN and on average, inference times were lower. ML model type 2, involving PCA, performed the worst overall when compared with the other two models, except in Experiment 2 and 3, where it performed better than the baseline model. The CNN models had the longest and most complex structures, particularly in the feature extraction section of the network. Nevertheless, CNNs proved powerful even with



(a) Before Regularisation - Overfitting Present.

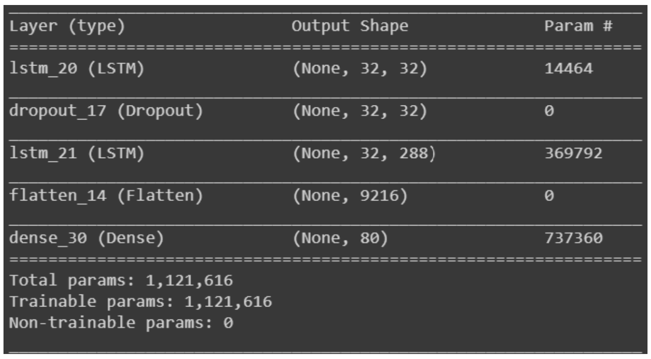


(b) After Regularisation - No Overfitting Present.

Figure 4.16: The Effect of Regularization on Model Training.

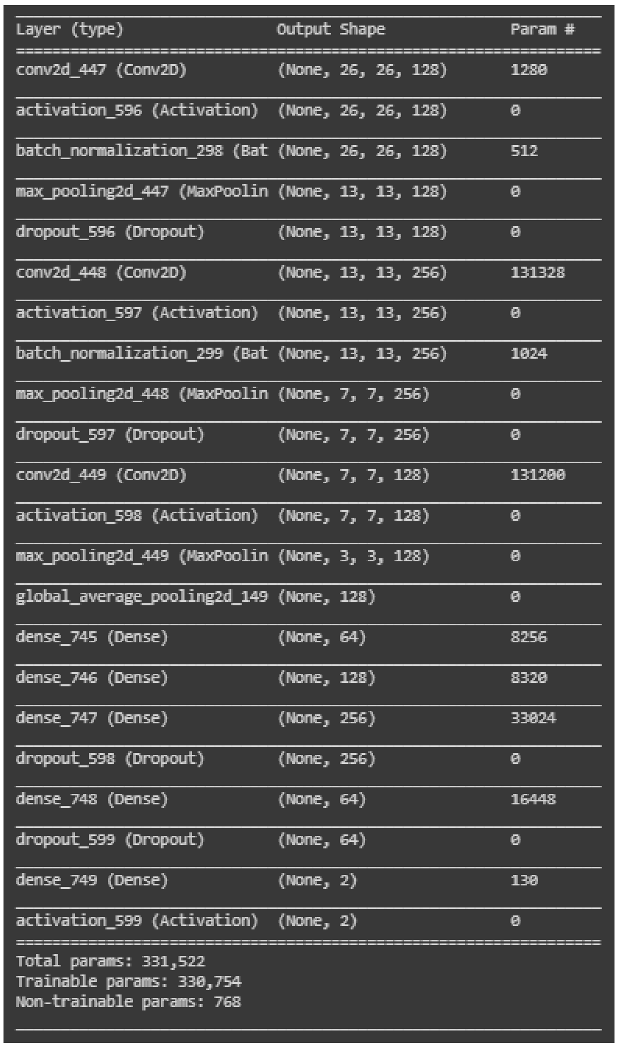
the few data points we possess. Lastly, looking at the LSTM model and Experiment 6, the performance was very poor overall when looking at the macro averaged scores, but not when looking at the dominant class only.

For the sake of completeness, Figures 4.17 and 4.18 present example model summaries of all model types explored in this study, as defined in Table 3.3. Particular attention should be paid to the number of trainable parameters per model type.



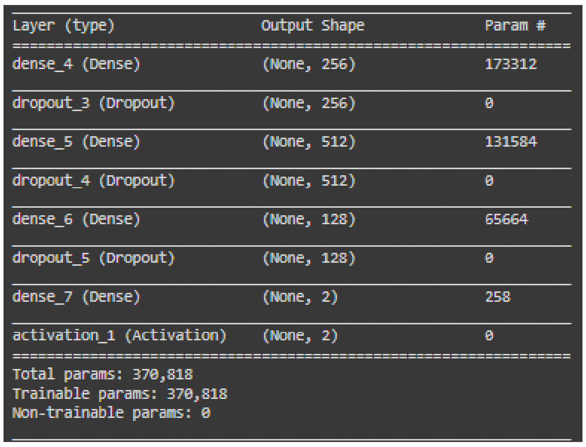
```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_20 (LSTM)               (None, 32, 32)            14464
_________________________________________________________________
dropout_17 (Dropout)         (None, 32, 32)            0
_________________________________________________________________
lstm_21 (LSTM)               (None, 32, 288)           369792
_________________________________________________________________
flatten_14 (Flatten)         (None, 9216)              0
_________________________________________________________________
dense_30 (Dense)             (None, 80)                737360
=================================================================
Total params: 1,121,616
Trainable params: 1,121,616
Non-trainable params: 0
```

Figure 4.17: Sample Regression LSTM Model Summary.



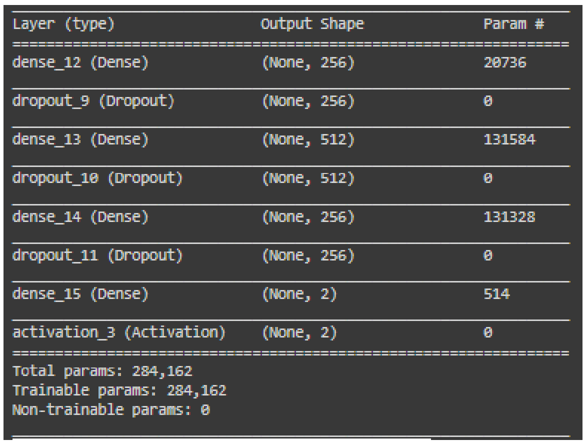
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_447 (Conv2D)          (None, 26, 26, 128)       1280
_________________________________________________________________
activation_596 (Activation)  (None, 26, 26, 128)       0
_________________________________________________________________
batch_normalization_298 (Bat (None, 26, 26, 128)       512
_________________________________________________________________
max_pooling2d_447 (MaxPoolin (None, 13, 13, 128)       0
_________________________________________________________________
dropout_596 (Dropout)        (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_448 (Conv2D)          (None, 13, 13, 256)       131328
_________________________________________________________________
activation_597 (Activation)  (None, 13, 13, 256)       0
_________________________________________________________________
batch_normalization_299 (Bat (None, 13, 13, 256)       1024
_________________________________________________________________
max_pooling2d_448 (MaxPoolin (None, 7, 7, 256)         0
_________________________________________________________________
dropout_597 (Dropout)        (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_449 (Conv2D)          (None, 7, 7, 128)         131200
_________________________________________________________________
activation_598 (Activation)  (None, 7, 7, 128)         0
_________________________________________________________________
max_pooling2d_449 (MaxPoolin (None, 3, 3, 128)         0
_________________________________________________________________
global_average_pooling2d_149 (None, 128)               0
_________________________________________________________________
dense_745 (Dense)            (None, 64)                8256
_________________________________________________________________
dense_746 (Dense)            (None, 128)               8320
_________________________________________________________________
dense_747 (Dense)            (None, 256)               33024
_________________________________________________________________
dropout_598 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_748 (Dense)            (None, 64)                16448
_________________________________________________________________
dropout_599 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_749 (Dense)            (None, 2)                 130
_________________________________________________________________
activation_599 (Activation)  (None, 2)                 0
=================================================================
Total params: 331,522
Trainable params: 330,754
Non-trainable params: 768
_________________________________________________________________
```

(a) Sample CNN Model Summary.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 256)               173312
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 258
_________________________________________________________________
activation_1 (Activation)    (None, 2)                 0
=================================================================
Total params: 370,818
Trainable params: 370,818
Non-trainable params: 0
_________________________________________________________________
```

(b) Sample Baseline Model Summary.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 256)               20736
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_13 (Dense)             (None, 512)               131584
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_15 (Dense)             (None, 2)                 514
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
Total params: 284,162
Trainable params: 284,162
Non-trainable params: 0
_________________________________________________________________
```

(c) Sample PCA Model Summary.

Figure 4.18: Sample Model Parameter Summaries of Classification Models (Binary case).

Chapter 5

Critical Evaluation and Discussion

5.1 Experimental Insights

The results presented in Section 4 carry valuable insights for the link between PPG and dengue's physiological parameters; focusing on dengue shock events. The series of experiments gradually built up the complexity of the analysis, from binary to multi-class classifiers for dengue shock detection, and finally to regression for the evaluation of PPG's predictive capabilities.

5.1.1 Dengue Shock Detection and Monitoring using PPG

The results from Experiments 1 to 5, as presented in Section 4, portray a strong correlation between recorded PPG signals and the physiological parameters observed in multiple patients.

Even with little data available for the training of our algorithms, Experiment 1 highly suggests that PPG signals carry enough information to distinguish between patients who were admitted with shock and those that were admitted with no heavy complications due to the disease. As our datasets were balanced when training and testing, the results can be representative of the model performance without further processing. A KFCV average accuracy of 90.9% in combination with a 90.9% recall average and a 91.2% precision are very encouraging, suggesting that PPG could be used by clinical professionals in evaluating the severity of a diseased patient, i.e. whether he is experiencing dengue shock, without the need of radiology or ultrasounds that can be costly procedures [18]. In the same experiment, the best CNN model was deployed onto three other patients whose signals were not used in training or testing. As Table 4.3 shows, the model was able to correctly classify PPG signals of patients being admitted with shock, reaching accuracies of up to 89% when the signal segments were extracted only 10 minutes after the PPG record start. On the exact same table however, when using the signal segments extracted 3 hours after the start of the PPG record, the classification outcomes were poorer, with the maximum accuracy being at 76%. There are many reasons why this could be the case. For example, it could be that the admitted patients responded well to the administration of intravenous fluids and other symptom alleviating practices, leading to their vitals reflecting this physiological improvement. Because of that, the classifier might have characterised some spectrogram windows as ”healthy”, whereas in reality the patients are unlikely to experience that in such short time-frame [20]. A better explanation for the fall in the accuracy could be the algorithm's poor generalisation abilities causing it to misclassify the different physiological phases during a shock. For the evaluation of this, more data would be required and possibly a more detailed characterisation of signal instances, pinpointing changes in physiological parameters in time. Unfortunately, the dataset provided for this study did not have this capacity.

Experiments 2 and 3 are cardinal, as the results allow us to evaluate the capabilities of PPG signals being used as a monitoring system, in a clinical environment, or even at home if circumstances allow. The results we have collected, given that the scope of this project was exploratory, are exceptional and suggest that PPG is effective for use in clinical decision support. In Experiment 2, we were able to achieve high KFCV average accuracies, at 91.5%, along with high recall and precision averages. If enough data is available for the design and implementation of a well generalising model, or computational resources are readily accessible for a personalised baseline to be established, PPG signals could be used in real time monitoring of dengue patients, looking out for imminent dengue shock events. This will not only lead to faster and more effective care for the patients, but it will also equip health professionals and clinics with necessary information to work pro-actively into managing the logistics related to patient support and allocate the necessary resources to effectively house the critically ill. The same is also valid for patients that were admitted with DSS to begin with, as it is equally important to monitor for further instances of shock. That would enable for the evaluation of the current approach for treatment and possibly suggest a more aggressive fluid administration [46]. Experiment 3 targetted just that scenario and while results are not as good as those of Experiment 2, having a KFCV average accuracy of 83.7%, they are still sufficient to suggest that a relationship can be established. Looking at the recall metric in Table 4.5, we can see that it is significantly lower than that of the previous two experiments. Even though they are not directly comparable, this suggests that this model misclassifies windows corresponding to post-clinical shocks. That is expected since the patient involved in training and testing was already admitted with DSS and therefore, the pre-clinical shock signal is likely to possess some similar features to the post-clinical shock signal. If we had a larger dataset we would be able to examine whether or not the performance of the classifier is linked to the volume of data used in training. Therefore, this can be part of future research.

Building up from binary classifiers, the results from Experiments 4 and 5, allow us to evaluate the information capacity of spectrogram windows. For example, while Experiment 4 resulted in a lower KFCV average accuracy and F1 scores compared to the individual binary classification Experiments 1 and 2, it was still able to classify the 4 classes with an average accuracy of 88% and an F1 score of 88.7%. This means that we can successfully classify and distinguish between signals of patients who were admitted with shock, those that were admitted without shock, pre-clinical shock signals and post-clinical shock signals. This suggests that the spectrogram of a patient who never developed DSS can be distinguished from that of a patient who is about to have a DSS episode in the near future. At the same time we can distinguish between spectrograms of a patient admitted with DSS and spectrograms extracted post to a clinical shock. Again, even though a lot of care was taken in standardising the data and avoiding data leakage, we cannot rule out the possibility that patient signals might possess bias that a classifier might pick onto. Therefore, while the findings support the ability of PPG to characterise and carry information of physiological parameters caused by dengue, further research with more data needs to validate these results.

Experiment 5 is the last multi-class experiment performed, combining all of the available classes from Experiments 1,2 and 3. The resulting KFCV average accuracy and F1 score is much lower than all of the 4 previous experiments. By looking at the confusion matrices in Figure 4.13, the main cause of this poor performance is the misclassification of post-clinical shock spectrograms of a patient who was admitted with shock. From the same figure, it is evident that these class' windows are either classified as the post-clinical shock windows of a patient who was not admitted with DSS or wrongly classified as the pre-clinical shock windows of a patient who was admitted with DSS. While the latter can be easily attributed to the fact that both post-clinical shock classes are very likely to posses similar features, the latter is more counter-intuitive. One explanation would be, once again, the low volume of data used in training. Even though the classification on the same 2 classes was successful in Experiment 3, the complexity of this experiment is much higher and therefore, the algorithm, given this data, failed to generalise sufficiently.

5.1.2 Dengue Shock Prediction using PPG

When considering the macro-averaged metrics, Experiment 6 results are very poor, with the classification appearing to be random. Due to the lack of data, we were not interested in the regression performance, due to the number of features present in each sample. That is because the high dimensionality of the feature vectors results in a higher complexity, which in turn results in poor performance when only few hundreds of data points are present; in our case just a few hundreds of spectrograms. Even when using PCA, with only 80 features describing our network, training mean squared errors were very high, as expected. Therefore, using a classifier allowed us to simplify the procedure, only caring whether or not an output characterised a pre or post-clinical shock window. While the results look disappointing, if we only focus on the class with the most representations in training, that is the pre-clinical shock class, given an input, our regression LSTM model was able to output a feature vector that our pre-trained classifier classified correctly in 99% of cases. While this is not enough to draw a complete and thorough conclusion, it may suggest that PPGs can indeed be used sequentially for predicting future events. Nevertheless, the results can be intriguing but future research needs to further investigate this relationship, using larger datasets and longer PPG records. Especially since LSTMs can be prone to overfitting [13]. Unfortunately for us, for the current implementation, only patient 003-2009 was eligible for testing a time-series model, which was not ideal for this purpose.

5.1.3 IR vs Pleth

Throughout our study, experiments were ran on both IR ADC and Pleth signals individually and then compared. As presented in Section 4, IR ADC signals outperformed Pleth signals in 4 out of 5 classification experiments. This can suggest that the former, is more information rich than the latter, allowing for more meaningful features to be extracted when using it as an input. As a result, IR ADC may be linked more effectively with the physiological characteristics of a patient. Interestingly however, Pleth outperformed IR ADC in 4 out 5 baseline models which might suggest that Pleth signals are less noisy outright, and just by using its time-frequency features we can still get adequate feature representations, without the need of further FE i.e. using a CNN. For the same reason, less complex models can perform better on Pleth than IR ADC. Therefore, while IR ADC signals result in higher classification performance, Pleth might be more useful when computational resources are not freely available. The opportunity cost that exists should be considered in future research but given the advancements in computing, there is no large limiting factor in using either of the two signals. To conclude, it would be nice to see other filtration and pre-processing methods implemented on both IR ADC and Pleth signals, and observe the effect on the performance of our models. The tight time frame of this project did not allow for such implementations, so it is suggested that this is explored in future research.

5.2 Feature Extraction and Machine Learning Review

Performing time-frequency analysis using STFT on the filtered PPG signals, and using the output spectrograms as an input to our ML models, proved to be an effective FE approach, resulting in the successful implementation of this study's experiments. The spectrum of frequencies chosen to form the input feature vectors were chosen based on empirical data and allowed us to avoid using redundant features which would lead to an increased dimensionality. The spectrograms themselves hold enough information to result in a successful link between signal windows and physiological parameters of dengue, deeming our pre-processing approach and initial FE process fruitful. The Baseline model, as the name suggests, was built as a reference point for model types 2 and 3. Its performance on all experiments is unexpectedly better than imagined, which can only validate the effectiveness of using the time-frequency analysis to extract information from PPG signals. In later stages, CNN FE resulted in improved experimental results overall, suggesting that the processed input spectrogram "images" could be manipulated further to pick up finer feature details. This not only proves that CNNs and deep learning can be powerful tools in clinical diagnosis and decisions support, but also confirms the abundance of information present in PPG. Both ANN and CNN models were effective, but CNNs have the leading edge which is attributed to their powerful FE capabilities.

No clear evaluation is possible on the effect of the signal window length on model performance. Taking as an example Experiments 2 and 3, the former, classified signal window inputs of 1.12 minute length, whereas in the latter, the window length was decreased to 1 minute. Experiment 3's results are poorer than Experiment 2's. However that could be very well attributed to the amount of data used in training (100 windows per class vs. 77 windows per class) and the complexity of the experiment. Even though it seems like window size did not cause the lower performance, further exploration on that matter is required.

It is intriguing to see that the results presented in the previous section portray the ineffectiveness of ML model type 2, using PCA, in Experiments 1,4 and 5 when compared to the Baseline and CNN models. In contrast, the PCA model performed impressively well in Experiment 2, having the same KFCV average accuracy as the best performing model, but lacked behind the CNN model in the precision and F1 metrics. If we were to consider the bigger picture of both models, in a clinical scenario, the PCA model might have been preferred over the CNN model. Focusing at the number of trainable parameters of the former compared to those of the latter, as illustrated in Figure 4.18c and 4.18a respectively, we can see that the PCA model has approximately $86,000$ less parameters. This quantifies the reduced complexity of the PCA model, making it attractive for use in wearable technology. Having that in mind, we could have selected the PCA model over CNN. The same is valid for Experiment 3, but this time the PCA and CNN models' results have a much larger gap between them. As a result, the CNN might still be preferred over the PCA model. An important distinction that needs to pointed out, is that in both Experiments 2 and 3, the classes used for training were formed using signals of the same patient in each experiment. As PCA was applied onto the population of the training set, having signals of different patients between classes, such as in Experiments 1,4 and 5, might have had an adverse effect on the results and that is why the PCA model might have underperformed in those specific experiments. In general, dimensionality reduction could be very beneficial in reducing the computational resources required and can aid in the deployment of more flexible applications, especially with regards to wearable technology and clinical decision support devices. Given our results, using PCA as a form of FE (and subsequently dimensionality reduction) before a ML algorithm could be effective in some cases. To estimate the level of generalising performance further exploration is required, preferably with more data.

Models in general benefited from regularisation as seen from the best performing model architectures. This implies the presence of noise in the data and in cases where IR ADC was used, regularisation was more aggressive, with higher dropout rates, suggesting that IR ADC signals were corrupted with more noise.

Even though stratified KFCV was used in training all of the models, in many cases the performance on the test set did not match that on the training and validation sets. There are many reasons on why that could be the case but one could be the amount of noise present in the data. Particularly, in Experiments 3, 4 and 5, it is obvious by their training and validation curves that the validation set was not fully representative of the test set. One reason for that could be that as we are elapsing through time, PPG signals can change a lot. For example a moment of stress experienced by a patient can lead to a much different time-frequency representation in one of the windows present in validation set.

We must also highlight that the amount of time it takes for a PPG signal to go through the pre-processing steps and the ML algorithms is of great importance. That's because computational efficiency and inference are vital in a clinical setting. While state-of-the-art algorithms do exist for efficient filtering, windowing and STFT calculations on wearable devices, deploying deep learning models onto wearable technology has its bottlenecks [38]. These are mostly related to parallel matrix multiplications and bandwidth restrictions [11]. If an alternative approach i.e. cloud computing, is not on the table, then more lightweight algorithms along with dimensionality reduction practices need to be employed. This was outside the scope of this experiment due to its early stage, but highly important to consider in future research on the topic.

Lastly, as it has been already said, the study would greatly benefit from more data. This would allow for more complex experiments to be carried out and better evaluation of the ones we already completed. Still, experiments were completed successfully and with very propitious results. Moreover, since this is the first study ever done on this specific topic, the results can be used as a guide for future research or even set the baseline for experiments to follow.

5.3 Remarks on the Clinical Dataset

One of the most time-consuming and challenging parts of this study was understanding, processing and structuring the raw clinical files of the dataset. A challenging part of this process was the extraction of the times of shock or events associated with shock e.g. ascites. This was of utmost importance for this study, as ML algorithms and all of our experiments are highly dependent on the accuracy of the registered event times. While the event lookup itself was a simple query, the date-times in the raw clinical and signal files did not match exactly. In some cases there were contradicting records, with one file registering a different start time of the PPG records from the other. It was then realised that the filenames themselves were automatically named after the date-time of the PPG record start. This is an example of the uncertainty in the dataset that made our work more challenging and required time-consuming experimentation and debugging. Of course, this is expected in a newly formed dataset.

Another concern of ours, was the method of registering clinical events, and symptoms, from the healthcare professionals. As this was predominantly a manual procedure, the human error may have had an influence over our results. In some experiments, even a 10 minute difference between the time of the event and the recorded time might have caused a severe disruption to the results. Patient 003-2012 can be an example showcase of an incorrect data entry, as seen on Figure 3.7b. While the PPG signal seems to flat-line in around the middle of the recording, the registered end time of the PPG was much later. Conclusively, we still had to assume that the records were accurate, but in some cases we chose to avoid data that looked corrupted or inaccurate.

5.4 Proposal for Future Research

Given the project's scope and exploratory nature, further study needs to be conducted for the validation of our results and more importantly, focus on the capability of the constructed models in generalising on a larger datasets, with more diverse feature distributions. Given the previous suggestion and looking at our study's results, a larger dataset can greatly influence the quality of modelling and creates opportunities for a more thorough exploration of the designed experiments; such as in predicting a shock using an RNN-based model. On top of that, it can enable the design of more complex experiments i.e. time-specific predictions of DSS. Adding to the benefits of larger datasets, more data would enable greater flexibility when it comes to choosing signal data to use in ML modelling, by allowing the use of SQIs in a more structured manner, setting SQI thresholds for rejecting low-quality signals; instead of choosing the best out of the ”worst”.

In future work, it is also principal that we consider data originating from different demographics. For example, there are substantial differences between infections in adults and children [54].

If we are to create models that generalise well for the wider population, this is a topic that should be researched in more depth, with children data. Moreover, as there are multiple PPG collection sites e.g., wrists and fingers, tests should be ran to compare these and the protocols employed in collection.

In the design and implementation stage of the project we were very much restricted by time and therefore, we weren't able to explore some interesting data processing and ML methodologies that were encountered during our research. First, more PPG filtering techniques could be explored, such as wavelet denoising, for artefact rejection. Even though we did not experience any setbacks with Butterworth filtering, in future work we should examine the effect of noise onto model performance in more depth.

It would be interesting to explore the length of signal segments forming the inputs of our ML algorithms. We have tested 1 and 1.2 minute segments to create our feature vectors and the results were positive. However, could we achieve similar performance with less features? Doing so would result in finer time analysis and lower ML inference times.

Moving on to FE, it would be interesting to see how different methods compare to our chosen approach. For example, even though STFT proved sufficient, wavelet analysis could be a better alternative given that it represents the signal in both time and frequency components. Another FE approach that might be worth exploring is the formation of ML input feature vectors comprising of a combination of SQIs, calculated during the early data processing stage. As mentioned earlier, not all SQIs would be appropriate for this purpose, but statistical features such as skewness and kurtosis do carry useful information to characterise different signal segments.

Looking back at out ML implementations, an alternative to LSTM-based models for the prediction of severe dengue parameters could be models utilising Gated Recurrent Units (GRUs). While GRU and LSTM cells exhibit similar behaviour and are both based off RNNs, GRUs cam be more easily optimized due to their simpler strucutre. Moreover, GRUs can perform relatively better on smaller datasets, which could be of benefit to our implementation. However, GRUs are relatively new and in theory, they can "remember" shorter sequences compared to

LSTMs, which can be important in a clinical scenario such as ours. Nevertheless, GRUs are definitely worth exploring in future research. [15]

Chapter 6

Conclusion

The novelty of the project is of no dispute. The newly formed OUCRU dataset was studied successfully, allowing us to explore multiple disciplines surrounding dengue's clinical decision support and PPG devices. From building data processing pipelines, to extracting PPG features and designing, as well as implementing machine learning algorithms, this project has a lot to give. Starting with the dataset itself, we were able to successively extract and structure raw data, not only allowing us to carry out necessary feature extraction steps, but also providing a well-rounded overview of the dataset's capabilities; essential for influencing further research. Our signal processing techniques, along with methodology employed in the analysis of the clinical and PPG data, were successfully designed and implemented, allowing for an in-depth exploration of ML algorithms for the classification and prediction of severe dengue. A set of well defined experiments were conducted meticulously, presenting the undisputed relationship between PPG and the physiological parameters experienced by dengue patients. Most importantly we have explored in-depth how ML algorithms can utilise PPG in the detection of various phases of pre and post DSS instances, with high sensitivity and precision. As a complementary step, we have evaluated the predictive capabilities of PPG, utilising their time-frequency features, with the help of LSTM networks. Even though results on this last experiment were underwhelming, the information obtained is of equal importance to the successful experiments, as the results highly encourage further research on verifying the correlation between sequential

PPG windows. The data at our disposal was limited but we managed to design and implement fruitful classification experiments, built and optimised ML models with good generalisation performances and extensively evaluated our results and methodology. Conclusively, the project can certainly act as a strong foundation for further research and encourage an era of inexpensive and effective clinical decision support developments to better manage dengue epidemics, provide more effective care and most importantly, minimise the loss of human lives.

As a final note, we should emphasise that while a diagnostic tool is of utmost importance, doctors and other health professionals are a vital part of the equation. We are not trying to substitute professionals, we are simply trying to contribute towards a more efficient, cost-effective, timely and accurate diagnosis.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation)*, pages 265–283, 2016.

[2] J. Allen. Photoplethysmography and its application in clinical physiological measurement. *Physiological measurement*, 28(3):R1, 2007.

[3] J. Allen, H. Liu, S. Iqbal, D. Zheng, and G. Stansby. Deep learning-based photoplethysmography classification for peripheral arterial disease detection: a proof-of-concept study. *Physiological Measurement*, 42(5):054002, 2021.

[4] A. Barroso, A. Lopez, S. Iodice, and J. S. Pujol. Imperial College Deep Learning Course - MatchLab, 2021.

[5] F. Bre, J. M. Gimenez, and V. D. Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158:1429–1441, 2018.

[6] J. Brownlee. A gentle introduction to k-fold cross-validation. *Machine Learning Mastery*, 2019, 2018.

[7] J. Brownlee. Data leakage in machine learning, 2021.

[8] S. Butterworth et al. On the theory of filter amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

[9] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, and H. Nazeran. A review on wearable photoplethysmography sensors and their potential future applications in health care. *International journal of biosensors & bioelectronics*, 4(4):195, 2018.

[10] P. Celka, M. Brucet, N. Granqvist, et al. Photoplethysmogram combined with the art of eastern pulse analysis: a novel way to assess body and mind energy balance. *Cardiol Vasc Res*, 5(1):1–13, 2021.

[11] A. R. Dargazany, P. Stegagno, and K. Mankodiya. Wearabledl: wearable internet-of-things and deep learning for big data analytics—concept, literature, and future. *Mobile Information Systems*, 2018, 2018.

[12] M. Elgendi. Optimal signal quality index for photoplethysmogram signals. *Bioengineering*, 3(4):21, 2016.

[13] T. Ergen and S. S. Kozat. Online training of lstm networks in distributed systems for variable length data sequences. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):5159–5165, 2018.

[14] C. for Disease Control and Prevention. About dengue: What you need to know. Technical report, Centers for Disease Control and Prevention. U.S. Department of Health Human Services, 2019. Accessed: 2021-07-21.

[15] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[16] M. Ghamari, C. Soltanpur, S. Cabrera, R. Romero, R. Martinek, and H. Nazeran. Design and prototyping of a wristband-type wireless photoplethysmographic device for heart rate variability signal analysis. 08 2016.

[17] I. Gotlibovych, S. Crawford, D. Goyal, J. Liu, Y. Kerem, D. Benaron, D. Yilmaz, G. Marcus, and Y. Li. End-to-end deep learning from raw sensor data: Atrial fibrillation detection using wearables. *arXiv preprint arXiv:1807.10707*, 2018.

[18] M. Guzman, A. Perez, O. Fuentes, and G. Kouri. Dengue, dengue hemorrhagic fever. 2008.

[19] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[20] J. S. S. Herencia. Lessons learned and recent advances in dengue research. *Dengue Fever in a One Health Perspective*, 2020.

[21] B. A. Hernandez Perez. The OUCRU datasets - Documentation, 2021.

[22] B. A. Hernandez Perez, S. Karolcik, and Others. Vital SQI Library for Python, 2021.

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] N. T. Huy, N. T. H. Thao, T. T. N. Ha, N. T. P. Lan, P. T. T. Nga, T. T. Thuy, H. M. Tuan, C. T. P. Nga, V. Van Tuong, T. Van Dat, et al. Development of clinical decision rules to predict recurrent shock in dengue. *Critical care*, 17(6):1–8, 2013.

[25] P. T. Inc. Collaborative data science. Montreal, QC, 2015. Plotly Technologies Inc.

[26] P. K. Lam, T. V. Ngoc, T. T. Thu Thuy, N. T. Hong Van, T. T. Nhu Thuy, D. T. Hoai Tam, N. M. Dung, N. T. Hanh Tien, N. T. Thanh Kieu, C. Simmons, et al. The value of daily platelet counts for predicting dengue shock syndrome: Results from a prospective observational study of 2301 vietnamese children with dengue. *PLoS neglected tropical diseases*, 11(4):e0005498, 2017.

[27] Y. Liang, M. Elgendi, Z. Chen, and R. Ward. An optimal filter for short photoplethysmogram signals. *Scientific data*, 5(1):1–12, 2018.

[28] L. Lindberg and P. Oberg. Photoplethysmography. part 2. influence of light source wavelength. *Medical & biological engineering & computing*, 29(1):48–54, 1991.

[29] E. Liu and Others. Machine Learning Tutorials - Model Selection, 2021.

[30] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[31] K. Mikolajczyk and A. Spiers. ELEC97113 - Computer Vision and Pattern Recognition 2020-2021, 2021.

[32] D. K. Ming, S. Sangkaew, H. Q. Chanh, P. T. Nhat, S. Yacoub, P. Georgiou, and A. H. Holmes. Continuous physiological monitoring using wearable technology to inform individual management of infectious diseases, public health and outbreak responses. *International Journal of Infectious Diseases*, 96:648–654, 2020.

[33] J. L. Moraes, M. X. Rocha, G. G. Vasconcelos, J. E. Vasconcelos Filho, V. H. C. De Albuquerque, and A. R. Alexandria. Advances in photopletysmography signal analysis for biomedical applications. *Sensors*, 18(6):1894, 2018.

[34] K. Morita. Dengue vaccines. *Nihon rinsho. Japanese journal of clinical medicine*, 66(10):1999–2003, 2008.

[35] S. L. Moulton, J. Mulligan, G. Z. Grudic, and V. A. Convertino. Running on empty? the compensatory reserve index. *Journal of Trauma and Acute Care Surgery*, 75(6):1053–1059, 2013.

[36] S. L. Moulton, J. Mulligan, A. Srikiatkhachorn, S. Kalayanarooj, G. Z. Grudic, S. Green, R. V. Gibbons, G. W. Muniz, C. Hinojosa-Laborde, A. L. Rothman, et al. State-of-the-art monitoring in treatment of dengue shock syndrome: a case series. *Journal of medical case reports*, 10(1):1–7, 2016.

[37] L. Nilsson, A. Johansson, and S. Kalman. Respiration can be monitored by photoplethysmography with high sensitivity and specificity regardless of anaesthesia and ventilatory mode. *Acta anaesthesiologica scandinavica*, 49(8):1157–1162, 2005.

[38] S. Nisar, O. U. Khan, and M. Tariq. An efficient adaptive window size selection method for improving spectrogram visualization. *Computational intelligence and neuroscience*, 2016, 2016.

[39] C. Olah. Understanding lstm networks. 2015.

[40] W. H. Organization et al. Dengue and severe dengue. Technical report, World Health Organization. Regional Office for the Eastern Mediterranean, 2021. Accessed: 2021-07-19.

[41] B. D. Patil. Introduction to wavelet. 2014.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[43] J. P. Phillips, M. Hickey, and P. A. Kyriacou. Evaluation of electrical and optical plethysmography sensors for noninvasive monitoring of hemoglobin concentration. *Sensors*, 12(2):1816–1826, 2012.

[44] R. Prabhu. Understanding of convolutional neural network (cnn)—deep learning. *Medium. Com*, pages 1–11, 2018.

[45] N. Pradhan, S. Rajan, A. Adler, and C. Redpath. Classification of the quality of wristband-based photoplethysmography signals. In *2017 IEEE international symposium on medical measurements and applications (MeMeA)*, pages 269–274. IEEE, 2017.

[46] S. Rajapakse. Dengue shock. *Journal of Emergencies, Trauma and Shock*, 4(1):120, 2011.

[47] A. Reiss, I. Indlekofer, P. Schmidt, and K. Van Laerhoven. Deep ppg: large-scale heart rate estimation with convolutional neural networks. *Sensors*, 19(14):3079, 2019.

[48] A. Y. Saleh and L. Baiwei. Dengue prediction using deep learning with long short-term memory. In *2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pages 1–5. IEEE, 2021.

[49] D. Sarma, S. Hossain, T. Mittra, M. A. M. Bhuiya, I. Saha, and R. Chakma. Dengue prediction using machine learning algorithms. In *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC)*, pages 1–6. IEEE, 2020.

[50] K. Shelley, S. Shelley, and C. Lake. Pulse oximeter waveform: photoelectric plethysmography. *Clinical monitoring*, 2, 2001.

[51] S. Shobitha, P. Amita, K. B. Niranjana, and M. A. M. Ali. Noninvasive blood glucose prediction from photoplethysmogram using relevance vector machine. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–4. IEEE, 2018.

[52] J. O. Smith. *Spectral audio signal processing.* W3K, 2011.

[53] S. Smith. *Digital signal processing: a practical guide for engineers and scientists.* Elsevier, 2013.

[54] L. J. d. Souza, L. B. Pessanha, L. C. Mansur, L. A. d. Souza, M. B. T. Ribeiro, M. d. V. d. Silveira, and J. T. D. Souto Filho. Comparison of clinical and laboratory characteristics between children and adults with dengue. *Brazilian Journal of Infectious Diseases*, 17:27–31, 2013.

[55] W. Storr. Butterworth Filter Design, 2014.

[56] M. Sunasra. Performance metrics for classification problems in machine learning. *Medium recuperado de: https://medium. com/thalusai/performance-metrics-for-classification-problems-in-machine-learningpart-i-b085d432082b*, 2017.

[57] G. A. Tadesse, H. Javed, N. L. N. Thanh, H. D. H. Thi, L. Thwaites, D. A. Clifton, T. Zhu, et al. Multi-modal diagnosis of infectious diseases in the developing world. *IEEE journal of biomedical and health informatics*, 24(7):2131–2141, 2020.

[58] H. Tjahjadi, K. Ramli, and H. Murfi. Noninvasive classification of blood pressure based on photoplethysmography signals using bidirectional long short-term memory and time-frequency analysis. *IEEE Access*, 8:20735–20748, 2020.

[59] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual.* CreateSpace, Scotts Valley, CA, 2009.

[60] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[61] V. Vizbara. Comparison of green, blue and infrared light in wrist and forehead photoplethysmography. *BIOMEDICAL ENGINEERING 2016*, 17(1), 2013.

[62] W. Waugh, J. Allen, J. Wightman, A. J. Sims, and T. A. Beale. Novel signal noise reduction method through cluster analysis, applied to photoplethysmography. *Computational and mathematical methods in medicine*, 2018, 2018.

[63] R. W. Wijshoff, A. M. van Asten, W. H. Peeters, R. Bezemer, G. J. Noordergraaf, M. Mischi, and R. M. Aarts. Photoplethysmography-based algorithm for detection of cardiogenic output during cardiopulmonary resuscitation. *IEEE Transactions on Biomedical Engineering*, 62(3):909–921, 2014.

[64] K. Wirsing. Time frequency analysis of wavelet and fourier transform. In *Wavelet Theory*. IntechOpen, 2020.

[65] V. Wiwanitkit. The importance of accurate diagnosis of dengue fever. *Future Virology*, 7(1):53–62, 2012.

[66] C. Yu, Z. Liu, T. McKenna, A. T. Reisner, and J. Reifman. A method for automatic identification of reliable heart rates calculated from ecg and ppg waveforms. *Journal of the American Medical Informatics Association*, 13(3):309–320, 2006.

Appendix A

Code Listings and Examples

Git software was used for the management and organisation of this study's code implementations. A very useful tool that was utilised is Sphinx, allowing for the automated generation of documentation for the code listings.

Example code listings and documentation can be found on:

`https://p-a-ha.github.io/fyp2021-ph720/_examples/tutorial/index.html`

For completeness, the project's Github Page can be seen on:

`https://github.com/P-A-Ha/fyp2021-ph720`

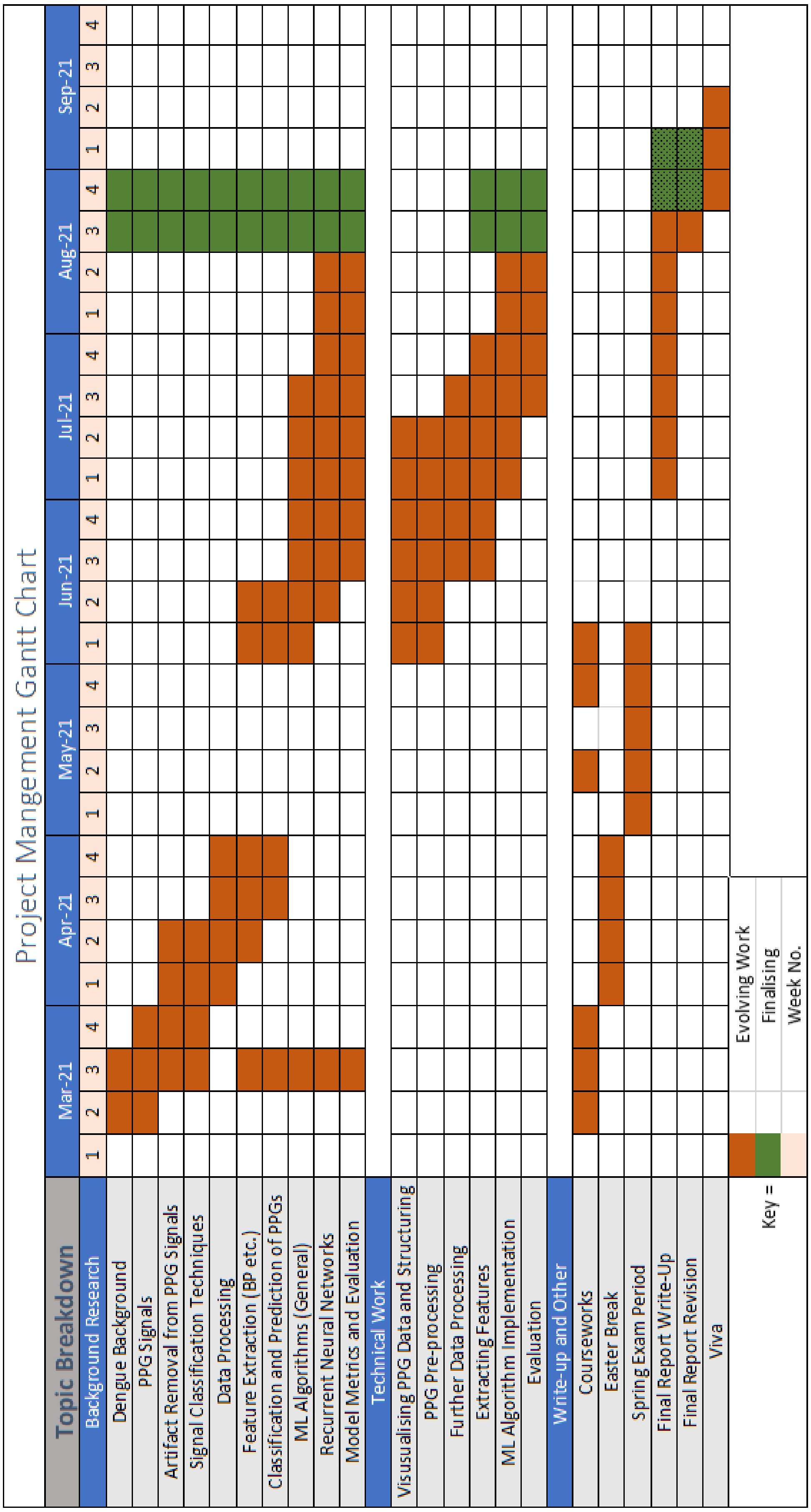
Appendix B

Project Management

The Agile methodology was employed throughout the project. This process can be summarised with the following principles.

- Measuring progress by evaluating software performance
- Reflect on practices regularly
- Simplify processes without compromising quality
- Adapt to challenges without drifting from the main goal
- Pay attention to the technical details of the project
- Consistency of work

I strictly adhered to the aforementioned strategy to successfully complete this study. A Gantt chart, demonstrated in the next page, was set for the control and management of our workflow and was followed reverently. For the file management I used Google Drive, Github and Overleaf to back-up all of the files that I created or used in this study. This procedure was carried out daily, to ensure no setbacks. The project was completed as it was originally planned, with all checkpoints being met.



Project Mangement Gantt Chart

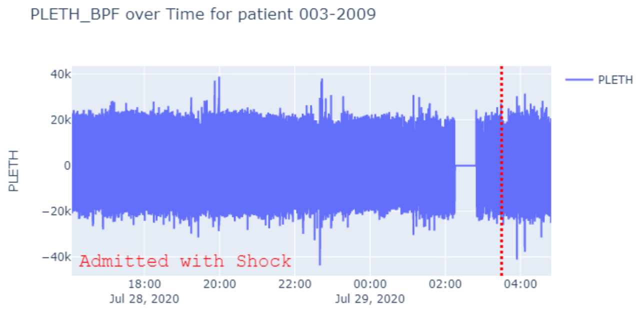
Topic Breakdown	Mar-21				Apr-21				May-21				Jun-21				Jul-21				Aug-21				Sep-21			
Background Research	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Dengue Background		🟧	🟧																				🟩	🟩				
PPG Signals		🟧	🟧	🟧																			🟩	🟩				
Artifact Removal from PPG Signals			🟧	🟧	🟧	🟧																	🟩	🟩				
Signal Classification Techniques			🟧	🟧	🟧	🟧																	🟩	🟩				
Data Processing					🟧	🟧	🟧	🟧															🟩	🟩				
Feature Extraction (BP etc.)			🟧			🟧	🟧	🟧					🟧	🟧									🟩	🟩				
Classification and Prediction of PPGs			🟧				🟧	🟧					🟧	🟧									🟩	🟩				
ML Algorithms (General)			🟧										🟧	🟧	🟧	🟧	🟧	🟧	🟧				🟩	🟩				
Recurrent Neural Networks			🟧											🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟩	🟩				
Model Metrics and Evaluation			🟧												🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟩	🟩				
Technical Work																												
Visusualising PPG Data and Structuring													🟧	🟧	🟧	🟧	🟧	🟧										
PPG Pre-processing													🟧	🟧	🟧	🟧	🟧	🟧										
Further Data Processing															🟧	🟧	🟧	🟧	🟧									
Extracting Features															🟧	🟧	🟧	🟧	🟧	🟧			🟩	🟩				
ML Algorithm Implementation																	🟧	🟧	🟧	🟧	🟧	🟧	🟩	🟩				
Evaluation																			🟧	🟧	🟧	🟧	🟩	🟩				
Write-up and Other																												
Courseworks		🟧	🟧	🟧						🟧		🟧	🟧															
Easter Break					🟧	🟧	🟧	🟧																				
Spring Exam Period									🟧	🟧	🟧	🟧	🟧															
Final Report Write-Up																	🟧	🟧	🟧	🟧	🟧	🟧	🟧	▩	▩			
Final Report Revision																							🟧	▩	▩			
Viva																								🟧	🟧	🟧		

Key =	
🟧	Evolving Work
🟩	Finalising
▭	Week No.

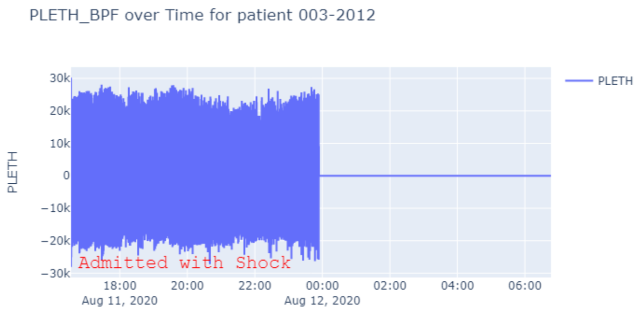
Figure B.1: Project Management Gantt Chart.

Appendix C

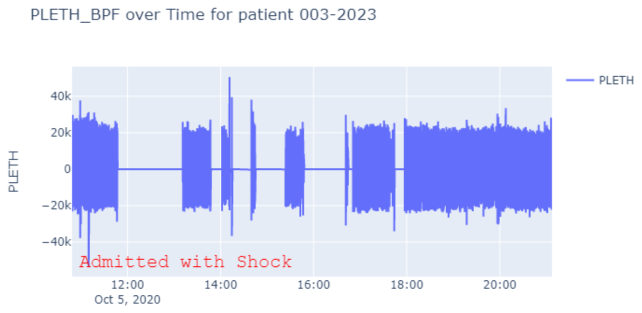
Plots of Pleth Signals



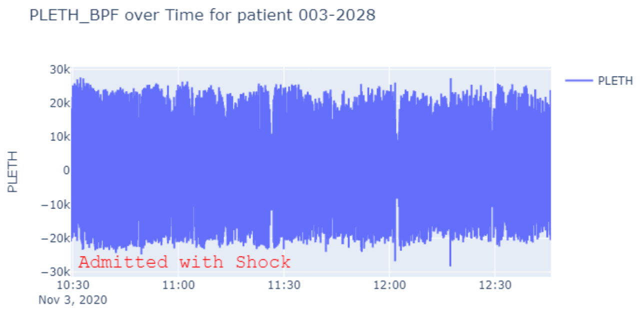
(a)



(b)

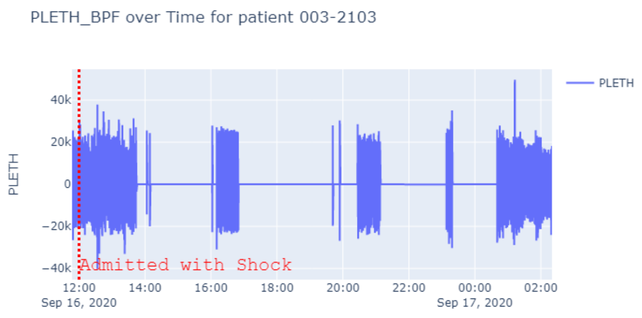


(c)

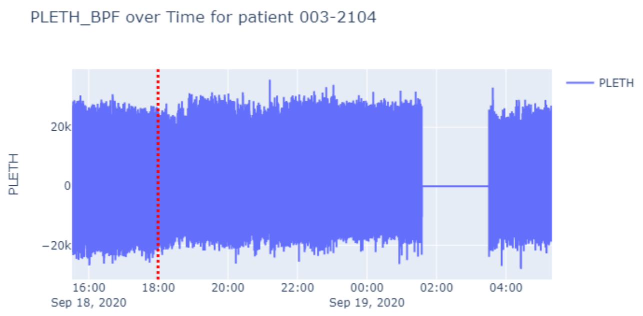


(d)

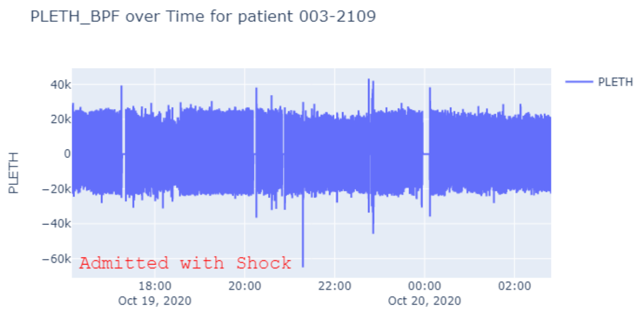
Figure C.1: Visualising the Patients' Pleth Signals along with the Shock Events Captured within the PPG Record - Part 1.



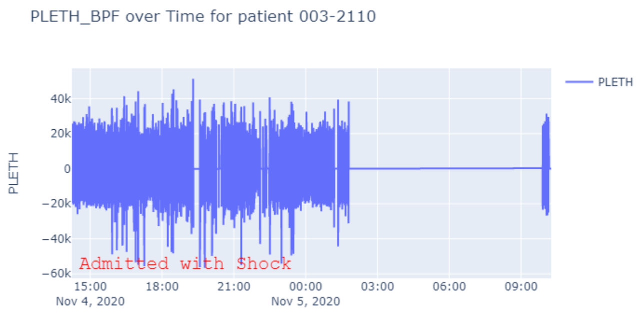
(a)



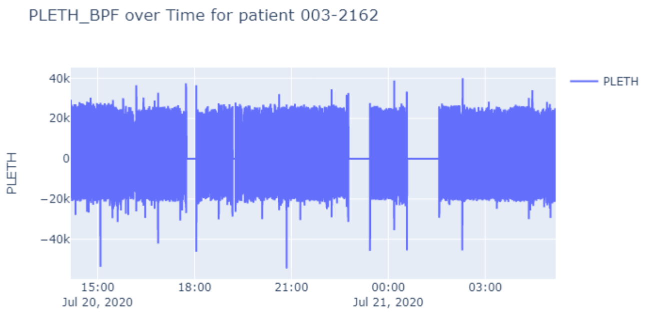
(b)



(c)



(d)



(e)

Figure C.2: Visualising the Patients' Pleth Signals along with the Shock Events Captured within the PPG Record - Part 2.

Appendix D

Etchics

Our study did not require any approval from Imperial College's Research Ethics Committee (ICREC). All conducted experiments were done diligently, paying attention not to breach any privacy barriers. Moreover, patient data was processed anonymously. Finally, the responsibility of conducting a study that is accurate, sensible and free of bias, governed our project.