2023

HRMobile: A lightweight, local architecture for heart rate measurement

Sam Morton
Dartmouth College, Samuel.H.Morton.21@dartmouth.edu

Sam Morton
Dartmouth College

HRMOBILE: A LIGHTWEIGHT, LOCAL ARCHITECTURE FOR REMOTE HEART RATE MEASUREMENT

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science

by

Sam Morton

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 2023

Examining Committee:

Nicholas Jacobson, Chair

Andrew Campbell

Soroush Vosoughi

F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

Abstract

Heart rate and heart rate variability (HRV) are important metrics in the study of numerous physical and psychiatric conditions. Previously, measurement of heart rate was relegated to clinical settings, and was neither convenient nor captured a patient's typical resting state. In effect, this made gathering heart rate data costly and introduced noise. The current prevalence of mobile phone technology and Internet access has increased the viability of remote health monitoring, thus presenting an opportunity to substantially improve the speed, convenience, and reliability of heart rate readings. Recent attention has focused on different methods for remote, non-contact heart rate measurement. Of these methods, video presents perhaps the best option for optimizing cost and convenience. This thesis introduces a lightweight architecture for estimating heart rate and HRV using a smartphone camera. The system presented here runs locally on a smartphone, requiring only a phone camera and 15s or more of continuous video of a subject's face. No Internet connection or networking is necessary. Building the system to run locally in this manner means that this software confers benefits such as greater user privacy, offline availability, reliability, cost effectiveness, and speed. However, it also introduces added constraints on computational complexity. With these tradeoffs in mind, the system presented here is implemented within an Android mobile app. The performance of our approach fell short of that of existing state-of-the-art methods in terms of mean absolute error (MAE) of heart rate estimation, achieving MAE during validation that was over $17x$

greater than other existing approaches. There are a number of factors which may contribute to this performance discrepancy, including limitations in the diversity of the data used with respect to gender, age, skin tone, and heart rate intensity. Further, remote photoplethysmographic (rPPG) signal generated by this architecture contains a large number of noise artifacts which are difficult to consistently remove through signal processing. This noise is the primary reason for the underperformance of this architecture, and could potentially be explained by model and feature engineering decisions which were made to address the risk of overfitting on the limited dataset used in this work.

Acknowledgements

I would like to thank several people for their support and contributions to this project. First, I would like to thank Professor Nicholas Jacobson for welcoming me into the AIM HIGH Lab and Mood Triggers project and for serving as my direct advisor throughout the duration of this project. Without his guidance and patience, this work would not have been possible. Further, I would like to thank Professors Andrew Campbell and Soroush Vosoughi. Professor Campbell agreed to serve as my primary departmental advisor for this project, and for that I am grateful. Professor Vosoughi helped me with great advice at different points throughout my undergrad and graduate careers at Dartmouth, and those conversations were always appreciated. I would also like to thank the other members of the Mood Triggers team: George Price, Lisa Oh, and Arnav Pondicherry, whose weekly input helped guide this project.

I would also like to offer a special acknowledgement to the contributions of Arnav Pondicherry to this project. Arnav laid the preliminary groundwork for the Android application used for testing the presented pipeline, implemented the Python plugin for Flutter, Pydroid, used in this project, and wrote seminal code incorporating the YoloV5 facial recognition model into Pydroid which was used in each iteration of this project.

Finally, I would like to thank my parents for supporting me throughout my undergrad and graduate careers.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Section 1.1

Motivation

This project unifies my interests in computer science and the application of technology to healthcare. Increasing public awareness of the importance of mental health has led to heightened interest in the field from both the technology industry and academia. The ability to conveniently, cheaply, and accurately measure physiological parameters such as heart rate and heart rate variability (HRV) could offer important new signal in monitoring both the physical and mental health of users. The contribution of this work is to design a system for remote, non-contact heart rate measurement which can be housed entirely within a smartphone. I hope this work will serve to further democratize access to methods for remote heart rate measurement and offer a more private, available architecture that can be built on in future iterations. Further, I hope this work serves to help the Mood Triggers team with their development targets going forward.

Section 1.2

Context

The healthcare industry is a magnet for technological innovation. Multifaceted yet inefficient, the pace of tech adoption in healthcare once lagged behind rate of expansion of the tech industry as a whole. (2). However, recent years have witnessed a burgeoning health tech sector emerge and begin to transform the nature of healthcare (3). From AI to blockchain, creative solutions for applying existing cutting-edge technologies to healthcare are emerging rapidly (4), (5). The sheer ubiquity of technology across the United States and worldwide is already changing the face of traditional healthcare, and this trend will likely accelerate going forward (3). Smartphones, already omnipresent throughout the developed world and growing in their degree of adoption in developing nations, give those with Internet access the ability to connect with healthcare providers anywhere (3). Furthermore, adoption of smartphones and wearable technology for health and fitness monitoring is rapidly evolving and becoming more accessible (3). Many regions worldwide do not have reliable Internet access due to cost or lack of infrastructure. Low Earth Orbit (LEO) satellite internet providers such as Starlink, OneWeb, and Kuiper bypass the need to install complex and expensive infrastructure (6). Not only will this revolutionize rural Internet access in the United States, but it will increase the rate at which developing nations are able to close the technological gap.

Telemedicine, often referenced interchangeably with telehealth, is a broad term encompassing all forms of remote access to healthcare services via the Internet. Though it is most commonly envisioned as the application of video conferencing technology to connect providers and patients in different geographic regions, telemedicine can also include remote health monitoring, education, and access to health records. Advance-

ments in smartphone and Internet access in the United States and across the world are likely to cause a revolution in healthcare access, though they may not necessarily fulfill the often-promised corresponding decrease in cost (7). However, the potential for telehealth to increase access to healthcare is indisputable, and its effects will be most pronounced amongst communities previously underserved by traditional healthcare. Further, the application of telemedicine in healthcare is not limited to physical health. There is a large and growing body of work concerning the application of telemedicine to mental health care (8). These applications include direct psychiatric care, automated interventions, remote behavior monitoring through questionnaires, and the automatic collection of physiological data such as heart rate and HRV (9).

1.2.1. Beneficiaries of telehealth

Increased access to healthcare via telemedicine has the potential to drastically improve health access for many groups in the United States, many of which have been previously underserved by traditional healthcare. Traditionally underserved communities that stand to benefit from increased telehealth adoption include Native Americans, rural populations, lower income Americans, minorities, and immigrants. In particular, Native American tribal nations are some of the most disadvantaged communities by traditional healthcare due to their often remote locations, lack of resources, minority status, and at times language barriers (10). Though hardly a solution to the problem of inadequate healthcare on reservations and amongst other underserved groups, the lower costs and increased ease of access associated with telemedicine may begin to help close this coverage gap seen in the United States.

1.2.2. Telepsychiatry

A subset of the larger field of telemedicine, telepsychiatry is one of the earliest forms of remote healthcare delivery, dating back to the 1950s (11). Consistent with the

field of telemedicine more generally, the most obvious application of telepsychiatry is the delivery of psychiatric services to remote communities which traditionally lack access (11). Likewise, there is an expectation that increased access to psychiatric care and a reduced demand for office space will decrease psychiatric costs (11). It is also true that it is possible to get closer to the full range of available psychiatric services remotely than is the case with traditional medicine, and thus the potential of telepsychiatry could be even greater than that of the rest of the field of telehealth.

An additional advantage of telepsychiatry is that its flexibility opens opportunities to providers to offer a larger range of care options to patients (11). For this reason, what was originally conceived as a solution for providing psychiatric services to underserved rural populations has seen an increase in adoption in urban areas (11). More recently, there has been a rapid increase in the diversity of telepsychiatric services offered. In addition to offering clinical patient care, providers have begun offering assessment and diagnostic services, psychosocial interventions, follow-up and home-based care, development of clinical care plans, care management, crisis intervention, neuropsychological testing, legal aid, forensic evaluations, and liaison services for other fields of medicine (11). This expansion of the scope of telepsychiatry to serve both urban and rural areas has aided in expanding the population of patients receiving psychiatric care to children, the elderly, and special populations such as military personnel and prisoners (11).

Though the potential benefits of telepsychiatry and telehealth more generally are numerous, increased reliance on technology introduces new risks to patient privacy. Hale and Kvedar note that the extent to which telehealth communication is covered by the Health Insurance Portability and Accounting Act (HIPAA) is unclear (12). This calls into question the legal protections granted to patients who participate in remote care. Further, protection of patient data from malicious actors may not

be guaranteed, particularly if a telehealth system is not implemented sufficiently carefully (12). The confluence of these factors often leads to a potential lack of trust in telehealth systems on the part of patients and a corresponding hesitance to participate in telemedicine. Though this of course slows down adoption of potentially helpful care, there is evidence to suggest that many patients are willing to accept this level of risk if the benefits of the care are sufficiently clear to them (12). With this in mind, it becomes important for developers of telehealth systems more broadly to build safe systems which can be used by providers to make the strongest possible safety guarantees to patients.

The potential benefits of telepsychiatry and telemedicine more generally were emphasized during the COVID-19 pandemic. In the case of such extreme circumstances, delivery of in-person psychiatric care decreased in order to lower COVID-19 exposure. Telepsychiatry was already available prior to the COVID-19 outbreak, and thus was well-positioned to make a positive impact during the pandemic (13). Telepsychiatry enabled providers to offer psychiatric care without risk of increasing spread of the disease. As a result, telepsychiatry adoption increased during the pandemic as patients sought to receive necessary care in the safest possible manner (14). However, it is important to note that telepsychiatry is still not as prevalent as it could be, even in the wake of the pandemic and efforts on the part of bodies such as the Centers for Medicare and Medicaid Services (CMS) to relieve some of the regulatory burden on telepsychiaty providers (14).

The widespread presence of technology also introduces options in the physical and mental health monitoring space. With increasing adoption of smartphones, wearable technology, and other devices in daily life, researchers in the private sector and academia are presented with the opportunity to study patient variables such as movement, sleep duration, heart rate, electrocardiogram, and skin temperature (15). These

factors can all be relevant to the study and development of psychiatric disorders. For example, accelerometer and microphone data can be used to analyze social activity. Heart rate and skin conductance are useful for inferring stress and anxiety disorders (15). Taken together, it is easy to envision complex and intelligent future systems which are able to monitor the mental health of willing participants with a high level of sophistication.

Additional approaches to remote monitoring have included AI chatbots (16). Such projects not only have the potential to provide patients with important at-will behavioral intervention and therapy, but also may serve as additional data sources for learning (16). Similar research is currently underway at Dartmouth within the AI and Mental Health: Innovation in Technology Guided Healthcare (AIM High) Lab with the Therabot project (17).

The popularity of smartphones and the Internet has also driven academic attention towards the delivery of surveys such as ecological momentary assessments (EMAs) via smartphones (18). EMAs involve the repeated sampling of subject behaviors, experiences, and feelings in real time, with the goal of minimizing bias and noise in samples (19). The application of properly designed EMAs to a sample population has the potential to yield important insights into mental health at an aggregate level, in addition to having the potential for serving as the basis for drawing individual insights. For example, EMAs were successfully applied in a longitudinal study of a population of college students during the COVID-19 pandemic and were combined with smartphone sensor data to provide insights into the relationship between data collected via smartphone sensors and the reported mental health experiences of students (18).

Supplementing surveys such as EMAs and passive sensing data collected from smartphones such as movement with physiological data has shown potential for offering enhanced signal for inferring stress, anxiety, and other disorders. Studies have

even attempted to link measures such as heart rate and HRV to conditions such as eating disorders (20). For example, it has been suggested that anxious and depressive disorders are associated with low HRV and future cardiovascular disease (21). Thus, the ability to pair passive sensing smartphone data and EMA data with heart rate or HRV readings may be relevant to modeling objectives and inferring conclusions from such data. However, there are questions about the feasibility of pairing traditional methods for collecting heart rate data with remote EMA surveys (22). Ground truth heart rate data are most commonly collected in a clinical setting. However, this is not realistic for large-scale EMA and passive sensing studies. Further, clinical settings are not normal daily environments for most subjects, and thus introduce bias. Supporting this, EMA studies that have been completed using wireless electrocardiography (ECG) patches for remote heart rate sensing report mixed results due to the added burden placed on subjects to wear the the sensors (22).

1.2.3. Remote heart rate monitoring

Use cases for remote, non-invasive methods for measuring heart rate and HRV are not limited solely to their application in remote physical and mental health monitoring. Both metrics have well established uses cases in sports and physical fitness. Measurement of heart rate and HRV has also been used to assess fatigue in individuals, which has implications for intervention in activities such as driving or operating heavy machinery (23).

Recent years have seen a proliferation of different strategies for remote measurement of heart rate. Some proposed methods include capacitively coupled ECG, Doppler radar, optical vibrocardiography (VCG), thermal imaging, heart rate from speech, and camera imaging (24). However, many of these methods, such as those applying Doppler radars or optical VCG, may be impractical for large-scale adoption due to their reliance on more expensive and less widely available technology. For

instance, the laser technology required for optical VCG is expensive (24). Methods applying doppler radar, of course, require the use of doppler radar, technology that may not always be accessible (24).

Approaches to remote, non-contact heart rate measurement leveraging cameras are some of the most promising. Camera based methods require only video of subjects within a few meters of the camera and with clear views of the subjects' skin. Cardiovascular pulse waves traveling through a human body stretch vessel walls (24). This happens periodically, and affects volumetric changes in the amount of blood or air contained within the body. These internal changes affect absorbance of light in human tissues and can be measured via PPG (24). As a result, these changes in human tissue light absorbance influence the red, blue, and green (RGB) color channels detected by a camera focused on human tissue (24). This color data, when collected from video, can detect cyclical color changes that are the result of cardiographic activity. Due to the fact that human faces are easily detectable and contain relatively large regions of unobstructed skin, camera based methods for heart rate measurement typically focus on collecting signal from faces. Further, camera requirements for color based methods are not stringent, and most smartphone cameras are capable of providing sufficiently high-resolution video at acceptable frame rates. Thus, this presents an opportunity for the measurement of human heart rate as easily as taking a short video of a subject's face.

At first glance, the development of a system for heart rate measurement using smartphones given the widespread availability of consumer wearable heart rate monitoring devices may seem redundant. However, it is important to note that there are multiple factors which make smartphone-based heart rate measurement more viable than wearable heart rate estimation. One such factor is cost and convenience. Many people already possess smartphones and thus can obtain heart rate measure-

ments simply by downloading software. This eliminates the need to purchase wearable devices, which are often expensive. Further, the problem of wearable device adoption and retention has garnered significant attention (25). Specifically, this means that even amongst the population of purchasers of wearable heart rate monitoring devices, many patients ultimately abandon the device for reasons including convenience, appearance, and comfort (25). Thus, this means smartphone-based heart rate management has a far larger population of potential users as it reducers barriers to entry relating to cost and convenience and does not require users to remember to don devices they may find uncomfortable or unflattering. Smartphones are far more ubiquitous than wearable devices, and the population of smartphone-owners is also the population of potential users of smartphone-based heart rate measurement systems.

1.2.4. Mood Triggers: Coupling psychiatric monitoring and heart rate

Under the direction of Professor Nicholas Jacobson, Dartmouth's AIM HIGH Lab is currently developing Mood Triggers, an Android app designed to help users identify sources of anxiety and depression and couple user inputs with physiological and behavioral data such as number of steps and sleep duration. Due to the connection between heart rate, HRV and conditions such as anxiety and depressive disorders, the ability to integrate remote, non-contact heart rate measurement with the app is highly desirable. Further, passive mobile phone sensing, or the collection of patient information without the need for user intervention, offers additional benefits as it allows for data sensing without dependence on active user participation. Given the potential of camera-based heart rate measurement architectures which can be implemented to work passively on smartphones, we opted to build such a system so that it may be deployed on mobile apps such as Mood Triggers.

Section 1.3

Related work

Heart rate measurement using smartphone cameras has received a lot of attention in recent years. This method of using cameras for estimating heart rate is commonly referred to as remote photoplethysmography (rPPG), as opposed to photoplethysmography (PPG), a common clinical method for computing heart rate. Timeseries mimicking PPG signal, from which heart rate can be derived, can be produced from videos of subject faces using either deep learning or transformations of the red, green, or blue (RGB) color channels obtained from video. This artificial, remote PPG signal will henceforth be referred to as rPPG signal. Of the RGB channels, many papers find that the green color channel is most useful for yielding rPPG signal (26) (27). However, numerous other studies disagree with this notion and find success using combinations of the three channels (28) (29) (30).

Given the diverse array of potential applications of remote heart rate measurement, there is a wide range of approaches to producing rPPG signal from video. For instance, a lot of recent work has turned its attention to the use of deep learning for the estimation of heart rate signals. Hasan et al, who released the MPSC-rPPG dataset used in this work, introduce a multi-task learning architecture for producing rPPG signal (1). Concerned more with the generation of realistic rPPG signal than directly computing heart rate, Hasan et al were able to produce smooth rPPG signal. However, their model requires fine-tuning on data from a given subject before it can reliably produce realistic signal for that subject. As their work uses the same dataset as this thesis, Hasan et al's approach was replicated for comparison to our framework (1).

Several studies applying advanced signal processing techniques to the production

of rPPG signal exist. Given the lesser reliance of papers in this category on complex machine learning models, many offer useful procedures for generating rPPG signal. Seminal to the field of remote heart rate measurement from video, authors Ming-Zher Poh, Daniel McDuff, and Rosalind Picard published two studies in 2010 (31) (28). First, they introduced the use of blind source separation to measure heart rate signal from video and were the first to achieve remote and non-contact heart rate measurement (31). This was a watershed paper which introduced new possibilities for the field of remote heart rate measurement. In that work, they also show that the calculation of the heart rate for multiple individuals in a given video is possible (31). In their followup paper, they extend the work introduced in (31) by altering the signal processing pipeline so that it is capable of computing metrics such as HRV and respiratory rate (28). In (28), Poh et al delineate a signal processing procedure for generating rPPG signal, which includes bandpass filtering for removing low and high frequency noise and computation of interbeat intervals (IBIs), which are defined as the time in between adjacent heart beats (28). Further, Poh et al apply a filter to remove noise attributable to motion at this step, incorporating the noncausal of variable threshold (NC-VT) algorithm (28). The NC-VT algorithm was originally proposed for processing signal before computing HRV, and was a relevant choice for Poh et al (32). In 2016, Wang et al attempted to aggregate knowledge gained from the numerous algorithms introduced for heart rate estimation with signal processing and unify them into a coherent mathematical framework (30). The result was "Algorithmic principles of remote PPG," which introduces methods for addressing skin reflectance, dimensionality reduction of color channels obtained from skin, and a unified algorithm named "plane-orthogonal-to-skin" (POS) (30). Of the signal processing approaches aggregated by Wang et al, the chrominance-based algorithm proposed by De Haan and Jeanne is particularly notable in its reach and its reported

performance (29). Incorporating all three color channels and an algorithm which weights color channels based on analytical findings regarding their relationship to ambient light, De Haan and Jeanne bring forth an important class of signal processing-based heart rate estimation frameworks (29).

Many papers specifically concern the use of smartphone cameras for the estimation of heart rate, given that modern phone cameras typically possess the necessary frame rates and resolutions for the generation of rPPG signal. In 2021, Qiao et al introduced a system for measuring heart rate based on denoising filtering and independent component analysis (ICA) (33). In 2019, Yu et al introduced a successful end-to-end deep learning approach for producing rPPG signal from face video and the subsequent calculation of heart rate (34). Also in 2019, Yu et al contributed an architecture directly addressing the problem of recreating BVP signal as closely as possible using deep spatio-temporal networks (35). This paper specifically targeted improvements in estimating HRV, as many previous works had focused entirely on outputting average heart rates. Producing only heart rate is insufficient for calculating HRV as HRV must be calculated directly from the interbeat intervals between individual heart rates. Further, Qiu et all introduced EVM-CNN, an architecture for remote heart rate calculation marrying Eulerian video magnification and a convolutional neural network (CNN) to output heart rate signal (36).

Another consideration gaining attention from researchers is the effect of motion, inconsistent or low lighting, and other noise on rPPG heart rate measurement. Specifically addressing these conditions is often referred to as measuring heart rate "under realistic conditions." In 2014, Li et al note that the performance of heart rate measurement systems from face videos decreases significantly in the presence of subject motion or variation in the degree of illumination (27). To mitigate these problems, they propose an architecture which builds upon typical face tracking methods and

applies normalized least mean square adaptive filtering to produce signal from which heart rate can be computed (27). Likewise, Lam and Kuno propose an algorithm with considerations made for motion and inconsistent illumination that selects suitable skin regions for rPPG signal generation (37). Tulyakov et al also cite the challenges brought forth by attempting heart rate measurement under realistic conditions. They introduce a framework for applying state-of-the-art matrix completion theory to identify the most optimal patches of skin for measuring heart rate simultaneously with the heart rate computation itself (38). However, it is important to note that many of these approaches may not be compatible with heart rate estimation locally on smartphones due to the size of the networks applied. Ultimately, the more limited processing power and RAM of smartphones must be considered if all heart rate computation is to be performed locally.

Some examples of the use of heart rate measurement systems on smartphones exist, but none explicitly perform all computation locally. Kwon et al introduce a system for heart rate extraction using smartphone cameras, but the system is not packaged into fully compatible smartphone software (26). Further, Maestre-Rendon et al develop an iOS app applying advanced signal processing techniques for heart rate calculation, but do not specify if they were successful in performing all the required steps locally on iOS smartphones (39).

Despite the reported success of the approaches noted above, most are difficult to replicate. In some cases, this is due to strict controls over the availability of data due to the sensitive nature of videos of human faces. In others, such as the system proposed by Poh et al, entire procedures are ill-defined or not defined at all (28). Some fail to address the effect skin tone may have on the performance of the proposed algorithms. For example, the heart rate measurement framework introduced by Maki et al utilizes the TokyoTech rPPG dataset (40). However, though the age

and gender breakdown of subjects is reported, variation of skin tone is not. Ernst et al propose an algorithm for mixing color channels to mitigate these differences (41). Further, Lee et al propose a deep learning architecture which uses self-supervised active learning to address potential performance discrepancies due to skin tone (42). However, it is also important to note that not all studies report finding significant performance differences in heart rate estimation between different skin tones. For example, Shirbani et al studied the effect of ambient lighting on skin tone and heart rate estimation and found that their approach only yielded significant performance discrepancies for different skin tones in low lighting (43). De Haan and Jeanne test their framework on a dataset containing subjects with a wide range of skin tones and found it was robust to variation in skin tone (29). Likewise, the POS algorithm introduced by Wang et al was also accurate on a range of skin tones (30). Thus, though some works in this field fail to address the impact skin tone may have on heart rate measurement performance, many state-of-the-art frameworks have also proven to be robust to skin tone differences.

Section 1.4

Data

This study utilized the MPSC-RPPG dataset (1). The dataset is comprised of 7 subjects (6 men, 1 woman) of varying age, skin tone, background, face movement, and presence of glasses (1). Skin tone of participants varied from lighter to darker skin tones, though the data were skewed towards those with darker skin tones. Videos were 5 minutes long and synced with ground truth blood volume pulse (BVP) data, from each subject's dominant wrist (1). BVP signal is PPG signal with high frequency noise filtered out. Given the noise filtering applied in this work, the rPPG signal produced by our model is directly comparable to BVP. Videos were captured at 30

frames per second under artificial ambient lighting conditions which varied slightly across each video (1). Wrist PPG data was sampled at 64 Hz. Further, the dataset included notes on how to align ground truth BVP with video. However, these notes were difficult to ingest automatically and thus were programmed directly into the model training and testing process. The data also include direct heart rate and inter-heart beat interval (IBI) ground truth, but insufficient clarity was provided to align these data with the video. Thus, ground truth BVP was used for all testing and validation.

The limited size of this dataset is noted and efforts were made to find options with more subjects. The sensitive nature of facial video and accompanying BVP data make them difficult to obtain. The data used for this work were the best available at the time.

1.4.1. Data augmentation

In an effort to augment this dataset, 3 different transformations were applied to subject video at the frame level: Gaussian noise, dimmed brightness, and increased brightness. This yielded a larger dataset containing 4 instances of each subject video: the original video, and a version for each of the transformations enumerated above. For the Gaussian noise transformation, noise was added to each frame with mean 0 and standard deviation 30. For the image dimming and brightening transformations, the brightness was reduced and increased by an intensity of 60. Further discussion of the effect of the augmented data on the framework presented here can be found in the Results and Discussion chapters.

Section 1.5

Content and organization

This thesis introduces a novel heart rate estimation framework which emphasizes efficient use of computational resources for end-to-end heart rate measurement locally on smartphones. The enumeration of this system will proceed as follows. It will introduce the overall system architecture and technologies used. Then, it will explore the simple Android application used for testing the local heart rate estimation procedure, outline the facial recognition and tracking methods applied, and detail the employed signal processing techniques. Finally, it will conclude with a reporting of results and subsequent discussion of model performance, the effect of the data augmentation approach on model performance, and the performance of the baseline implementation taken from (1).

Chapter 2

Architecture

Section 2.1

Overview

The framework introduced in this thesis follows a similar structure to existing architectures which apply signal processing techniques to measure heart rate rather than more complex deep learning methods. The procedure begins with capturing face video within a mobile application implemented for Android smartphones. Following video capture, the file is saved locally and analyzed by the face detection and tracking system. The face tracking system uses the YoloV5 facial recognition model to isolate a region of interest (ROI) within the subject's face and aggregates color channel data for each frame (44). To enhance the speed of this face detection and color channel data collection, a framewise object tracker was used to update the position of the ROI bounding box in successive frames without having to apply the YoloV5 model to each frame (45). For the purpose of this work, the forehead was chosen as the target ROI for all subjects due to its size and ease of detection. This produces a matrix of size F x 3, where F refers to the number of frames captured in the video, and 3 refers to the 3 color channels collected. To optimize for user privacy to the greatest extent

possible, the video is deleted at this time, leaving only the raw RGB data. The signal processing step consists of preprocessing the raw RGB channels, feature engineering, regression via a lightweight gradient boosting model to output rPPG signal, and final postprocessing. Heart rate and HRV can then be computed from both the rPPG signal generated by our pipeline and the ground truth BVP signal provided by the dataset to validate the model.

The signal processing steps applied before and after the data are fed to the gradient boosting regressor closely follow the steps outlined by Poh et al in (28). Any deviations from that framework are noted in this chapter. The primary contribution of this thesis is the lightweight pipeline for heart rate estimation presented here. The architecture has been named HRMobile, and encompasses the entire system from video capture to the calculation of user heart rate.

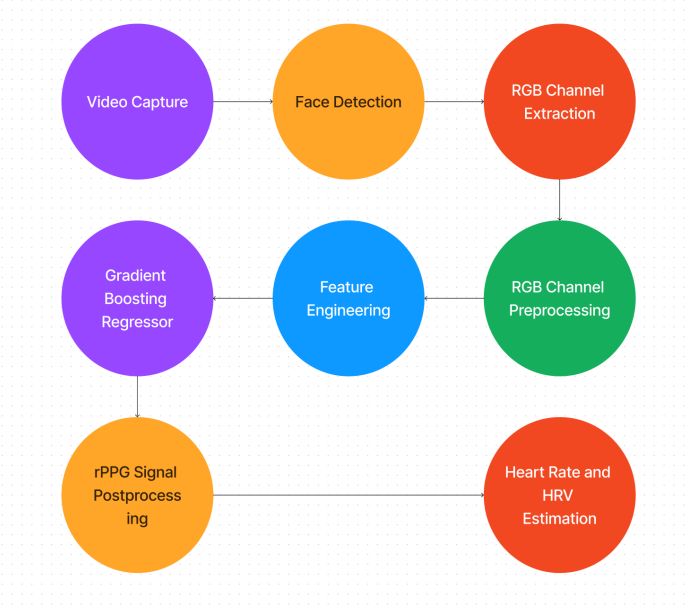

Figure 1: Diagram of the HRMobile architecture.

Section 2.2

Mobile application

This work was completed as part of the Mood Triggers project within Professor Nicholas Jacobson's AIM HIGH Lab. A prototype mobile app was developed to test this framework before incorporating it into the primary Mood Triggers application. The test app was developed for use on Android smartphones using Flutter, an open-source mobile application framework developed by Google. It was important to use this technology stack as it enabled the implementation of a custom Flutter plugin for running Python code within an Android app. Given the machine learning components inherent to HRMobile, the ability to run Python code was essential. Thus, it was possible to implement our framework in such a manner where all computation was performed locally within the Android app.

Section 2.3

Facial recognition and tracking

The proposed heart rate calculation architecture applies existing computer vision models for facial recognition within smartphone video. As has been emphasized thus far, efficient use of computational resources was important to this project. Given that facial recognition was one of the greatest potential sources of complexity for our proposed framework, particular attention was paid to selecting a lightweight model for facial recognition. The YoloV5 facial recognition model was selected for this project due to its performance and optimization for use in mobile phones (44). TensorFlow, a Python machine learning framework developed by Google, was used to deliver YoloV5 within our mobile app (46). In practice, the model was applied at the frame level to return bounding boxes for each detected face within a frame. This framewise facial

detection procedure allowed for flexibility with regards to the number of frames to which facial detection was applied. To accelerate the RGB channel data collection process, the facial recognition model was not applied at each frame. Instead, facial recognition was applied to frames at a regular interval to return bounding boxes for the foreahead ROI, and ROI bounding boxes for intermediate frames were returned via the object tracking algorithm proposed by Danelljan et al, which can return ROI boxes faster than YoloV5 when given an initial bounding box (47). The DLib Python machine learning library's implementation of Danelljan et al's work was used for this step (45). Given the intent for the HRMobile framework to deliver heart rate measurement capability within a smartphone, the decision to use the object tracking algorithm was prudent to reduce the time complexity of the process.

The process for raw RGB extraction from facial video proceeds as follows. The OpenCV Python computer vision library is employed for general image processing tasks (48). After the video is saved locally, the face detection process begins. The procedure iterates through frames until a face is identified. For all frames, the algorithm assumes a single user's face will be present in the video. To ensure the system is not susceptible to infinitely searching for faces in video when there are none to be detected, an iteration limit is set, whereby if no face is found within the limit of frames, the procedure exits and reports failure to detect faces. Once a face is detected and a bounding box for the face is returned, forehead detection is applied to return a bounding box specifically for a clear region of the user's forehead. Python's *face recognition* package, a wrapper for DLib, was used to identify facial landmarks such as the forehead (45). This forehead bounding box is then used as the target ROI for RGB channel data extraction.

The use of the DLib object tracker accelerated the speed of RGB data extraction and introduces flexibility with regards to how often the YoloV5 facial recognition

model is run (44) (45). This allows for an optional "face renewal" parameter, P to be set, meaning YoloV5 can be run every P frames as the procedure iterates through the video. A lower value for P means the raw RGB channel collection process will have greater time complexity, but may also have less potential for noise introduced by error from the object tracker. Each time the facial recognition model is run for a given frame, the DLib tracker for tracking the ROI bounding box is re-initialized to keep it up-to-date. For each frame to which the facial recognition model is not applied, the position of the facial bounding box is inferred using the object tracker (45).

Algorithm 1 details the high-level procedure for obtaining raw RGB channel data using iterative facial recognition and object tracking through the duration of a given video. For each frame, it either uses the YoloV5 facial recognition model or the DLib object tracker to obtain a bounding box for the forehead ROI for the current frame and collect the raw RGB channel data from the frame in question. To collect raw RGB channel data from a given frame, the image is first cropped according to the rectangular coordinates defined in the corresponding bounding box. After cropping the image, each individual color channel is separated and spatially averaged to obtain a single framewise value per channel. This method for generating RGB channels via spatially averaged ROI was first introduced by Poh et al in (28). Following the procedure outlined in Algorithm 1, the video can be safely deleted to help enhance privacy.

Algorithm 1 GetRGBChannelData. Given captured video of face, detect forehead boundaries, initialize ROI tracker, and collect raw RGB channels for each frame through spatial averaging.

Input: captured face video, "face renewal" parameter P
Returns: Raw RGB channel data

$i \leftarrow 0$
$channels \leftarrow []$

for $frame \in video$ **do**

  if $i \mod P = 0$ **then**
    $bbox \leftarrow findForeheadBbox(frame)$
    $tracker \leftarrow initializeNewTracker(frame, bbox)$

  else
    $bbox \leftarrow getBboxFromTracker(tracker, frame)$

  end if

  $croppedFrame \leftarrow cropFrameUsingBbox(frame, bbox)$
  $r, g, b \leftarrow spatiallyAverageEachChannelIndividually(croppedFrame)$
  $channels.append((r, g, b))$
end for
return $channels$

Section 2.4

Signal processing

The HRMobile signal processing procedure for translating raw RGB channel data to heart rate saw many iterations. Given the preference in this project for solutions which rely primarily on signal processing to limit computational complexity, this work is greatly influenced by the framework introduced by Poh et al (28). As a result, the proposed data preprocessing and feature engineering steps are guided by those employed by Poh et al (28). Unfortunately, several key procedures identified in their work are ill-defined, and thus make their exact pipeline difficult to replicate. However, several steps enumerated in (28) proved useful. Lack of reproducibility is common across many proposed heart rate measurement architectures in the field, and

thus our approach blends ideas shared by these works to produce an outline which is more transparent and easier to implement.

The RGB values generated in the previous step consist of 3 features, one for each color channel, where each sample corresponds to a video frame. Each color channel value represents the spatial average of the channel in question for the forehead of the user (28). Several preprocessing tasks were applied to the raw color channels before a feature engineering step.

2.4.1. RGB data preprocessing

The data preprocessing steps presented here are inspired by those of Poh et al, but diverge at certain key junctions (28). Our three RGB color channel features are defined as follows: $r(t)$, $g(t)$, and $b(t)$, where t refers to the frame number (28). First, each feature is detrended using a 3rd degree polynomial (28). Then, each feature is normalized using the following formula, which is also enumerated in (28):

$$v'(t) = \frac{v(t) - \mu_{v(t)}}{\sigma_{v(t)}} \tag{1}$$

where $v(t)$ corresponds to any of $r(t)$, $g(t)$, or $b(t)$ (28).

Further preprocessing steps were considered. One important observation about the RGB channel data is that they are quite noisy. To address this, two denoising techniques were tested. The first was a bandpass filter which wrapped the Scipy Python library's butter filter (49). The bandpass filter is another preprocessing step employed by Poh et al (28). However, Bousefsaf et al report success using wavelet filtering to clean rPPG signal before measuring heart rate (50). Wavelet transforms are a class of signal processing algorithms that can be useful for denoising (51). One feature of Wavelet transforms is that they can be applied iteratively to reduce a certain amount of noise from a given signal at a time. Algorithm 2 presents a

denoising method utilizing the PyWavelets module to experiment with applying the different wavelet methods at different depths (51). The algorithm iteratively applies a wavelet filter to the given signal array. Following this process, the algorithm linearly interpolates the signal before returning so that the returned signal contains the same number of samples as the original.

Given the focus of this project on efficient use of computational resources, the application of both the bandpass and wavelets filters during the data preprocessing step was deemed redundant. Following experimental training runs applying each method individually, applying a wavelet filter using the Daubechies 2 wavelet with 1 iteration was selected to serve as the denoising step within the data preprocessing pipeline.

Algorithm 2 WaveletFilter. Filter given signal using 'DB2' wavelet with *depth* number of iterations.

Input: signal array to be denoised: *signal*, number of iterations to apply wavelet: *depth*
Returns: filtered signal as array

$sig \leftarrow signal.copy()$
$origLength \leftarrow sig.length$

for $d \in 1 \dots depth$ **do**
    $sig \leftarrow applyWavelet(sig, \text{“}db2\text{”}, d)$ ▷ PyWavelet wavelet transform function
end for
$sig \leftarrow linearInterpolation(sig, origLength)$ ▷ linearly interpolate *sig* back to original length
return *sig*

2.4.2. Feature engineering

Base features. The gradient boosting model does not have an inherent conception of time or periodicity. This was intentional as I was concerned about overfitting an explicitly temporal model such as a recurrent neural network to specific temporal patterns exhibited in the small subject set which would not generalize well to the

population at large. However, it was deemed important for the gradient boosting model to be able to have some concept of the previous state of the color signals. Thus, the *velocity*, *acceleration*, and *memory* features defined below represent an effort to encapsulate the temporal nature of the data in the features, rather than the model architecture itself.

Following the data preprocessing tasks, a feature engineering step was included to expand the information fed to the gradient boosting regression model. First, raw RGB signals are upsampled from the video frame rate of 30 to the ground truth sample rate of 64 to better align the training data and targets. This step simplifies the model testing process given the necessity of accurately aligning ground truth signal with signal generated from video. The following features were generated directly from the raw RGB channels: *velocity*, *acceleration*, and *memory* features for each channel. *Velocity* features approximate the first derivative of each channel by taking the difference between adjacent samples in each channel. Thus, there are three velocity features. *Acceleration* features approximate the second derivative of each channel by twice differencing adjacent samples. Thus, there are also three acceleration features.

Memory features are parametric. Memory features are lagged versions of each of the color channels. Thus, there were N memory features for each color channel, and each represented a lag amount L. This meant that the first memory feature for color channel c would lag L samples behind the current value of c, the second memory feature for c would lag $2L$ samples behind the current value of c, etc. The values of N and L had to be experimentally derived. This process is enumerated further in the Hyperparameter Optimization section. Following experimentation, it was determined that there would be 8 such memory features for each color channel c, named $\{c\}\_mem\_feat0$ ... $\{c\}\_mem\_feat8$ respectively, and that each successive memory feature would carry a lag L of 12.

The raw color channels, velocity features, acceleration features, and memory features comprise the "base features." In total, there are 33 base features. With the exception of the color channels, the creation of each of these features shortens the raw signal. Thus, some samples had to be discarded from the raw data to maintain uniform length for all features. In practice, this involved discarding the first 2 samples of the raw RGB channels and the first sample of the velocity features to align the lengths of the raw RGB, velocity, and acceleration features. Then, the first M remaining samples were removed from each of these features so that their length matched the length of the memory features. $M = N \times L$, or the number of memory features multiplied by the lag amount for each memory feature. In total, this means that the first $8 \times 12 + 2 = 98$ samples are discarded in the feature engineering step. However, note that the initial upsampling step in data preprocessing linearly increased the raw data sampling rate from the expected camera frame rate of 30 to the ground truth sampling rate of 64, and thus these discarded samples account for only ~ 1.53 seconds of video.

Chrominance feature. De Haan and Jeanne introduce a chrominance-based heart rate measurement architecture which applies analytical formulas derived from the interaction of ambient light with raw RGB channels to produce rPPG signal (29). The chrominance algorithm proceeds as follows. Detrended and normalized RGB channels are passed to the method and are each further normalized to produce 3 signals: r_n, g_n, and b_n (29). They are then combined into two signals, x_s and y_s (29). Then, each of r_n, g_n, b_n, x_s, and y_s are bandpass filtered with frame rate 64, minimum bandpass frequency 0.7, and maximum bandpass frequency 4.0 (28). This produces the vectors r_f, g_f, b_f, x_f, y_f. Finally, the constant α is computed and used to combine the normalized and bandpass filtered color channels into the final rPPG output (29). Direct application of the chrominance algorithm's output to computing heart rate

was attempted, but the rPPG signal produced was too noisy to consistently and accurately measure heart rate. However, experiments with the chrominance rPPG signal as a feature indicated that it added useful information to the dataset and improved HRMobile's performance. Thus, the chrominance rPPG signal was added to the feature set for the HRMobile gradient boosting regressor.

ICA feature. ICA is a signal processing technique used in statistics and machine learning to separate signal into its constituent components. ICA is able to remove motion artifacts from raw RGB channels by separating color changes representing informative signal from noise (28). Following the normalization step in their heart rate estimation pipeline, Poh et al suggest decomposing normalized RGB channels using independent component analysis (ICA) (28). Poh et al apply ICA as an intermediate step in generating rPPG signal. They pair ICA with a selection step in which they choose the ICA component with the highest power spectrum peak for further processing (28). When applied in isolation, we found that the processing steps outlined by Poh et al, including the use of ICA, were insufficient for producing appropriately denoised signal for estimating heart rate. However, we found that the ICA component with the highest power spectrum peak did provide signal useful to the gradient boosting regression model. Thus, the ICA process outlined in (28) was adapted for the feature engineering step and included as a 34th feature for our model.

Section 2.5

Gradient boosting regression model

The model utilized for this architecture is a gradient boosting regressor which learns from tabular data and does not support a built-in conception of time or periodicity. This was a conscious decision given the limitations introduced by the size of the

dataset. We were wary of implementing an explicitly temporal or spatio-temporal model as we thought such an architecture may suffer performance degradation as a result of learning cyclical artifacts specific to the subjects in the dataset that may not be generalizable to the larger population. Thus, though some deep learning approaches are well-established for modeling problems such as ours, they were not considered for this project. However, the inclusion of some temporal information in the form of the *velocity*, *acceleration*, and *memory* features was deemed appropriate. This was also consistent with our intention to limit the complexity of HRMobile, as deep learning models tend to demand more computational resources than more traditional machine learning techniques.

It is important to also note that it was necessary to simplify the learning task at hand and focus on heart rate. Thus, the model was optimized for heart rate estimation, and was not directly concerned with accuracy in calculating HRV. However, the model's performance in estimating HRV is still included in the Results section for discussion. Furthermore, the model presented below is highly parametric and the values of these hyperparameters were determined through automated hyperparameter optimization and experimentation. All hyperparameters whose values are mentioned below to have been determined experimentally are elaborated on in the Hyperparameter Optimization section.

2.5.1. Model selection

Given the specifications of our modeling objective, two machine learning architectures which are compatible with our tabular dataset were investigated: gradient boosting and random forest regression. However, it is important to note that the chosen model was required to strike a difficult balance in that it was necessary to limit its ability to learn temporal artifacts specific to subjects in the small dataset but also optimize for the best possible ability to produce generalized rPPG signal. To achieve this, a custom

objective function was implemented for the model. This was necessary due to the fact that common loss functions such as mean squared error (MSE) are only indirectly relevant to the model's ability to accurately measure heart rate. This is because the calculation of heart rate from model output relies on a peak detection procedure and we were primarily concerned with the model's ability to produce signal with as similar a number of detected peaks as the ground truth as possible. Further, heart rate is computed from batches of signal corresponding to S seconds of video. Thus, this required a loss function which considered these batches of samples corresponding to S seconds of video, rather than sample-wise error. MSE concerns only the sample-wise differences between model output and ground truth, and thus does not necessarily penalize the model for producing signal with excessive noise that may confuse the peak detection algorithm.

The metric of primary concern was the model's error in estimating heart rate, and therefore a loss function which reflected this was required. Many out-of-the-box APIs offering gradient boosting and random forest regression do not support custom loss functions. However, the LightGBM and XGBoost Python libraries are popular options which implement both gradient boosting and random forest regression and offer the necessary features (52) (53). Crucially, these models have relatively low computational complexity and thus were good options for this project.

LightGBM was initially selected as the framework of choice. However, LightGBM's random forest implementation does not support custom objective functions. Following initial testing, a LightGBM gradient boosting regressor equipped with a custom objective function outperformed an equivalent LightGBM random forest regressor which used MSE loss, and thus gradient boosting regression became the machine learning model of choice. Further, it was discovered that XGBoost is more compatible with our Android mobile app than LightGBM. Given that XGBoost largely

offers the same set of features in use by this project as LightGBM, XGBoost gradient boosting regression with a custom loss function was selected for use over LightGBM. Going forward, only the XGBoost model implemented will be discussed.

2.5.2. Heart rate computation

The procedure for computing heart rate error was taken directly from Poh et al's work and proceeds as follows (28). The algorithm accepts two vectors: the ground truth signal and corresponding model predictions. For each, it applies a peak detection method to obtain signal peaks in both the ground truth and predicted signals. The procedure relies on the assumption that detected peaks in the signal correspond to heart beats. A minimum peak prominence threshold was experimentally derived in the Hyperparameter Optimization section to attempt to eliminate spurious peaks representing noise rather than heart beats. The procedure then computes interbeat intervals (IBIs) using the peaks detected in the ground truth and predicted signals, which represent the estimated time in between two heart beats, in seconds. Thus, if the IBI between two adjacent signal peaks is 0.5, this means that 0.5 seconds elapsed between those two heart beats. Finally, a heart rate estimate is computed by dividing 60 by the mean interbeat interval.

Algorithm 3 HeartRateError. Given a pair of vectors representing ground truth and model predictions respectively, return absolute model error.

Input: ground truth array: y_true; prediction array: y_pred; Integer sample rate: fr
Returns: Float absolute HR error

$true\_peaks \leftarrow find\_peaks(y\_true)$ ▷ outputs index locations of ground truth peaks
$pred\_peaks \leftarrow find\_peaks(y\_pred)$
$true\_ibis \leftarrow \frac{diff(true\_peaks)}{fr}$ ▷ interbeat intervals for true signal, taken by differencing the $true\_peaks$ array and dividing by fr
$pred\_ibis \leftarrow \frac{diff(pred\_peaks)}{fr}$
$true\_hr \leftarrow \frac{60}{average(true\_ibis)}$ ▷ true heart rate based on ground truth
$pred\_hr \leftarrow \frac{60}{average(pred\_ibis)}$
return $abs(true\_hr - pred\_hr)$

2.5.3. Learning objective

The custom objective for our modeling problem needed to satisfy three conditions to maintain compatibility with both XGBoost and the learning task: 1) it needed to be differentiable, 2) it needed to return the gradient vector, the first-order partial derivatives of the function with respect to the model's predictions, and the hessian, the diagonal of the square matrix of second-order partial derivatives of the function with respect to the model's predictions in the case of XGBoost, and 3) it needed to quantify the difference in predicted heart rate between generated time series from a sequence of samples and the corresponding ground truth. However, this presents two dilemmas. First, our method for measuring heart rate from a given rPPG signal is non-differentiable. Second, a custom loss function would be required to calculate the loss on batches of sequential samples and produce a batch-wise loss, rather than a sample-wise loss. This would require a complex training process. MSE could satisfy these requirements, but does not directly proxy the calculation of heart rate and thus was sub-optimal.

The differentiable loss function for comparing predicted and ground truth sig-

nals would ultimately need to be a proxy for heart rate calculation, given the non-differentiable nature of the heart rate estimation method. Though finding such a suitable proxy was nontrivial, Cuturi et al's Soft-DTW Algorithm was selected to fulfill this requirement (54). Soft-DTW is a differentiable version of the dynamic time warping algorithm for comparing timeseries. Given that Soft-DTW quantifies the similarity of two timeseries, it was a suitable choice for proxying our heart rate calculating function (54). Once Soft-DTW was identified, there remained the requirement for the custom loss function to return both the gradients and hessians of the function with respect to the prediction vector. However, the computational complexity of automatically differentiating the second-order derivatives of Soft-DTW proved to be too great, and were thus were omitted. This required an alternative approach to returning the diagonal of the hessian matrix. Returning a vector of ones in place of the hessian was a an option, but was suboptimal in that valuable information for the model training process would be lost. Instead, the solution was to ensemble the Soft-DTW function with the lower complexity MSE loss function. This procedure of combining two loss functions is a viable option in the case that one seeks to optimize for multiple objectives with a single model. Though the hessians returned from the MSE function are likely less useful than those of Soft-DTW, this was a suitable alternative since the hessian for Soft-DTW was unavailable. Thus, the hessian returned from the Soft-DTW function was set to a vector of ones and the returned gradients and hessians from both Soft-DTW and MSE were combined with a weighted sum for the final output of the loss function. The respective weights of the Soft-DTW and MSE components of the custom loss function were implemented as hyperparameters and determined experimentally.

Algorithm 4 CustomLoss. Return the gradient and hessian for the custom loss function.

Input: Ground truth array: y_true; predictions array: y_pred; MSE loss weight: mse_weight; DTW loss weight: dtw_weight; hyperparameter split size (integer): $split\_size$
Returns: Arrays tuple: $(gradient, hessian)$

$mse\_grad, mse\_hess \leftarrow mse\_loss\_function(y\_true, y\_pred)$
$dtw\_grad \leftarrow$ initialize array of zeros of length $y\_pred.length$
$dtw\_hess \leftarrow$ initialize array of zeros of length $y\_pred.length$

for i in $1 \ldots split\_size$ **do**
    $true\_curr \leftarrow y\_true[i * split\_size : (i+1) * split\_size]$
    $pred\_curr \leftarrow y\_pred[i * split\_size : (i+1) * split\_size]$
    $curr\_dtw\_grad, curr\_dtw\_hess \leftarrow dtw\_loss\_function(true\_curr, true\_pred)$ ▷ (54)
    $dtw\_grad[i * split\_size : (i+1) * split\_size] \leftarrow curr\_dtw\_grad$
    $dtw\_hess[i * split\_size : (i+1) * split\_size] \leftarrow curr\_dtw\_hess$
end for

$combined\_grad \leftarrow mse\_weight * mse\_grad + dtw\_weight * dtw\_grad$
$combined\_hess \leftarrow mse\_weight * mse\_hess + dtw\_weight * dtw\_hess$
return $(combined\_grad, combined\_hess)$

2.5.4. Model training

The custom loss function enumerated above necessitated the implementation of a batched training algorithm for the gradient boosting regression model. To achieve this, sequences of consecutive samples of length N for a given subject were aggregated. The specific length of these sequences were implemented as a hyperparameter and determined experimentally. These sequences of consecutive samples were then treated as collective units and the training-testing split was performed across these collections of samples, rather than between individual samples. Once the split was complete, the sample sequences were reconnected into training and testing sets, while preserving their order.

Following this initial batching step, there is an additional, nested batching step controlled by a *batches* hyperparameter, the value of which is determined experimen-

tally. This batches parameter could be set to a value greater than or equal to 1, and effectively repeats the batching step enumerated above, but only on the training set. This creates *batches* separate training sets. Going forward, these batches as controlled by the batches parameter and will be denoted as *sub-batches*. The reason for this framework is to allow for further tuning of the model for performance specifically on the non-differentiable heart rate estimation algorithm.

For each sub-batch, the model is trained using the custom loss function. Following this training step, the model's performance is evaluated via the non-differentiable heart rate error function. Then, the ground truth for the given sub-batch is replaced with the residuals formed from differencing the model's heart rate predictions and the ground truth for the current sub-batch. The current version of the model is now re-trained on these residuals without re-initializing its parameters. Finally, with each successive sub-batch, the model is trained on the new sub-batch starting with the parameters from the previous iteration. This procedure allowed us to tailor the model for performance in computing heart rate to the greatest extent possible in the absence of a differentiable method for computing heart rate. By fine-tuning the model on the predicted heart rate signal's residuals after each batched training step, we were able to penalize the model for poor performance in producing signal for accurate heart rate measurement *during* the training process, despite the fact that the heart rate function's non-differentiable nature precluded us from using it directly as a loss function.

Algorithm 5 GradientBoostingRegressorTrainingAlgorithm

Input: data as matrix: *data*, parameters as defined in Hyperparameter Optimization section
Returns: none

$splits \leftarrow package(data)$ ▷ Package dataset into array of sequential sequences of samples, each of length *split_size*
$training\_splits, test\_splits \leftarrow trainTestSplit(splits, test\_size)$ ▷ Randomly select splits to make up training and testing sets
$train\_sets \leftarrow []$ ▷ for holding *sub-batches* of the training set

for i in $0 \ldots batches$ **do** ▷ *batches hyperparameter*
  $batch\_size \leftarrow integer(\frac{training\_splits.length}{baches})$
  $curr\_train\_set \leftarrow selectBatch(training\_splits, barch\_size)$ ▷ Select *batch_size* splits from the set of training splits, combine them into a single training set, preserving the order of the splits, and remove the selected splits from *training_splits*
  $train\_sets.append(curr\_train\_set)$
end for

$model \leftarrow$ initialize XGBoost regressor with *hyperparams*

for $(train\_data, train\_labels)$ in $train\_sets$ **do**
  $model.train(train\_data, train\_labels)$
  $y\_pred \leftarrow model.predict(train\_data)$
  $y\_true \leftarrow train\_label$
  $num\_batch\_splits \leftarrow$ integer$(\frac{y_pred.length}{split_size})$
  $residuals \leftarrow$ initialize array of ones of length $y\_pred.length$

  for i in $1 \ldots num\_batch\_splits$ **do** ▷ Collect heart rate residuals for model finetuning
    $pred\_curr \leftarrow y\_pred[i * split\_size : (i+1) * split\_size]$
    $label\_curr \leftarrow y\_true[i * split\_size : (i+1) * split\_size]$
    $residuals[i * split\_size : (i+1) * split\_size] \leftarrow label\_curr - pred\_curr$
  end for
  $model.train(train\_data, residuals)$ ▷ Fine-tune the model on heart rate error
end for

2.5.5. Hyperparameter optimization

Many aspects of the model architecture presented here, such as the custom loss function and training process, are highly parametric. Further, the underlying XGBoost model is very customizable. It therefore would would have been difficult to test a suitable number of hyperparameter combinations manually. Thus, Bayesian hyper-

parameter optimization was applied to find performant combinations of parameters. The specific parameters tested include the following: *number of estimators*, the number of boosting iterations, *split size*, the number of consecutive samples selected in the first batching step of the training algorithm, *learning rate*, *early stopping rounds*, *MSE weight*, the weight attributed to MSE loss in the custom loss function, *DTW weight*, the weight attributed to Soft-DTW loss in the custom loss function, *batches*, the number of sub-batches applied during the training process, *predicted peaks prominence*, the minimum prominence for peaks detected in the non-differentiable heart rate function, *true peaks prominence*, *maximum depth*, the maximum tree depth in the model, *maximum bin*, the maximum number of bins inside which features will be bucketed, *number of samples per subject*, *number of memory features*, the number of memory features created during the feature engineering process, and *memory feature lag amount*, the number of samples skipped in the creation of each successive memory feature. The specific values yielded by the Bayesian optimizer are enumerated in Table 1.

Of particular note is the *split size* parameter. This is due to the fact that it also affects the patient experience in using a smartphone app equipped with this architecture. The split size determines the number of samples the system requires in order to optimally determine heart rate. The parameter value of 960 indicates that the optimal amount of video required of the user is 15 seconds. This is due to the fact that samples collected from video are upsampled to a frame rate of 64, the sample rate of the ground truth BVP signal, and thus there are $64 \times 15 = 960$ samples required for heart rate estimation. Note that the user may submit video longer than 15 seconds, but the system is designed to simply use 960 samples, if that many samples are available.

Table 1: Hyperparameters for gradient boosting regressor.

Hyperparameter	**Value**
Number of estimators	188
Split size	960
Learning rate	0.001
Early stopping rounds	16
MSE weight	0.2
DTW weight	0.8
Batches	5
Predicted peaks prominence	0.28
True peaks prominence	0.32
Max depth	6
Max bin	235
Number of samples per subject	3000
Number of memory features	8
Memory feature lag amount	12

2.5.6. Model evaluation

Model evaluation is performed via the non-differential heart rate estimation function. Though the model is not being trained to optimize this function directly, its performance on the heart rate estimation function is the primary objective of this work. Thus, for each boosting round of the gradient boosting model, the model's performance was measured by the heart rate estimation function. Early stopping was utilized to end end training early if heart rate estimation performance on the test set began to degrade. The specific early stopping value was optimized during the hyperparameter optimization step.

Subject-wise cross validation was employed to test the generalizability of the model to the greatest extent possible, given the limitations introduced by the number of subjects included in the dataset. In this cross-validation step, the model was trained on a dataset excluding a single subject at a time, and validated on a test set including only the subject excluded from the training data. This method provided the greatest possible insight into the model's performance on subjects not included in its training

dataset.

Chapter 3

Results

The HRMobile framework was successfully developed and implemented within an Android mobile app. The mobile app is capable of capturing video, detecting a user's face, and displaying heart rate and HRV measurements. Further, this architecture can be implemented passively for automated and frequent heart rate measurement without the requirement for user intervention.

Given the limitations introduced by the number of subjects included in the dataset, it was important to take steps to ensure the greatest possible generalizability of the model. Thus, hyperparameters such as the maximum tree depth and number of samples per subject were limited in an attempt to address the risk of overfitting. Moreover, we were sensitive to the possibility for the model to overfit on specific spatial or temporal factors inherent to one or more members of the subject population that may not be generalizable to the human population as a whole. Model validation was therefore performed with particular attention paid to the model's performance on subjects not seen in training.

Hasan et al, the authors of the dataset used in this work, analyze their model's performance in terms of their ability to accurately detect heart rate peaks in their model-generated rPPG signal (1). Thus, they report their error in terms of true

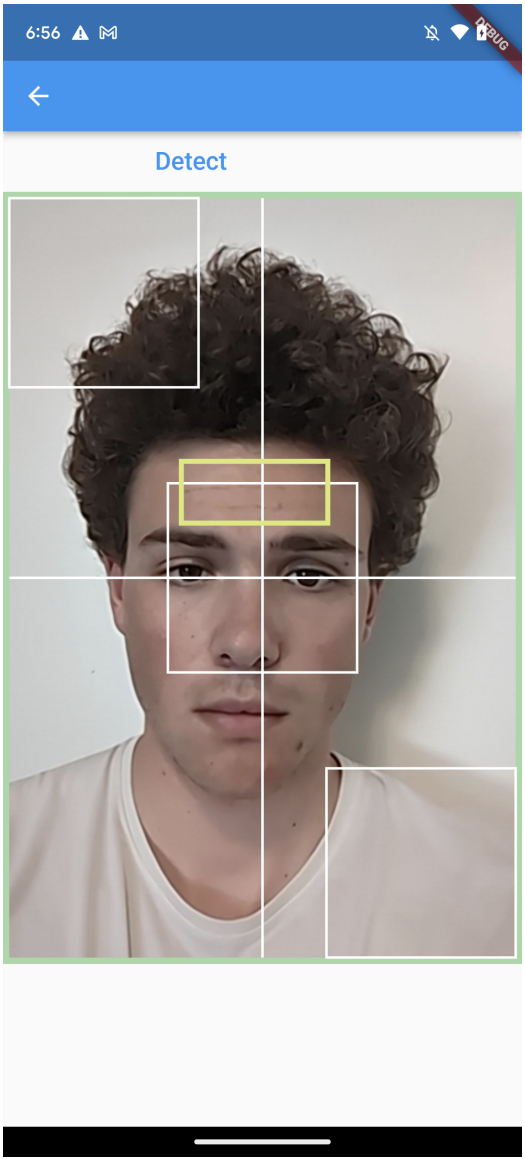

Figure 2: Forehead detection within the Android app described as part of the HRMobile architecture. First, the face is detected, and then facial landmarks are used to isolate the forehead. The forehead is denoted by the rectangle centered on the subject's forehead.

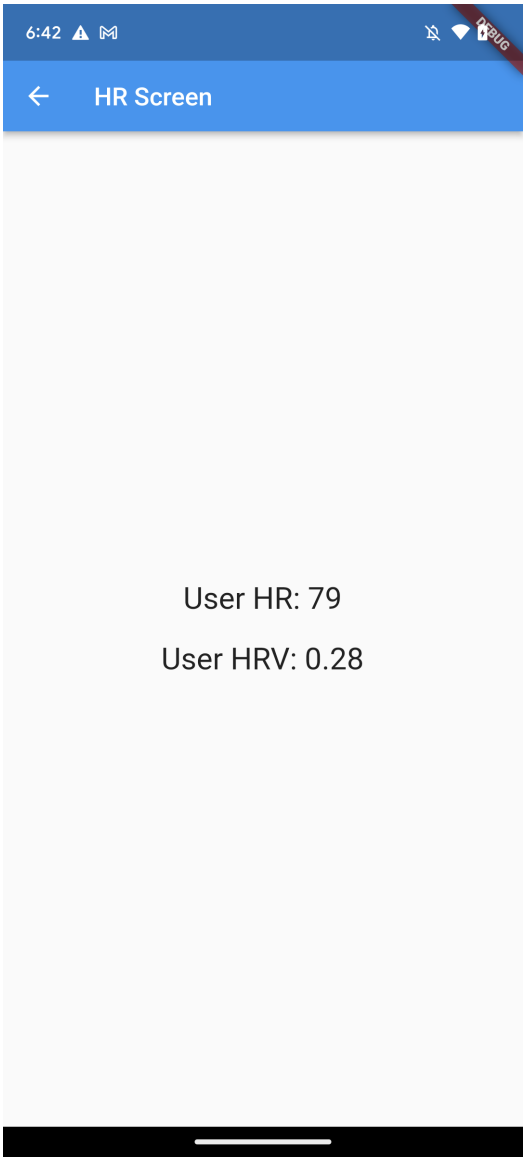

Figure 3: Heart rate display screen shown following the application of the HRMobile architecture to user face video.

positive, false positive, and false negative rates for peak detection. For our purposes, we felt that reporting error in terms of computed heart rate was more intuitive. Thus, we report mean absolute heart rate error and mean absolute HRV error. In accordance with (1), we also report the mean absolute difference in detected peaks in model-generated signal and ground truth signal.

Though HRV was not considered as part of the model training and validation processes, the model's performance with respect to HRV is noted in the results. HRV was calculated using the root mean square of successive differences (RMSSD) formula.

Section 3.1

Validation with augmented data

We augmented our dataset to include 3 frame-wise transformations of each subject video, in addition to the original version of each subject video. The three transformations were: adding Gaussian noise to each frame with mean 0 and standard deviation 30, dimming each frame by 60 units, and brightening each frame by 60 units. This yielded 4 groups of data in our augmented dataset. Augmenting the dataset in this manner allowed two experiments to occur. First, we could test whether adding these transformed versions of the data to the training set improved the accuracy of the model. Second, we could use these data transformations to simulate video taken in noisy or sub-optimal lighting conditions, and test model performance in these situations. Thus, 3 rounds of model cross-validation were performed: **(1)** cross-validation using just the original data for training, **(2)** cross-validation using the original data and Gaussian noise augmented data for training, and **(3)** cross-validation using the original data, Gaussian noise augmented data, dimmed data, and brightened data for training. In all three cases, testing was performed on all 4 groups of data during cross-validation, and results are reported for each.

Given that there were 7 subjects included in the dataset, subject-wise cross validation was applied by iteratively by training the model on 6 subjects and testing on the seventh. For each subject, we have 4 groups of data: the original video, video with Gaussian noise added to each frame, dimmed video, and brightened video. Thus, for the cross-validation step in which subject S is held out, the original data for subject S is excluded from training, as is each set of augmented data for subject S. For all other subjects, the inclusion or exclusion of each grouping of augmented data was determined by whether it was cross-validation run **(1)**, **(2)**, or **(3)**. For all cross-validation runs, all metrics were validated iteratively on 15-second batches of video for each test subject. This means that, for a given training run, model performance on a given subject is reported as an average for each error metric across all 15-second batches for that subject.

The validation results for the HRMobile framework indicate that there is room for improvement in its heart rate prediction accuracy prediction accuracy. The best heart rate validation error achieved by HRMobile was 16.64 bpm. Tables 2, 3, and 4 contain cross-validation results for each of the 3 cross-validation rounds respectively. For each cross-validation round, the model was tested on the original data in addition to the transformed data. Cross-validating the model on the transformed data in addition to original data allowed us to test the model's ability to perform when noise or imperfect lighting was introduced to the data. Each successive cross-validation round included more of the transformed data in training to test whether the addition of the transformed data to the training set improved model performance.

Tables 5, 6, and 7 convey the range in HRMobile's performance within a given subject. This is possible because cross-validation results per subject are averages of the model's performance across batches corresponding to 15s of video for a given subject. Sometimes, the variation in the model's performance across these batches

for a given subject is relatively small, but sometimes it may be quite large. The data presented in tables 5, 6, and 7 are drawn from the same cross-validation run as tables 2, 3, and 4, but simply represent a more granular look at HRMobile's performance at the subject level.

Error metric	Transformation applied	Result
MAE heart rate	None	16.64 bpm
MAE HRV	None	511 ms
MAE peak difference	None	4.11 peaks
MAE heart rate	Gaussian noise	19.82 bpm
MAE HRV	Gaussian noise	589 ms
MAE peak difference	Gaussian noise	5.39 peaks
MAE heart rate	Dimmed lighting	23.96 bpm
MAE HRV	Dimmed lighting	1106 ms
MAE peak difference	Dimmed lighting	7.10 peaks
MAE heart rate	Brightened lighting	27.51 bpm
MAE HRV	Brightened lighting	1445 ms
MAE peak difference	Brightened lighting	7.37 peaks

Table 2: Gradient boosting regression model cross-validation error when trained on the original dataset with augmented data used only for testing. Test results reported for both original data and all data augmentations.

Section 3.2

Baseline model

The deep learning architecture presented in (1) by Hasan et al was implemented as a benchmark model to compare to HRMobile. Hasan et al introduce both this architecture and the MPSC-rPPG dataset used in this study in (1). Hasan et al's architecture comes in two versions, both of which are implemented here. The first combines a convolutional neural network (CNN) with a fully connected multi-task learning (MTL) head for rPPG signal generation, and is trained and validated on data from a single subject. Thus, it is effectively a personalized rPPG generation

Error metric	Transformation applied	Result
MAE heart rate	None	19.26 bpm
MAE HRV	None	608 ms
MAE peak difference	None	5.12 peaks
MAE heart rate	Gaussian noise	23.50 bpm
MAE HRV	Gaussian noise	916 ms
MAE peak difference	Gaussian noise	6.35 peaks
MAE heart rate	Dimmed lighting	28.01 bpm
MAE HRV	Dimmed lighting	1223 ms
MAE peak difference	Dimmed lighting	9.41 peaks
MAE heart rate	Brightened lighting	29.32 bpm
MAE HRV	Brightened lighting	1167 ms
MAE peak difference	Brightened lighting	8.35 peaks

Table 3: Gradient boosting regression model cross-validation error when trained on the original dataset and Gaussian noise augmentation, but not for dimmed and brightened augmentations. Test results reported for both original data and all data augmentations.

Error metric	Transformation applied	Result
MAE heart rate	None	17.54 bpm
MAE HRV	None	603 ms
MAE peak difference	None	4.76 peaks
MAE heart rate	Gaussian noise	23.06 bpm
MAE HRV	Gaussian noise	733 ms
MAE peak difference	Gaussian noise	4.93 peaks
MAE heart rate	Dimmed lighting	23.74 bpm
MAE HRV	Dimmed lighting	1002 ms
MAE peak difference	Dimmed lighting	6.90 peaks
MAE heart rate	Brightened lighting	27.33 bpm
MAE HRV	Brightened lighting	1085 ms
MAE peak difference	Brightened lighting	6.78 peaks

Table 4: Gradient boosting regression model cross-validation error when trained on the original dataset, Gaussian noise augmentation, dimmed augmentation, and brightened augmentation. Test results reported for both original data and all data augmentations.

Transformation applied	Subject	Mean HR MAE	Mean HRV MAE	Mean Peak MAE
None	1	11.27 ± 16.11	0.47 ± 0.3	3.63 ± 4.39
	2	10.75 ± 7.27	0.5 ± 0.29	3.32 ± 3.01
	3	17.4 ± 13.2	0.47 ± 0.41	4.0 ± 3.09
	4	23.25 ± 7.04	0.32 ± 0.42	6.33 ± 2.49
	5	31.9 ± 15.02	0.61 ± 0.51	7.87 ± 3.46
	6	10.14 ± 8.22	0.41 ± 0.18	3.05 ± 2.1
	7	16.1 ± 10.19	0.77 ± 0.78	4.18 ± 2.79
Gaussian noise	1	19.47 ± 19.27	0.94 ± 0.71	5.41 ± 4.2
	2	13.94 ± 11.0	1.08 ± 1.24	4.23 ± 2.83
	3	17.08 ± 13.1	0.55 ± 0.5	3.42 ± 1.87
	4	19.87 ± 10.01	0.75 ± 0.49	6.0 ± 2.94
	5	33.32 ± 15.8	0.76 ± 0.7	9.0 ± 3.84
	6	11.19 ± 8.15	0.51 ± 0.21	3.18 ± 2.21
	7	19.21 ± 15.37	1.14 ± 1.67	4.05 ± 2.64
Dimmed lighting	1	26.17 ± 26.57	1.37 ± 1.23	6.37 ± 3.71
	2	18.5 ± 13.1	1.03 ± 0.96	5.59 ± 2.87
	3	20.13 ± 15.18	1.0 ± 0.97	5.16 ± 3.83
	4	13.46 ± 8.95	1.23 ± 1.21	3.33 ± 2.49
	5	58.35 ± 24.58	1.51 ± 1.33	19.0 ± 5.51
	6	16.83 ± 11.68	0.33 ± 0.28	4.86 ± 3.86
	7	16.22 ± 14.13	1.56 ± 1.5	4.36 ± 2.37
Brightened lighting	1	30.82 ± 19.38	1.55 ± 1.24	8.7 ± 4.41
	2	24.97 ± 15.03	1.84 ± 2.11	6.41 ± 3.93
	3	25.86 ± 23.15	1.09 ± 1.11	7.47 ± 3.5
	4	18.25 ± 4.91	0.92 ± 1.18	6.33 ± 0.94
	5	61.03 ± 15.7	1.45 ± 1.04	18.74 ± 3.69
	6	12.16 ± 10.58	0.53 ± 0.3	3.0 ± 2.71
	7	21.37 ± 16.07	0.66 ± 0.89	5.14 ± 2.93

Table 5: Mean and standard deviation (STD) validation error at the subject level when model is trained on original data and no augmentated data. Validation results shown for testing on original data and all augmentation sets. Mean and STD are taken across all batches of video for each subject, representing the range of HRMobile's accuracy within a single subject across batches.

Transformation applied	Subject	Mean HR MAE	Mean HRV MAE	Mean Peak MAE
None	1	12.19 ± 10.62	0.85 ± 0.82	3.07 ± 2.84
	2	7.45 ± 4.96	0.58 ± 0.29	1.91 ± 1.83
	3	15.71 ± 10.64	0.45 ± 0.4	4.0 ± 3.21
	4	23.65 ± 10.85	0.44 ± 0.16	6.67 ± 4.5
	5	40.39 ± 13.47	0.73 ± 0.43	10.52 ± 3.47
	6	16.38 ± 8.86	0.63 ± 0.32	4.59 ± 2.99
	7	19.05 ± 10.42	0.57 ± 0.47	5.09 ± 2.54
Gaussian noise	1	18.76 ± 15.44	1.11 ± 1.15	5.56 ± 2.97
	2	13.08 ± 10.28	0.74 ± 0.66	3.14 ± 2.26
	3	16.82 ± 17.41	1.18 ± 0.77	4.05 ± 3.5
	4	23.97 ± 12.91	0.13 ± 0.16	5.67 ± 2.36
	5	50.36 ± 10.79	0.99 ± 0.49	12.96 ± 2.79
	6	25.59 ± 20.68	1.45 ± 1.79	9.64 ± 5.35
	7	15.91 ± 9.47	0.82 ± 0.79	3.45 ± 2.29
Dimmed lighting	1	21.26 ± 19.44	0.72 ± 0.59	5.22 ± 4.01
	2	24.42 ± 15.98	1.79 ± 1.36	8.36 ± 3.27
	3	25.99 ± 18.27	1.65 ± 1.94	8.68 ± 3.57
	4	8.41 ± 8.94	1.18 ± 0.38	6.33 ± 1.89
	5	53.3 ± 26.6	1.24 ± 1.14	20.74 ± 4.24
	6	29.6 ± 16.91	1.48 ± 1.29	8.05 ± 4.41
	7	33.07 ± 23.17	0.51 ± 0.46	8.5 ± 5.85
Brightened lighting	1	22.79 ± 20.94	1.08 ± 1.06	5.74 ± 5.2
	2	25.11 ± 16.9	1.69 ± 1.59	7.5 ± 3.71
	3	29.14 ± 24.72	0.92 ± 1.24	7.68 ± 4.43
	4	15.73 ± 3.82	0.38 ± 0.19	5.33 ± 2.87
	5	69.4 ± 21.89	1.55 ± 1.33	20.09 ± 5.0
	6	25.2 ± 20.0	1.11 ± 1.5	6.77 ± 4.86
	7	17.84 ± 16.01	1.43 ± 2.16	5.32 ± 3.42

Table 6: Mean and STD validation error at the subject level when model is trained on original data and Gaussian noise augmentation. Validation results shown for testing on original data and all augmentation sets. Mean and STD are taken across all batches of video for each subject, representing the range of HRMobile's accuracy within a single subject across batches.

Transformation applied	Subject	Mean HR MAE	Mean HRV MAE	Mean Peak MAE
None	1	13.04 ± 14.69	0.6 ± 0.32	4.19 ± 3.34
	2	10.3 ± 9.11	0.71 ± 0.44	2.68 ± 2.1
	3	15.75 ± 10.04	0.47 ± 0.33	4.0 ± 3.91
	4	20.32 ± 9.52	0.66 ± 0.21	6.0 ± 2.45
	5	31.9 ± 14.35	0.62 ± 0.78	8.22 ± 3.94
	6	16.58 ± 10.46	0.72 ± 0.35	4.55 ± 2.59
	7	14.89 ± 10.58	0.44 ± 0.27	3.68 ± 2.82
Gaussian noise	1	20.03 ± 18.22	0.99 ± 0.85	5.48 ± 4.09
	2	14.12 ± 11.89	0.91 ± 0.87	3.86 ± 3.22
	3	21.42 ± 14.29	0.64 ± 0.58	4.95 ± 2.91
	4	37.59 ± 8.24	0.7 ± 0.45	3.0 ± 0.82
	5	34.21 ± 17.14	0.56 ± 0.31	8.78 ± 4.87
	6	19.54 ± 16.54	0.73 ± 0.62	5.18 ± 3.97
	7	14.49 ± 9.73	0.59 ± 0.37	3.27 ± 3.28
Dimmed lighting	1	22.58 ± 20.84	0.99 ± 0.79	6.0 ± 4.86
	2	23.12 ± 14.66	1.29 ± 1.25	6.55 ± 3.56
	3	20.75 ± 15.75	1.08 ± 0.8	5.47 ± 3.38
	4	7.88 ± 2.85	0.53 ± 0.15	4.67 ± 3.3
	5	53.59 ± 23.76	1.47 ± 1.53	16.96 ± 5.95
	6	13.65 ± 11.81	0.56 ± 0.32	3.95 ± 3.66
	7	24.59 ± 29.82	1.09 ± 0.77	4.68 ± 2.7
Brightened lighting	1	22.22 ± 16.05	1.0 ± 0.73	5.26 ± 3.95
	2	25.6 ± 17.16	1.21 ± 1.69	7.36 ± 4.08
	3	25.07 ± 19.56	0.83 ± 0.86	6.32 ± 4.32
	4	17.79 ± 6.82	0.7 ± 0.6	3.33 ± 2.05
	5	60.25 ± 27.04	1.86 ± 1.49	16.0 ± 6.54
	6	16.27 ± 11.64	0.76 ± 0.67	4.41 ± 3.74
	7	24.12 ± 16.17	1.23 ± 1.59	4.77 ± 3.23

Table 7: Mean and STD validation error at the subject level when model is trained on original data, Gaussian noise data augmentation, dimmed data augmentation, and brightened data augmentation. Validation results shown for testing on original data and all augmentation sets. Mean and STD are taken across all batches of video for each subject, representing the range of HRMobile's accuracy within a single subject across batches.

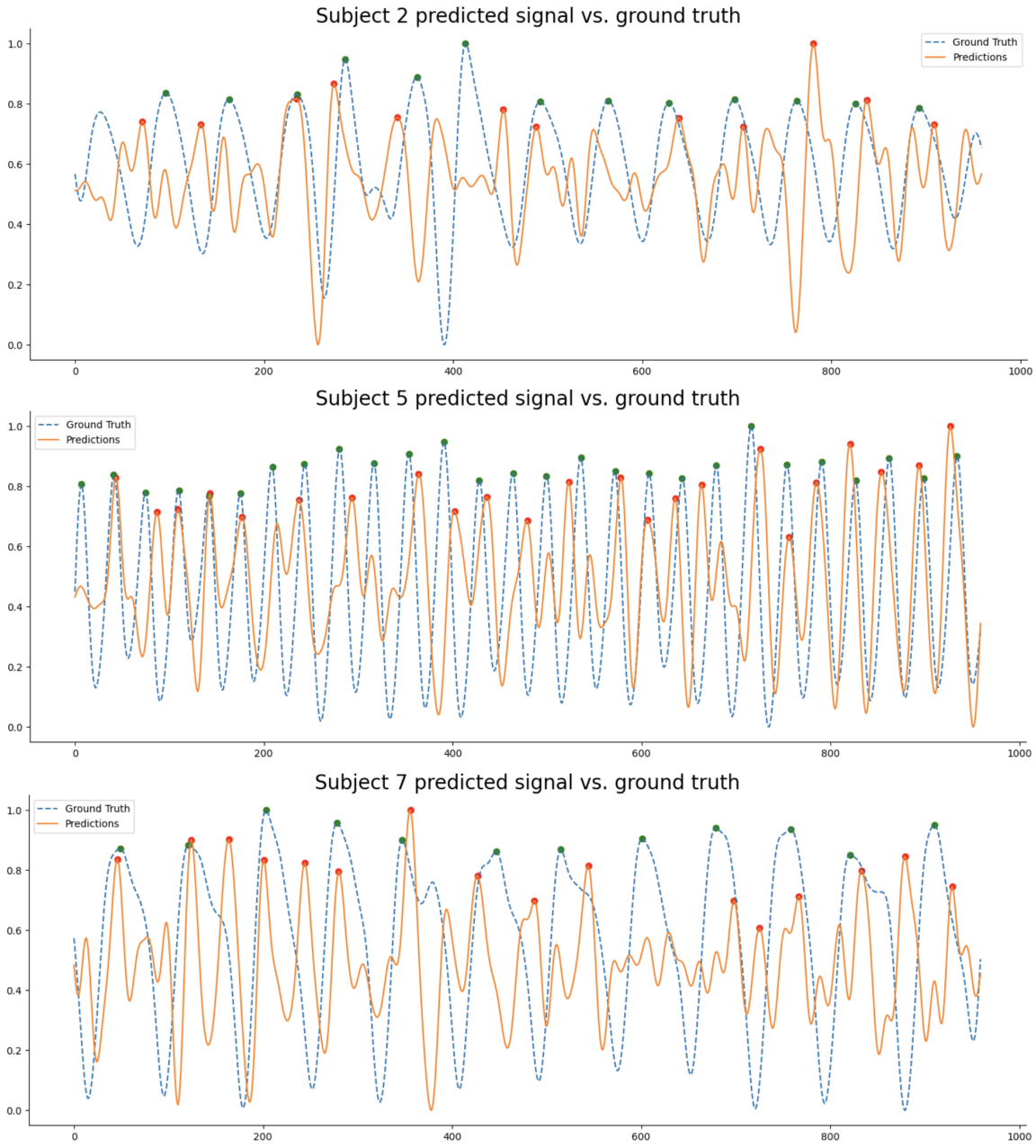

Figure 4: HRMobile generated rPPG signal versus ground truth BVP signal for three selected subjects for 15 second periods of video.

model for a given subject. This version of Hasan et al's work will be referenced as the Personalized Baseline henceforth. Of course, training and validating a personalized model for a single subject is not generalizable for application to any other subject. However, the Personalized Baselines for each subject generated very accurate rPPG signal, and thus are used to demonstrate an upper limit for possible accuracy of any model seeking to generate rPPG signal.

A second version of Hasan et al's work was implemented for direct comparison with the subject-wise cross-validation approach taken to test the accuracy of HRMobile. This approach will be referenced as the Generalized Baseline going forward. First, it is important to note that the Generalized Baseline discussed here is not a direct application of Hasan et al's work as they do not present any architecture which is capable of generating rPPG signal for a subject not included at any point in their training or fine-tuning steps. This is because they primarily concern themselves with building architectures capable of high degrees of accuracy, albeit with dependence on fine-tuning their model with ground truth with a subject before prediction is performed on that subject. Figure 5 depicts Hasan et al's architecture, showing their CNN with MTL heads for subjects 1 through N (1). Their most generalized version of this model still requires training on all subjects for which rPPG signal will be generated. This is because their more generalized architecture consists of the "shared" CNN model, paired with MTL head networks corresponding to each subject in its training set. This combined CNN-MTL model is trained by iteratively selecting batches from one subject at a time, and computing gradients for the "shared" CNN model and the MTL head corresponding to the current subject. The weights other subject MTL heads are frozen for this step. Predictions come from the MTL heads. Thus, to generate rPPG signal for subject S, video from subject S is fed to the "shared" network, the shared network output is passed to the MTL head

H_S corresponding to subject S, and model output is received from H_S. For our adaptation of this approach as our Generalized Baseline, we remove the MTL heads from this framework. Thus, we keep the training process and shared CNN intact, but predictions now come directly from the shared CNN. This allowed us to apply the same subject-wise cross-validation approach used to test HRMobile to Hasan et al's architecture, using our Generalized Baseline.

The Personalized and Generalized baselines adapted the Hasan et al architectures enumerated above (1). Each architecture was implemented with TensorFlow (46). Optimization was performed using a stochastic gradient descent optimizer with a learning rate of 0.005 for both models. Likewise, the combined MSE and sign loss function described by Hasan et al was utilized for both models (1). Convergence rates for both the Personalized and Generalized Baselines were found to differ from those described by the authors. Specifically, the Personalized Baseline models were trained with 12,000 iterations, and the Generalized Baseline models were trained with 1,000 iterations.

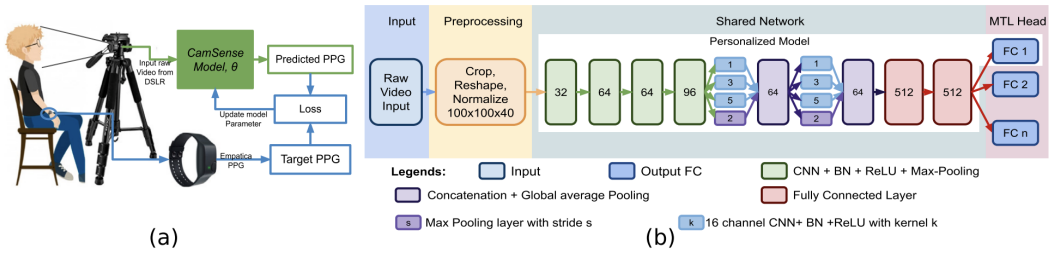

Figure 5: Architecture introduced by Hasan et al. Depicted are the shared CNN model connected with MTL heads for subjects 1 through N in the training data (1).

Example rPPG output from the Personalized Baseline models is shown in figures 6 and 7. Personalized Baseline output was found to produce the most accurate signal amongst the set of models implemented for this work, achieveing an average heart rate error of 5.21 bpm. This was expected, as the Personalized Baselines train Hasan

et al's framework on a single subject for prediction on that subject. It is the least generalized of the models implemented for this work, but the most accurate. To make these results directly comparable to those of HRMobile, the same error metrics reported for HRMobile's cross-validation (heart rate MAE, HRV MAE, and detected peaks MAE) are applied to both the Personalized Baseline and Generalized Baseline models. Tables 8 and 9 show the performance of these baseline models. Since the Personalized Baseline for each subject should only be applied to the subject it is trained on, cross-validation for the Personalized Baseline is not possible. Thus, we report the average for each error metric across the Personalized Baselines for each subject, as well as the individual performance of each Personalized Baseline.

Example rPPG output from the Generalized Baseline models is shown in figures 8 and 9. Generalized Baseline model performance varied significantly, achieving a mean heart rate error of 7.51 bpm at best and mean heart rate error of 62.32 bpm at worst. This skewed the average heart rate error for the Generalized Baseline approach to be worse than that of HRMobile's, achieveing an error of 26.31 bpm. Though this range in results is surprising, it could be explained by the composition of each cross-validation round and how representative it was of the tested subject for each round. Given the size of the dataset, it is possible that this could have an outsized impact on the cross-validation results. Corroborating this hypothesis is the fact that subject 5, for whom the generalized model performed the worse, had a notably higher ground truth heart rate than the rest of the subject population.

The RAM consumption and runtime were also measured for the Personalized and Generalized Baseline models for comparison to HRMobile's resource consumption. Both baseline models had comparable runtimes to HRMobile, but consumed far more RAM. Specifically, both the Personalized and Generalized Baselines typically consumed between 500 and 650 MB RAM.

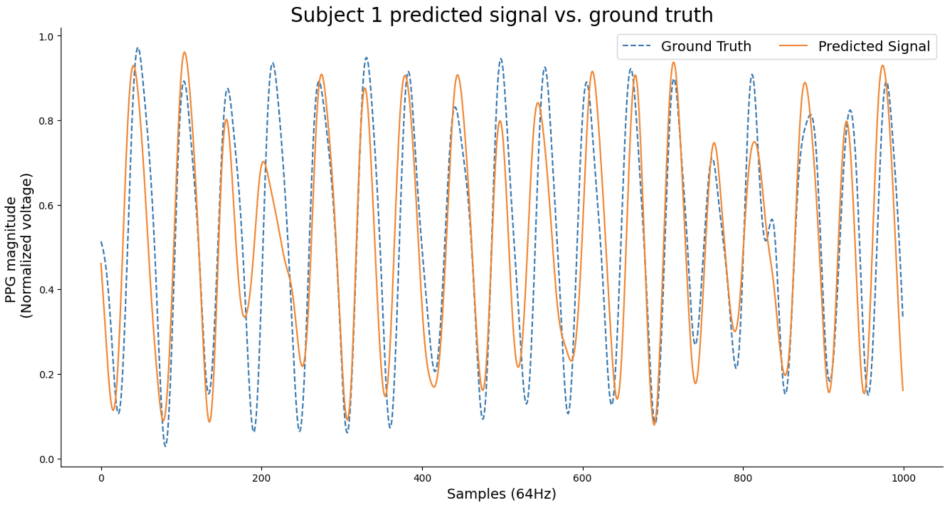

Figure 6: Personalized Baseline rPPG generated for subject 1.

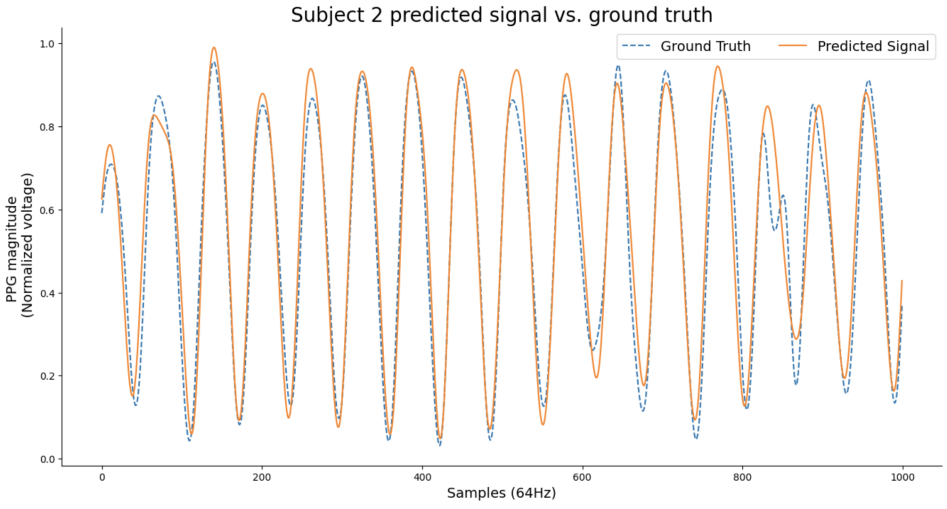

Figure 7: Personalized Baseline rPPG generated for subject 2.

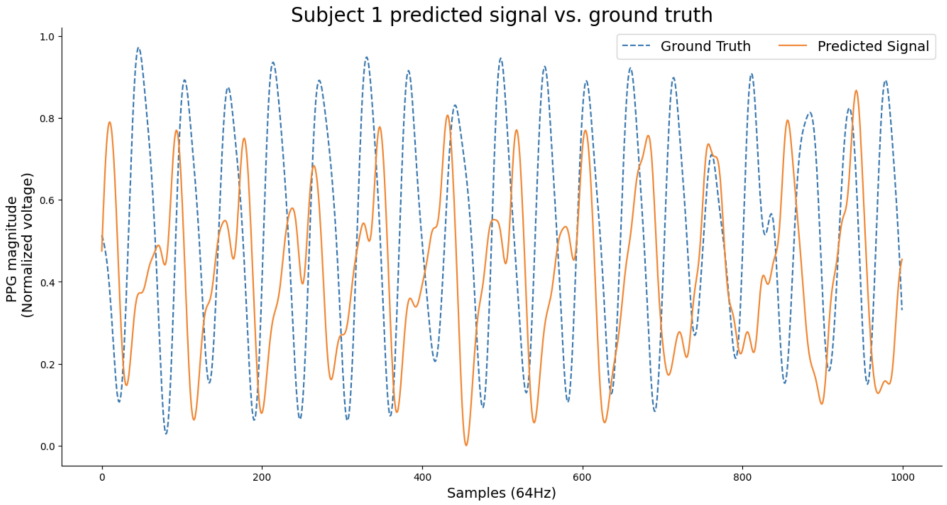

Figure 8: Generalized Baseline rPPG generated for subject 1.

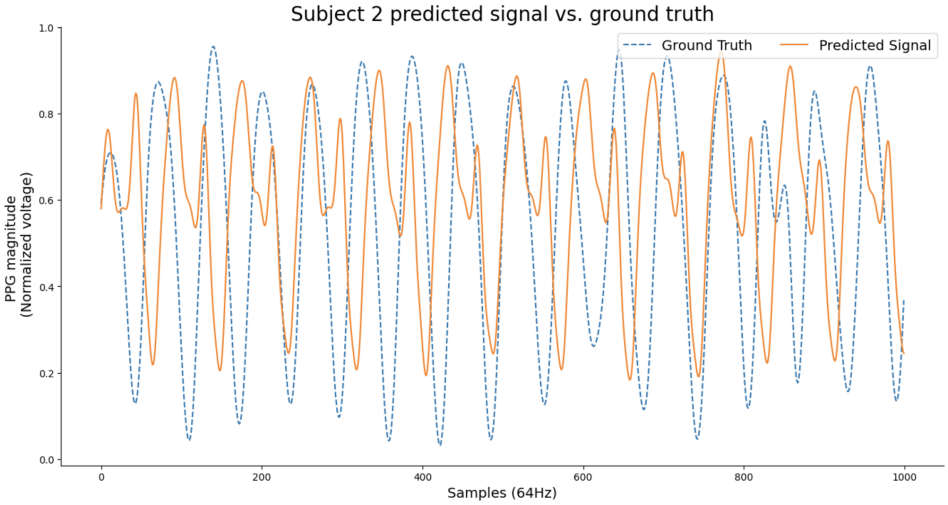

Figure 9: Generalized Baseline rPPG generated for subject 2.

Subject	Error metric	Result
Average	MAE heart rate	5.21 bpm
	MAE HRV	210 ms
	MAE peak difference	1.42 peaks
1	MAE heart rate	4.20 bpm
	MAE HRV	239 ms
	MAE peak difference	1.11 peaks
2	MAE heart rate	5.65 bpm
	MAE HRV	267 ms
	MAE peak difference	1.74 peaks
3	MAE heart rate	7.99 bpm
	MAE HRV	294 ms
	MAE peak difference	2.06 peaks
4	MAE heart rate	4.29 bpm
	MAE HRV	264 ms
	MAE peak difference	1.26 peaks
5	MAE heart rate	5.24 bpm
	MAE HRV	88 ms
	MAE peak difference	1.53 peaks
6	MAE heart rate	0.60 bpm
	MAE HRV	39 ms
	MAE peak difference	0.21 peaks
7	MAE heart rate	8.50 bpm
	MAE HRV	278 ms
	MAE peak difference	2.05 peaks

Table 8: Validation metrics across Personalized Baseline models. First row contains averages across all Personalized Baselines; all other rows show results for Personalized Baseline corresponding to each subject. These are the most accurate but least generalizable baselines.

Subject	Error metric	Result
Average	MAE heart rate	26.31 bpm
	MAE HRV	280 ms
	MAE peak difference	6.59 peaks
1	MAE heart rate	26.08 bpm
	MAE HRV	25 ms
	MAE peak difference	6.32 peaks
2	MAE heart rate	16.75 bpm
	MAE HRV	349 ms
	MAE peak difference	4.53 peaks
3	MAE heart rate	17.45 bpm
	MAE HRV	393 ms
	MAE peak difference	4.35 peaks
4	MAE heart rate	20.12 bpm
	MAE HRV	397 ms
	MAE peak difference	5.21 peaks
5	MAE heart rate	62.32 bpm
	MAE HRV	312 ms
	MAE peak difference	15.63 peaks
6	MAE heart rate	33.94 bpm
	MAE HRV	319 ms
	MAE peak difference	8.53 peaks
7	MAE heart rate	7.51 bpm
	MAE HRV	165 ms
	MAE peak difference	1.58 peaks

Table 9: Validation metrics across Generalized Baseline models. First row represents average metric results; all other rows show results for each subject. Individual subject results were obtained from subject-wise cross-validation.

Section 3.3

Model complexity

The gradient boosting regression model performed well in maintaining limited demand for computational resources. Specifically, the deployed model runs in less than 0.5 seconds and uses less than 1 MB RAM, meaning that this work succeeded in developing an architecture for heart rate estimation entirely within the confines of a smartphone while limiting its resource demand. Though the performance of the model leaves room for improvement, this proof-of-concept demonstrates that it is possible to develop such an architecture.

As a trial run for HRMobile deployed in the Android test app, I recorded video of my own face within the application before and after exercise. Though it is not possible to obtain clinical ground truth data for these tests, the application did report biologically plausible values for my heart rate and reported a realistic increase in heart rate following exercise.

Chapter 4

Discussion

Section 4.1

Comparison with seminal works in the field

The performance of the architecture presented in this paper is not consistent with results reported elsewhere. Given that the gradient boosting regressor was optimized for performance in computing heart rate rather than HRV, the model's heart rate performance will be the focus of this discussion. Two papers by Poh et al and De Haan and Jeanne respectively were selected to compare HRMobile's performance to that of seminal works in the field (28) (29). Poh et al (28) report their method's heart rate performance as MAE and achieve a result of 0.95. De Haan and Jeanne (29) report their method's heart rate performance in root mean squared error (RMSE) and achieve a value of 0.4. These performance metrics correspond to heart rate estimation on subjects in a laboratory setting, similar to that of the subjects in the MPSC-rPPG dataset used in this study (1). Thus, the cross-validation tests run on the original subject data, as opposed to those run on the augmented data, are the ones which should be compared to the results from the reference papers. The heart rate MAE achieved by HRMobile was 16.64, which is significantly higher than the error reported

by the seminal papers mentioned.

The most significant contributors to this model underperformance are most likely the limitations introduced by the lack of size and diversity of the dataset and the presence of noise artifacts in model-produced rPPG signal which were confused for heart rate peaks in the heart rate estimation algorithm. These false peaks often led to an overestimation of heart rate. Further, the size and shape of these false peaks was not consistent enough to remove them through postprocessing the model's output. Lastly, the gradient boosting regression architecture selected for this work may simply not have been powerful enough to capture the necessary dynamics in the data to produce accurate rPPG signal.

Though a requirement for this project was that it limit its demand of computational resources, methods which rely only on signal processing and do not employ machine learning would likely also satisfy this demand. Many existing works in this field, however, leave room for improvement with regards to reproducibility. Though this work does not achieve state-of-the-art performance, we hope that the materials and methods enumerated throughout this project are properly explained so that future work may build upon them.

Section 4.2

Conclusions from data augmentation experiments

The data augmentation experiments involved the extension of the 7-subject MPSC-rPPG dataset by adding Gaussian noise, dimming brightness, or increasing brightness at the frame level for all subjects (1). These 3 data augmentation techniques added 3 "sets" to our dataset. This allowed us to test two hypotheses: **(1)** that expanding the training dataset using these data augmentation techniques would improve the model's

performance, and **(2)** that model performance would degrade when tested on subject video with decreased video quality introduced from Gaussian noise, increased lighting, or decreased lighting. Tables 2, 3, and 4 display the results of these tests. In Table 2, the model was trained on only the original MPSC-rPPG data, but tested on the original data and all sets of augmented data (1). Table 3 shows the results when the model was trained on both the original dataset and the Gaussian noise augmented data, but tested on the original data and all sets of augmented data. Finally, Table 4 demonstrates the results when the model is trained on the original data and all data augmentations, and then tested on all sets of the data.

The results from tables 2, 3, and 4 indicate that hypothesis **(1)**, that augmenting the training dataset would improve model performance, largely did not hold. Across the cross-validation runs, the model trained only on the original dataset achieved the best performance in computing heart rate on the original data, scoring heart rate MAE of 16.64. For cross-validation runs in which augmented data were added to the training set, the model's heart rate performance on non-augmented data was worse, earning heart rate MAE of 19.26 when trained with just the Gaussian noise added to the training set, and earning heart rate MAE of 17.54 when all all data augmentations were added to the training set.

One notable result is that adding Gaussian noise to the training set actually worsened performance when the model was tested on the videos with Gaussian noise added. When the model was trained on only the original data, it achieved a heart rate MAE error of 19.82 on the Gaussian noise augmented data. When the Gaussian noise augmented data were added to the training data, the same metric jumped to 23.50. However, it should be noted that adding the dimmed and brightened videos to the training data slightly improved the model's performance in those adverse lighting conditions. Specifically, the model, when trained only on the original data, scored

heart rate MAEs of 23.96 and 27.51 bpm on the dimmed and brightened videos respectively. When the dimmed and brightened videos were added to the training data, those errors dropped slightly to 23.74 and 27.33 bpm. The reason for the lack of a performance boost from the data augmentations could be that they did not broaden the set of subjects seen by the model in training. Instead, they effectively added noisier versions of the existing videos to the training set. While this may have served to improve performance as it exposed the model to frames with less perfect video quality and lighting conditions, adding this noise may also have simply served to confuse the original signal detected by the model.

The second hypothesis tested by working with the augmented dataset was that HRMobile's performance would drop when presented with video characterized by less optimal quality and lighting than the laboratory conditions seen in the original data (1). This was expected, given that the model setup and data available were tailored for performance in controlled conditions, devoid of noise introduced by subject movement, lighting changes, or inferior video quality. This hypothesis was shown to be correct by the tests run on the model with augmented data. In all cross-validation runs across all combinations of original data and augmented data included for training, the model performed worse on the augmented data than on original MPSC-rPPG data (1). For the reasons enumerated above, this was the expected outcome. Fine-tuning this architecture for improved performance in adverse and noisy conditions, such as those simulated by the augmented data, could be an avenue for future improvements to the HRMobile framework.

The best validation error achieved by HRMobile on the original dataset of 16.64 implies that there is still much room for improvement in developing a framework for heart rate measurement which can be deployed in a smartphone app. However, we hope that this work unwraps a blueprint for the development of such a model.

Signal processing methods are lightweight and could be good candidates for such a framework. Though they are more complex, the baseline method implemented in this work indicates that deep learning approaches show promise for producing accurate rPPG signal, particularly if they are personalized to a subject. With this in mind, a future model could replace either the feature engineering step or gradient boosting regression step laid out as part of this first iteration of HRMobile.

Section 4.3

Comparison to baseline models

The implementation of Hasan et al's work in the form of the Personalized and Generalized Baseline models demonstrates both the promise of deep learning approaches for heart rate measurement and the challenges associated with the task (1). The Personalized Baseline models demonstrated relatively high degrees of accuracy, averaging a heart rate error of just 5.21 bpm. Figures 6 and 7 show the visual similarity of Personalized Baseline rPPG output to ground truth BVP signal. However, the cross-validation results of the Generalized Baselines show that it is difficult to achieve consistent and accurate rPPG generation across a population of subjects. For some subjects, the Generalized Baseline performed quite well, achieving a minimum heart rate error of 7.51 bpm. However, it was extremeley inaccurate for some subjects, earning a maximum heart rate error of 62.32 bpm. This wide range of results for the Generalized Baseline meant that its MAE cross-validation heart rate error was 26.31 bpm, worse than the cross-validation error achieved by HRMobile. I believe that the strong performance of the Generalized Baseline on some subjects indicates that deep learning approaches for heart rate measurement are an interesting avenue for further research and development. It is likely that the wild swings in model performance amongst subjects is due to the composition of the training set when a subject is tested

on during cross-validation. More specifically, if a subject's ground truth is unrepresentative of the subjects contained in the training set for that cross-validation round, the Generalized Model's performance could suffer. These results also indicate that HRMobile's performance on the MPSC-rPPG dataset is actually an improvement on the average Generalized Baseline performance. Finally, the significantly lower RAM consumed by HRMobile than the Personalized and Generalized Baselines represents a further success of this work.

Section 4.4

Limitations

The most significant limitation present in this work is the size and demographic makeup of the dataset. With only 7 subjects, it would be impossible for the dataset to cover a sufficiently extensive range of ages, genders, and skin tones. The subject population for our dataset skewed towards younger males with darker skin tones and was ∼86% male. These demographic challenges, when coupled with the range of possible heart rates, make for an insufficient dataset for a truly generalizable model. Further, this data limitation also informed the model selection for this project, as there was concern regarding overfitting to the particular BVP patterns of the subjects in the dataset. It is possible that machine learning architectures specifically designed for spatiotemporal objectives such as heart rate estimation may be superior to the model employed for this project if given sufficiently large and diverse datasets. Another limitation of this work is the fact that the gradient boosting model is merely indirectly tasked with optimizing for heart rate estimation performance due to the non-differentiable nature of the heart rate estimation function. If a differentiable objective, which could directly or more accurately estimate heart rate, were discovered, this should yield significant improvements in model performance. A further limita-

tion of the project is its potentially disproportionate focus on the GBDT regressor's performance in calculating heart rate, while focusing less on HRV until the final reporting of results. It is possible that a custom objective function tailored towards HRV or early stopping criteria taking HRV into account could enhance the model's performance. Regardless, it is likely true that this focus on heart rate at the expense of HRV led to lower overall performance on the HRV prediction task. It must also be noted that the model introduced in this work is trained exclusively on motionless subjects at rest with sufficient lighting. It does not control for factors such as motion, improper lighting, or factors that change the appearance of a subject's skin, such as makeup, perspiration, or discoloration due to physical activity or medical factors.

Section 4.5

Future directions

There are a number of options for future research which may yield improvements on the baseline architecture presented in this work. The most obvious of which would be the discovery or creation of a larger, more diverse dataset. Another possibility is relaxing the requirement for all computation to be on the smartphone while simultaneously retaining user privacy. This could be achieved by producing the raw RGB channels from the video of the user's face locally on-device and deleting the video after the raw channels are generated. This could allow for transmission of the raw channels to a server for ingestion into a more complex model while preserving the anonymity and privacy of the user. However, such an approach would be accompanied by drawbacks including greater costs and lower reliability.

An interesting avenue for future research could be personalization of the HRMobile architecture. Hasan et al report promising results from their multi-task learning architecture when the model is fine-tuned on data from a given subject (1). This

fine-tuned approach outperformed the same model when applied to a subject that was not seen in training (1). This suggests that fine-tuning a model on data from a given subject could yield greater accuracy in generating rPPG signal. However, such a framework would need a method for acquiring subject ground truth to inform this fine-tuning step. This would likely require reliance on incumbent methods for measuring heart rate such as clinical settings or wearable devices. In either case, a calibration step in which smartphone video of a subject's face were synced with a clinical or wearable heart rate sensor would be possible. For added robustness, the calibration step could also allow the subject to introduce factors such as dimmed lighting, motion, facial hair, or glasses with corresponding labels.

Further model personalization could be achieved through adding demographic information such as race, ethnicity, gender, age, or skin tone information to the model. This was not possible in this study as this information was not included in the dataset (1). Therefore, this approach was not built into Hasan et al's model personalization step (1). However, it is possible that demographic data could influence the model's performance. As such, future research into the incorporation of demographic information in a model for rPPG signal generation could have the potential to yield performance improvements.

The promising performance of the Personalized and Generalized Baseline models adapted from (1) also indicate potential for the personalization and fine-tuning concepts. If the multi-task learning head networks were re-introduced to the Generalized Baseline model implemented for comparison to HRMobile, it may not require a significant amount of additional ground truth from a subject for fine-tuning the model on that subject. If this were possible, accurate and generalized rPPG signal could be produced. The only problem with this is obtaining ground truth for a subject not in the original training dataset is nontrivial. However, it may be possible using

techniques such as that suggested in Jacobson and Bhattacharya (55). Jacobson and Bhattacharya discuss a remote heart rate measurement system which calculates heart rate and HRV by having the user press a finger over the rear-view camera of their phone for 30 seconds (55). If sufficient rPPG signal ground truth could be collected using Jacobson and Bhattacharya's method, Hasan et al's work could potentially be deployed and fine-tuned remotely, thereby achieveing the generation of generalized and accurate rPPG signal.

The results from the data augmentation experiments highlight additional opportunity for model improvement in the context of poor video quality or sub-optimal lighting. By demonstrating performance degradation of HRMobile when Gaussian noise, dimmed lighting, or brightened lighting are added to the data, we have shown that there is room for improving the model when it is presented with video of sub-optimal quality. Improvements to the model's performance in the face of sub-optimal video quality would add substantial robustness to the HRMobile framework if deployed in a mobile app for public use.

There may be further improvement to be discovered in HRV estimation. The model presented in this work is specifically optimized for performance on producing rPPG signal for estimating heart rate. This rPPG signal is also used to estimate HRV, but HRV is not the primary learning objective of the model. Prioritizing HRV in the model development process could yield better performance with respect to HRV. Likewise, the custom loss function implemented for the gradient boosting model is merely an approximation of our non-differential heart rate estimation function. A custom loss function which better approximates HRV could ultimately improve performance in estimating HRV. It is possible that either a differentiable function for calculating heart rate or HRV could be discovered, or that there exist better, differentiable proxies of the non-differentiable heart rate and HRV functions employed

in this work. Such discoveries could yield improvements in model performance as they would allow for direct optimization of the model for estimating heart rate and HRV.

Finally, greater adoption of existing methods which report superior performance to the architecture presented in this work could yield performance improvements. Though replication of many state-of-the-art signal processing methods for heart rate estimation is difficult, it is possible that better implementations of these proposals exist and could improve the performance of the machine learning based approach presented here.

Chapter 5

Conclusion

The development of a lightweight architecture for heart rate estimation from video taken on smartphone cameras was the central objective of this thesis. As an added contribution, I sought to maintain the greatest user privacy controls possible through requiring our pipeline to consume limited computational resources. This project was completed in collaboration with the Mood Triggers team within Professor Nicholas Jacobson's AIM HIGH Lab to enhance their data collection and UX with physiological parameters such as heart rate and HRV. An Android app was developed using the Flutter framework for testing this pipeline. Within this Flutter framework, a Python plugin was implemented to enable the incorporation of Python code within the app. This permitted the execution of facial recognition, signal processing, and machine learning code within the test app. Ultimately, the system presented here consisted of the YoloV5 facial recognition model for discovering facial ROIs for RGB channel extraction, feature engineering, and the use of a gradient boosting regression model implemented with XGBoost. Through testing, I discovered that our architecture is not as accurate as other state-of-the-art methods. However, I hope that readers find the architecture presented here more transparent and easier to replicate than peer works in the field. There are a number of interesting avenues to explore for

improving on the baseline performance presented here, and I hope that some of them may enhance the ability of the Mood Triggers team to incorporate heart rate and HRV into their psychiatric research going forward.

Bibliography

[1] Z. Hasan, S. R. Ramamurthy, and N. Roy, "MPSC-rPPG Dataset." IEEE Dataport, 2021.

[2] R. Agarwal, D. Z. Sands, and J. D. Schneider, "Quantifying the economic impact of communication inefficiencies in us hospitals," *Journal of Healthcare Management*, vol. 55, no. 4, pp. 265–282, 2010.

[3] B. Meskó, Z. Drobni, É. Bényei, B. Gergely, and Z. Győrffy, "Digital health is a cultural transformation of traditional healthcare," *Mhealth*, vol. 3, 2017.

[4] H. W. Loh, C. P. Ooi, S. Seoni, P. D. Barua, F. Molinari, and U. R. Acharya, "Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022)," *Computer Methods and Programs in Biomedicine*, p. 107161, 2022.

[5] M. Attaran, "Blockchain technology in healthcare: Challenges and opportunities," *International Journal of Healthcare Management*, vol. 15, no. 1, pp. 70–83, 2022.

[6] O. B. Osoro and E. J. Oughton, "A techno-economic framework for satellite networks applied to low earth orbit constellations: Assessing starlink, oneweb and kuiper," *IEEE Access*, vol. 9, pp. 141611–141625, 2021.

[7] J. S. Ashwood, A. Mehrotra, D. Cowling, and L. Uscher-Pines, "Direct-to-consumer telehealth may increase access to care but does not decrease spending," *Health Affairs*, vol. 36, no. 3, pp. 485–491, 2017.

[8] J. G. Perle and B. Nierenberg, "How psychological telehealth can alleviate society's mental health burden: A literature review," *Journal of Technology in Human Services*, vol. 31, no. 1, pp. 22–41, 2013.

[9] S. Lawes-Wickwar, H. McBain, and K. Mulligan, "Application and effectiveness of telehealth to support severe mental illness management: systematic review," *JMIR mental health*, vol. 5, no. 4, p. e8816, 2018.

[10] C. S. Kruse, S. Bouffard, M. Dougherty, and J. S. Parro, "Telemedicine use in rural native american communities in the era of the aca: a systematic literature review," *Journal of medical systems*, vol. 40, pp. 1–9, 2016.

[11] S. Chakrabarti, "Usefulness of telepsychiatry: A critical evaluation of videoconferencing-based approaches," *World journal of psychiatry*, vol. 5, no. 3, p. 286, 2015.

[12] T. M. Hale and J. C. Kvedar, "Privacy and security concerns in telehealth," *AMA Journal of Ethics*, vol. 16, no. 12, pp. 981–985, 2014.

[13] F. Di Carlo, A. Sociali, E. Picutti, M. Pettorruso, F. Vellante, V. Verrastro, ..., and M. di Giannantonio, "Telepsychiatry and other cutting-edge technologies in covid-19 pandemic: Bridging the distance in mental health assistance," *International Journal of Clinical Practice*, vol. 75, no. 1, p. e13567, 2021.

[14] H. Li, A. Glecia, A. Kent-Wilkinson, D. Leidl, M. Kleib, and T. Risling, "Transition of mental health service delivery to telepsychiatry in response to covid-19: A literature review," *Psychiatric Quarterly*, vol. 93, no. 1, pp. 181–197, 2022.

[15] M. Sheikh, M. Qassem, and P. A. Kyriacou, "Wearable, environmental, and smartphone-based passive sensing for mental health monitoring," *Frontiers in Digital Health*, vol. 3, p. 662811, 2021.

[16] P. Rathnayaka, N. Mills, D. Burnett, D. De Silva, D. Alahakoon, and R. Gray, "A mental health chatbot with cognitive skills for personalised behavioural activation and remote health monitoring," *Sensors*, vol. 22, no. 10, p. 3653, 2022.

[17] N. C. Jacobson, M. Heinz, and J.-W. Kim, "Designing a generative mental health chatbot using artificial intelligence," in *ANNALS OF BEHAVIORAL MEDICINE*, vol. 55, pp. S558–S558, OXFORD UNIV PRESS INC, 2021.

[18] J. F. Huckins, A. W. DaSilva, W. Wang, E. Hedlund, C. Rogers, S. K. Nepal, ..., and A. T. Campbell, "Mental health and behavior of college students during the early phases of the covid-19 pandemic: Longitudinal smartphone and ecological momentary assessment study," *Journal of medical Internet research*, vol. 22, no. 6, p. e20185, 2020.

[19] S. Shiffman, A. A. Stone, and M. R. Hufford, "Ecological momentary assessment," *Annu. Rev. Clin. Psychol.*, vol. 4, pp. 1–32, 2008.

[20] A. S. Juarascio, R. J. Crochiere, T. M. Tapera, M. Palermo, and F. Zhang, "Momentary changes in heart rate variability can detect risk for emotional eating episodes," *Appetite*, vol. 152, p. 104698, 2020.

[21] J. M. Gorman and R. P. Sloan, "Heart rate variability in depressive and anxiety disorders," *American heart journal*, vol. 140, no. 4, pp. S77–S83, 2000.

[22] J. Yang and K. N. Kershaw, "Feasibility of using ecological momentary assessment and continuous heart rate monitoring to measure stress reactivity in natural settings," *Plos one*, vol. 17, no. 3, p. e0264200, 2022.

[23] A. Aryal, A. Ghahramani, and B. Becerik-Gerber, "Monitoring fatigue in construction workers using physiological measurements," *Automation in Construction*, vol. 82, pp. 154–165, 2017.

[24] J. Kranjec, S. Beguš, G. Geršak, and J. Drnovšek, "Non-contact heart rate and heart rate variability measurements: A review," *Biomedical signal processing and control*, vol. 13, pp. 102–112, 2014.

[25] G. Shin, M. H. Jarrahi, Y. Fei, A. Karami, N. Gafinowitz, A. Byun, and X. Lu, "Wearable activity trackers, accuracy, adoption, acceptance and health impact: A systematic literature review," *Journal of Biomedical Informatics*, vol. 93, p. 103153, 2019.

[26] S. Kwon, H. Kim, and K. R. S. Park, "Validation of heart rate extraction using video imaging on a built-in camera system of a smartphone," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2174–2177, IEEE, 2012.

[27] X. Li, J. Chen, G. Zhao, and M. Pietikainen, "Remote heart rate measurement from face videos under realistic situations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4264–4271, 2014.

[28] M. E. Poh, D. J. McDuff, and R. W. Picard, "Advancements in noncontact, multiparameter physiological measurements using a webcam," *IEEE transactions on biomedical engineering*, vol. 58, no. 1, pp. 7–11, 2010.

[29] G. De Haan and V. Jeanne, "Robust pulse rate from chrominance-based rppg," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 10, pp. 2878–2886, 2013.

[30] W. Wang, A. C. Den Brinker, S. Stuijk, and G. De Haan, "Algorithmic principles of remote ppg," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, pp. 1479–1491, 2016.

[31] M.-Z. Poh, D. J. McDuff, and R. W. Picard, "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation," *Optics express*, vol. 18, no. 10, pp. 10762–10774, 2010.

[32] J. Vila, F. Palacios, J. Presedo, M. Fernandez-Delgado, P. Felix, and S. Barro, "Time-frequency analysis of heart-rate variability," *IEEE Engineering in Medicine and Biology Magazine*, vol. 16, no. 5, pp. 119–126, 1997.

[33] D. Qiao, F. Zulkernine, R. Masroor, R. Rasool, and N. Jaffar, "Measuring heart rate and heart rate variability with smartphone camera," in *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*, pp. 248–249, IEEE, 2021.

[34] Z. Yu, W. Peng, X. Li, X. Hong, and G. Zhao, "Remote heart rate measurement from highly compressed facial videos: an end-to-end deep learning solution with video enhancement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 151–160, 2019.

[35] Z. Yu, X. Li, and G. Zhao, "Remote photoplethysmograph signal measurement from facial videos using spatio-temporal networks," *arXiv preprint arXiv:1905.02419*, 2019.

[36] Y. Qiu, Y. Liu, J. Arteaga-Falconi, H. Dong, and A. El Saddik, "Evm-cnn: Real-time contactless heart rate estimation from facial video," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1778–1787, 2018.

[37] A. Lam and Y. Kuno, "Robust heart rate measurement from video using select random patches," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3640–3648, 2015.

[38] S. Tulyakov, X. Alameda-Pineda, E. Ricci, L. Yin, J. F. Cohn, and N. Sebe, "Self-adaptive matrix completion for heart rate estimation from face videos under realistic conditions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2396–2404, 2016.

[39] J. R. Maestre-Rendon, T. A. Rivera-Roman, A. A. Fernandez-Jaramillo, N. E. Guerron Paredes, and J. J. Serrano Olmedo, "A non-contact photoplethysmography technique for the estimation of heart rate via smartphone," *Applied Sciences*, vol. 10, no. 1, p. 154, 2019.

[40] Y. Maki, Y. Monno, K. Yoshizaki, M. Tanaka, and M. Okutomi, "Inter-beat interval estimation from facial video based on reliability of bvp signals," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6525–6528, IEEE, 2019.

[41] H. Ernst, M. Scherpf, H. Malberg, and M. Schmidt, "Optimal color channel combination across skin tones for remote heart rate measurement in camera-based photoplethysmography," *Biomedical Signal Processing and Control*, vol. 68, p. 102644, 2021.

[42] E. Lee, E. Chen, and C.-Y. Lee, "Meta-rppg: Remote heart rate estimation using a transductive meta-learner," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII*, vol. 16, pp. 392–409, Springer International Publishing, 2020.

[43] F. Shirbani, N. Hui, I. Tan, M. Butlin, and A. P. Avolio, "Effect of ambient

lighting and skin tone on estimation of heart rate and pulse transit time from video plethysmography," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 2642–2645, IEEE, 2020.

[44] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, A. V, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, J. Hajek, L. Diaconu, Marc, Y. Kwon, oleg, wanghaoyang0106, Y. Defretin, A. Lohia, ml5ah, B. Milanko, B. Fineran, D. Khromov, D. Yiwei, Doug, Durgesh, and F. Ingham, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," Apr. 2021.

[45] D. E. King, "Dlib-ml: A machine learning toolkit," 2009.

[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, ..., and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[47] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference*, (Nottingham), BMVA Press, September 2014.

[48] G. Bradski, "The opencv library," 2000.

[49] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, ..., and P. Van Mulbregt, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[50] F. Bousefsaf, C. Maaoui, and A. Pruski, "Continuous wavelet filtering on webcam photoplethysmographic signals to remotely assess the instantaneous heart rate," *Biomedical Signal Processing and Control*, vol. 8, no. 6, pp. 568–574, 2013.

[51] G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O'Leary, "Pywavelets: A python package for wavelet analysis," *Journal of Open Source Software*, vol. 4, no. 36, p. 1237, 2019.

[52] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, and T.-Y. ..., Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, vol. 30, Curran Associates, Inc., 2017.

[53] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA), pp. 785–794, ACM, 2016.

[54] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *International conference on machine learning*, pp. 894–903, PMLR, 2017.

[55] N. C. Jacobson and S. Bhattacharya, "Digital biomarkers of anxiety disorder symptom changes: Personalized deep learning models using smartphone sensors accurately predict anxiety symptoms from ecological momentary assessments," *Behaviour Research and Therapy*, vol. 149, p. 104013, 2022.