UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Tuukka Bogdanoff

COMPARATIVE EVALUATION OF THE APPLICABILITY OF SELF-ORGANIZED OPERATIONAL NEURAL NETWORKS TO REMOTE PHOTOPLETHYSMOGRAPHY

Master's Thesis
Degree Programme in Computer Science and Engineering
November 2023

Bogdanoff T. (2023) Comparative Evaluation of the Applicability of Self-Organized Operational Neural Networks to Remote Photoplethysmography. University of Oulu, Degree Programme in Computer Science and Engineering, 88 p.

ABSTRACT

Photoplethysmography (PPG) is a widely applied means of obtaining blood volume pulse (BVP) information from subjects which can be used for monitoring numerous physiological signs such as heart rate and respiration. Following observations that blood volume information can also be retrieved from videos recorded of the human face, several approaches for the remote extraction of PPG signals have been proposed in literature. These methods are collectively referred to as remote photoplethysmography (rPPG). The current state of the art of rPPG approaches is represented by deep convolutional neural network (CNN) models, which have been successfully applied in a wide range of computer vision tasks.

A novel technology called operational neural networks (ONNs) has recently been proposed in literature as an extension of convolutional neural networks. ONNs attempt to overcome the limitations of conventional CNN models which are primarily caused by exclusively employing the linear neuron model. In addition, to address certain drawbacks of ONNs, a technology called self-organized operational neural networks (Self-ONNs) have recently been proposed as an extension of ONNs.

This thesis presents a novel method for rPPG extraction based on self-organized operational neural networks. To comprehensively evaluate the applicability of Self-ONNs as an approach for rPPG extraction, three Self-ONN models with varying number of layers are implemented and evaluated on test data from three data sets representing different distributions. The performance of the proposed models are compared against corresponding CNN architectures as well as a typical unsupervised rPPG pipeline. The performance of the methods is evaluated based on heart rate estimations calculated from the extracted rPPG signals.

In the presented experimental setup, Self-ONN models did not result in improved heart rate estimation performance over parameter-equivalent CNN alternatives. However, every Self-ONN model showed superior ability to fit the train target, which both shows promise for the applicability of Self-ONNs as well as suggests inherent problems in the training setup. Additionally, when taking into account the required computational resources in addition to raw HR estimation performance, certain Self-ONN models showcased improved efficiency over CNN alternatives. As such, the experiments nonetheless present a promising proof of concept which can serve as grounds for future research.

Keywords: artificial intelligence, atrial fibrillation, healthcare, machine learning

Bogdanoff T. (2023) Vertaileva arviointi itseorganisoituvien operationaalisten neuroverkkojen soveltuvuudesta etäfotopletysmografiaan. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 88 s.

TIIVISTELMÄ

Fotopletysmografia on laajasti sovellettu menetelmä veritilavuuspulssi-informaation saamiseksi kohteista, jota voidaan käyttää useiden fysiologisten arvojen, kuten sydämensykkeen ja hengityksen, seurannassa. Seuraten havainnoista, että veritilavuusinformaatiota on mahdollista palauttaa myös ihmiskasvoista kuvatuista videoista, useita menetelmiä fotopletysmografiasignaalien erottamiseksi etänä on esitetty kirjallisuudessa. Yhteisnimitys näille menetelmille on etäfotopletysmografia (remote photoplethysmography, rPPG). Syvät konvolutionaaliset neuroverkkomallit (convolutional neural networks, CNNs), joita on onnistuneesti sovellettu laajaan valikoimaan tietokonenäön tehtäviä, edustavat nykyistä rPPG-lähestymistapojen huippua.

Uusi teknologia nimeltään operationaaliset neuroverkot (operational neural networks, ONNs) on hiljattain esitetty kirjallisuudessa konvolutionaalisten neuroverkkojen laajennukseksi. ONN:t pyrkivät eroon tavanomaisten CNN-mallien rajoitteista, jotka johtuvat pääasiassa lineaarisen neuronimallin yksinomaisesta käytöstä. Lisäksi tietyistä ONN-mallien heikkouksista eroon pääsemiseksi, teknologia nimeltään itseorganisoituvat operationaaliset neuroverkot (self-organized operational neural networks, Self-ONNs) on hiljattain esitetty lajeennuksena ONN:ille.

Tämä tutkielma esittelee uudenlaisen menetelmän rPPG-erotukselle pohjautuen itseorganisoituviin operationaalisiin neuroverkkoihin. Self-ONN:ien soveltuvuuden rPPG-erotukseen perusteelliseksi arvioimiseksi kolme Self-ONN -mallia vaihtelevalla määrällä kerroksia toteutetaan ja arvioidaan testidatalla kolmesta eri datajoukosta, jotka edustavat eri jakaumia. Esitettyjen mallien suorituskykyä verrataan vastaaviin CNN-arkkitehtuureihin sekä tyypilliseen ohjaamattomaan rPPG-liukuhihnaan. Menetelmien suorituskykyä arvioidaan perustuen rPPG-signaaleista laskettuihin sydämensykearvioihin.

Esitellyssä kokeellisessa asetelmassa Self-ONN:t eivät johtaneet parempiin sykearvioihin verrattuna parametrivastaaviin CNN-vaihtoehtoihin. Self-ONN:t kuitenkin osoittivat ylivertaista kykyä sovittaa opetuskohteen, mikä sekä on lupaavaa Self-ONN:ien soveltuvuuden kannalta että viittaa luontaisiin ongelmiin opetusasetelmassa. Lisäksi, kun huomioon otetaan vaaditut laskentaresurssit raa'an sykkeen arvioinnin suorituskyvyn lisäksi, tietyt Self-ONN -mallit osoittivat parempaa tehokkuutta CNN-vaihtoehtoihin verrattuna. Näin ollen kokeet joka tapauksessa tarjoavat lupaavan konseptitodistuksen, joka voi toimia perustana tulevalle tutkimukselle.

Avainsanat: tekoäly, eteisvärinä, terveydenhuolto, koneoppiminen

TABLE OF CONTENTS

FOREWORD

This Master's thesis was composed in the center for machine vision and signal analysis (CMVS) in the University of Oulu for the purpose of evaluating the applicability of self-organized operational neural networks to remote photoplethysmography. I would like to thank the supervisors of my thesis, Dr. Miguel Bordallo López and Dr. Xiaobai Li, for suggesting the topic of the thesis and their guidance during the writing process. I would also like to thank Constantino Álvarez Casado for his help in implementing the presented experimental setup and providing the code for the Face2PPG pipeline as well as Zhaodong Sun for his help in implementing the supervised models by providing the code for the used negative maximum cross-correlation loss function in addition to example code for training the models.

Oulu, November 7th, 2023

Tuukka Bogdanoff

LIST OF ABBREVIATIONS AND SYMBOLS

AF	atrial fibrillation
AFL	atrial flutter
AUC	area under the receiver operating characteristic curve
BP	backpropagation
BPM	beats per minute
BVP	blood volume pulse
CNN	convolutional neural network
DRMF	Discriminative Response Map Fitting
ECG	electrocardiography
ELU	exponential linear unit
FIR	finite impulse response
FP	forward propagation
FPS	frames per second
GIS	greedy iterative search
GOP	generalized operational perceptron
GPU	graphics processing unit
GWN	Gaussian white noise
HCI	human-computer interaction
HR	heart rate
HRV	heart rate variability
IBI	inter-beat interval
IR	infrared
KLT	Kanade-Lucas-Tomasi
LED	light-emitting diode
LSTM	long short term memory
MAC	multiply-accumulate operation
MAE	mean absolute error
MLP	multi-layer perceptron
MSE	mean square error
NaN	not a number
NIR	near infrared
NLMS	Normalized Least Mean Square
NMCC	negative maximum cross-correlation
NN	neural network
NPC	negative Pearson's correlation
OBF	Oulu Bio-Face
OMIT	Orthogonal Matrix Image Transformation
ONN	operational neural network
PCC	Pearson's correlation coefficient
PPG	photoplethysmography
PSD	power spectral density
ReLU	rectified linear unit
RGB	red, green, blue
RMSE	root mean square error

RNN	recurrent neural network
ROI	region of interest
rPPG	remote photoplethysmography
SD	standard deviation
Self-ONN	self-organized operational neural network
SGD	stochastic gradient descent
SNR	signal to noise ratio
SR	sinus rhythm
SVM	support vector machine
VRAM	video random access memory
*CP**	target classification performance
F_s	sampling frequency
m_y	mean of the vector of ground truth HR values
$m_{\hat{y}}$	mean of the vector vector of predicted HR values
N	number of vector elements
N_{BP}	number of backpropagation runs
P	projection matrix
P_k^l	pool operator of neuron k in layer l
Q	orthonormal matrix
q_1	first column of the matrix Q
T	temporal dimension
w_{ki}^l	kernel i of neuron k in layer l
x_k^l	input map of neuron k in layer l
y	vector of ground truth HR values
$\hat{y}$	vector of predicted HR values
y_i^l	output map of neuron i in layer l
y_i	ground truth heart rate value
$\hat{y}_i$	predicted heart rate value
Δ_k^l	delta error at neuron k in layer l
ϵ	learning factor
Ψ_{ki}^l	nodal operator associated with kernel i of neuron k in layer l
$\mathbf{\Psi}$	composite nodal operator
θ	operator set
$\{\theta_N^*\}$	operator set library

1. INTRODUCTION

Measuring the heart activity of a patient is important in a variety of medical contexts. Typically, a patient's heart activity is measured by using either electrocardiography (ECG) or photoplethysmography (PPG). ECG and PPG signals can be used to calculate the average heart rate (HR) of the patient, as well as more detailed information about heart activity such as the inter-beat interval (IBI) and heart rate variability (HRV) measures which can help physicians in the diagnosis of a multitude of conditions. However, utilizing these methods requires the use of dedicated equipment and continuous physical contact, which makes them inapplicable or inconvenient in many situations and in prolonged use.

In recent years, many methods have been suggested for measuring PPG signals remotely from facial videos to overcome the burden of using contact sensors in measuring heart activity. These remote methods for retrieving PPG signals are collectively referred to as remote photoplethysmography (rPPG). Since most rPPG methods only require the use of a standard RGB (red, green, blue) camera, they can overcome the need for physical contact and specialized equipment to measure PPG signals, making the information from PPG signals more readily available in a variety of contexts, including telemedicine with the use of a webcam.

Methods utilizing neural networks, specifically convolutional neural networks (CNNs), represent the current state of the art in rPPG analysis. A neural network is a computational system composed of connected artificial neurons that loosely models the functions in biological nervous systems. Due to their ability to automatically infer discriminative features for a task based on the data they are given, neural networks have been successfully applied to an extensive range of use cases.

However, the calculations performed in a typical neural network are mostly based on simple linear functions, and the desired nonlinearity is achieved only by applying a nonlinear activation function to the result of a linear combination in each neuron. Due to relying extensively on simple linear calculations, neural networks must often consist of a large number of layers of neurons in order to achieve the desired performance on problems with complex nonlinear solution spaces. This means that a significant amount of computational power may be necessary to utilize the developed neural network model, which may limit the usability of the achieved model on consumer hardware.

Operational neural networks (ONNs) are a novel type of neural network model proposed in 2020 by Kiranyaz et al. [1]. ONNs extend upon the principle of conventional neural network models such as CNNs by allowing each neuron in the network to utilize any set of linear or nonlinear operations instead of relying entirely on a linear neuron model followed by a nonlinear activation. As such, the so-called "operational" neuron model utilized by ONNs forms a superset of the linear neurons utilized by most conventional neural network models, including CNNs. The authors of [1] demonstrate that due to utilizing the more general operational neuron model, ONNs can achieve performance superior to traditional CNNs with a smaller number of layers in challenging problems such as image denoising or image synthesis.

However, the ONNs as formulated in [1] also exhibit certain shortcomings which can limit their performance and scalability to large-scale problems. To address these issues, an extension of ONNs called self-organized operational neural networks (Self-

ONNs) were proposed in [2]. The formulation of the Self-ONN model follows the authors' observations of the particular importance of "nodal" operators found in ONNs, which in their original formulation exhibit limitations that can potentially hinder performance. The Self-ONN models utilize so-called generative neurons, which are neurons utilizing a composite nodal operator as opposed to the conventional nodal operators of ONNs. The defining characteristic of composite nodal operators is that they can be optimized during training without limitations to maximize performance for the given learning task. The authors of [2] demonstrate that Self-ONNs can exhibit superior learning performance compared to both ONN and CNN alternatives in the same experimental setups as proposed in [1].

As ONNs and Self-ONNs aim at overcoming the shortcomings of more traditional neural network models, such as the CNNs which represent the current state of the art in rPPG analysis, there is a clear incentive to explore the potential performance of these novel neural network models to the problem of estimating PPG signals remotely. However, as ONNs and Self-ONNs are rather novel proposals and yet to see widespread use, there is currently no publicly documented research evaluating the performance of these types of neural network approaches in rPPG analysis.

The main contribution of this thesis is a novel framework for remote PPG signal extraction utilizing self-organized operational neural network models. The networks are trained end-to-end for the extraction of PPG signals from facial videos and their performance is evaluated in comparison to current state of the art methods in a variety of experimental setups. The proposed framework is based on and compared against a supervised rPPG model presented in [3] and an unsupervised rPPG pipeline presented in [4]. The viability of an approach based on Self-ONNs for the task of rPPG extraction is evaluated extensively by constructing multiple Self-ONN models with varying complexity and comparing them against corresponding CNN alternatives. The approaches are evaluated based on the accuracy of heart rate estimates calculated from the extracted signals as well as the efficiency of the achieved solution.

The thesis is structured as follows: Chapter 2 presents an overview of the principles and applications of both conventional and remote photoplethysmography. The chapter describes suitable data sets for evaluating rPPG systems and presents existing approaches for rPPG extraction proposed in literature, which are divided into unsupervised and supervised methods. Chapter 3 describes operational neural networks as proposed in [1], as well as the Self-ONNs proposed in [2], presenting the motivation, theoretical foundation, implementation as well as current and potential applications of ONNs and Self-ONNs. Chapter 4 describes the implementation of the methods used in the experiments presented in this thesis, including the proposed novel rPPG extraction pipeline based on Self-ONNs. Chapter 5 describes the experimental setup and the metrics used in evaluating the methods. The obtained experimental results and a detailed analysis of and what they signify is presented in Chapter 6. Chapter 7 provides further discussion of the results and addresses certain problems and limitations in the setup in addition to providing directions for future research. Finally, the thesis ends with closing statements and conclusions in Chapter 8.

2. REMOTE PHOTOPLETHYSMOGRAPHY

Remote photoplethysmography (rPPG) refers to the process of remotely obtaining photoplethysmograph (PPG) signals. This chapter discusses the principles and applications of conventional PPG and rPPG as well as previous and current research on the topics.

2.1. Photoplethysmography

Photoplethysmography (PPG) is an optical technique for measuring blood volume changes in in the microvascular bed of tissue [5]. It can serve as a non-invasive means for measuring cardiac activity from the surface of the skin, also capturing information about other physiological phenomena such as respiration [5, 6, 7]. PPG systems are affordable and easy to set up, requiring only a light source for illuminating the tissue and a photodetector for measuring the variations in illumination related to changes in blood volume, which has led to their wide adoption in a variety of contexts [5]. The physiological origins of the components present in a PPG signal are not fully understood, but it is nonetheless generally accepted that they contain valuable information pertaining to cardiovascular function [5].

PPG signals are typically measured from the finger as this measurement site has been found to produce a signal with high amplitude compared to other locations, although numerous other measurement sites have been proposed for use in situations in which measuring from the finger is inconvenient [8]. The light source used in PPG usually consists of red and infrared (IR) light-emitting diodes (LEDs) [8].

The waveform of PPG signals are affected by numerous factors, not all of which are related to underlying physiological phenomena, such as the subject's posture and movement, the contact force of the sensor as well as ambient temperature [8]. As such, the choice of a suitable measurement site and signal processing steps is crucial in obtaining a high quality signal with relevant information. Numerous approaches for processing PPG signals have been proposed in attempts to reliably extract meaningful information from them [5, 6, 8], and the exact means of PPG measurement and processing ultimately depend on the specific application at hand [5, 7].

Photoplethysmography has found wide use in a variety of clinical settings in particular. Clinicians routinely apply PPG techniques for monitoring blood oxygen saturation, heart rate, blood pressure, cardiac output, heart rate variability and respiration, among others [5, 6]. The physiological signs that can be measured using PPG can provide clinicians with valuable information in diagnosing a variety of cardiovascular diseases and monitoring autonomic function [5]. PPG is also routinely utilized in the monitoring of patients during anaesthesia [5].

Nowadays, PPG sensors can also often be embedded in wearable devices [8, 7], and consumer interest in wearable health devices utilizing PPG, such as smart watches and fitness trackers, has increased significantly in recent years [9]. Such wearable devices often utilize green light as a light source [7] and employ several signal processing steps to mitigate the effect of motion artifacts introduced in their intended pervasive daily use [7, 9]. While devices worn on the wrist are the most popular [9], several other sites for wearable PPG devices, such as ears, have been proposed [8, 9]. Ring sensors worn on the finger can provide high quality signals which are comparable to benchmark PPG and ECG sensors in detecting beat-to-beat pulsations [8].

The remote measurement of PPG signals was first proposed in 2008 by Verkruysse et al. [10] based on the observation that blood volume pulse (BVP) information can also be observed on the face and extracted from simple RGB videos showing the face of the subject. The prospect of obtaining PPG signals remotely is significant as it could extend the applicability of PPG technology to a much wider context, including applications in which it would be inconvenient or infeasible for the subject to continuously wear a PPG sensor, such as telemedicine settings in which a patient communicates with a clinician in a video call. Remote PPG signals also encompass many of the features of conventional PPG signals and could be applied by individuals as an inexpensive means for the self-screening of various conditions, such as atrial fibrillation (AF) [11], which is often asymptomatic but potentially life-threatening. Practical applications of rPPG have become increasingly feasible due to recent advances in computer vision as well as the ubiquity of RGB cameras in modern everyday life. Interest in rPPG research has increased significantly since the inception of the concept, with the number of related publications showing a steep upward trend in the previous decade [12].

Remote PPG signals can also be applied in emotion analysis from facial videos since the heart rate and heart rate variability features extracted from PPG and rPPG signals can complement the information obtained by analysing facial expressions [12]. This is an example of a human-computer interaction (HCI) use case in which using contact sensors for obtaining the relevant physiological data would be inconvenient compared to a remote approach.

Since the inception of rPPG, numerous approaches for the remote extraction of PPG signals from facial videos have been proposed [13, 4]. A typical pipeline for remote PPG measurement includes the steps of detecting the face in the video, selecting a region of interest (ROI) suitable for rPPG measurement from the face, extracting the rPPG signal from the color variations in the selected region, and finally, post-processing the extracted signal to remove undesired artifacts and frequencies [12, 4]. Multiple alternatives for each step have been proposed in literature [4, 12]. After these steps, HR features can be calculated from the signal by means of frequency analysis or peak detection, for example, and in the case of peak detection, HRV features can also be computed. Figure 1 shows an example of a typical rPPG extraction pipeline as described above.

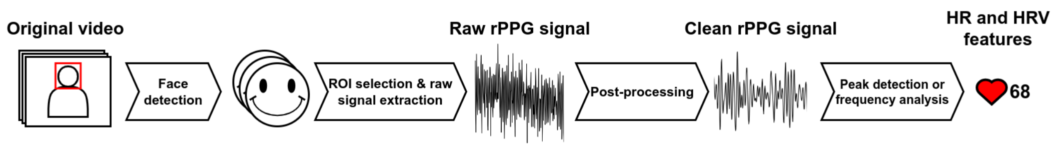

Figure 1. A general pipeline for remote PPG extraction from facial videos.

Approaches for rPPG extraction are often divided into unsupervised methods and supervised methods. A supervised method may combine one or more of the above steps in a single model, which is trained in a supervised manner using relevant physiological signals. During the training process, a supervised model can implicitly learn to perform several steps of the general pipeline described above. Unsupervised methods, on the other hand, do not rely on supervisory signals in developing the rPPG extraction process, and all of the above steps are usually defined explicitly. The principles and differences of unsupervised and supervised rPPG methods are discussed further later in this chapter and examples proposed in literature are given of both.

2.2. Unsupervised Methods

In 2008, Verkruysse et al. [10] were the first to demonstrate the feasibility of measuring PPG signals remotely from facial videos. The authors showed that PPG signals can be extracted from videos recorded of the human face from a distance of several meters under normal ambient light as the light source using a consumer level digital photo camera in movie mode. The authors demonstrate that at a distance of 1.5 meters, the signal to noise ratio (SNR) of the green channel signal measured of the face was sufficient for extracting up to four harmonics of the fundamental HR frequency, thus allowing the extraction of not only the HR but also the signal waveform [10].

In 2014, Li et al. [14] proposed a framework for remote heart rate measurement from facial videos utilizing face tracking and Normalized Least Mean Square (NLMS) adaptive filtering. While existing methods proposed for remote heart rate measurement from facial videos had been demonstrated to perform well on data recorded under strictly controlled conditions in which illumination variations or spontaneous movements by the subjects were not present, research demonstrating the applicability of remote HR estimation to more realistic scenarios had not yet been conducted. Based on existing research, the authors proposed a new framework that could account for the effects of illumination variations and subjects' motions, thus demonstrating the feasibility of remote HR extraction from facial videos under more realistic conditions.

The proposed framework for processing facial videos is divided into four distinct steps of region of interest (ROI) detection and tracking, illumination rectification, non-rigid motion elimination and temporal filtering. In the first step, a Viola-Jones face detector is used to detect the face rectangle in the first frame of the video and the Discriminative Response Map Fitting (DRMF) method [15] is used to detect 66 landmarks on the face. 9 of the landmarks are used to determine the ROI so that the eye region is excluded from the ROI and the ROI boundary is indented from the face boundary. Instead of detecting the ROI separately in each frame, the changes in the location of the ROI detected in the first frame are tracked through the the following frames of the video using the Kanade-Lucas-Tomasi (KLT) algorithm [16] to counter the variations caused by rigid head movement. The raw blood volume pulse signal is then defined as the mean of the green channel values inside the ROI of each frame. In the second step, the raw pulse signal is filtered using a NLMS filter using the mean green value extracted from a background region as a reference in order to account for changes in illumination. In the third step, to mitigate the effects of non-rigid movements such as changes in facial expression, the filtered signal is divided into segments of equal length and 5% of the segments with the largest standard deviation in all the testing samples are removed, after which the remaining segments are concatenated back together. In the final step, the resulting signal is filtered using a detrending filter proposed by Tarvainen et al. [17], a moving average filter, and a Hamming window based finite impulse response band pass filter with a cutoff frequency set to [0.7, 4] Hz. Finally, the power spectral density (PSD) of the filtered pulse signal is calculated using Welch's method [18] and the heart rate measured from the video is defined as $60f_{HR}$ where f_{HR} is defined as the frequency with the maximal power response as observed in the PSD.

The authors evaluate the performance of their network on two data sets: the VideoHR database, which was compiled by the authors themselves, and the multi-modal MAHNOB-HCI database [19]. The VideoHR database consists of videos recorded under controlled settings in which illumination variations or body movements do not occur. The MAHNOB-HCI database was in turn chosen because it contains videos recorded under more diverse and realistic human-computer interaction scenarios. The authors also compared their method against four previous methods proposed in literature by [20], [21], [22] and [23]. The authors find that all tested methods, including their own, were able to achieve almost perfect performance for HR estimation on the VideoHR database due to the controlled nature of the data. However, when testing on the MAHNOB-HCI database, the authors' method significantly outperformed all previous methods in all metrics used by the authors, achieving a root mean square error (RMSE) of 7.62, mean error rate percentage of 6.87$\%$ and Pearson's correlation coefficient of 0.81 for the task of predicting average HR in 30 second clips extracted from videos in the MAHNOB-HCI database. The authors also demonstrate that each of the steps performed in their framework improves the performance of the model.

In their paper, Álvarez Casado and Bordallo López [4] propose and evaluate a number of unsupervised approaches for extracting blood volume pulse signals from facial videos. The authors propose a novel framework for rPPG extraction based on a generic unsupervised pipeline, which the authors aim to improve with novel techniques. The main contributions to the pipeline by the authors are a new stabilization method for faces based on rigid mesh normalization to increase robustness to variations in pose and movement, a dynamic ROI selection scheme based on statistical and fractal analysis for choosing the best facial regions from which to measure the signals, and a novel method for rPPG extraction from RGB signals called Orthogonal Matrix Image Transformation (OMIT) based on QR decomposition.

The pipeline employed by the authors is a typical example of an unsupervised rPPG extraction pipeline, which can generally be split into three main phases: selecting the face regions for the measurements, extracting the rPPG signals from the color variations in the selected areas, and the computation of heart rates and other relevant parameters. The modular pipeline utilized by the authors consists of a total of 8 modules: a database interface for loading the data, detection and alignment of the face, ROI selection, extracting the raw RGB signal as the mean value of the ROI pixels, pre-processing of the extracted RGB signal, RGB to PPG transformation (rPPG), frequency analysis for HR computation, and evaluation of the HR estimation performance [4].

The authors base their rPPG pipelines on one included in version 0.0.4 of the PyVHR framework [24], to which the authors refer as the "baseline" pipeline. The authors then aim to improve upon the selected baseline by introducing changes to several steps, proposing three new versions called the "improved", "normalized" and "multi-region" pipelines. The improved pipeline introduces a few changes to the baseline in order to enable better reproducibility and fairer comparison to other methods. The normalized pipeline improves on the previous approaches by employing a skin segmentation method based on geometric normalization. Finally, the multi-region pipeline aims to improve on the normalized pipeline by leveraging the fact that the employed skin segmentation method produces a set of facial regions which can be processed separately. The authors develop a method for automatically choosing the regions which exhibit the most BVP information based on statistical parameters and discard the regions which are deemed less relevant [4].

The PyVHR framework includes implementations for several reference rPPG methods such as POS [25], CHROM [26], GREEN [10], PCA [27], ICA [22], 2SR [13], LGI [28] and PBV [29], which are all employed and evaluated by the authors of [4] as part of each of the four proposed pipelines. The authors also test an rPPG method proposed in [30], which utilizes the chroma channel a in the CIE Lab color space.

In addition, the authors propose a novel approach for RGB to PPG transformation, Orthogonal Matrix Image Transformation (OMIT), which is also integrated in the four pipelines and compared against previously proposed alternatives. The proposed approach attempts to improve robustness against noise and artifacts in the input compared to previous rPPG methods. The OMIT method consists of the three key steps of calculating a reduced QR decomposition with Housenholder reflections, calculation of a subspace projection matrix, P, and finally, using P to project the input RGB data to a subspace orthogonal to the first column q_1 of the matrix Q, whose columns represent the orthonormal basis of the input matrix [4].

The authors of [4] evaluate all 10 rPPG methods mentioned as part of all four proposed pipelines. The authors evaluate the methods on six publicly available benchmark data sets: PURE [31], COHFACE [32], the LGI-PPGI-Face-Video-Database [28], the two datasets, UBFC1 and UBFC2, forming the UBFC-RPPG Video dataset [33] as well as MAHNOB-HCI [19]. The authors evaluate the proposed pipelines based on HR prediction accuracy as measured by the mean absolute error (MAE) and Pearson's correlation coefficient (PCC) between the predicted and ground truth heart rate envelopes. For MAE, the standard deviation (SD) is listed in addition to the mean.

The best experimental results in all six data sets were achieved using the multi-region pipeline which incorporates all the changes proposed by the authors. Of the different rPPG conversion methods, the authors report CHROM and POS to perform the best on uncompressed data, while their proposed OMIT method performs the best on heavily compressed data, outperforming the other methods on the compressed data in the MAHNOB data set. The authors also note that the level of performance that the pipelines can achieve depends strongly on the nature of the data in question, as achieved error levels on UBFC and PURE reach below 2 BPM, the best result on LGI-PPGI just below 4 BPM and the average errors on the highly compressed COHFACE and MAHNOB data sets between 8 and 12 BPM [4].

The multi-region pipeline proposed by the authors outperforms its alternatives when presented with videos recorded under natural conditions, especially on data which shows variations in facial expressions, head movements and changes in illumination.

However, achieved performance improvements are only modest on data which shows subjects remaining still in front of the camera, such as PURE and UBFC. The authors also demonstrate the performance improvements gained using the multi-region pipeline on natural data by comparing its results against the baseline pipeline on the four different scenarios in LGI-PPGI, resting, rotation, talking and gym, separately, showing significant performance improvements on the more natural talking and gym scenarios, with an especially drastic improvement in performance seen in the gym scenario. The authors also compare their multi-region pipeline against state of the art unsupervised and supervised methods, showing that it can outperform the previously proposed unsupervised methods and achieve performance comparable to state of the art supervised methods with lower computational cost and no requirement of training [4].

2.3. Supervised Methods

Unlike unsupervised rPPG extraction pipelines, in which each phase of the pipeline is defined explicitly by the developer, supervised methods leverage a training process on relevant data to learn to perform one or more of the required steps automatically. Most supervised methods applied in rPPG extraction are based on a machine learning approach called neural networks (NNs). The current state of the art in rPPG extraction is represented by so-called end-to-end approaches based on deep neural networks, which attempt to combine all the necessary steps required to achieve the desired rPPG signal from a facial video into a single model. The best performance in the task of remote PPG extraction is achieved by a type of neural network called convolutional neural networks (CNNs), which have been shown to perform well in a variety of computer vision applications. CNNs are also serve as the basis for the operational neural networks (ONNs) proposed in [1], which in turn are the basis of the self-organized operational neural networks (Self-ONNs) proposed in [2]. The different types of neural network models and their characteristics will be covered in more detail in Chapter 3.

A neural network is trained by providing samples of input-output pairs and updating the parameters of the network iteratively based on a target function calculated between them. The training process aims to minimize the value of the target function, which is achieved by calculating the gradient of the function and performing backpropagation to compute the contribution of each parameter in the model so that their values can be updated accordingly. In an end-to-end approach for rPPG extraction from faces, the input is typically a facial video while the ground truth output is a PPG signal recorded from the finger. Through the training process, the network learns all the steps required to produce the desired outputs from its inputs simultaneously and automatically, which eliminates the need for the feature engineering typical of unsupervised methods. However, the computational requirements of deep neural networks may be high compared to unsupervised approaches, and the quality of the learned representation of any supervised model is entirely dependent on the quality of the training samples provided.

Yu et al. [3] proposed a number of spatiotemporal neural network models for the purpose of remote PPG extraction. The authors propose two model architectures based on 3D CNNs, which treat the input videos as a 3D volume, and three long short term memory (LSTM) neural network models. The implementations based on LSTMs also incorporate a 2D CNN model for extracting spatial features, forming a time series to serve as input for the actual LSTM models [3]. One of the 3D CNN models proposed by the authors, PhysNet-3DCNN-ED, employs an encoder-decoder structure in its architecture. This model also served as the basis for the models evaluated in the experimental section of this thesis and is described further in Section 4.2. A diagram of the architecture of the model can be seen in Figure 6, in which it is referred to as 'CNN-deep'.

The models are trained by the authors in an end-to-end manner using samples in the OBF database, with the included PPG signals serving as the ground truth for the input videos. The videos are preprocessed to contain only the face and resized to 128 by 128 pixels. Both the input videos and the ground truth PPG signals are resampled to a sampling rate of 30 Hz. The entire videos are not input into the models at once, and four different lengths of 32, 64, 128 and 256 frames for the training clips are used in the experiments instead. The authors test two different loss functions when training their models: the mean square error (MSE) and negative Pearson's correlation (NPC) losses [3].

The authors evaluate the performance of the proposed models based on HR and HRV features calculated from the output rPPG and ground truth signals using peak detection. The metrics chosen for evaluating HR and HRV prediction performance are standard deviation (SD), root mean square error (RMSE), Pearson's correlation coefficient (PCC) and mean absolute error (MAE). The authors also evaluate their models in experimental settings for atrial fibrillation detection and emotion recognition, for which the evaluation metrics of accuracy and specificity are used [3].

The models are first evaluated in a 10-fold cross validation setup on the OBF database, for which performance in HR and HRV estimation as well as AF detection accuracy based on HRV features is reported. The best performance for HR prediction was achieved by the proposed PhysNet-3DCNN-ED model when the length of the training clips was set to 128 and the NPC loss was used, resulting in an RMSE of 1.812 and PCC of 0.992 for the HR estimates. The authors also extract ten-dimensional HRV features for use in an AF detection setup on OBF, for which the authors report an accuracy of 80.22 % and specificity of 81.87 %. This model was then evaluated on the MAHNOB-HCI data set to evaluate its ability to generalize to data outside the OBF database used in training the model. The authors report an SD of 7.84, MAE of 5.96, RMSE of 7.88 and PCC of 0.76 for the task of HR prediction on the MAHNOB-HCI data set [3].

The model was also tested in an emotion recognition setup on MAHNOB-HCI. The authors extract the same ten dimensional HRV features as for AF detection, which are used to detect three levels of arousal and valence as well as nine categories of emotions. The authors report accuracies of 46.86 % and 44.02 % in the 3-class valence and arousal prediction tasks, respectively, and an accuracy of 29.79 % in the 9-class emotion classification task [3].

Sun et al. [11] developed an approach based on deep learning for a specific application of rPPG: atrial fibrillation (AF) detection from facial videos. The authors base their method on a slightly modified version of the PhysNet-3DCNN-ED model proposed in [3]. To train their model, the authors use a binary time series indicating the systolic peak locations detected in the PPG signal measured from the subject as the ground truth for the corresponding facial videos as opposed to utilizing the entire PPG waveform as in [3]. The choice is made to prevent the model from learning unnecessary features inherent to PPG signals, such as diastolic peaks and artifacts, instead retaining only the information that is required for calculating HR and HRV features, i.e., the timing of the systolic peaks. To that end, the authors also propose a novel loss function called Wasserstein distance. The authors find that utilizing their newly proposed loss to train the proposed model leads to better performance in HR and IBI prediction compared to alternative loss functions suitable for training on a binary time series. In addition, a model trained using the systolic peaks using their newly proposed loss function outperforms an equivalent model trained on the entire PPG waveform [11].

The authors record and conduct their experiments on the full version of the Oulu Bio-face (OBF) dataset, which consists of 100 facial videos recorded of healthy subjects and 100 recorded of AF patients along with the corresponding PPG and ECG recordings. The subset consisting of recordings of healthy subjects is referred to as OBF-H and the subset recorded of AF patients as OBF-P. For classification purposes, the samples of AF patients were labeled as showing atrial fibrillation (AF), sinus rhythm (SR) or atrial flutter (AFL), with certain complex cases labeled as 'other'. 73 of these videos were labeled as AF, 61 as SR, 11 as AFL, and 24 were labeled as other and excluded from classification experiments [11].

The authors test the applicability of their method to AF detection in two classification setups. In the first experimental setup, videos of healthy subjects and videos of AF patients labeled as containing AF are used in a binary classification scheme of healthy vs. AF. In the second setup, videos of patients labeled as SR and videos labeled as AF are used to classify SR vs. AF. The authors also test the feasibility of their approach for AFL detection in a setup for classifying samples labeled as SR against samples labeled as AFL. To perform classification, 20 HRV features are extracted from the signals and used to train a support vector machine (SVM) classifier. In the first two setups, subject-independent 10-fold cross-validation is used. In the SR vs. AFL classification experiment, 6 patients with SR and 6 with AFL are selected randomly to form the set of training data while another 5 patients with SR and 5 with AFL are chosen for the test set [11].

The authors report an accuracy of 96.00 %, sensitivity of 95.36 % and specificity of 96.12 % for the first experiment of classifying healthy subjects against samples containing AF when utilizing their newly proposed deep learning pipeline. For reference, the authors list the classification performance achieved using the recorded ECG signals and the same HRV features, which resulted in an accuracy of 99.38 %, sensitivity of 99.63 % and specificity of 99.24 %. In the second experiment of classifying samples of patients with AF vs. patients with SR, the proposed approach achieved an accuracy of 95.23 %, sensitivity of 98.53 % and specificity of 91.12 %, with the corresponding reference ECG results for the experiment measuring an accuracy of 97.89 %, sensitivity of 99.31 % and specificity of 95.85 %. The authors also demonstrate the feasibility of their approach in AFL detection with an accuracy of 88.43 %, sensitivity of 90.04 % and specificity of 87.04 % in the AFL vs. SF classification task, with the corresponding reference ECG signal resulting in an accuracy of 91.96 %, sensitivity of 92.31 % and specificity of 91.36 %. The experiments producing these results were conducted using 30-second clips extracted from the samples. The authors also compare their method against other rPPG pipelines proposed in [3], [34], [25] and [29] in the same classification setups and find that their proposed approach outperforms the alternatives in all settings based on all the metrics used, i.e., accuracy, sensitivity, specificity, F1 score and area under the receiver operating characteristic curve (AUC) [11].

2.4. Data Sets Suitable for Remote PPG

To evaluate a pipeline for rPPG extraction, suitable data consisting of videos showing the face and at least one corresponding relevant physiological signal, such as PPG or ECG signals, is required. This section lists some data sets of facial videos that contain ground truth recordings of cardiac activity in the form of PPG or ECG signals and have been used for evaluating the performance of rPPG extraction frameworks in literature.

2.4.1. Oulu Bio-Face (OBF)

The Oulu Bio-Face (OBF) database [35] consists of facial videos and reference physiological signals recorded of 100 healthy subjects with no history of heart disease as well as 6 subjects diagnosed with atrial fibrillation (AF). According to the authors, the OBF database is the first example of a database built specifically for the purpose of evaluating methods of measuring physiological signals remotely. Of the healthy subjects, 61 were male and 39 were female, 39 wore eyeglasses, and the average age was 31.6 years. 32 of the healthy subjects were white, 37 were Asian and the remaining 31 were of other ethnic background. The subjects with AF had an average age of 68.1 years, were all white, and were split evenly between male and female subjects. In addition, half of them wore eyeglasses. Two recording sessions took place for each subject, lasting five minutes each. In the case of healthy subjects, five minutes of exercise by climbing the stairs took place between the two recording sessions, resulting in a higher heart rate during the second session. For the subjects with atrial fibrillation, the first session was recorded before they underwent cardioversion treatment for their condition, meaning that they showed symptoms of AF during recording, and the second session took place three hours after the cardioversion treatment took place, at which point the patients show a regular sinus heart rhythm. All the videos were recorded under controlled conditions in artificial light, with subjects facing the camera throughout the videos. The facial videos were recorded on a Blackmagic URFA mini RGB camera at a resolution of 1920 by 1080 pixels at a frame rate of 60 frames per second. In addition to RGB video, near infrared (NIR) video was recorded using a specialized camera during each session at a resolution of 640 by 480 pixels and a frame rate of 30 frames per second. Three ground truth physiological signals were recorded during each recording session: an ECG signal, a respiration signal and a BVP signal. The ECG signal was recorded at a sampling rate of 256 Hz, the respiration signal at a sampling rate of 32 Hz, and the BVP signal had a sampling rate of 128 Hz. The cameras and all biosensors were synchronized by passing a visual cue to the cameras and pressing a trigger button on the biosignal data acquisition device simultaneously at the beginning and end of each recording.

Figure 2 shows examples of three facial videos from OBF to demonstrate the recording conditions. The presented examples have been postprocessed to conceal the subjects' identities.

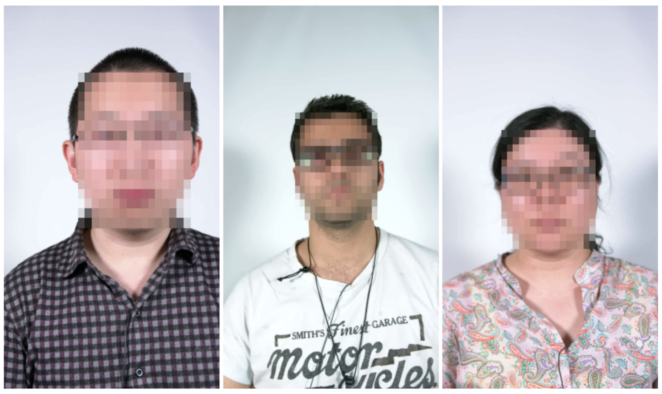

Figure 2. Examples of three facial videos from the OBF database.

2.4.2. COHFACE

The COHFACE data set [32] consists of 160 videos recorded of 40 healthy subjects. 12 of the subjects were women and 28 were men, and the average subject age was 35.6 years. The videos were recorded of the subjects sitting still in front of a webcam under two different types of lighting conditions: "studio" lighting, during which the blinds in the room were closed and the subject's face was illuminated using a spotlight, and "natural" lighting, during which all lights in the room were turned off and the blinds were open. Two sessions were recorded of each subject under both lighting conditions, resulting in a total of four recordings of each subject. Figure 3 shows examples of the two lighting conditions for one subject, with the facial features blurred for privacy. Each video was approximately 60 seconds long and recorded using a consumer grade webcam, Logitech HD Webcam C525, with a resolution of 640 by 480 pixels at 20 frames per second. In addition, blood volume pulse and respiration signals are recorded of the subjects during each session using measuring equipment manufactured by Thought Technologies. The BVP signals were recorded at a sampling rate of 256 Hz and the respiration signals were recorded at a sampling rate of 32 Hz. The physiological signals were synchronized with the corresponding video sequence during recording by a software suite provided by the manufacturer of the physiological signal measurement equipment.

2.4.3. LGI-PPGI-Face-Video-Database

The LGI-PPGI-Face-Video-Database [28] consists of videos recorded of 25 subjects, although only the recordings of 6 of the subjects have been made publicly available.

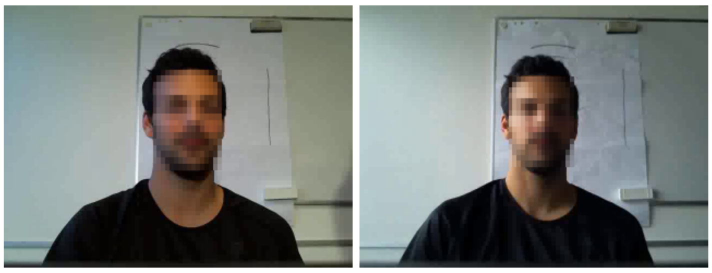

Figure 3. Examples of the two lighting conditions under which the data in COHFACE was recorded, with an example of natural light on the left and studio light on the right.

20 of all the participating subjects are male and 5 are female. Of the 6 subjects in the publicly available part of the data, five subjects are male and one is female. The age of the subjects ranges between 25 and 42 years and the majority race is white. Four videos were recorded of each subject under varying conditions, resulting in a total of 100 videos. The first of the recording sessions involves a resting scenario with minimal changes in face and head position or illumination. In the second session, illumination also remains static, but the subjects are instructed to move their face and head. The third session is recorded during exercise on a bicycle ergometer with no specific instructions given to the subjects. The fourth session is recorded during a conversation in an urban environment with natural variations in illumination. The length of the videos is one minute except for the sessions recorded during exercise, which are five minutes long. All of the videos were recorded using a consumer grade webcam, the Logitech HD C270, at a frame rate of 25 frames per second. PPG ground truth signals were recorded for each session using a CMS50E finger pulse oximeter at a sampling rate of 60 Hz.

Figure 4 shows examples of the four different recording sessions for a subject from the LGI-PPGI-Face-Video-Database, demonstrating the differences in lighting conditions and the pose of the subject. The facial features in the presented examples were blurred to protect the subject's privacy.

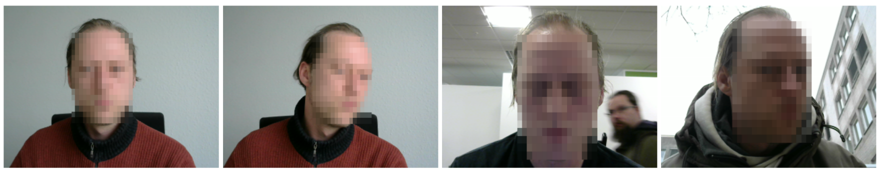

Figure 4. Examples of the four different recording scenarios in the LGI-PPGI-Face-Video-Database in the following order from left to right: resting, rotation, gym and talking.

2.4.4. MAHNOB-HCI

The MAHNOB-HCI database [19] is a multimodal set of data collected primarily for the purpose of emotion recognition research. However, as the database contains facial videos as well as corresponding synchronized ECG recordings of the subjects, which can be used for extracting ground truth HR and HRV features, the database is also suitable for evaluating the performance of rPPG extraction frameworks. The database consists of synchronized recordings of face videos, audio signals, eye gaze data and various physiological signals recorded in a multimodal setting. The modalities of interest in an rPPG experimental setting are the color facial video, which was recorded using an Allied Vision Stingray F-046C color camera at a resolution of 780 by 540 pixels at 60 frames per second, and the ECG signals, which were recorded using three electrodes at a sampling rate of 1024 Hz but later downsampled to 256 Hz for more efficient processing. The videos in the database are stored in a compressed H.264 format, which poses challenges for the remote extraction of PPG signals. The samples were recorded of 27 subjects of whom 11 were male and 16 were female. The age of the subjects ranged between 19 and 40 years [19].

3. OPERATIONAL NEURAL NETWORKS

Operational neural networks (ONNs) are a novel neural network technology proposed by Kiranyaz et al. in 2020 [1]. The ONN networks are based on the operational neuron, which is an extension of the convolutional neurons used in conventional CNNs. While the traditional convolutional neural networks are homogeneous with their configuration based on the linear neuron model, ONNs can be heterogeneous, consisting of operational neurons with any set of operators. By proposing ONNs as a more generalized alternative to the conventional CNNs by extending on them with a neuron model which can encompass nonlinear operations, the authors attempt to overcome the limitations imposed by the linear neuron model which limits the performance of CNNs in problems with highly complex and nonlinear solution spaces [1].

In addition, an extension of ONNs called self-organized operational neural networks (Self-ONNs) has recently been proposed in [2]. The newly proposed Self-ONNs attempt to overcome certain observed shortcomings in the ONN model by employing so-called generative neurons, which allow the model to self-organize the operators of the network during training in an unrestricted manner with backpropagation. This results in the operators being tailored to the specific task at hand as opposed to being selected from a fixed operator library, which can result in improved performance for a number of applications as well as improved scalability for large scale problems with high complexity due to eliminating the computationally expensive search process for the best operators.

This chapter discusses the motivation and background behind ONNs as well as their theoretical foundation and practical implementation as proposed in [1]. In addition, the current and potential applications of ONNs are discussed, and the advantages and limitations of ONNs against other existing alternatives are evaluated. The chapter concludes by providing a similar review of Self-ONN models, covering their motivation and background, theoretical foundation, practical implementation as well as applications, advantages and shortcomings.

3.1. Background and Motivation

ONNs are a novel type of neural network which is based on CNNs and a novel neural network model called generalized operational perceptrons (GOPs), both of which are ultimately based on multi-layer perceptrons (MLPs) which are universal approximators loosely inspired by biological neurons. By solving an optimization problem to achieve the output weights of the network, the conventional fully connected and feed-forward neural networks such as MLPs have been shown to be capable of approximating any continuous function as long as the neural units of which they consist are able to perform nonlinear piece-wise continuous mappings of their inputs and the network has a sufficient number of layers, resulting in high enough computational capacity. However, such artificial neural networks traditionally exhibit a homogeneous structure and a neuron model which is rather a crude approximation of the biological neurons inspiring it, only being capable of performing the linear transformation and weighted sum operations. The real biological neural systems, on the other hand, are heterogeneous and consist of a large variety of specialized neuron types which differ in their structural, biochemical and electrophysiological properties. As such, although the typical homogeneous neural networks are capable of learning to approximate the corresponding responses of training samples, they may be unable to learn the underlying functional form which maps the inputs to the outputs of the problem. Such homogeneous networks based on the traditional neuron model may often fail to produce satisfactory results for problems with highly complex and nonlinear solution spaces. Extensions of MLPs, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and long-short term memories (LSTMs) also inherit the limitations caused by the traditional neuron model [1].

To address the limitations of the traditional neural networks, a novel feed-forward and fully connected neural network model, called generalized operational perceptron (GOP), has been proposed. These models attempt to model the biological neuron more accurately with a neuron model which incorporates features of biological neurons not represented in the traditional neuron model, such as variations in the synaptic connections between nerve cells, which are nonlinear in general. The GOP neuron model is an extension of the MLP neuron which adds a "nodal" operation, which corresponds to the synaptic connections in the dendrites of biological neurons, and a "pool" operation, corresponding to an spatial and temporal integration of incoming signals in the soma of a biological neuron. The GOP neurons directly adopt the "activation" operation of MLPs, which in a biological sense corresponds to the pooled potentials exceeding a certain limit and the consequent release of a series of action potentials in the axon of the nerve cell. The GOP neurons generalize the fixed linear model of MLPs and allow any set of nonlinear transformations to be used in defining the transformations applied to the input signals at the neuron level. The GOP neurons are a superset of MLP neurons, also known as linear perceptrons, which allow for more compact and efficient neural network architectures with superior performance resulting from better encoding of the input signals due to combining linear and nonlinear operations. In fact, some studies have shown that GOPs can achieve respectable results on many challenging problems which MLPs entirely fail to learn to solve such as "two spirals", "N-bit parity" for $N > 10$ and "white noise regression". As a superset of MLPs, a GOP network is capable of reverting to a conventional MLP, but only in the case that the learning process defining the operators of the neurons determines that the native operators of an MLP best fulfill the learning criterion [1].

GOPs are also the reference point for the ONNs proposed by the authors since they share the underlying philosophy of generalizing the homogeneous networks utilizing only the linear neuron model with a heterogeneous model utilizing an "operational" neuron model capable of encapsulating any set of linear or nonlinear operators. Much like GOPs are a generalization of MLPs, the neuron model of ONNs attempts to generalize the linear neuron model conventionally employed in CNNs with any linear or nonlinear operator. As an extension of MLPs, conventional CNNs similarly rely on linear transformations to produce their output, and because of this, in many challenging applications only deep CNNs with a very large number of layers and substantial computational complexity are capable of learning meaningful representations and achieving satisfactory performance [1].

3.2. Theoretical Foundation

Operational neural networks are based on the principles behind two previously proposed types of neural networks: CNNs and GOPs. ONNs retain the two defining properties of weight sharing and limited connectivity of CNNs but extend upon using linear convolutions exclusively by adding the nodal and pool operators as found in GOPs. Much like GOPs are a superset of MLPs, the proposed ONNs become a superset of CNNs by definition [1].

Conventional convolutional neural networks are based on the same neuron model as MLPs, but employ two additional restrictions of weight sharing and limited connections. In a fully connected MLP without these restrictions, every pixel in the feature map of a layer would be connected to every pixel in the corresponding feature map of the previous layer, leading to an exceedingly large number of trainable parameters which are infeasible to optimize efficiently [1]. Employing these two restrictions, which exploit the fact that the most relevant information to a given region is most likely found in the neighboring pixels, has allowed CNNs to be applied efficiently to applications of computer vision in particular. The output of a convolutional layer serving as the input map of the next layer neuron, x_k^l, is calculated by cumulating the output maps, y_i^{l-1}, of the previous layer convolved with the kernels, w_{ki}^l of the current layer [1]. The corresponding mathematical formula for 2D convolutional layers is presented in Equation 1 [1].

$$\begin{gathered}
x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} \text{conv2D}\left(w_{ki}^l, y_i^{l-1}, \text{ 'NoZeroPad' }\right) \\
\therefore x_k^l(m,n)\Big|_{(0,0)}^{(M-1,N-1)} = \sum_{r=0}^{2}\sum_{t=0}^{2} \\
\left(w_{ki}^l(r,t) y_i^{l-1}(m+r,n+t)\right) + \cdots
\end{gathered} \tag{1}$$

ONNs extend upon this principle by adding the nodal and pool operations. To calculate the output of an operational layer, x_k^l, the final output maps of the previous layer, y_i^{l-1}, are first pooled and then operated on using the kernels, w_{ki}^l, of the current layer, leading the computation to take the form presented in Equation 2 as defined in [1].

$$\begin{gathered}
x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} \text{oper2D}\left(w_{ki}^l, y_i^{l-1}, \text{ 'NoZeroPad' }\right) \\
x_k^l(m,n)\Big|_{(0,0)}^{(M-1,N-1)} = b_k^l + \\
\sum_{i=1}^{N_{l-1}} \left(P_k^l \begin{bmatrix} \Psi_{ki}^l\left(w_{ki}^l(0,0), y_i^{l-1}(m,n)\right), \ldots, \\ \Psi_{ki}^l\left(w_{ki}^l(r,t), y_i^{l-1}(m+r,n+t), \ldots\right), \ldots \end{bmatrix} \right)
\end{gathered} \tag{2}$$

In Equation 2, P_k^l denotes the pool operator of neuron k in the current layer l and Ψ_{ki}^l denotes the nodal operator associated with kernel number i of the same neuron. The authors of [1] also point out that by comparing Equations 1 and 2, it can be seen that when the pool operation is summation and the nodal operator is multiplication, i.e., $P_k^l = \Sigma$ and $\Psi_{ki}^l(y_i^{l-1}(m,n), w_{ki}^l(r,t)) = w_{ki}^l(r,t) y_i^{l-1}(m,n)$, for every neuron in an ONN, the resulting homogeneous ONN is identical to a corresponding CNN, indicating that ONNs form a superset of CNNs as mentioned.

Figure 5 presents an illustration of the operation of CNN and ONN models as defined in the equations above, with three 3x3 convolutional layers presented in the upper diagram and three 3x3 operational layers presented in the diagram below it. The diagrams illustrate how the *k*th neuron of layer *l* of the network operates on the previous layer outputs with a size of 22 by 22 to produce its output.

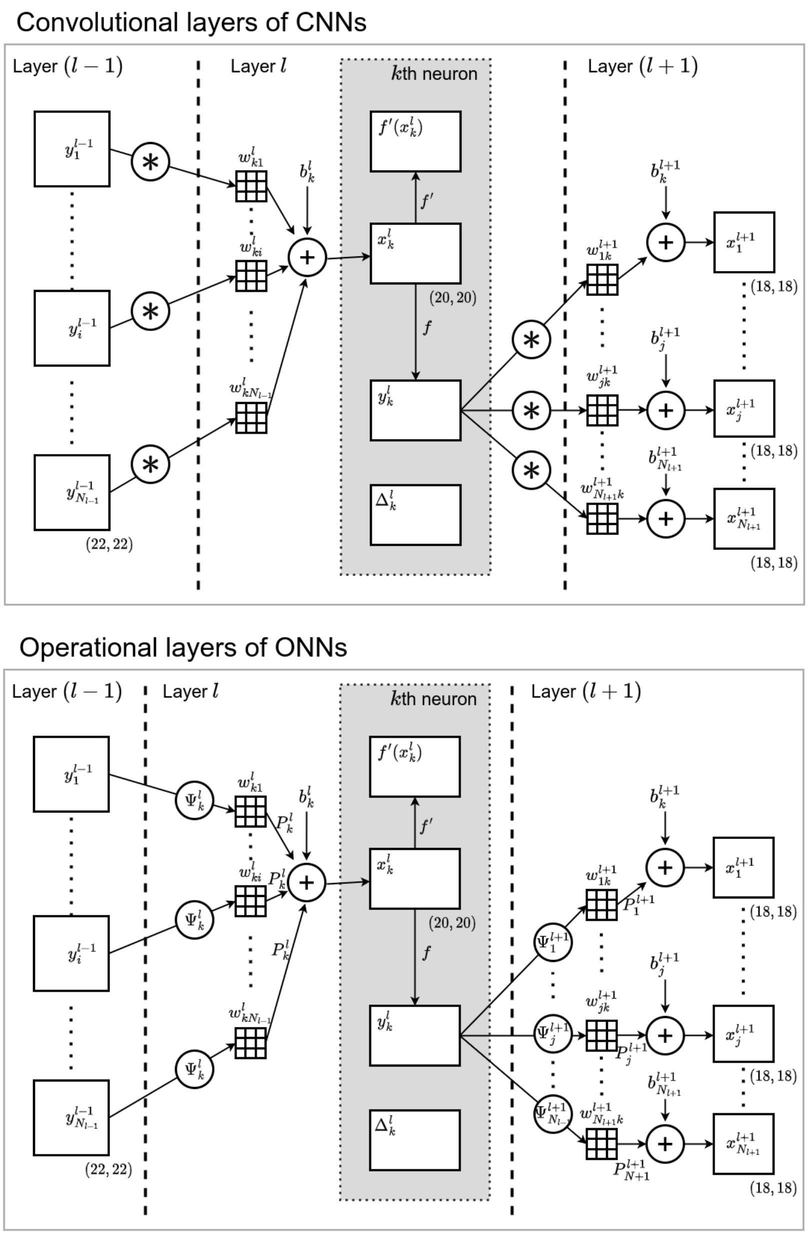

Figure 5. A figure demonstrating the operation of the *k*th neuron of layer *l* of a CNN in the upper diagram and ONN in the diagram below, illustrating its relation to the previous and following layers.

3.2.1. Training with Backpropagation

The BP training of ONNs as formulated by [1] consists of four phases: (1) computation of the delta error, Δ_1^L, at the output layer, (2) inter-BP between two operational layers, (3) intra-BP in an operational neuron and (4) computation of the weight (operator kernel) and bias sensitivities for the purpose of updating them at each iteration of BP. Phase 3 is also responsible for any pooling operations, i.e., up- or downsampling, if they are applied at the neuron. The training of ONNs with backpropagation (BP) is based on the same principles as training CNNs. Some steps in backpropagation training, such as the entirety of phases 1 and 3, are common between ONNs and CNNs, but certain differences arise due to the added nodal and pool operations of ONNs.

3.3. Implementation

To implement the proposed ONN model in practice, the authors of [1] present implementations for the forward propagation (FP) and backpropagation (BP) operations based on the four phases described earlier. The authors also present a method for searching for the optimal operator set for each neuron by performing short training sessions with BP utilizing potential operator set candidates. After this, the ONN with the best operators can be trained on the designated set of train data for a given problem.

The pseudocode for the modified backpropagation algorithm for ONNs as described in [1] is presented in Algorithm 1. Some details have been omitted here for brevity, but the algorithm still captures the underlying principle behind the process of training ONNs with backpropagation. The full details of the algorithm, including the relevant mathematical equations, can be found in the original publication [1]. In the algorithm, *iterMax* and *CP** are examples of stopping criteria for the training process, with *iterMax* denoting a maximum number of iterations and *CP** referring to a target classification performance [1].

Algorithm 1. Backpropagation algorithm for ONNs

```
    Input  : ONN, Stopping Criteria (iterMax, CP*)
    Output: BP trained ONN* = BP(ONN,iterMax,CP*)
 1  Initialize all parameters (e.g., randomly)
 2  until a stopping criterion is reached iterate:
 3      for each item in the train dataset do:
 4          FP: Forward propagate from the input layer to the output layer to find
              the outputs and required derivatives and sensitivities for BP for each
              neuron $k$ at each layer $l$.
 5          BP: Compute delta error at the output layer and backpropagate the
              error back to the first hidden layer to compute delta errors of each
              neuron $k$, $\Delta_k^l$, at each layer $l$.
 6          PP: Find the bias and weight sensitivities.
 7          Update: Update the weights and biases with the sensitivities found in
              the previous step, scaled with a learning factor $\epsilon$.
 8      end
 9  end
10  return ONN*
```

For the purpose of determining the best set of operators for each operational neuron, the authors of [1] utilize the greedy iterative search (GIS) algorithm [36, 37]. The GIS algorithm performs a pruned search for the best operators of one layer at a time, which means that each neural layer will be homogeneous with all of its neurons sharing the same operator set. The operator set library, denoted by $\{\theta_N^*\}$, consists of N operator sets, each of which represents a unique combination of a nodal, pool and activation operator. With each pass, GIS searches for the best operator set from the library for a particular layer, while the operator sets of other layers remain fixed. To that end, a small number of short BP runs with randomly initialized parameters are performed with each possible operator set assigned to the given layer in turn. The operator set that leads the ONN to perform the best is assigned to the layer, and GIS moves on to the next layer until the search has been performed for every layer in the network. The authors assert that only few passes of GIS should suffice to determine a near-optimal ONN configuration since with each pass, the best operator set always remains assigned to each layer [1].

Algorithm 2 presents the pseudocode for two-pass GIS for forming a near-optimal ONN, ONN*(θ), as presented in [1]. In the algorithm, $\{\theta_N^*\}$ refers to the operator set library as described earlier, and N_{BP} denotes the number of backpropagation runs performed for testing each operator set for a given layer during each pass of GIS. Much like in Algorithm 1, *iterMax* and *CP** represent the stopping criteria of maximum iterations and target classification performance set for the problem at hand, and the GIS algorithm is set to stop early if the learning objective set for the problem, i.e., target *CP**, is met during the search [1].

Algorithm 2. Two-pass GIS

```
   Input  : $\{\theta_N^*\}$, Stopping Criteria (iterMax, CP*), $N_{BP}$
   Output: ONN*($\theta$)
1  Initialization: Form an ONN with neurons having operator set (nodal, pool,
     activation) randomly selected from $\{\theta_N^*\}$.
2  for GIS-pass = 1:2 do:
3      for output layer l = L:1 do:
4          for $\forall\theta_i \in \{\theta_N^*\}$ do:
5              Assign the operator set of each neuron in the $l$th layer of the ONN
                 to $\theta_i \rightarrow$ ONN$(l, \theta_i)$
6              Perform: $N_{BP}\times$ BP(ONN$(l, \theta_i)$, iterMax, CP*) and Record:
                 ONN*($\theta_i$) that achieves the best performance
7          end
8          Assign $\theta_i^*$ as the operator set of each neuron in the $l$th layer of the
             ONN $\rightarrow$ ONN$(l, \theta_i^*)$
9          if CP* is reached in any BP run, break GIS
10     end
11 end
12 return ONN*($\theta$), the best performing ONN
```

3.4. Applications

As the proposed ONNs constitute a superset of CNNs, in principle they can be applied to all the same problems as conventional CNNs as well as additional problems in which CNNs may not be applicable due to their constraints. As CNNs are firmly established as a powerful tool for a variety of applications, especially in computer vision, it is clear that the potential applications for ONNs are similarly numerous. In addition, ONNs may be capable of superior performance compared to CNNs with the same configuration in many applications, although the actual benefit of opting for an ONN approach instead of a CNN depends on the specific problem in question.

To demonstrate the performance of their proposed ONN model, the authors of [1] draw direct comparisons between the performance achieved by ONN and CNN models in experimental setups for four tasks: image denoising, image synthesis, face segmentation and image transformation. The authors also impose certain restrictions in the experiments in an attempt to demonstrate the learning capabilities of ONNs. The restrictions are low resolution, which means limiting the resolution to 60 by 60 pixels, compact model, which means using a compact ONN configuration of *In* × *16* × *32* × *Out* and only optimizing the two hidden layers using GIS, scarce train data, which means using only 10 % of the data for the denoising and segmentation problems and evaluating using tenfold cross-validation, multiple regressions, which means that a single network is trained to regress multiple images in the two regression problems of image synthesis and transformation, as well as shallow training, which means limiting the maximum iterations of BP training. The authors also apply the same restrictions to the tested CNN models at first, but later relax the constraints to assess whether CNNs can achieve the same learning performance with a more complex configuration [1].

In the image denoising task, the models are trained to output the original images, which are downsampled and converted to grayscale, from their counterparts with added Gaussian white noise (GWN). The data used for this experiment consisted of 1500 images from the Pascal VOC database [38]. In the image synthesis task, the models are trained to generate 8 target images from 8 source images consisting of Gaussian white noise. The target images consisted of 80 randomly selected images from the Pascal VOC dataset, 8 for each of the 10 folds. In the face segmentation task, the models were trained to output the segmentation mask corresponding to the face from facial images. The experiment was conducted on the 1000 images from the FDDB face detection dataset [39], each containing one or more human faces. Finally, in the image transformation task, the models are trained to convert four source images to four corresponding target images. The data consisted of close-up face images, mostly obtained from the FDDB face detection dataset. In the first fold, the authors attempt to train the models to also perform the opposite transformation, i.e., convert the target images to their corresponding source images [1].

The authors conclude that ONNs exhibit superior learning performance compared to CNNs in all of the proposed problems. The difference in performance also increases as the problem becomes more difficult. In image denoising, the average performance on the train and test partitions as measured by SNR between the ground truth and output images were 5.59 and 5.32 dB for the ONN model, and 4.1 and 3.96 dB for the CNN, respectively. In the image synthesis task, which constitutes a more difficult problem, the average SNR level was 9.91 dB for the ONN and 5.02 dB for the CNN, which gives a difference of about 5 dB. In image transformation, which is considered the most difficult of the problems, the average SNR level is 10.42 dB for the ONNs and -0.083 for the CNNs, producing a difference of more than 10 dB. The CNN with the default configuration failed to produce the transformations in all 10 folds, which was also the case when a CNN configuration with 4 times the complexity, called CNNx4, was used even for a significantly simplified task of performing only one image transformation as opposed to four [1].

For the face segmentation task, two distinct ONN models were tested in addition to the CNN configuration. The two ONN models utilized the best and third best operator sets as determined by GIS on the first fold and were referred to as ONN-1 and ONN-3, respectively. The average performance as measured by F1 score between the ground truth and predicted segmentation masks was 58.58 % for the CNN, 87.4 % for ONN-1 and 79.86 % for ONN-3 on the train partition and 56.74 % for the CNN, 47.96 % for ONN-1 and 59.61 % for ONN-3 on the test partition [1].

3.5. Advantages and Limitations

As the experimental results collected by the authors which are summarized in the previous section suggest, the proposed operational neuron model shows significant advantages over the conventional CNNs with the most glaring difference appearing in the image transformation task for which even a CNN with twice the number of hidden neurons and about 4 times the number of parameters compared to the default configuration failed to produce acceptable results despite the ONN model being able to learn a solution. In fact, analysis of the two-pass GIS process revealed that there were at least 16 other potential ONN operator set configurations that would outperform the CNN. However, it is important to note that the experimental setup was intentionally constructed in a way that highlights the advantages of ONNs, and the experiments were also conducted with fairly small models and limited amount of data, serving as an early proof of concept. The actual performance benefit of opting for an ONN approach as opposed to CNNs, for example, ultimately depends on the specific problem at hand. It is also worth noting that the training of ONNs also poses some challenges of its own compared to CNNs. This section takes a closer look at the potential advantages and limitations of ONNs in terms of performance and efficiency, comparing against CNNs in particular.

In addition to performance, the measured efficiency of an ONN model against a corresponding CNN depends on its computational complexity. In their original publication, the authors of [1] present a detailed analysis of the computational complexity of an ONN model against a corresponding CNN, evaluating the complexity of both forward propagation (FP) as well as backpropagation. The authors assert that the difference in computational complexity between an ONN and a corresponding CNN depends entirely on the choice of the set of operators for each neuron or layer. In the worst-case scenario that can result with the operators options chosen for the experiments, i.e., choosing sinc, median and tanh as the operator set, the FP of an ONN will be 8.68 slower compared to the corresponding CNN. However, in the example experimental setups presented in their publication, the pool operator chosen by GIS turned out to always be summation, which would bring the worst-case figure to 2.704 times. In practice, the authors observed a deterioration in FP speed by 1.4 to 1.9 times for ONNs when comparing to an equivalent CNN. Through thorough analysis of the BP algorithm for ONNs, the authors also assert that the increase in the computational complexity of BP for ONNs compared to CNNs occurs due to the increased complexity of computing the initial FP as the complexity of each BP phase is the same for both network types. In practice, the authors observed the speed of BP iterations, with FP included, to be between 1.5 and 4.7 times slower for ONNs compared to the speed of a BP iteration in a corresponding CNN [1].

Based on the analysis presented above, the added nonlinearity of operational layers comes at the cost of added computational complexity, which is to be expected. However, based on the results achieved in the example experiments in [1], the resulting increase in learning performance is also significant, which results in ONNs producing a more efficient solution for the tested problems despite the increase in complexity compared to CNNs. The difference in efficiency can be observed most clearly in the image transformation task, in which the ONN significantly outperformed the CNNx4 model despite being more than twice as fast. In fact, as implementing deeper and more complex CNNs have become more viable with advances in modern hardware, most benefits of opting for an ONN approach instead would most likely come from the added efficiency associated with achieving the desired performance with a shallower architecture. However, it is also important to note that if the number of layers in an ONN is increased to a point in which a CNN with the same configuration is also sufficient for solving the specific training task at hand, the operator set of the ONN can revert to the native operators of the CNN as determined by GIS during the training process, which leads the ONN to also having the same computational complexity as the corresponding CNN. As such, the operator sets which increase complexity should not be chosen by GIS at least in principle unless they result in actual performance benefits for fitting the train data.

A challenge associated with ONNs worth noting is that as an ONN generally has higher capacity to fit its training data compared to an equivalent CNN, it is also more susceptible to overfitting. This was also demonstrated by the results of the face segmentation experiments presented in [1], which saw the ONN with the best operator set as determined by GIS, ONN-1, have about 8 % lower average generalization performance compared to its CNN counterpart on the test partition as measured by the F1 score despite scoring 29 % higher on the train partition on average. On the other hand, the ONN with the third best operator set, ONN-3, outperforms the CNN on both the train and test partitions, but the authors acknowledge that it may also suffer from overfitting due to exhibiting a significant gap in performance between the train and test sets [1]. These results highlight the importance of ensuring that the train data is sufficiently representative of the problem at hand when training an ONN model. While susceptibility to overfit is a problem shared by all supervised models with high capacity, the problem is magnified in the case of ONNs as unlike CNNs with a fixed operator set, the train data is also used for determining the set of operators used in the operational neurons. As such, in the worst case a set of operators which increases computational complexity over the native operators of a CNN may be chosen for the ONN by GIS during training, even if the choice were to lead the model to overfit. Such a scenario is detrimental to the applicability of the model to outside data as well as efficiency compared to a CNN alternative, as in such a case, the added complexity of the nonlinear operations degrades generalization performance as opposed to improving it. If the operator set is already determined based on the train data, it limits the possibilities to address the problem with certain techniques to combat overfitting such as early stopping because the best operators on the train data may not be the best in general, and may cause the model to overfit while also unnecessarily introducing added computational complexity. To address this problem, the authors suggest performing GIS over a validation set instead as a means to combat overfitting [1]. The added capacity of ONNs compared to CNNs should also be taken into account in designing the architecture of the network, as shifting from convolutional to operational layers implies that a smaller number of layers should suffice in general, and using a large number of operational layers may lead the model to overfit.

While ONNs can exhibit superior performance compared to CNNs due to their heterogeneous and nonlinear set of operators, finding the right operator set for ONNs poses certain challenges. Choosing an operator set suitable for a given task is crucial and a defining factor of ONN performance and efficiency. However, as implied by its name, the greedy iterative search algorithm chosen by the authors of [1] for choosing the set of operators does not provide an optimal solution as it cannot evaluate all possible operator set assignments to the layers. How the operator set library is formed is also a significant factor contributing to the efficiency of the achieved ONN model. On the one hand, constructing a library with many possible operator sets can provide more options to specifically tailor the operations to the task at hand. However, increasing the number of possible operator sets also complicates the task of choosing the best operator set, and some of the given options might even turn out to not be worth considering due to their high complexity or only providing negligible increases in performance compared to other alternatives. On the other hand, constructing an operator set library with fewer options can simplify the process of determining which of the operators are best suited for the given task but also limits the number of available options and may not contain the best operator sets for a given problem. As the choice of operator set is so crucial to the efficiency of ONNs, more research should be conducted to determine which operator set libraries and search strategies can yield the best results in practice.

3.6. Self-Organized Operational Neural Networks

Self-organized operational neural networks (Self-ONNs) have recently been proposed in [2] as an extension to the operational neural network model in an attempt to address certain limitations of the conventional ONNs proposed in [1]. Although the superior learning performance of ONNs compared to CNNs was both theorized and demonstrated in a few proof-of-concept experimental setups in [1], the ONN model exhibits certain limitations which limit its applicability to real-world large scale problems. The most significant limitation of conventional ONNs is the limited heterogeneity resulting from the usage of a single operator set for every neuron in a hidden layer, which leads each neuron to only use a single nodal operator for every kernel connection to the neurons in the previous layer. Another major limitation of ONNs is that the learning performance of an ONN model is heavily dependent on which operators, especially nodal operators, are present in the predetermined operator set library. This means that if a suitable operator set for the problem at hand is missing from the library, the learning performance of the ONN model will deteriorate. In addition, the GIS algorithm, which is applied for determining the operator sets used with the ONN model, is computationally demanding and requires many runs of backpropagation. The best operator sets as determined by GIS may also not be optimal for the problem in question, which further may further limit learning performance during training. Overall, applying the GIS algorithm significantly limits the scalability of conventional ONNs to complex setups with large-scale data sets or deep network architectures [2].

To address the limitations and drawbacks of conventional ONNs mentioned above, the authors of [2] propose the self-organized ONNs with generative neurons. As implied in the name, a Self-ONN can self-organize the operators of the network during training. The proposed Self-ONNs leverage the so-called "generative neuron" model, which eliminates the need for a predetermined operator set library as well as any search process for finding the optimal nodal operator. While the neurons in conventional CNNs and ONNs have static nodal operators, which can limit their performance in fitting a certain set of train data, the newly proposed generative neurons are able to create any arbitrary nodal function for each kernel element of each connection, addressing the problem of utilizing only a single nodal operator for all kernel connections in conventional ONNs. A generative neuron is a neuron with a composite nodal operator in which each kernel element of each kernel connection can have any arbitrary nodal function, $\mathbf{\Psi}$, which does not necessarily have to be a well-defined function such as linear, exponential or sinusoid. The nodal functions are generated iteratively during the training process without restrictions in a manner that maximizes the learning performance on the given set of train data by backpropagating the error at the output layer through the operational layers of the Self-ONN.

3.6.1. Generative Neurons

Generative neurons, which are the main difference between Self-ONNs and conventional ONNs, are neurons with a "composite nodal operator", which is created iteratively during training with backpropagation without restrictions. To generate a composite nodal operator capable of encompassing any arbitrary nodal functions without requiring too many trainable parameters, the authors leverage the fact that any function can be approximated using Taylor polynomials, which also have reasonable computational complexity. The Taylor approximation of the function f(x) near the point $x = a$ can be expressed as

$$\mathrm{f}(x) = \mathrm{f(a)} + \frac{\mathrm{f}'(\mathrm{a})}{1!}(x-a) + \frac{\mathrm{f}''(\mathrm{a})}{2!}(x-a)^2 + \frac{\mathrm{f}'''(\mathrm{a})}{3!}(x-a)^3 + \cdots \quad (3)$$

in which f' is the first, f'' is the second and f''' is the third derivative of f(x). As such, the composite nodal operator function can be formed by utilizing the *Q*th order truncated Taylor approximation as follows:

$$\Psi(\mathbf{w}, \mathbf{y}) = w_0 + w_1(y-a) + w_2(y-a)^2 + \cdots + w_Q(y-a)^Q \quad (4)$$

where $w_q = \frac{f^{(q)}(\mathrm{a})}{q!}$ is the *q*th parameter of the *Q*th order polynomial. During the training process, these parameters are optimized to form an approximation of the nodal operator function which best fits the train data for each kernel element of each connection to the previous layer neurons [2].

An issue with this approach of defining the nodal operator is that the presented approximation is only valid near the point $y = a$, and the approximation becomes coarser the farther the points are from a. However, the nodal operators operate over the outputs of the neurons in the previous layer, which are in turn bounded by the generative range of their respective activation functions. As such, if the activation function is a sigmoid, for example, the outputs y are bound within the range of [0, 1], and in this case, approximating the nodal operator functions for the middle point of the range, $a = 0.5$, a polynomial with a high enough degree is capable of approximating any arbitrary function within the range of the previous layer outputs. If the activation function is chosen as the hyperbolic tangent (*tanh*), which is bound in the range of [-1, 1], the choice of $a = 0$ is made and the *Q*th order Taylor approximation in Equation 4 simplifies to the Maclaurin series as follows:

$$\Psi(\mathbf{w}, \mathbf{y}) = w_0 + w_1 y + w_2 y^2 + \cdots + w_Q y^Q \quad (5)$$

In addition, the bias coefficient w_0 can be omitted as each neuron already contains a bias term, b_i^l, which can compensate for any offset by a constant [2].

3.6.2. Forward Propagation and Backpropagation for Self-ONNs

The formula for the forward propagation (FP) in Self-ONNs is similar to the one of conventional ONNs presented in Equation 2 but with two key differences. Firstly, the nodal operator functions with a single kernel element, $\Psi_i^{l+1}\left(y_k^l(m+r,n+t), w_{ik}^{l+1}(\mathrm{r},\mathrm{t})\right)$, is replaced with the approximation given by the composite nodal operator $\Psi_i^{l+1}\left(y_k^l(m+r,n+t), \boldsymbol{w_{ik}^{l+1}}(\mathrm{r},\mathrm{t})\right)$ as expressed by the *Q*th order Taylor approximation in Equation 4 or the Maclaurin series in Equation 5 if the *tanh* activation function is used. Secondly, the scalar kernel parameter, $w_{ik}^{l+1}(\mathrm{r},\mathrm{t})$, of in ONN neurons becomes a *Q*-dimensional array, $\boldsymbol{w_{ik}^{l+1}}(\mathrm{r},\mathrm{t})$, and the composite nodal operator function as expressed in Equation 4 or 5 is used for all the neurons in the network. As such, the individual nodal operators, Ψ_i^{l+1}, can be simplified to a unified expression of a single composite nodal operator, $\boldsymbol{\Psi}$. The composite nodal function for the kernel element $\boldsymbol{w_{ik}^{l+1}}(\mathrm{r},\mathrm{t})$ is presented in Equation 6 below [2].

$$\begin{aligned}
&\boldsymbol{\Psi}\left(y_k^l(m+r,n+t), \boldsymbol{w_{ik}^{l+1}}(\mathrm{r},\mathrm{t})\right)\\
&\quad = w_{ik}^{l+1}(\mathrm{r},\mathrm{t},1)y_k^l(m+r,n+t)\\
&\quad + w_{ik}^{l+1}(\mathrm{r},\mathrm{t},2)y_k^l(m+r,n+t)^2\\
&\quad + \cdots\\
&\quad + w_{ik}^{l+1}(\mathrm{r},\mathrm{t},\mathrm{Q})y_k^l(m+r,n+t)^Q
\end{aligned} \tag{6}$$

In Equation 6, the bias term $w_{ik}^{l+1}(\mathrm{r},\mathrm{t},0)$ is omitted as per reasoning presented previously. As a result, the kernel of a generative neuron becomes a 3-dimensional matrix in which $w_{ik}^{l+1}(\mathrm{r},\mathrm{t},\mathrm{q})$ represents the *q*th weight of the kernel element (r,t) [2].

Backpropagation for Self-ONNs follows the same basic steps as backpropagation for conventional CNN and ONN models as described in Section 3.2.1. The main differences between the backpropagation algorithms of Self-ONNs and ONNs arise from the presence of the composite nodal operator in the generative neurons of Self-ONNs. The backpropagation process for ONNs presented in Algorithm 1 also applies for Self-ONNs, with the main differences being in how the required values are calculated at each step. All the steps of backpropagation for Self-ONNs and the relevant equations are described in detail in [2].

3.6.3. Applications and Performance

To evaluate the performance of the proposed self-organized operational neural networks, the authors of [2] compare Self-ONN models against CNN and ONN alternatives in the same experimental setups as presented in [1], i.e., image synthesis, denoising, face segmentation, and image transformation. The same constraints to the training setup as described in [1] are also applied. The problems and constraints are described in Section 3.4 of this thesis. To keep the comparison between Self-ONNs and the other neural network models fair, the authors use a Self-ONN architecture of *In* × *6* × *10* × *Out* with *Q* set to 7 in all layers. This results in the Self-ONN model having approximately the same number of trainable parameters as the CNN and ONN alternatives. It is also worth noting that as a result, the Self-ONN models also have three times fewer hidden neurons than the CNNs or ONNs at 16 against 48. In addition, the pool function used in the Self-ONN models was fixed to summation and the activation function was fixed to hyperbolic tangent for simplicity. The Self-ONN models were trained using stochastic gradient descent (SGD) without momentum and with a fixed learning parameter, while an adaptive learning rate was used for training the CNN and ONN models. To choose the Self-ONN model used for each problem, three runs of BP were performed and the model achieving the minimum MSE loss was chosen by the authors [2].

In the image denoising task, the Self-ONN models outperformed the tested CNN and ONN models on both the train and test sets. The authors report the average performance levels on the train and test partitions as measured by SNR between the ground truth and output images as 5.67 dB and 5.61 dB for the CNN models, 5.68 dB and 5.46 dB for ONNs, and 7.05 dB and 6.15 dB for Self-ONNs, respectively. As such, the Self-ONNs outperform the other models by more than 0.5 dB on the test data. In the image synthesis task, however, the Self-ONNs outperform ONNs in only two of the ten experimental folds. In this experimental setup, the average SNR level was 5.02 dB for CNNs, 9.91 dB for ONNs and 8.73 dB for Self-ONNs. For the face segmentation task, the conventional ONN model chosen for comparison was the ONN-3 model which achieved the best performance for the task in the experiments of [1] as described previously in Section 3.4. The average F1 scores for the face segmentation task were 58.58 % for the CNN model, 79.86 % for ONN-3 and 96.6 % for the Self-ONN on the train set, and the average F1 scores on the test set were 56.74 % for the CNN, 59.61 % for ONN-3 and 62 % for the Self-ONN. For the image transformation task, the average SNR levels were 0.5 dB for CNNs, 9.5 dB for ONNs and 10 dB for Self-ONNs. The CNNx4 model and the best ONN model as described in [1] were used in the comparison of the performance on the image transformation task [2].

3.6.4. Advantages and Limitations

The experimental results presented in [2] suggest that Self-ONN models are able to produce better performance in various problems compared to CNN and ONN alternatives with a similar number of trainable parameters. The Self-ONN models seem to retain the desirable features of ONNs in that they significantly outperform the tested CNN alternatives, particularly as the learning task becomes more complex, while also addressing the inherent limitations of the ONN models as proposed in [1]. This section discusses the advantages and limitations associated with Self-ONN models in terms of performance and efficiency against CNN and ONN alternatives.

Although opting for a Self-ONN model resulted in improved results compared to both CNN and ONN models in most cases in the experiments of [2], the authors also note cases in which the conventional ONN model outperform the proposed Self-ONN approach. The authors mention that in the image synthesis task, Self-ONNs outperform ONNs in only two of the ten experimental folds, which also led the ONNs to achieve a higher average SNR. Another example is the image transformation task, as although the Self-ONN models outperform the ONNs in this task on average, they do so by only a narrow margin of 0.5 dB and the ONN models are able to outperform Self-ONNs in three of the ten transformations.

The authors list two reasons for the conventional ONNs outperforming the Self-ONNs in these cases. The first reason is that the nodal operators chosen for the conventional ONN models by GIS provide a near-optimal solution for the task at hand, in which case the Qth order approximation utilized by Self-ONNs cannot reach as good a level of performance. The second reason is the ONN models containing three times as many neurons as the Self-ONN models to compensate for the fact that the generative neurons utilized by Self-ONNs require Q times as many parameters as conventional ONN neurons, which can potentially lead to deterioration in the performance of a Self-ONN model compared to an ONN counterpart with a similar number of trainable parameters. The authors also note that when using the same *1 × 16 × 32 × 1* configuration as the ONN models, the Self-ONNs manage an average SNR level of 10.27 dB in the image synthesis task, surpassing the SNR achieved by ONNs by over 3.5 dB [2].

Although the conventional ONNs were able to outperform the Self-ONN approach in a few instances, the authors of [2] assert that opting for Self-ONNs can still generally be expected to lead to better performance compared to ONNs, as demonstrated by the results of the majority of their experiments. While nearly optimal nodal operators could be determined for the ONN models in some of the experimental scenarios, it is generally not reasonable to assume a near-optimal operator set to be available within the fixed operator set library used with conventional ONNs. In such a case, the nodal operators of generative neurons which are tailored specifically to the learning task at hand during backpropagation can lead to significantly improved learning performance. In addition, despite a near-optimal operator set being present in the operator set library, the best operators for a given task may not always be correctly assigned by GIS due to its suboptimal nature. It also worth noting that opting for a Self-ONN approach also allows for constructing deeper architectures which would be computationally infeasible for ONNs due to the computational complexity of determining the operators using GIS, which makes them applicable to a wider range of more complex learning problems on larger amounts of data compared to ONNs [2].

In addition to the performance levels they achieve, the usefulness of Self-ONNs in practical applications also depends on the computational resources required for their implementation. Although the results listed in [2] indicate that Self-ONN models can generally achieve superior performance compared to CNN and ONN models with a similar number of parameters, to determine whether they lead to a more efficient solution compared to their CNN or ONN counterparts, their computational complexity must be also be calculated and compared against these alternatives. In their publication, the authors of [2] evaluate the computational complexity of the FP and BP operations of their proposed Self-ONN models and draw comparisons against parameter-equivalent CNN and ONN models. They state that when the pool operator is fixed as the sum operation, i.e., $P_i^l = \Sigma$, as assumed in their study, the FP operation of a Self-ONN as based on Equation 2 can be expressed as

$$\begin{aligned} x_k^l &= b_k^l + \sum_{i=1}^{N_{l-1}} \text{oper2D}\left(w_{ki}^l, y_i^{l-1}, \text{'NoZeroPad'}\right) \\ x_k^l(m,n)\Big|_{(0,0)}^{(M-1,N-1)} &= b_k^l \\ &+ \sum_{i=1}^{N_{l-1}} \left(\sum_{r=0}^{K_x-1} \sum_{t=0}^{K_y-1} \mathbf{\Psi}\left(w_{ki}^l(\mathbf{r},\mathbf{t}), y_i^{l-1}(m+r, n+t)\right)\right) \end{aligned} \tag{7}$$

in which $\mathbf{\Psi}$ is the composite nodal operator and $w_{ki}^l(\mathbf{r},\mathbf{t})$ is the Q-dimensional array associated with kernel element $(\mathbf{r},\mathbf{t})$. By placing the qth order 2D kernel $w_{ki}^l\langle q\rangle (q = 1..Q)$ composed of the kernel elements $w_{ik}^{l+1}(\mathrm{r},\mathrm{t},\mathrm{q})$ in Equation 7, the equation can be simplified as follows:

$$x_k^l = b_k^l + \sum_{q=1}^{Q} \left\{ \sum_{i=1}^{N_{l-1}} \text{conv2D}\left(w_{ki}^l\langle q\rangle, \left(y_i^{l-1}\right)^q, \text{'NoZeroPad'}\right) \right\}. \tag{8}$$

The authors note that once the powers-outputs, $\left(y_i^{l-1}\right)^q$, have been computed for $q = 1..Q$ for each hidden neuron in the network, Equation 8 becomes only $(Q \times N_{l-1})$ independent 2D convolution operations, which are parallelizable and as such, take the same time for a single convolution when parallelized. The authors conclude that in a parallelized implementation, the FP operation of a Self-ONN has approximately the same computational complexity as the FP of a CNN with the same number of parameters. In addition, the authors calculate the total number of multiply-accumulate operations (MACs) for the CNN and Self-ONN models used in their experiments. The number of trainable parameters was 39749 thousand for the CNN and 32481 thousand for the Self-ONN, and the corresponding total MACs were 78246 and 79923 million, respectively. Based on the backpropagation formulae for Self-ONNs derived, the authors also conclude that the BP operations for Self-ONN also have approximately the same computational complexity as BP for a CNN with the same configuration in a parallelized implementation [2].

A challenge associated with opting for a Self-ONN model instead of a parameter-equivalent CNN is the increased susceptibility to overfit the training data due to the increase in learning capacity. As mentioned previously, this problem was already observed with ONNs in the experiments of [1] in the face segmentation task, in which the model that performed the best on the train data, ONN-1, did not achieve the best performance on the test set. The authors of [2] note that the Self-ONN model trained for the same face segmentation task also exhibits significant overfitting. This is apparent from the fact that it achieves a significantly higher F1 score compared to the CNN and ONN alternatives on the train data, while the margin by which it outperforms the two other models is rather small on the test data. In addition, the authors note that the Self-ONN achieves its lowest MSE loss of 0.324 on the test set at epoch 21 of training, after which the loss on the test set gradually increases, which is a clear indication of overfitting. However, the authors note that the overfitting problem can easily be addressed by employing early stopping based on the loss achieved on a separate validation set, which is established practice in machine learning research

[2]. Although the overfitting problems that may arise when using Self-ONN models can largely be addressed with the same approaches as those already in use with other supervised machine learning approaches, it is worth noting that Self-ONN models generally possess more learning capacity compared to not only a CNN architecture with the same number of neurons, but even a CNN architecture with the same number of trainable parameters, which should be taken into account when attempting to design a Self-ONN architecture of appropriate computational complexity to a given problem.

4. IMPLEMENTATION

This chapter describes a novel framework for remote PPG extraction from facial videos utilizing self-organized operational neural networks. In order to gain insight to how the depth of a Self-ONN model affects its performance in the proposed experimental settings, three Self-ONN architectures with varying number of neural layers are proposed and evaluated. To draw a comparison between the proposed Self-ONN models and conventional CNN-based approaches, three CNN models with the corresponding number of layers are also implemented and tested under the same experimental settings. A state of the art convolutional neural network proposed in [3] was chosen as the basis for the neural network architectures described in this chapter.

4.1. Materials

The experiments described in this thesis were conducted using three data sets of facial videos: OBF, COHFACE and the publicly available portion of the LGI-PPGI-Face-Video-Database. Samples from the OBF data set, which were recorded under strictly controlled conditions, were used for training and validating the neural network models. A subset of the OBF data set was also used as test data in evaluating the models. The samples from COHFACE and LGI-PPGI were used as test data in a cross-database evaluation setting to evaluate the performance of the models on unseen data that represents a variety of realistic real world scenarios and as such pose more challenges for an rPPG extraction framework.

4.1.1. Preprocessing

All videos used in the the experiments were preprocessed to only contain the face before being input to the neural network models. For detecting the face in the videos, a Single Shot Multibox Detection network [40] implemented in the OpenCV library and pretrained for face detection was used. This face detection method was chosen to maintain better comparability with the results of Álvarez Casado and Bordallo López, 2022, who employed the same face detection method in their proposed improved unsupervised rPPG pipelines [4].

As the face detection given by the model could be a rectangle with any given aspect ratio, the face was instead defined as a square with its center at the center of the original face detection and a side equal to the larger of the dimensions of the original face detection for the use with the neural network models. This step was performed because all inputs of the neural network models must have the same resolution, and this approach avoids the warping of the face which occurs when stretching a rectangular face detection to a square shape. After defining the square region containing the face, the cropped face detection was resized to a size of 128 by 128 pixels using nearest-neighbor interpolation.

For the videos in the OBF data set, face detection was only performed on the first frame as the coordinates of the face were assumed to remain approximately constant throughout the video, due to the subjects having been instructed to remain as still as possible during recording. When preprocessing the videos from COHFACE and LGI-PPGI, face detection was performed on every frame as the face of the subject moves over time throughout the recording session.

For the purpose of training the neural network models, the ground truth blood volume pulse (BVP) labels in the OBF data set were resampled to the corresponding video sampling rate of 60 Hz. The sampling rate, or FPS (frames per second), of any of the videos was not changed during preprocessing. After preprocessing the videos by extracting the face from each frame, the resulting processed videos and BVP labels were saved to disk in the h5 format for efficient loading during the training and evaluation of the network models.

4.2. Methods

This section describes an end-to-end approach for rPPG extraction from facial videos with self-organized operational neural network models utilizing three-dimensional operational layers. To gain more perspective into the benefits and potential limitations of utilizing Self-ONNs for the task of rPPG extraction, three Self-ONN architectures with varying number of layers are proposed. In addition, to comprehensively evaluate the performance of the Self-ONN models against comparable CNN architectures, three CNN models with the corresponding numbers of layers are also trained and evaluated in the same experimental settings. The models are based on the architecture of the "PhysNet-3DCNN-ED" model proposed in [3], employing an encoder-decoder structure. From this chapter onward, the term "operational layer" refers to Self-ONN layers with generative neurons unless otherwise specified.

All the neural networks described in this chapter were implemented using the Python language and the PyTorch library [41]. The implementation of the self-organized operational neural network models was based on the PyTorch-based FastONN library, which provides efficient graphics processing unit (GPU) enabled implementations for ONN and Self-ONN models in Python [42]. However, while the FastONN library contains implementations for 1-dimensional and 2-dimensional Self-ONN layers, as of writing it lacks an implementation for the 3-dimensional Self-ONN layers required for the purposes of this thesis. As such, the 3-dimensional Self-ONN layers used in implementing the 3D-SelfONN models proposed in this thesis were implemented by the author based on the 1D and 2D Self-ONN layer implementations of FastONN by replacing the 1D or 2D operations used in the library with their corresponding 3D equivalents found in PyTorch. Another point worth mentioning about the used Self-ONN layers is that they assume their input values to be within the range of [-1, 1], which means that the Self-ONN models must utilize sigmoid or hyperbolic tangent activation functions instead of more currently prevalent activation functions such as the rectified linear unit (ReLU) or exponential linear unit (ELU) functions.

The implemented Self-ONN and CNN models are similar to the ones used in [3] and [11], utilizing 3D convolutional or operational layers to operate on the three-dimensional volumes formed by the frames of the videos. The models mostly consist of 3x3x3 layers, each followed by batch normalization and an activation function except for the final layer. In addition, average pooling layers are employed to decrease the size of the inputs in the spatiotemporal domain. The architectures also contain a temporal encoder-decoder structure which was proposed in [3] and found to improve rPPG extraction performance by the authors.

To evaluate how the number of layers affects the performance of Self-ONN models in the proposed experimental settings, three Self-ONN models with varying depth were implemented: a baseline "deep" model, and two shallower ones. To evaluate how the same variations in depth affect CNNs, three CNN models with corresponding numbers of layers were also implemented for comparison. The nonlinearity parameter Q was set to 3 for every operational layer in the implemented Self-ONN models, which also results in the layers of the Self-ONNs having three times as many trainable parameters as a CNN layer with the same number of neurons would have [43]. To make the Self-ONN models as comparable as possible to their CNN counterparts, the number of neurons in each operational layer of the Self-ONN models was adjusted so that each Self-ONN model has approximately as many trainable parameters as its corresponding CNN alternative as suggested in [2] and [43]. Otherwise, the corresponding Self-ONN and CNN architectures were made as similar as possible with the only other differences being that the Self-ONN models utilize 3D operational layers as opposed to 3D convolutional layers.

The neural network architecture used in [3] and [11] was chosen as the baseline for the neural networks implemented for the purposes of this thesis. The two other architectures were formed from this "deep" architecture by removing a number of the neural layers with a kernel of size 3x3x3 as well as the batch normalization and activation functions following them, resulting in shallower architectures with reduced computational complexity. In the shallowest architecture, one of the deconvolution layers in the decoder section was also removed. An important point to note is that the shallower networks retain all the spatiotemporal dimension reducing operations of the deeper models. As such, the encoder-decoder structures in the shallower models attempt to reduce the dimensionality of their inputs by as much as the deeper models do but with fewer neural layers. For the rest of the thesis, the three proposed Self-ONN models will be referred to as SelfONN-deep, SelfONN-reduced and SelfONN-shallow, and the CNN models will be referred to as CNN-deep, CNN-reduced and CNN-shallow, respectively, in decreasing order by number of layers.

All activations in both the Self-ONN and CNN models utilize the hyperbolic tangent function as suggested in [2]. Although [3] and [11] utilized the ReLU activation function in their implementation, the activation function was kept the same between the two types of neural models in the experiments of this thesis to make the models as comparable as possible. When other aspects of the model architectures are fixed, the results can provide more insight into the specific effects of the tradeoff between the number of neural units and the nonlinearity enabled by the generative neurons of Self-ONNs on the performance of the model.

Figure 6 illustrates the architectures of all the implemented neural networks in detail, with every operation performed by the models represented by a box shape. In the boxes representing convolutional or operational layers in the diagram, the numbers on the top indicate the kernel size of the layer and number at the bottom is the number of output channels for the layer. The first of the numbers on the bottom represents the number of output channels in the case of a CNN model and the latter indicates the number of output channels in the corresponding Self-ONN model. Each convolutional or

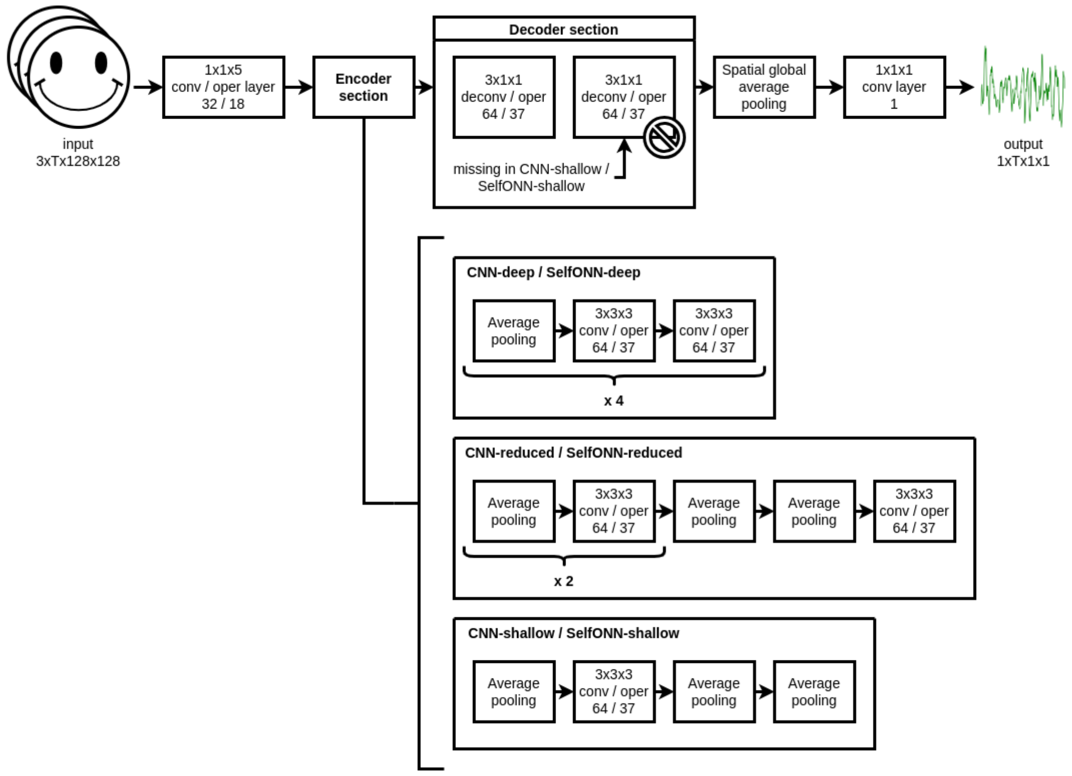

Figure 6. A diagram illustrating the implemented neural network architectures.

operational layer is also followed by a batch normalization operation and an activation function, which are omitted from the diagram for simplicity.

As demonstrated in Figure 6, all the implemented models are based on the same pipeline of functional parts. The variations in the depth of the proposed architectures are achieved only by adjusting the number of operational or convolutional layers included in the encoder section of the models, except in the case of SelfONN-shallow and CNN-shallow, in which one of the deconvolution layers in the decoder section is also omitted. All the networks accept an input of shape 3xTx128x128, where the temporal dimension T was fixed as 256 during training and the experiments. For every model, the first operation is performed on the input by a 1x1x5 convolutional or operational layer, followed by an encoder section which decreases the spatial and temporal dimension of the input. The reduction of dimensionality is achieved using four average pooling operations, of which the first and last only downsample the spatial dimensions and the second and third downsample both the spatial and temporal dimensions. The encoder is followed by a decoder section which increases the temporal dimension back to the original length. After the decoder section, a spatial global average pooling operation is performed to reduce the spatial dimension to 1x1. Finally, a 1x1x1 convolutional layer decreases the number of channels from 64 or 37 to 1, producing the desired 1-dimensional output rPPG signal with a length of T. The final output layer was kept as a convolutional layer even for the Self-ONN models, while all other neural layers in the Self-ONN models were operational layers.

5. EXPERIMENTAL SETUP

This chapter describes the experimental setup used for evaluating the performance of the proposed neural network approaches for rPPG extraction described in Chapter 4. To draw comparisons between the proposed Self-ONN-based rPPG method and existing alternatives proposed in literature, the three Self-ONN models were compared against the CNN models described in Chapter 4 as well as an unsupervised rPPG extraction pipeline described in [4]. The performance of each rPPG approach is evaluated based on HR estimation accuracy as opposed to estimating the more challenging HRV features.

The experiments were conducted using three suitable data sets: the OBF database, the COHFACE data set and the publicly available portion of the LGI-PPGI-Face-Video-Database. The videos from the OBF database were split into subject-independent train, validation and test sets. The train set was used for training the neural network models, the validation set was used for validation and model selection, and the test set was used to evaluate the performance of the models on unseen data. 20 of the 100 subjects were chosen randomly for the validation set and another 20 for the test set, and the videos recorded of the remaining 60 formed the train set. As two videos were recorded of each subject, the train set came to contain a total of 120 videos, and the validation and test sets contained 40 videos each.

In addition to evaluating the performance of the models on the test set consisting of OBF samples, the proposed neural networks are also evaluated on two outside data sets which were not used in training, COHFACE and LGI-PPGI, resulting in a kind of cross-dataset experimental setup. The videos contained in the COHFACE and LGI-PPGI data sets represent more challenging scenarios for an rPPG framework compared to the videos from the OBF database used in training the models. As such, the performance of the proposed models on these data sets provides insight to whether the models are capable of learning a means of representing remote PPG signals that is generally applicable to more realistic settings despite having been trained on samples recorded under constrained conditions.

Figure 7 illustrates the process of training, validating and evaluating the neural network models. First, the data in the OBF database is split into the dedicated train, validation and test sets. The videos and corresponding PPG labels in the train set are then used to iteratively update the weights of the neural network model based on the loss function. After 20 epochs of training, the weights of the epoch that achieves the lowest average loss on the validation set are selected for use during evaluation. Finally, the performance of the model is evaluated separately on the previously reserved test data from OBF, the COHFACE data set, and the LGI-PPGI-Face-Video-Database. The evaluation process consists of calculating the output rPPG signals from the input videos using the selected model and calculating HR estimates from them. Ground truth HRs are calculated from the PPG signals associated with the videos, and an evaluation metric is calculated between the HR estimate resulting from the rPPG signal and the ground truth HR value. The performance of the model can then be assessed based on the value of the calculated metric. Each of the phases presented in the figure is described in detail later in this chapter.

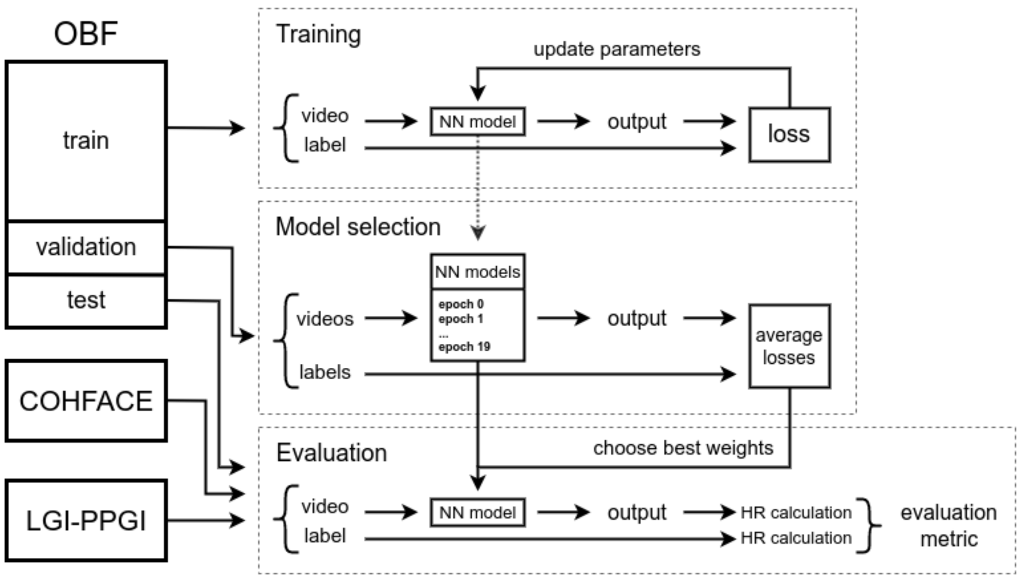

Figure 7. A figure illustrating the process of training, model selection and evaluation of the neural network models as well as the data used in each of these phases.

5.1. Training the Neural Network Models

5.1.1. Data

The train set described above, consisting of preprocessed samples from the OBF data set, was used for training the implemented neural networks. The resampled ground truth PPG signal was used as the label, which the models attempt to estimate based on their inputs. The same sets of data were used in training and validating all six of the implemented neural network models.

5.1.2. Loss Function

The function chosen to measure the loss between the estimates given by the models and the ground truth labels was chosen as negative maximum cross-correlation (NMCC). As the name implies, this loss function is calculated by first forming the cross-correlation between the two signals and taking the additive inverse of the maximum value of the resulting series. The cross-correlation between two 1-dimensional vectors x and y can be defined as follows:

$$z[k] = (x * y)(k - N + 1) = \sum_{l=0}^{N_x - 1} x_l y^*_{l-k+N-1} \quad \text{for } k = 0, 1, ..., N_x + N_y - 2 \tag{9}$$

where N_x and N_y denote the number of elements in x and y, respectively, $N = \max(N_x, N_y)$, and $y_m = 0$ when m is outside the range of y [44]. Based on the above, NMCC can be defined as

$$\text{NMCC} = -\max(z[k]) \tag{10}$$

This loss function was chosen over the negative Pearson's correlation (NPC) loss function suggested in [3] primarily due to its robustness against slight phase shifts between the ground truth PPG signals and the corresponding latent blood volume pulse information present in the facial videos. In initial tests, a set of neural network models was also trained using NPC loss. In these experiments, although the train loss of the models did converge, the NPC loss on the validation data did not decrease with continued training, which suggests that the neural networks were unable to learn a useful representation of the BVP signals. Figure 8 demonstrates this by showing the average train and validation NPC losses achieved by the CNN-deep model over 15 epochs of training during these initial experiments. The issue was resolved when opting to use the NMCC loss function instead. The problem caused by phase shifts could also have been overcome by synchronizing the facial videos with the corresponding ground truth PPG signals based on cross-correlation. However, the NMCC loss was adopted instead due to its versatility and to avoid this additional preprocessing step.

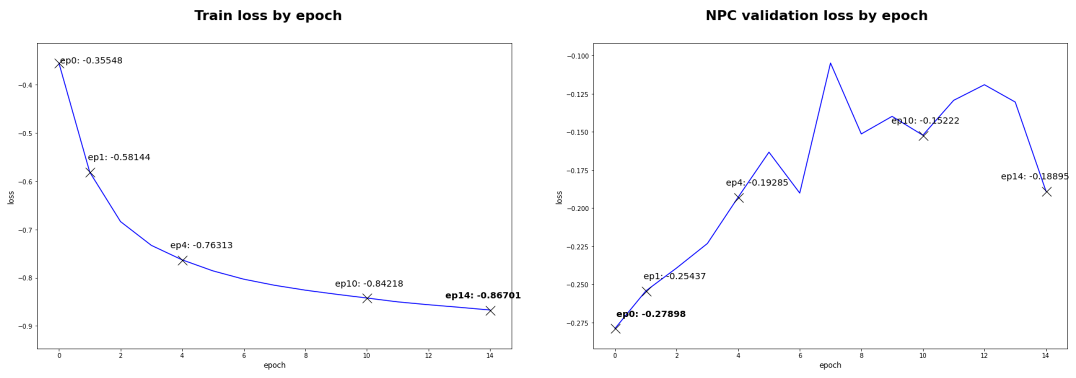

Figure 8. Average train and validation loss for the CNN-deep model during initial tests in which NPC was used as the loss function.

In order to maximize the information relevant to heart rate estimation in the estimated PPG signals, only the cross-correlation of the frequency components corresponding to realistic heart rates was considered when calculating the loss. The range of relevant frequencies was set as [40, 250] beats per minute (BPM) or [0.67, 4.17] Hz. This was achieved by calculating the cross-correlation efficiently in the frequency domain by taking the Fourier transfer of the estimated and ground truth PPG signals, multiplying the Fourier transfer of the predicted signal with the complex conjugate of the Fourier transfer of the ground truth signal and setting the frequency components outside the desired range to zero in the resulting frequency domain cross-correlation. Finally, the desired time domain cross-correlation was achieved by calculating the inverse Fourier transfer of the result. The final loss function value is the additive inverse of the maximum value of the resulting cross-correlation. The resulting value was also adjusted for the ratio of energy between the relevant and non-relevant frequency regions.

5.1.3. Train Setup

All the models were trained on a consumer graphics processing unit (GPU) with 8 GB of dedicated video random access memory (VRAM), the Nvidia GeForce RTX 3070, by utilizing the CUDA computing capabilities of the PyTorch library. Each of the implemented neural network models was trained on the same data for a total of 20 epochs. An AdamW optimizer, a variant of the Adam optimizer with weight decay, was used to speed up the training process [45]. The learning rate of the optimizer was set to 10^{-4}. A batch size of 1 was utilized when training all the models. Each training sample and the corresponding ground truth label were also normalized before before each iteration of backpropagation.

In order to train the model with backpropagation using a GPU, the batch of samples used to perform one iteration of backpropagation must fit in the dedicated VRAM of the GPU along with the weights of the trained model. Due to this memory constraint, the model could not be trained using the full 5 minute facial videos from the OBF data set. Instead, the models were trained using segments with a length of 256 frames which were extracted from the videos in the train data at random locations. For each epoch of training, 70 iterations of backpropagation were performed using such random samples from each video in the train set. The number of iterations was calculated as the length of the video divided by the length of the random segments, rounded to the nearest integer. This was to ensure that during each epoch, the models are trained using approximately 5 minutes of video material from each video in the train data, as they would be if training on the entire videos at once.

5.1.4. Training Results

Figure 9 presents the average train and validation NMCC loss achieved by the models after each epoch of training. It is important to note that the train losses presented in Figure 9 are the averages of the train losses calculated during training process itself. In other words, the train loss figure presents the average of the losses calculated on the randomly selected segments after each iteration of backpropagation during a given epoch. As such, the plot does not illustrate the performance of the models on the entire train set after each epoch, and the train data used for calculating the results differs slightly between models due to the random selection of video segments for training. In the case of the plot on the right, the NMCC loss was calculated on the entire videos from the validation set after each full epoch of training, and the averages of these values were plotted over the epoch numbers. The numbering of epochs in the plots starts from 0, which represents the state of the models after one full epoch of training, and ends at 19, which in turn is the value after the full 20 epochs.

As seen in the plots on the left in Figure 9, all the models were able to converge during training. The NMCC losses on the train data for the deepest models, SelfONN-deep and CNN-deep, were -0.962 and -0.957 during the final epoch, respectively.

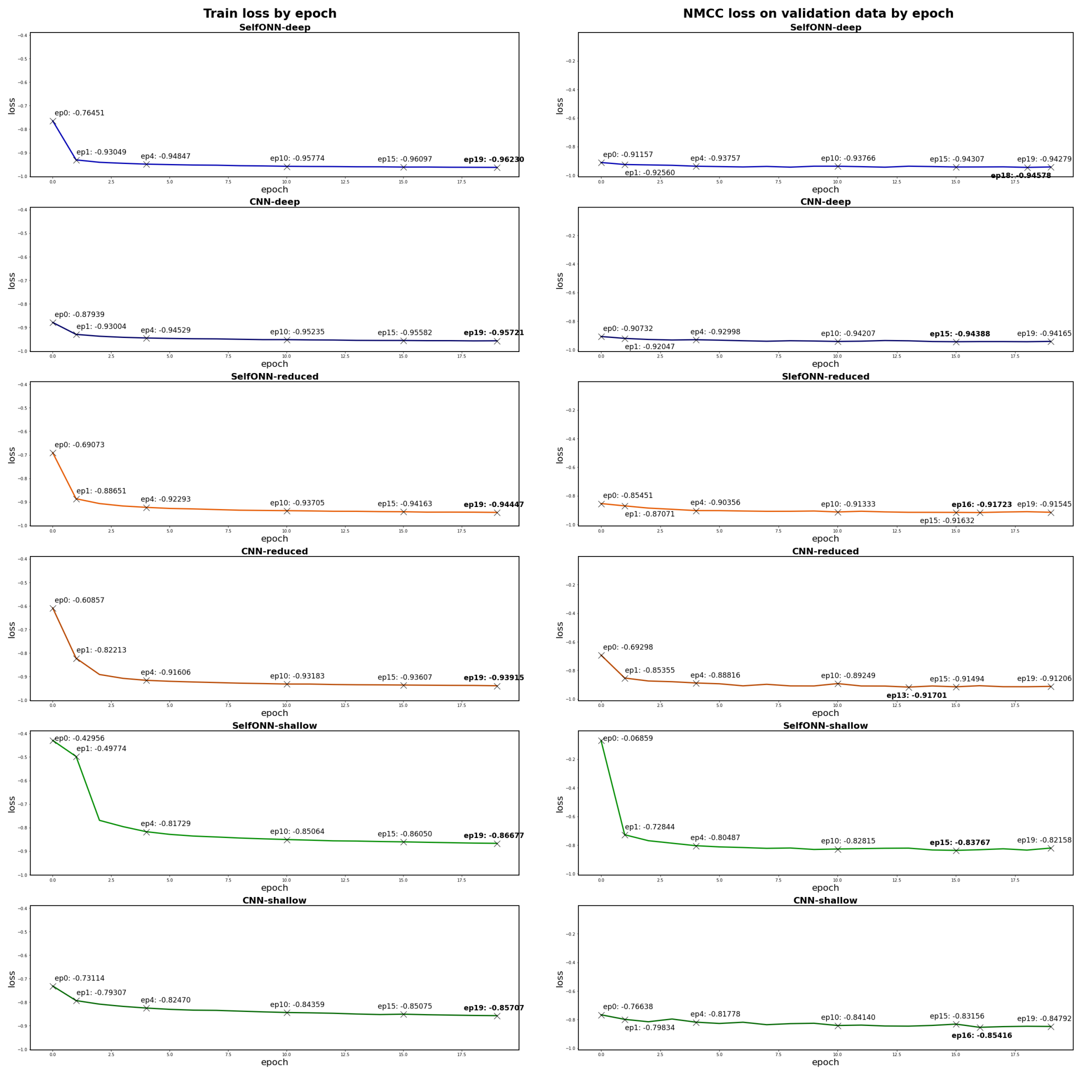

Figure 9. Average train and validation loss after each epoch of training.

SelfONN-reduced and CNN-reduced achieved train losses of -0.944 and -0.939, respectively. The train losses achieved by SelfONN-shallow and CNN-shallow were significantly lower than those of the other models at -0.867 and -0.857, respectively, after the full 20 epochs of training. As such, every Self-ONN model was able to achieve a lower train loss compared to its corresponding CNN counterpart, which indicates a better fit of the train data based on the target function.

Using a separate set of data for validating the neural network models has two main purposes: ensuring that the models learn to generalize to unseen data during training as well as choosing the best weights achieved during training based on validation loss. The validation loss plots on the right side of Figure 9 indicate a clear downward trend in the NMCC loss on the validation data for each model when training for longer, which implies that the models do not overfit the train data and are able to generalize to the unseen data from the validation set. In each validation loss plot, the epoch that achieves the lowest NMCC loss on the validation set is highlighted in bold along with the corresponding validation loss value. For each model, the weights of the epoch that achieved the lowest validation loss were used when conducting further experiments.

The lowest validation loss of all the models is achieved by the SelfONN-deep model during epoch 18 at -0.946. The second best validation loss is achieved by CNN-deep on epoch 15 at -0.923. Both SelfONN-reduced and CNN-reduced achieved an NMCC loss of -0.917 on the validation set at best during epochs 16 and 13, respectively. The best validation losses achieved by the shallow models were -0.838 on epoch 15 for SelfONN-shallow and -0.854 on epoch 16 for CNN-shallow. As such, while the train loss values indicated that the Self-ONN models always achieved a better fit of the train data compared to their CNN counterparts, only SelfONN-deep achieves a better validation loss compared to its CNN counterpart, with the validation loss achieved by SelfONN-reduced and CNN-reduced being the same, and CNN-shallow outperforming SelfONN-shallow on the validation set.

The minimum validation losses achieved by the models and the corresponding epochs are also listed in Table 1 along with the number of trainable parameters in each model and the train losses achieved after the full 20 epochs of training. Together, these values provide insight into the capacity of each model to fit, and potentially overfit, the train data with which it is presented.

Table 1. The train and validation losses achieved by the neural network models along with the number of trainable parameters and epoch achieving the lowest validation loss.

Model	Trainable parameters	Train loss	Validation loss	Val loss epoch
SelfONN-deep	860063	-0.962	-0.946	18
CNN-deep	858497	-0.957	-0.923	15
SelfONN-reduced	305063	-0.944	-0.917	16
CNN-reduced	304577	-0.939	-0.917	13
SelfONN-shallow	70631	-0.867	-0.838	15
CNN-shallow	70529	-0.857	-0.854	16

5.2. Extracting Remote PPG Signals from the Test Data

The first step in evaluating the performance of the frameworks was extracting the remote PPG signals from all the facial videos in the COHFACE and LGI-PPGI data sets and the OBF test data using the neural network models described in Chapter 4 and the Face2PPG unsupervised rPPG extraction pipeline. For all the neural network models, the weights during the epoch that achieved the lowest NMCC loss on the validation set were used when conducting further experiments. The samples in the test data were preprocessed as described in Section 4.1.1.

The inference of rPPG signals from the videos in the test set using the neural networks was performed on a GPU, much like during training and validation. However, as explained in Section 5.1.3, the rPPG signals cannot be inferred from the entire facial videos at once when computing the results on the GPU used in these experiments due to limitations in VRAM capacity, which is also true in the case of the test data. To facilitate the extraction of rPPG signals from the whole facial videos on a GPU, the preprocessed videos from the test data were input to the model in segments of 256 frames at a time, and the results were concatenated to form the final inferred rPPG signal of the entire video.

However, as the number of frames in the videos is not divisible by the chosen segment length of 256 in general, it was also necessary to take into account how to handle the remainder of frames that could not form another segment with the desired length. In the implementation used in these experiments, the number of segments in a video was in most cases set to the floor division of the video length in frames by the segment length of 256. The remainder of frames were considered as part of the last segment, resulting in the final segment of frames being longer than the others. However, in this approach, the length of the final segment may theoretically be as long as $2T - 1 = 2 \times 256 - 1 = 511$ where T is the length of the video segment in frames. This means that the final sample could exhaust the limited VRAM of the GPU, which was also observed during preliminary experiments. To overcome this problem, the number of segments that are extracted from the video was adjusted so that the length of the final segment would not exceed T+T/2 = 384. If the length of the final segment were to exceed this number of frames with the approach above, the number of segments extracted from the video was instead increased by one, so that the remainder forms its own segment instead of being considered part of the previous segment. The reason why the remainder is not always treated as a segment of its own is because the input of the neural network must be sufficiently long in the time domain due to the temporal dimension reducing operations performed by the models.

To draw a comparison of the proposed supervised rPPG extraction frameworks to an unsupervised method, rPPG signals were also extracted from the test data using the Face2PPG rPPG extraction framework proposed in [4]. Of the multiple unsupervised remote PPG extraction pipelines proposed by the authors, the "improved" pipeline utilizing the POS rPPG extraction method [25] was chosen as an example of an unsupervised method in this thesis. All experimental results reportedly achieved using Face2PPG in this thesis utilize this particular pipeline. The Face2PPG pipeline implementation used in this thesis was provided by the authors of the publication in the form of a Python repository.

After the initial extraction of rPPG signals from the videos in the test data, the resulting signals were postprocessed to exclude frequency components irrelevant to heart activity in order to make it easier to extract the desired heart rate features from them. In the case of the neural network models, the postprocessing of the output signals only consisted of filtering using a bandpass FIR (finite impulse response) filter with cutoff points at 0.75 and 4 Hz, corresponding to 45 and 240 BPM. The filter was designed using a Hamming window with a variable length that was determined as $2F_s - 1$ for each signal, where F_s denotes the sampling frequency of the signal in question. The used Face2PPG implementation already incorporated similar postprocessing steps as part of the pipeline, which was applied to the videos as-is with no additional postprocessing steps taken. The PPG signals used for calculating the ground truth heart rates were also filtered using a bandpass filter similarly to the signals extracted using the neural network models in order to counter the effects of possible noise present in the recorded signals.

5.3. Extracting Heart Rates

After postprocessing, the resulting signals were divided into 10 second windows for which the average heart rate was calculated. The windows were extracted at intervals of 1 second, resulting in 9 seconds of overlap between consecutive windows. The same process of heart rate extraction was applied to all the signals in the test set, extracted rPPG signals and ground truth BVP signals alike, resulting in two vectors of heart rates for each video in the test set: a vector of estimated HRs calculated from the rPPG signal and a vector of ground truth HRs calculated from the BVP label.

To calculate the heart rate for the windows extracted from the signals, two different heart rate estimation methods were employed and evaluated: one based on peak detection and one based on spectral analysis. The results for both methods are presented because depending on the method used to achieve the rPPG signal, one method may produce significantly better results than the other approach, which, on the other hand, may not be suitable for the given signal at all.

The peak detection based heart rate calculation method utilized the function ppg_process from the ppg module of the neurokit2 physiological signal processing library for Python [46]. This function is specifically designed to detect peaks in PPG signals, applying a number of preprocessing steps to enhance the input signal before peak detection. The function also outputs an envelope of heart rate values for the input signal, but this output was not utilized in the experiments of this thesis. Instead, only the peak detections given by the function were used, and the heart rate was defined as the reciprocal of the average interval between consecutive peak detections for each window. This method was also used for calculating all ground truth heart rate values because it was found to perform well in detecting the systolic peaks in the PPG signals recorded from the finger, as this is the type of signal it was originally designed for.

The peak detection function from the neurokit2 library was chosen because it was found to be fairly robust to the particularly challenging nature of remotely extracted PPG signals, producing sensible results without additional modifications to the implementation. Other similar solutions were also tested, such as those found in the heartpy [47] and systole [48] libraries, but the results achieved using the implementations from these libraries were found to be unsatisfactory when calculating heart rates for rPPG signals as they produced NaN (not a number) values as opposed to valid heart rates for certain signal windows, for example.

The other method that was used for calculating heart rates for each window from all rPPG signals was the implementation of Welch's method provided by the authors of [4] which they used to achieve their results. This method was evaluated to maintain comparability with the results of their publication, but also because peak detection performs relatively poorly on the signals produced by the unsupervised Face2PPG pipeline, which means that exclusively using peak detection for heart rate calculation would underestimate the potential performance of this rPPG extraction method. However, it is worth noting that unlike methods based on peak detection, approaches based on spectral analysis such as Welch's method cannot be used for calculating the more challenging HRV features which require information of peak locations.

Due to certain characteristics of the ground truth PPG signals provided with the OBF data set, which was also used in training the neural network models, the implementation of the Welch's method was modified slightly when calculating HRs from the rPPG signals extracted using the neural network models. The implementation was modified so that instead of simply returning the maximum value of the power spectral density (PSD) curve as given by the Welch algorithm, peak detection was performed on the PSD, and in the presence of more than one peak, the two peaks with the most power were compared. If the peak at the higher frequency was found to be the second harmonic of the first, i.e., approximately double the frequency, within a margin of 10 BPM, then the lower of the peak frequencies was chosen for the HR output. In the presence of only one peak, the frequency associated with the peak was chosen as is, and if for some reason no peak could be detected in the PSD by the peak detection function used, the frequency associated with the maximum power was chosen as the HR. This additional step was performed in an attempt to prevent the function from outputting the second harmonic of the heart rate instead of the desired HR value and was found to improve the test performance of the neural network models. However, the implemented workaround was not perfect and some problems still occurred with the used Welch's method implementation particularly when calculating HRs from the rPPG signals extracted using the neural networks. When calculating HRs from signals extracted using the Face2PPG pipeline, the original unmodified implementation of the algorithm as provided by the authors was used, as the described modification was not found to improve the performance of the Face2PPG pipeline. The reasoning behind the modification and the problems associated with the Welch's method implementation are discussed in more detail in Section 7.2 of the next chapter.

5.4. Evaluation Criteria

The performance of each rPPG extraction method was evaluated by comparing the average window HR values calculated from the rPPG signals they produced against the HRs calculated from the corresponding ground truth PPG signals. Three metrics which are commonly applied in similar evaluation settings were used in evaluating the tested models: mean average error (MAE), root mean square error (RMSE), which is defined as the square root of the mean square error (MSE), and Pearson's correlation coefficient (PCC). The equations for MAE, MSE and PCC are presented below.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{11}$$

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{12}$$

$$\text{PCC} = \frac{\sum (y - m_y)(\hat{y} - m_{\hat{y}})}{\sqrt{\sum (y - m_y)^2 \sum (\hat{y} - m_{\hat{y}})^2}} \tag{13}$$

In Equations 11 and 12, N is the number of elements in the HR vectors, i.e., the number of windows for which HR was calculated, $\hat{y}_i$ denotes the predicted HR value for a given window, calculated from the rPPG signal, and y_i denotes the corresponding ground truth HR value. In Equation 13, $\hat{y}$ denotes the vector of predicted HR values from a video, y is the corresponding vector of ground truth HRs, $m_{\hat{y}}$ is the mean of the vector $\hat{y}$ and m_y is the mean of the vector y.

For each video in the test set, MAE, MSE and PCC were calculated between the predicted HR vector resulting from the rPPG signal and the ground truth HR vector calculated from the BVP label. The mean values of these metrics were calculated for each data set, and the resulting average performance metric values are reported in Chapter 6 with the exception of MSE whose square root, RMSE, is reported instead. For MAE, the standard deviation (SD) is reported in addition to the mean.

6. RESULTS AND ANALYSIS

This chapter presents the performance of all the tested rPPG extraction methods on the three sets of test data based on the evaluation metrics described in Section 5.4. In addition to listing the performance of each method, the underlying reasons and significance of the experimental results are discussed, providing an analysis of which methods perform best on the different types of test data and which factors contribute to the performance of the methods. The results on each set of test data are analysed individually, discussing which methods achieve the best results on the particular type of data represented by the set in question and why each method performs well — or poorly — on the particular type of data. Close attention is given to the newly proposed Self-ONN models, which are directly compared against the other tested methods with particular attention given to the corresponding CNN models as the most directly comparable alternatives.

The performance metrics for each method on the OBF test data are listed in Table 2, the results on the COHFACE data set in Table 4 and the results on LGI-PPGI in Table 5. In each table, the best result based on each metric is highlighted in bold. For MAE and RMSE, a lower value is better as they measure error, whereas for PCC a higher value is better as it measures correlation between the HR predictions and ground truth HRs. Since two HR calculation methods were applied to the rPPG signals extracted using each of the 7 rPPG extraction methods, the results for a total of 14 unique remote HR estimation pipelines were collected.

In addition to the quantitative metrics described in Section 5.4, the performance of the rPPG extraction pipelines is demonstrated with plots displaying example results of rPPG extraction and HR calculation on select videos from the test data. Figures 10 and 11 display results on the OBF test data, Figures 12 and 13 contain examples from the COHFACE data set, and Figures 14 and 15 display examples on videos from the LGI-PPGI data set.

The left column of plots in the figures show the rPPG signals, as well as the corresponding ground truth PPG signal on the bottom, with peak detections computed using neurokit marked with red dots. The plots on the right display the heart rates for each time window calculated from the corresponding rPPG signal on the left, forming an envelope of average window heart rates. The heart rates calculated using neurokit peak detections are plotted in black, and the heart rate envelope achieved with Welch's method is plotted in red. The bottom plot displays the ground truth HR envelope, calculated from the PPG signal to its left using the peak detections given by neurokit. All the signals are plotted after having undergone every signal processing step described in this chapter, including the signal cleaning performed by the ppg_process function of neurokit. It is worth noting that the steps performed by neurokit were not applied when using Welch's method for HR calculation, but the signals are plotted after having also undergone this final processing step in order to display the peak detections correctly.

In the titles of the plots on the left, the average HR for the whole video is shown, calculated as the reciprocal of the average IBI based on the peak detections given by neurokit as shown in the plots. For the plots on the right, the average HRs shown in the legend are the mean values of the corresponding HR envelopes, i.e., the mean of the average window HRs.

A general point worth noting when assessing the experimental results achieved using the neural network models is that although the signals given by the networks resemble the real PPG signals recorded from the finger on which they were trained, the rPPG signals that they provide do not actually represent any real physiological signal that could be recorded from the subject. The neural networks output a signal that is similar to a real PPG signal because it is what they were trained to do, but in reality the changes in blood volume that appear on the face differ significantly from the corresponding changes measured from the finger. This means that instead of extracting the inherent blood volume pulse signal that could be measured from colour changes on the face, the neural networks attempt to estimate what the corresponding blood volume changes in the finger could look like based on this information. This arises from the inherent problem in the training setup that the "ground truth" PPG signal recorded from the finger does not actually represent the true underlying blood volume changes occurring on the face and as such could be considered not to represent a true ground truth for the problem at hand. Despite this, the PPG signals recorded from the finger are the closest thing to an accurate ground truth for the problem that can easily be measured, which is why it has been widely adopted in the training of supervised models for rPPG extraction.

6.1. Performance on OBF

As indicated in Table 2, the best results on the OBF test data based on every used metric was achieved by the deepest CNN model when using neurokit peak detection for calculating HRs, with a MAE of 0.2784 ± 0.3097, RMSE of 0.9495 and PCC of 0.9629, clearly outperforming every other tested method. In fact, not only are these the best results achieved on the OBF test data, but they are also the best results achieved by any of the tested methods on any data set, producing the only RMSE value below 1 BPM recorded during the experiments. The best performing Self-ONN model based on MAE and PCC was SelfONN-deep with a MAE of 1.229 ± 3.341 and PCC of 0.8797, and the best RMSE of the Self-ONN models was achieved by SelfONN-reduced at 3.786. The shallower neural networks also produce good results on this data set, although the performance of the neural network generally degrades as the complexity of the model is reduced. In addition, the CNN models generally perform better than the Self-ONN models of the same depth. For almost every neural network model, calculating HRs using neurokit peak detection produced better results compared to Welch's method based on every metric on this test data, with the only exceptions being the MAE and PCC of the SelfONN-shallow model and PCC of the CNN-shallow model. For the unsupervised Face2PPG pipeline, Welch's method produced significantly better results compared to neurokit. When comparing the best results achieved by each rPPG extraction method, Face2PPG with Welch's method outperforms the two shallowest neural network models paired with neurokit as the HR calculation method based on MAE and RMSE, with the rest of the neural network models outperforming the best result achieved by unsupervised Face2PPG pipeline when they are paired with neurokit as the HR calculation method.

As all the data in the OBF database is quite homogeneous, the OBF test data is similar to the train data used in training the supervised neural network models, which

Table 2. Performance of each method in estimating heart rates on the OBF test set with the best result for each metric highlighted in bold.

rPPG extraction method	Evaluation metric			HR calculation method
	MAE ± SD	RMSE	PCC	
SelfONN-deep	6.716 ± 16.086	19.84	0.6696	Welch
CNN-deep	0.8288 ± 1.0946	4.803	0.9065	Welch
SelfONN-reduced	5.686 ± 14.492	18.42	0.7453	Welch
CNN-reduced	1.043 ± 2.163	5.083	0.8812	Welch
SelfONN-shallow	2.382 ± 6.619	9.838	0.8064	Welch
CNN-shallow	4.114 ± 10.477	14.74	0.7507	Welch
Face2PPG	1.928 ± 2.278	6.548	0.6794	Welch
SelfONN-deep	1.229 ± 3.341	4.396	0.8797	neurokit
CNN-deep	**0.2784 ± 0.3097**	**0.9495**	**0.9629**	neurokit
SelfONN-reduced	1.245 ± 2.614	3.786	0.8665	neurokit
CNN-reduced	0.6765 ± 1.5267	2.308	0.9096	neurokit
SelfONN-shallow	2.726 ± 5.208	6.817	0.7124	neurokit
CNN-shallow	2.512 ± 5.826	7.155	0.7744	neurokit
Face2PPG	4.829 ± 4.517	8.324	0.4058	neurokit

also consisted of samples from OBF. As such, the neural networks have an inherent advantage over the unsupervised Face2PPG pipeline on this set of test data. This can also be observed in the results presented in Table 2 as the neural networks — apart from the two shallowest models — achieve better performance compared to Face2PPG based on all used metrics. As the shallow models have less capacity to fit the train data due to their significantly lower computational complexity, they also see less benefit from being trained on similar data compared to the other neural networks when evaluated on the OBF test set. It is also worth noting that because the samples in OBF were recorded under quite ideal conditions for an rPPG extraction framework, every tested method, including Face2PPG, achieves quite respectable performance on this test data, achieving lower MAE and RMSE and a higher PCC than on any other test set.

The peak detection method of neurokit outperforming Welch's method in HR calculation for the rPPG signals calculated from the OBF test data using the neural networks is also partially explained by the similarity of this test data to the train data. During the training process, the neural networks have been presented with similar video samples, with real PPG signals recorded from the finger as the ground truth. Because of this, the networks learn to produce outputs that resemble real PPG signals as long as the input is similar enough to the data on which they were trained. As such, the rPPG signals extracted from the videos in the OBF test data using the networks resemble real PPG signals with pronounced peaks and overall similar shape, which makes the neurokit PPG peak detection method effective in correctly detecting the systolic peaks in these signals as the function was originally designed for peak detection from PPG signals recorded from the finger. However, there are also certain other reasons for neurokit peak detection producing better results compared to Welch's method which will be discussed later in this section as well as Section 7.2.

The good performance of the neural network models and the neurokit peak detection algorithm is also demonstrated in Figure 10 which shows an example result from a video in the OBF test data. As seen in the plots on the left, the rPPG signals produced by the neural networks, especially the two deepest ones, closely resemble a real PPG signal recorded from the finger. Although these rPPG signals do not follow the ground truth exactly, the peak locations are almost exactly correct. Much like the HR envelopes calculated using neurokit peak detection, the envelopes calculated using Welch's method also follow the ground truth HR envelope closely in the example presented in the figure.

The example results achieved using the neural networks in Figure 10 are in rather stark contrast to the rPPG signal extracted using the unsupervised Face2PPG pipeline shown in the second to last plot on the left. This signal is shaped much more irregularly with less pronounced peaks, which can in turn result in false positive peak detections or missed peak detections. The Face2PPG example result shows an example of a false peak detection and how it affects the HR prediction result achieved using neurokit. As seen in the black HR envelope in the plot on the right, a single false peak detection results in the HR predictions for multiple windows being calculated as too high since the false detection appears in several consecutive 10 second windows. On the other hand, the heart rates calculated from the Face2PPG signal using Welch's method give quite accurate results, which suggests that despite the irregular shape of the signal, the signal nonetheless contains the relevant frequency information related to heart activity.

The peak detections in Figure 10 and the metrics achieved using neurokit peak detection in Table 2 suggest that rPPG signals extracted using the neural networks could also be applicable for calculating the more challenging HRV features in addition to simple heart rates at least in this particular example, which requires the accurate calculation of the intervals between consecutive heartbeats. Since the inter-beat intervals were used for calculating the heart rates from the neurokit peak detections, the excellent results achieved with the neural networks suggest that the intervals between peaks are calculated accurately. In addition, the peak detections seen in Figure 10 also seem to follow the corresponding peak locations in the ground truth signal closely. However, it is worth noting that although the neural network results show promise for HRV calculation, no HRV features were actually calculated during the experiments.

On the other hand, based on the results seen in Figure 10 and Table 2, the signals given by Face2PPG would most likely not be reliable for calculating HRV features as these signals are prone to missed or false peak detections due to their irregular shape. In addition, although the average inter-beat intervals can often give quite accurate HR predictions for rPPG signals extracted using Face2PPG, using the intervals between consecutive peak detections in the signal for calculating HRV features seems questionable based on Figure 10.

Due to the similarity between the OBF test data and the train data used for training the neural network models, the neural network models which achieve the best fit of the train data, i.e., the lowest train loss, could be expected to perform the best on this particular set of test data. By observing the results achieved by the CNN models on the OBF data in Table 2 and comparing them against the loss that each CNN model achieved during training as seen in Table 1, it can be seen that an increase in the depth of the architecture resulted in a better fit of the train data, and that the lower the train loss achieved by the CNN model was, the better the performance on the OBF test data.

However, when observing the train losses achieved by the Self-ONN models in Table 1 and comparing them against the results achieved on the OBF test data in Table 2, it can be seen that in the case of the Self-ONN models, a lower loss on the train data does not necessarily translate to better performance on the OBF test set. For example, SelfONN-deep fails to outperform both its equivalent CNN model of CNN-deep as well as the shallower CNN-reduced and SelfONN-reduced models based on RMSE of the HRs calculated using neurokit despite achieving the best fit on the train data and lowest validation loss of any neural network model as seen in Table 1. In addition, SelfONN-reduced was in turn outperformed by CNN-reduced despite also achieving a better fit of the training target than its CNN counterpart.

As the Self-ONN models have higher capacity compared to their CNN counterparts in theory, which also translated to a lower train loss in practice, overfitting to the train and validation data sets could be a plausible explanation for why they fail to outperform their respective equivalent CNN models on unseen data. For the SelfONN-deep model in particular, the very late epoch chosen based on the validation loss, specifically the second to last epoch at 18, suggests that there is a possibility that the SelfONN-deep model might have even achieved lower validation loss with continued training. However, as the Self-ONN models achieve better performance on the OBF test set compared to their CNN counterparts based on the NMCC loss function used as the training target as seen in Table 3, overfitting the train data based on the used loss function is most likely not the cause for the Self-ONN models falling short in performance.

As the SelfONN-models consistently achieve worse, or at best similar, performance compared to their parameter-equivalent CNN counterparts when presented with test data that is similar to the data used in training, based on the results seen in Table 2, it cannot be concluded that opting for Self-ONN models instead of CNNs improves rPPG extraction performance in the presented experimental setup based on the used metrics. However, as the Self-ONN models consistently achieve better *learning* performance, i.e., a better fit of the train data based on the target function, their poor results on the OBF test set suggest that there are inherent problems in the training setup as opposed to the Self-ONN models being inherently less fit to the problem at hand. In fact, during conducting the experiments presented in this thesis, several such issues related to the training setup — both the training target and the nature of the train data — were detected. These problems can significantly hinder the rPPG extraction performance achieved by the neural networks — particularly when evaluated based on HR estimation accuracy. In the rest of this section, these issues will be addressed in more detail and some insight will be provided into why the problems related to the training setup affect the Self-ONN models more than their CNN counterparts.

A problem observed with the data in the OBF data set is that the ground truth PPG signal recordings, which were also used as the training target for the neural network models, contain very prominent second harmonic components, which can cause problems when calculating heart rates. As a frequency-based method, Welch's method is particularly susceptible to errors resulting from the higher harmonic components in the signal. These problems were already briefly touched upon in Section 5.3 when discussing the workarounds added to the Welch's method implementation to address the problems and will be covered in more detail in Section 7.2. This is also an underlying reason for why Welch's method produced a higher error compared to neurokit for all the tested neural network models as seen in Table 2. In the time domain, the second harmonic components correspond to prominent diastolic peaks, which can in turn be erroneously detected by the used neurokit peak detection function as systolic peaks.

As the neural networks were trained to reconstruct the entire signal waveform with these PPG signals with strong second harmonic components as ground truth, the rPPG signals produced by these models inherit the problems associated with the ground truth data, i.e., the strong second harmonic components and diastolic peaks, which in turn can result in increased error rates. During the experiments, it was observed that the Self-ONN models are more capable of learning these problematic features compared to their CNN counterparts. This can be seen particularly clearly from the example rPPG signals and their corresponding HR estimates presented in Figure 11 and can also be observed as particularly high error values associated with the HR estimates given by Welch's method for the signals extracted using the SelfONN-deep and SelfONN-reduced models as seen in Table 2. This explains why the Self-ONN models produce higher errors compared to the CNN models despite being able to achieve lower train loss levels, as learning the higher harmonic features is appropriate considering the training target of reproducing the ground truth PPG waveform as closely as possible but unnecessary for the end goal of calculating heart rates from the extracted rPPG signals.

When comparing the example results achieved with SelfONN-deep and CNN-deep in Figure 11, it can be seen that a reproduced diastolic peak in the rPPG signal produced by SelfONN-deep is erroneously detected by neurokit, which results in too high average HR estimates for a number of consecutive windows. Similar problems can be observed with the signals extracted using SelfONN-deep and SelfONN-shallow, but also with CNN-reduced and CNN-shallow. In addition, the Welch's method implementation completely fails to produce reasonable HR estimates for the signals produced by SelfONN-deep and SelfONN-reduced and also produces significant errors when calculating HRs from the signals extracted with SelfONN-shallow and CNN-shallow. In contrast, when presented with the rPPG signal produced by the unsupervised Face2PPG pipeline, which has not been trained to reproduce PPG signals from the OBF database, Welch's method performs reasonably well in estimating the average heart rates correctly. However, neurokit peak detection produces slightly less desirable results with Face2PPG, as the produced signal is shaped less regularly for the same reason, resulting in erroneous peak detections.

Table 3 lists the NMCC losses associated with the signals extracted using the neural network models from the OBF test data. These numbers further suggest problems with the train target, as SelfONN-deep and SelfONN-reduced outperform their respective CNN counterparts based on the target NMCC loss function even on this test data in addition to the train and validation sets, but fail to achieve lower HR error rates on the OBF test set despite this as established earlier.

As many of the erroneous HR calculations associated with the signals extracted using the Self-ONN models in particular result from the inherent characteristics of the ground truth PPG signals in the OBF database, this raises the question of whether the ground truth HR values calculated from said PPG signals with neurokit and used

Table 3. The average NMCC losses achieved by the neural network models on the OBF test data.

Model	Loss
SelfONN-deep	-0.958
CNN-deep	-0.951
SelfONN-reduced	-0.936
CNN-reduced	-0.923
SelfONN-shallow	-0.862
CNN-shallow	-0.877

in evaluating the models could also be erroneous. During the experiments, it was observed that in the case of certain PPG signals from OBF with particularly strong diastolic peaks, the neurokit peak detection method could indeed mistakenly detect diastolic peaks as systolic peaks, resulting in erroneous ground truth HR calculations. An example of such a case is presented in Figure 16 which is discussed in detail later in Section 7.2 along with other problems observed with the heart rate calculations. However, such cases were rare and occurred only in few signals, and even in these signals only few of such erroneous detections were made. As such, the ground truth HR values for OBF as calculated using neurokit peak detection were mostly correct, but these small errors should be born in mind when analyzing the results on the OBF test set. Despite their strong diastolic peaks, the ground truth PPG signals contained in OBF are still more regularly shaped than rPPG estimates given by the neural networks, and as such did not usually cause problems for the neurokit peak detection function used as it was specifically designed for use with PPG signals recorded from the finger in the first place.

6.2. Performance on COHFACE

On the COHFACE data set, none of the tested methods clearly outperformed the others, with all of the methods achieving quite similar performance based on the used metrics as seen in Table 4. The best performance was achieved neither by the deepest nor the shallowest neural networks but rather by the CNN-reduced model, with the rest of the tested methods performing slightly worse. The CNN-reduced model achieved the best MAE and PCC of all the tested methods when using neurokit for HR calculation with a MAE of 10.37 ± 5.27 and RMSE of 13.34 and the best PCC at 0.07990 with Welch's method as the HR calculation method. Much like with the OBF test data, using neurokit for HR calculation generally outperformed Welch's method as neurokit peak detection produced better results than Welch's method for every neural network model based on RMSE with the only exception of CNN-deep. Interestingly, using neurokit peak detection for HR calculation also produced better results for Face2PPG compared to Welch's method with a MAE of 12.67 ± 7.07, RMSE of 16.17 and PCC of 0.01173. This is unlike the results on the other two sets of test data on which Welch's method clearly outperformed neurokit in accurately calculating heart rates from the rPPG signals extracted with Face2PPG. When comparing the HR estimation performance of Self-ONNs to CNNs on the COHFACE data set, with neurokit peak detection used for HR calculation, Self-ONN models generally outperformed the CNN models of the same depth except when comparing SelfONN-reduced and CNN-reduced. However, for the results achieved using neurokit, the differences in performance were small — not only between neural networks of the same depth but between all the tested methods. The Face2PPG pipeline was outperformed by the neural networks, with each neural network model producing a better result than the best result achieved using Face2PPG — albeit narrowly. As each method achieved quite similar results on COHFACE, it is rather difficult to draw conclusions about the performance of the proposed Self-ONN models against the other tested methods based on these results.

Table 4. Performance of each method in estimating heart rates on the COHFACE data set with the best result for each metric highlighted in bold.

rPPG extraction	Evaluation metric			HR calculation
method	MAE $\pm$ SD	RMSE	PCC	method
SelfONN-deep	14.91 ± 6.20	19.77	-0.006944	Welch
CNN-deep	10.89 ± 4.94	14.98	0.03400	Welch
SelfONN-reduced	20.43 ± 8.89	28.03	0.03479	Welch
CNN-reduced	13.45 ± 6.91	20.40	**0.07990**	Welch
SelfONN-shallow	20.10 ± 11.38	29.37	0.01329	Welch
CNN-shallow	19.01 ± 10.29	27.92	0.01559	Welch
Face2PPG	13.90 ± 6.84	20.67	0.005546	Welch
SelfONN-deep	10.98 ± 5.06	13.54	0.001720	neurokit
CNN-deep	12.22 ± 7.95	15.88	-0.01350	neurokit
SelfONN-reduced	11.39 ± 4.73	14.14	-0.02040	neurokit
CNN-reduced	**10.37 ± 5.27**	**13.34**	0.07375	neurokit
SelfONN-shallow	11.44 ± 5.07	14.30	-0.003261	neurokit
CNN-shallow	11.46 ± 4.94	14.34	-0.002377	neurokit
Face2PPG	12.67 ± 7.07	16.17	0.01173	neurokit

The results achieved by the models on COHFACE are very different from the results on OBF discussed earlier, which is explained by the differences in the nature of data in these data sets. Each method produced significantly poorer results on COHFACE than they achieved on OBF, which suggests that the data in COHFACE is overall more challenging in nature compared to OBF. Although the scenarios under which the video samples in COHFACE were recorded are in fact quite similar to OBF, with the subjects remaining still and looking at the camera at all times, the actual recordings are heavily compressed and of a relatively low quality in general with a resolution of only 640 by 480 pixels and frame rate of only 20 FPS, which is the lowest in any of the data sets used in the experiments. This is in stark contrast to high quality data sets such as OBF, which was recorded at a full HD resolution at 60 FPS. The overall low quality of the data may make it challenging to accurately extract rPPG signals from the videos and could explain why the performance of all the tested methods is similarly underwhelming on this particular data set.

In addition, the relative portion of the frame area taken up by the face is generally lower in COHFACE compared to the OBF database, for example, which implies that the relative face area resolution compared to OBF is lower still. To compare this, the average size of the rectangular face detection in the first frame of each video as given directly by the used Single Shot Multibox Detection network was calculated for both data sets. For OBF, the average width and height of a face detection were 456 and 607 pixels respectively, whereas for COHFACE the corresponding dimensions were only 146 and 205 pixels. The average area taken by the face detection was 279367 pixels for OBF, or 13.5 % of the frame resolution, and 30230 pixels for COHFACE, or 9.8 % of the frame resolution. Although each face detection was resized to 128 by 128 pixels before being input to the rPPG extraction networks, the original resolution of the detection affects the quality of the resulting resized image as results can degrade especially when the resolution of the original detection becomes low enough to be close to the target resolution when the video is heavily compressed to begin with.

The reason for the neural networks performing slightly better than the unsupervised Face2PPG pipeline could be due to the neural networks having been trained on the OBF data set which, despite all its differences to COHFACE, does represent similar overall recording settings in which the subjects sit still facing the camera. This familiarity with somewhat similar data may have provided the supervised methods with an edge over the unsupervised Face2PPG, but due to the observed overall difficulty of the COHFACE data set, the performance of even the neural network models falls far short of the results on OBF.

The generally low quality and high compression of the video samples in the COHFACE data set could be a significant limiting factor in how accurately heart rates could feasibly be estimated from the videos. In their paper, Álvarez Casado and Bordallo López [4] compare the results achieved by different rPPG extraction approaches on this data set, and the best result they report based on MAE is achieved by their proposed multi-region pipeline at 7.5 ± 3.5 when employing the RGB to rPPG extraction method proposed in [28]. The result is rather poor compared to the results that the authors report on most other data sets despite the seemingly easy nature of the COHFACE data set which was recorded under controlled conditions. This suggests that too much of the information relevant to heart activity could be lost due to the compression and low quality of the data in order to reliably achieve excellent results such as those on OBF presented in Table 2.

Figure 12 presents an example result calculated from a video in the COHFACE data set. The video was recorded under artificial lighting, which could be considered the easier of the two scenarios in the data set. From the extracted rPPG signals on the left, it can be seen that the signals extracted by the neural network no longer resemble real PPG signals recorded from the finger as was the case for the example video from the OBF test set. On the other hand, the signal given by Face2PPG does resemble the results achieved by the unsupervised pipeline on OBF. Although none of the signals given by any of the methods resembles a real PPG signal, it can be seen that the deeper neural networks attempt to recreate the typical features of PPG signals more closely than the other methods by producing more pronounced peaks with the characteristic shape of a systolic peak, whereas steep peaks are absent in the signals given by the shallower models which in turn resemble the signal achieved using Face2PPG. The deeper the neural network model, the more steep peaks it produces and the higher the peaks are.

Although the signals in Figure 12 are not as cleanly shaped and the peaks are less pronounced than in the results on OBF, the neurokit peak detection algorithm generally produced more accurate HR estimates compared to Welch's method for this example, which was also the trend on the entire COHFACE data set. This can mostly be explained by some observed underlying problems in the used implementation of the Welch's method algorithm, which can be seen in the figure as sudden increases and decreases in the heart rates calculated with Welch's method. These issues are discussed in more detail in Section 7.2 along with other problems in the implementation recognized during experiments.

Similar conclusions could be drawn from Figure 13, which shows example results calculated from a video recorded of the same subject as in the example shown in Figure 12, but this time under natural lighting. Similar characteristics can be observed in the shape of the rPPG signals, and much like in the previous example, using neurokit for HR calculation generally produced better results. In addition, the aforementioned issues observed with Welch's method are even more prominent in this example, especially in the case of SelfONN-shallow and CNN-shallow.

Unlike the example result on OBF in Figure 10, no method could be expected to be feasible for reliably extracting HRV features on COHFACE based on Figure 12 or Figure 13. Although the average heart rates are calculated quite accurately in Figure 12, the actual locations of the peaks and the individual inter-beat intervals do not appear consistent or accurate enough for reliable calculation of HRV features in any of the rPPG signals plotted on the left. The irregular shape of the signals and the resulting inconsistencies in IBIs could be explained in part by the neural networks struggling with data that is different from the data on which they were trained as well as the generally challenging nature of the data or the relatively low quality of the video recordings.

6.3. Performance on LGI-PPGI

The best performance on the LGI-PPGI data set was achieved by the Face2PPG pipeline with Welch's method as the HR calculation method, significantly outperforming every other tested method based on every evaluation metric as seen in Table 5 by achieving a MAE of 6.967 ± 5.415, RMSE of 13.98 and PCC of 0.4003. The best performing neural network model based on all metrics was SelfONN-shallow, which achieved a MAE of 11.30 ± 8.11 and PCC of 0.2576 with Welch's method and the best RMSE value of the supervised models at 19.90 with neurokit peak detection. Comparing the two HR calculation methods, neither is clearly better than the other with Welch's method resulting in a better RMSE compared to neurokit in the case of 4 of the 7 tested methods. However, Welch's method very clearly outperforms neurokit in calculating HRs from the signals extracted with Face2PPG. The Self-ONN models generally achieve better performance compared to the CNN models of the same depth although the differences are not very large. In the case of both Self-ONN and CNN models, the shallower the model, the better the achieved performance was based on MAE and RMSE regardless of which HR calculation method was used, with the only exception being SelfONN-deep outperforming SelfONN-reduced when comparing results calculated with neurokit.

Table 5. Performance of each method in estimating heart rates on the LGI-PPGI data set with the best result for each metric highlighted in bold.

rPPG extraction	Evaluation metric			HR calculation
method	MAE ± SD	RMSE	PCC	method
SelfONN-deep	16.24 ± 12.04	24.72	0.03465	Welch
CNN-deep	16.08 ± 15.47	26.24	0.1277	Welch
SelfONN-reduced	12.90 ± 8.96	22.05	0.1918	Welch
CNN-reduced	13.75 ± 13.63	23.75	0.1443	Welch
SelfONN-shallow	11.30 ± 8.11	21.60	0.2576	Welch
CNN-shallow	12.44 ± 7.96	23.21	0.2538	Welch
Face2PPG	**6.967 ± 5.415**	**13.98**	**0.4003**	Welch
SelfONN-deep	15.71 ± 13.82	23.16	-0.02141	neurokit
CNN-deep	20.01 ± 17.39	28.50	0.09374	neurokit
SelfONN-reduced	15.72 ± 13.57	23.29	0.05821	neurokit
CNN-reduced	15.19 ± 15.40	23.93	0.09676	neurokit
SelfONN-shallow	12.85 ± 11.60	19.90	0.1276	neurokit
CNN-shallow	13.55 ± 11.11	20.56	0.05849	neurokit
Face2PPG	14.27 ± 12.65	21.42	0.1447	neurokit

The very noticeable difference in the results achieved on LGI-PPGI compared to the other two data sets used in testing can be explained by the nature of data it contains. While the videos in the OBF and COHFACE data sets were recorded under quite strictly controlled conditions in which the subjects remained mostly still and looked at the camera, the LGI-PPGI database also contains samples recorded under more realistic and natural scenarios, in which the subjects move their heads, talk, or exercise, for example. This makes some of the data in LGI-PPGI very dissimilar to the videos from the rather homogeneous OBF database in particular, of which the set of train data used for training the supervised models is made up.

Fitting a homogeneous set of train data which lacks examples of varied and challenging scenarios does not adequately prepare the neural networks to handle the more varied data from LGI-PPGI, and as such, the supervised models struggle to produce good results when they are presented with some of the more challenging samples from this data set. This inability to generalize to new samples which are recorded under more varied conditions represents a type of overfitting which stems from the lack of variation in the train data. The unsupervised Face2PPG pipeline, on the other hand, clearly outperforms all of the tested supervised methods due to not relying on any train data at all and as such not learning to make assumptions regarding the data with which it is presented as the supervised models do.

Unlike on COHFACE, on which it was approximately equally difficult to achieve good performance with every method, in the results on LGI-PPGI there is a very clear disparity between the results achieved by the methods based on how much the method bases its results on the OBF train data set. This implies that the deepest models, which have the most capacity to fit the train data and as such learn to make the most assumptions about the data with which they are presented, struggle the most when presented with samples from a completely different distribution of data. In contrast, the shallower models fare better as their lesser capacity to fit the train data also implies a lesser ability to overfit the train data, resulting in better generalization performance. In turn, the Face2PPG pipeline, which as an unsupervised method was neither trained nor tuned on the OBF train data, performs the best as mentioned earlier. The result is also in stark contrast to the results on the test set comprised of samples from the OBF data which are very similar to the train data, providing an advantage to the the supervised models with high capacity.

However, a somewhat surprising result is that SelfONN-deep was able to achieve better performance compared to CNN-deep on LGI-PPGI based on MAE and RMSE despite fitting the OBF train data more closely based on the NMCC loss function as seen in Table 1. Similarly, SelfONN-reduced and SelfONN-shallow also achieved a lower train loss and better performance than their respective CNN counterparts on LGI-PPGI based on these metrics when comparing results achieved with Welch's method HR calculation. This suggests that what the Self-ONN models learned in order to fit the train data they were presented with could be more suited for rPPG extraction on even the more varied data of the LGI-PPGI data set when compared against their CNN counterparts. The results might imply that the Self-ONN models are more capable of learning the fundamentals related to remote PPG extraction from facial videos, even from limited train data, and less prone to learning to take shortcuts based on the specific characteristics of the train data compared to CNN models. However, it is difficult to make conclusive statements based on these rather limited results, especially when the differences in performance are quite small. As such, more research should be conducted to further evaluate these properties of Self-ONN models.

Another point of comparison worth noting between the results on COHFACE and LGI-PPGI is that despite the difficult realistic scenarios included in LGI-PPGI, the Face2PPG pipeline achieves much better performance on LGI-PPGI compared to COHFACE. The reason for this is most likely a combination of two factors. Firstly, the processing steps in the unsupervised Face2PPG pipeline seem to be well suited to the purpose of extracting rPPG signals from the face, being capable of producing meaningful results even on data that is challenging for supervised methods. The second factor is the higher quality of the data in LGI-PPGI compared to COHFACE. While the data in COHFACE was highly compressed and of poorer quality in general, causing every tested method to struggle on the data set, the data in LGI-PPGI is significantly higher in quality, which could explain why Face2PPG can achieve better results on it compared to COHFACE despite its seemingly more challenging nature. As an unsupervised method, the pipeline does not make assumptions about the type of data with which it is presented, and as such the 'difficult' nature of the data in LGI-PPGI is does not affect its performance much compared to the supervised methods, making the fidelity of the data a more significant limiting factor in how well it can perform.

The reason why Welch's method generally outperforms neurokit peak detection in calculating heart rates on LGI-PPGI can be explained by the inability of the tested methods to produce neatly shaped rPPG signals resembling real PPG signals with pronounced peaks on this kind of challenging data, which in turn makes the peak detection algorithm prone to missed or false positive detections. The Face2PPG pipeline was never trained or designed to produce signals that resemble the PPG signals recorded from the finger like the neural networks were. The neural networks, on the other hand, struggle to extract the kind of output signals that they were trained to produce due to the dissimilarity of this data to the data on which they were trained. Frequency-based HR calculation methods, such as Welch's method, are more robust to the shape of the signal as long as the fundamental frequency of the signal corresponds to a real heart rate. The better results achieved using Welch's method, especially with Face2PPG, suggest that although detecting the systolic peaks and their locations in the signals extracted from the videos in LGI-PPGI is challenging, the frequency content in the signals still contains information relevant to heart activity.

Figure 14 shows an example result on a video from the LGI-PPGI data set. The video is an example of the easiest scenario in the database in which the subject faces the camera with minimal head movement. In the case of this relatively easy example, every method performs quite well and especially the results achieved using Welch's method are quite promising. The HR envelopes calculated with neurokit peak detection, on the other hand, show some clear deviations from the ground truth HR envelope resulting from missed or false positive peak detections in many of the rPPG signals. From the rPPG plots on the left, it can be seen that the neural network models attempt to produce realistic PPG signals with pronounced peaks, but struggle as the peaks in the rPPG signals they produce are not very pronounced or high in amplitude. Although the video was recorded under an almost ideal setting for the neural networks to perform well, the nature of the data still differs enough from the train data for the performance of the supervised methods to suffer, which is a phenomenon that could also be observed on the COHFACE data set. The signal produced by Face2PPG is similar to the ones it produced on data from the other sets. It does not show any clear systolic peaks, but the frequency content in the signal is relevant to the subject's heart activity based on the HR envelope calculated with Welch's method.

Figure 15 contains an example result on a video of the same subject as the previous example in Figure 14 but recorded under what could be considered the most challenging of the scenarios in which the subject talks naturally in an urban environment with natural variations in illumination. Unlike the previous example, on which every method performed relatively well, this example shows clear differences in the accuracy of the HR estimations achieved with each tested method. This example follows the pattern of the results observed on the entire LGI-PPGI data set as shown in Table 5 as Face2PPG clearly outperforms the supervised methods, and the shallower neural networks outperform the deeper ones. This further suggests that the videos recorded under more natural conditions, such as the example in this figure, which deviate from the samples in the train data are the reason behind the overall poor performance of the deeper neural networks on this data set — both compared to their results on other data as well as the results of other methods on LGI-PPGI.

Based on the peak detection results seen in either of the example figures, no method could reasonably be considered applicable to the extraction of HRV features, much like was the case in the examples from COHFACE in Figures 12 and 13. The peak locations in the rPPG signals are irregular and do not correspond to the ground truth, and the peak detection algorithm struggles to detect the peaks correctly in the first place due to the inconsistencies in the shape of the signals.

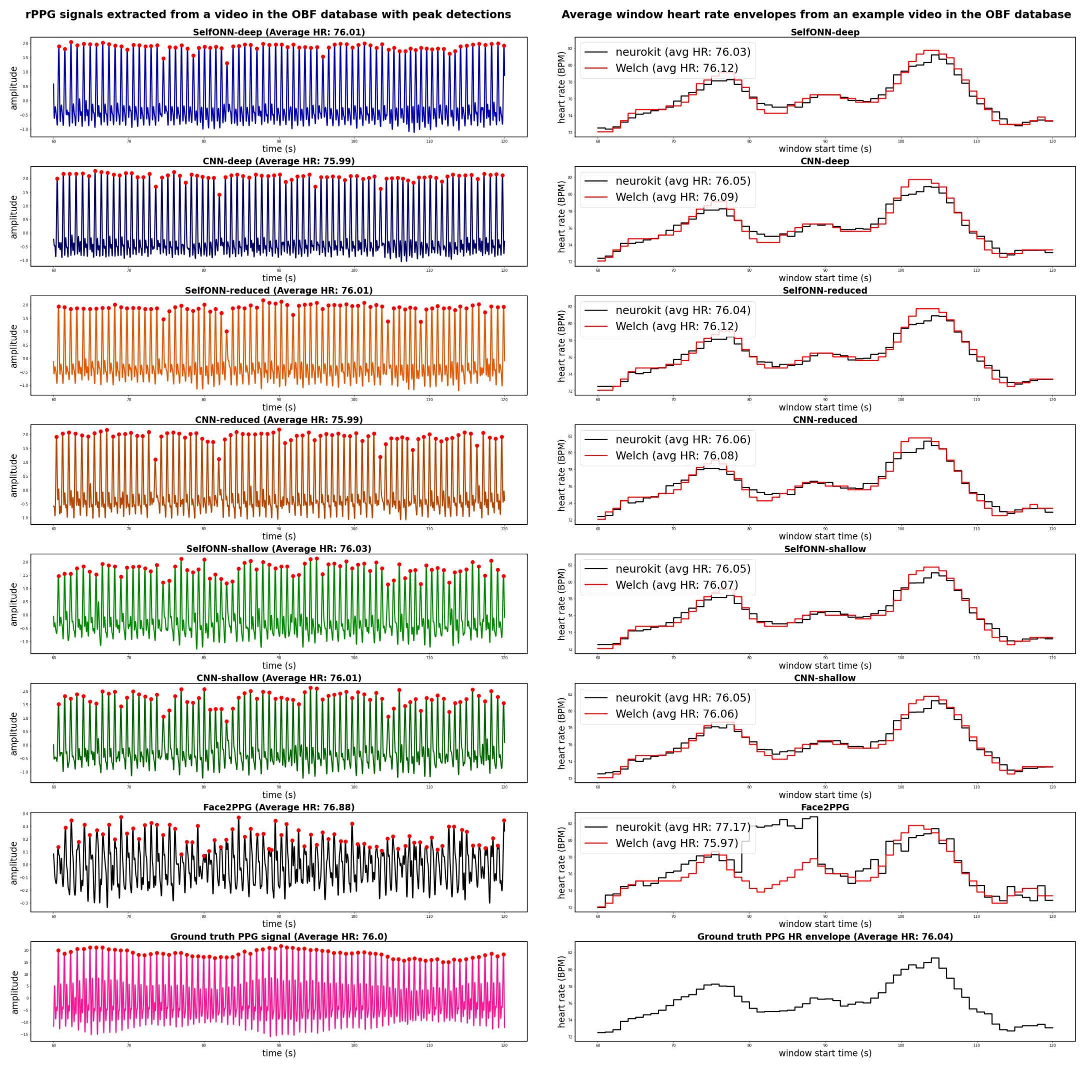

Figure 10. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a video from the OBF database on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

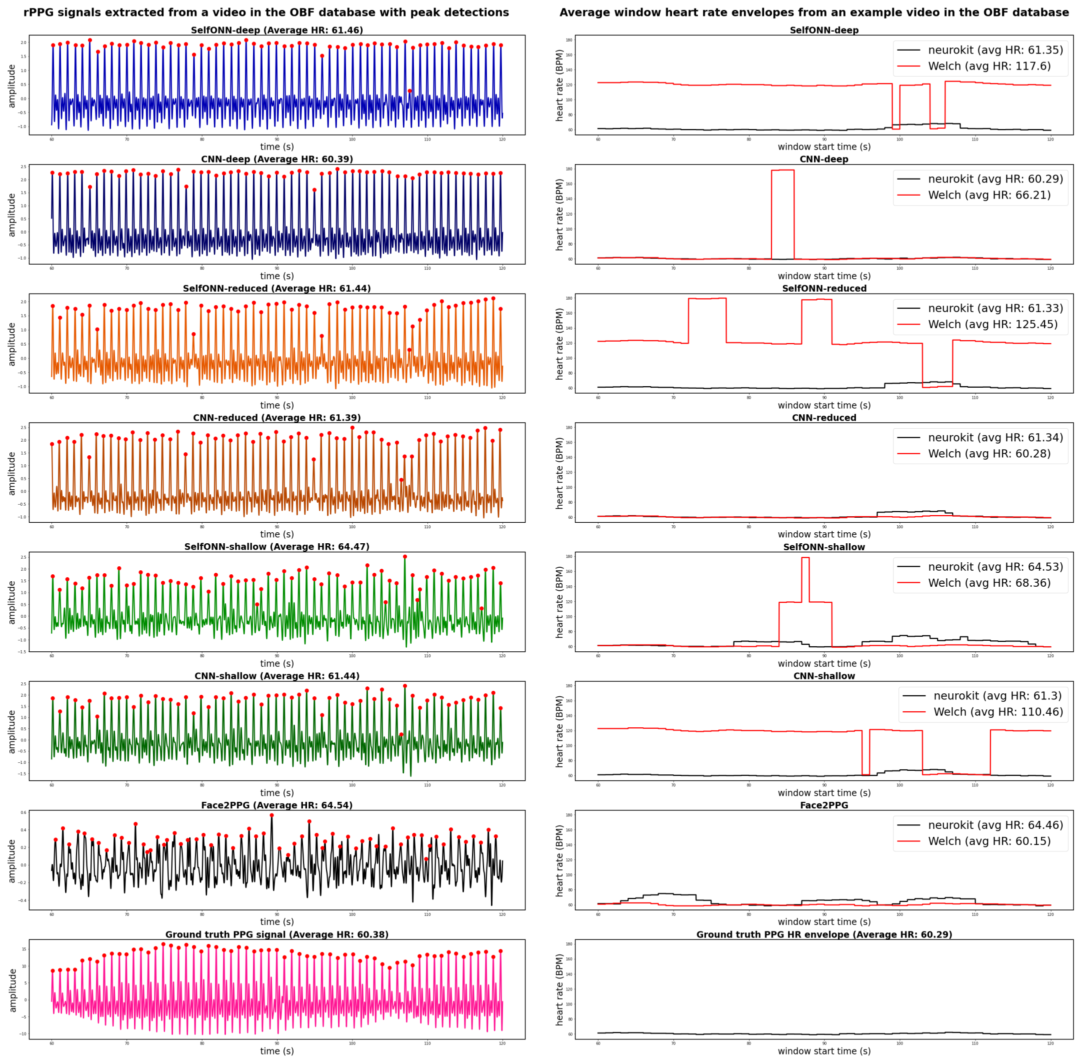

Figure 11. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a video from the OBF database on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

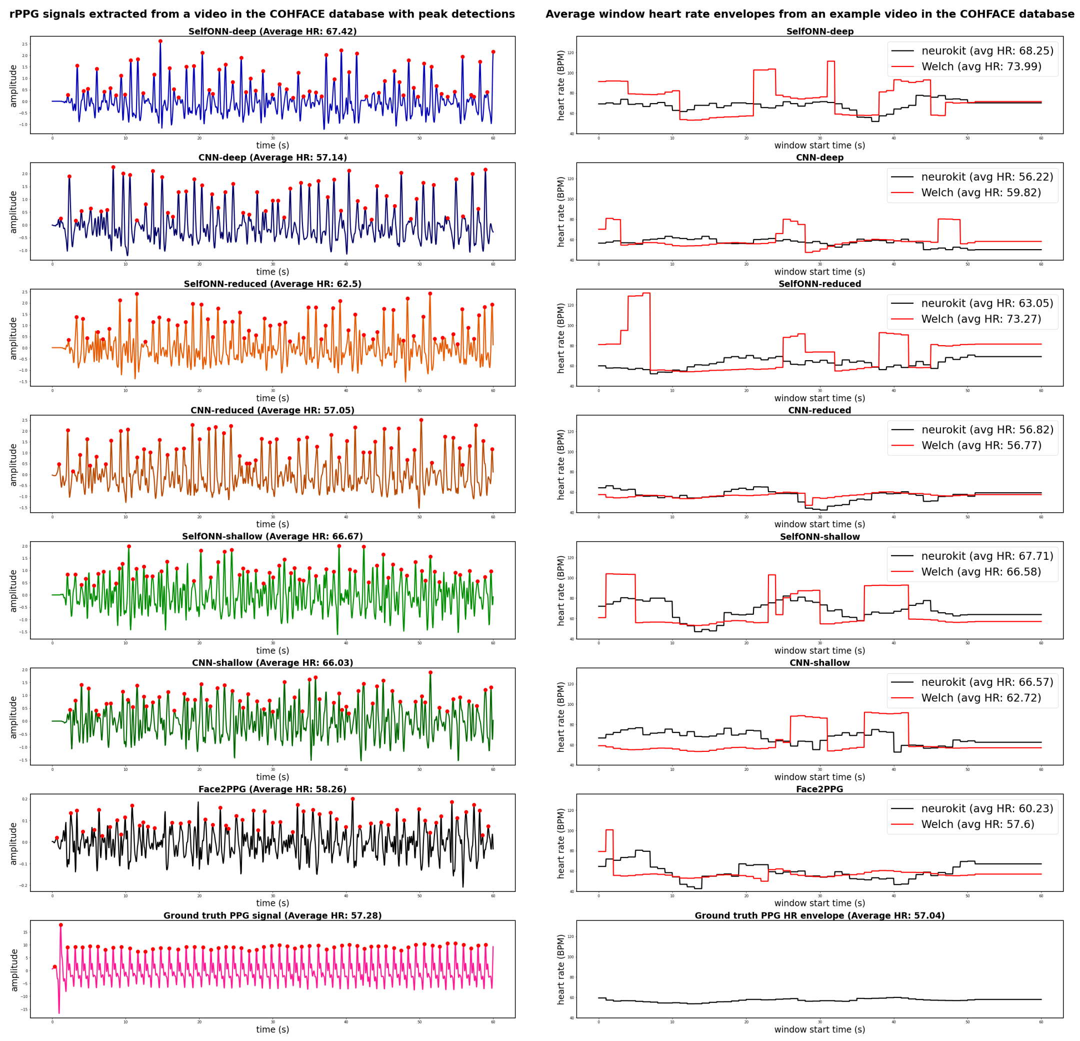

Figure 12. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a video from the COHFACE data set on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

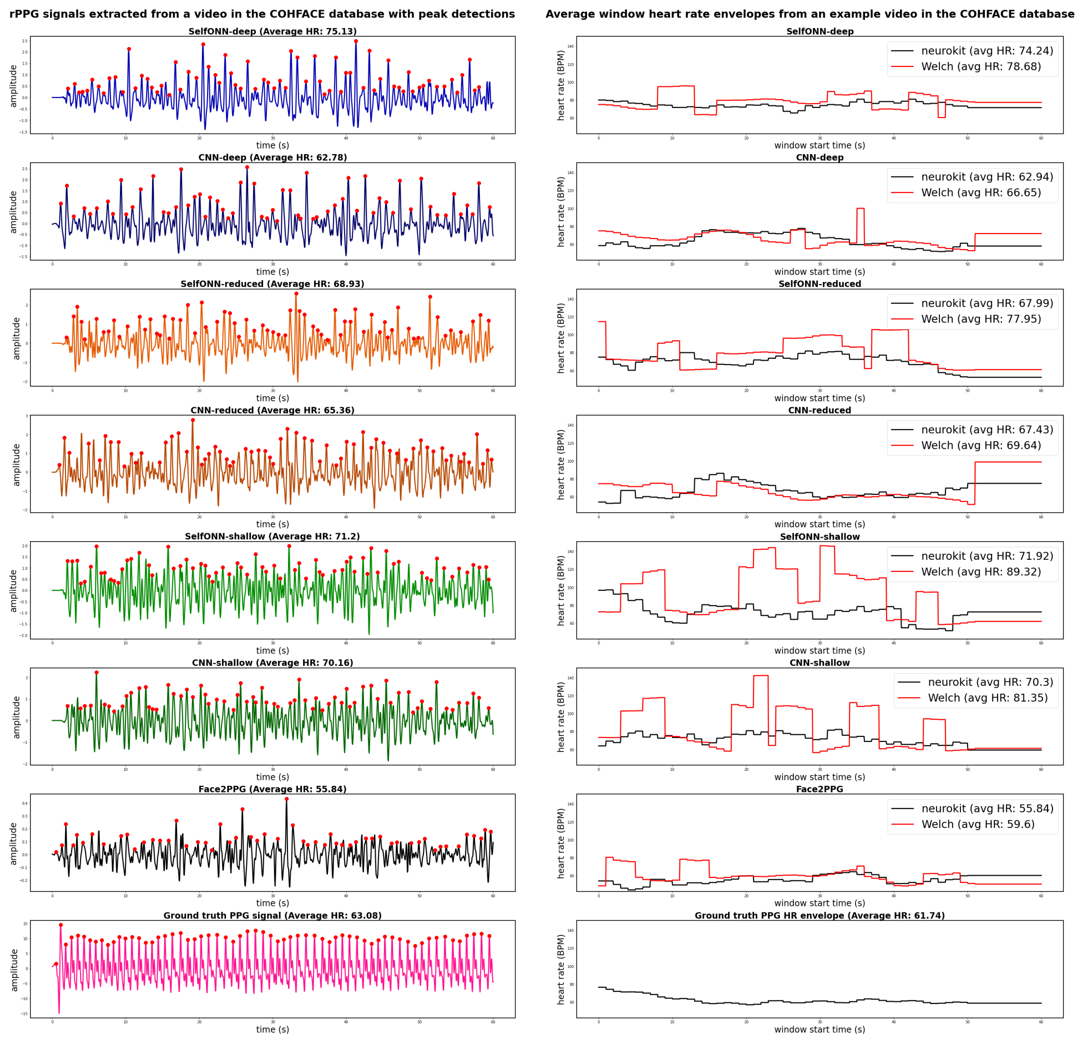

Figure 13. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a video from the COHFACE data set on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

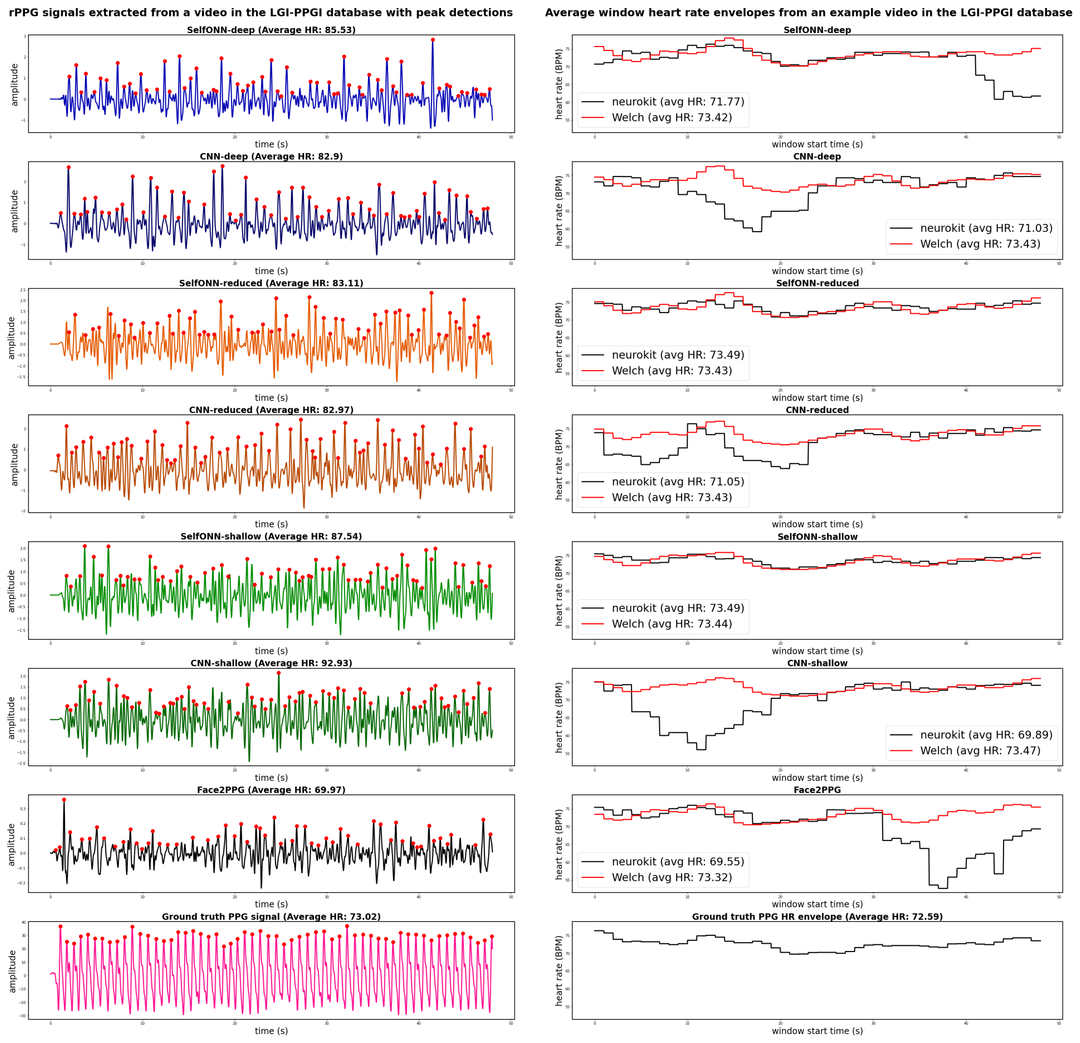

Figure 14. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a resting scenario video in LGI-PPGI on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

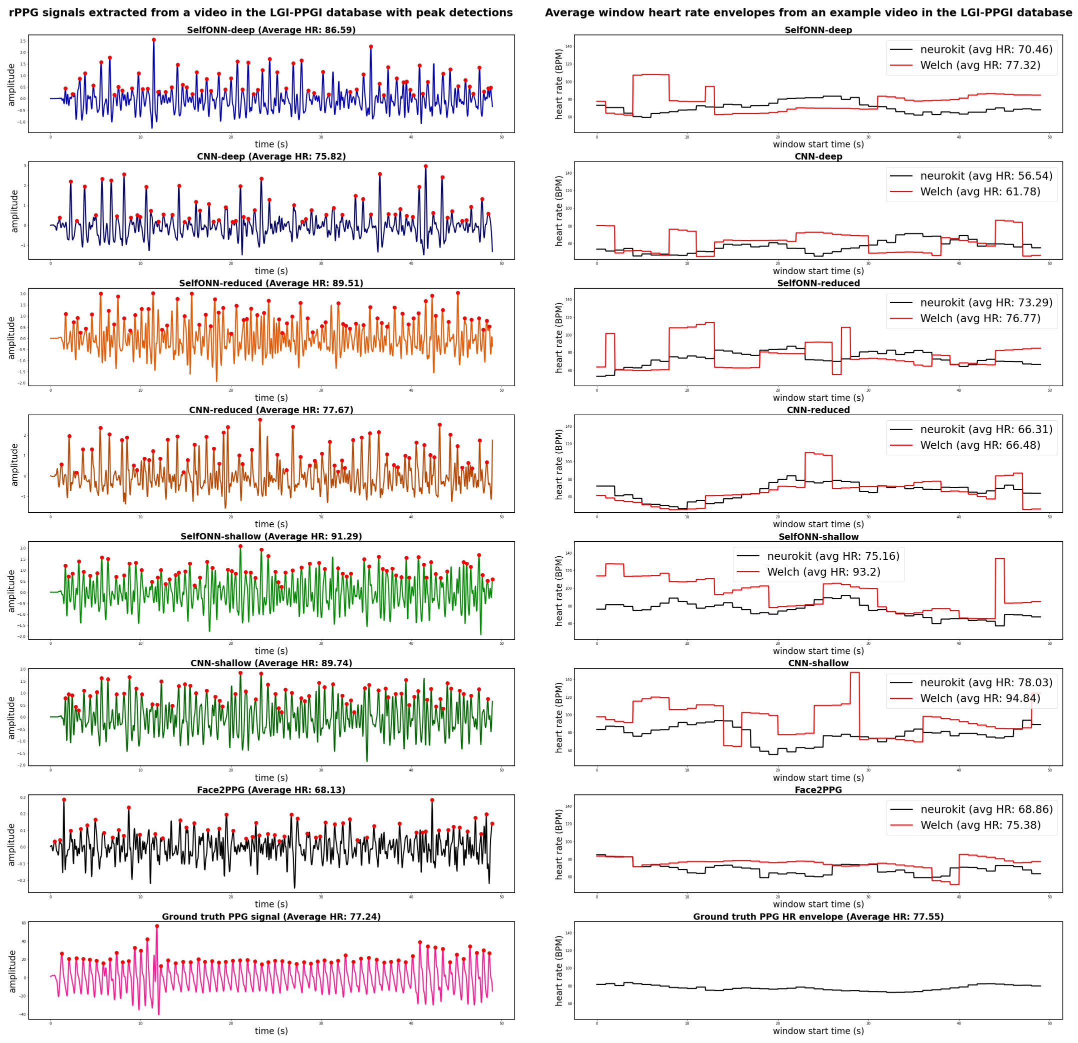

Figure 15. A figure showing the rPPG signals and ground truth PPG signal as well as the corresponding peak detections associated with a talking scenario video in LGI-PPGI on the left along with the respective HR envelopes achieved with neurokit and Welch's method plotted on the right.

7. FURTHER DISCUSSION

This chapter provides some further discussion based on the results presented in Chapter 6, covering certain topics not addressed in the previous chapter. While the analysis presented in Chapter 6 focused on the individual performance of each method on each set of test data, the discussion in this chapter focuses less on the individual results on each test set and aims to draw more general conclusions, particularly about the efficiency of Self-ONN models against CNN alternatives, in addition to addressing certain issues common to the applied methods. The potential efficiency gains achievable by opting for a Self-ONN approach are discussed first, after which certain problems recognized in the implementation are addressed. The chapter then concludes by discussing potential directions for future research.

7.1. Efficiency of ONN Models against CNN Models

While the previous analysis was mostly only based on the performance of the tested methods as measured by the evaluation metrics described in Section 5.4, this section aims to provide insight to the efficiency of the Self-ONN models against their CNN counterparts, taking into account the time and computational resources required by the methods to produce their outputs. In some ways, the efficiency of the models was already evaluated previously by comparing how the depth of the model affects the results of the Self-ONN models compared to CNNs. However, as each Self-ONN model was designed to have approximately the same number of parameters as its CNN counterpart as detailed in Section 4.2 and also seen in Table 6, and comparisons between the parameter-equivalent neural network alternatives were already covered previously in Chapter 6, this section places more focus on the time required by rPPG extraction using each neural network model as well as the time taken to train the models as seen in Table 6. Based on these metrics, the efficiency of the models is evaluated by analysing how the computational complexity and time requirements translate into HR estimation performance in the case of each type of neural network model.

In order to gain insight into the efficiency of each neural network model, the average inference time for 3600 frames of video, which is equivalent to 60 seconds at 60 fps, the time taken by training each model with the setup described in Section 5.1.3, as well as the number of trainable parameters for each neural network are presented in Table 6. The average inference times are calculated based on all the videos in the test data, although using any set of data should also produce comparable results. In addition to these three metrics, which illustrate the computational resource requirements of the model, Table 6 also lists the HR estimation RMSE achieved by each model on each data set used in testing as well as the total HR estimation RMSE on all the test data, with neurokit peak detection as the HR calculation method. It should be noted that as the OBF test data contains 40 samples, COHFACE contains 164 and LGI-PPGI contains only 24, the samples from COHFACE form the majority in the test data, which is reflected in the total test data RMSE measures presented. The measures of computational complexity together with the HR estimation error metrics allow for quick assessment of how an increase or decrease in the computational complexity of the models translates to increases or decreases in error on the different types of data in the case of the tested CNN and Self-ONN models.

Table 6. Metrics illustrating the computational complexity of the tested neural network models against HR estimation RMSE on the test data.

Model	Trainable parameters	Extraction time [s]	Training time [h]	RMSE on test data
SelfONN-deep	860063	3.48	419	OBF: 4.396 COHFACE: 13.54 LGI-PPGI: 23.16 Total: 13.85
CNN-deep	858497	3.74	12.5	OBF: 0.9495 COHFACE: 15.88 LGI-PPGI: 28.50 Total: 16.34
SelfONN-reduced	305063	2.30	186	OBF: 3.786 COHFACE: 14.14 LGI-PPGI: 23.29 Total: 14.26
CNN-reduced	304577	2.43	6.83	OBF: 2.308 COHFACE: 13.34 LGI-PPGI: 23.93 Total: 13.75
SelfONN-shallow	70631	2.24	164	OBF: 6.817 COHFACE: 14.30 LGI-PPGI: 19.90 Total: 14.04
CNN-shallow	70529	2.40	6.08	OBF: 7.155 COHFACE: 14.34 LGI-PPGI: 20.56 Total: 14.19

By observing the signal extraction times presented in Table 6, it can be seen that every Self-ONN model can produce its output in less time than the corresponding CNN alternative. This suggests that a Self-ONN based approach has potential to provide a more efficient solution to the problem at hand compared to CNN alternatives, most likely resulting from the reduced number of neural units in the Self-ONN models which is enabled by the added nonlinearity provided by the generative neurons. In the experimental results presented in this thesis, there were cases in which a Self-ONN achieved very similar performance when compared with its CNN counterpart. Most noticeably, SelfONN-shallow and CNN-shallow achieved very similar performance on every set of test data used. In such cases, the Self-ONN model is more efficient in extracting a useful signal from the given videos as it can provide similar accuracy based on the metrics used with less time required for signal extraction.

However, while each Self-ONN model requires less time for signal extraction than its CNN counterpart, the time required for training the Self-ONN models is greater by an order of magnitude compared to the CNN models with the setup described in Section 5.1.3 as seen in Table 6. In fact, even training the shallowest Self-ONN model took more than 13 times the time required to train the deepest CNN. This implies that in an iterative development process of Self-ONN architectures can require significantly more time and resources compared to CNN alternatives, which should also be taken into account when deciding which of the approaches to utilize. However, once the investment in developing and training the architecture has been made, a Self-ONN approach can potentially lead to a more efficient end result due to the reduction in output computation time as seen in Table 6.

When comparing the metrics associated with SelfONN-reduced in Table 6 against the CNN model with the shortest required extraction time, CNN-shallow, it can be seen that SelfONN-reduced can achieve better performance on COHFACE and significantly better performance on the OBF test set based on RMSE while still managing a shorter signal extraction time. The performance of SelfONN-reduced on LGI-PPGI, however, was poorer compared to CNN-shallow, most likely due to overfitting as detailed earlier in Chapter 6, which leads the model to have a slightly higher overall RMSE. However, the OBF and COHFACE test sets are more similar to the data used in training the neural networks than LGI-PPGI, and as such, SelfONN-reduced achieved better performance on the type of data which was represented in the train set. If the set of train data were constructed to contain more varied examples representing realistic scenarios, SelfONN-reduced could potentially see more benefit. In addition, the total RMSE levels achieved by the models is still quite close, but SelfONN-reduced requires less time for signal extraction, which implies better overall efficiency, at least based on the time required to compute the signals. However, the reduced computation time comes at the cost of significantly more trainable parameters as seen in Table 6.

Next, a closer look is taken at the performance achieved by SelfONN-shallow, which is compared against its CNN counterpart in CNN-shallow as well as the ’reduced’ models as the most directly comparable alternatives in order to get an idea of the potential efficiency gains achievable with a Self-ONN approach when the complexity of the model is severely restricted. When comparing the RMSE values associated with the heart rates calculated with neurokit as seen in Table 6, it can be seen that the results achieved using SelfONN-shallow and CNN-shallow are very similar, with SelfONN-shallow only slightly outperforming CNN-shallow on every set of test data in this setting. However, as SelfONN-shallow requires less time for computing its output signal, and both models have approximately the same number of trainable parameters, SelfONN-shallow can be considered to provide the more efficient solution of the two. SelfONN-shallow also achieves similar total RMSE to CNN-reduced as seen in Table 6 despite requiring less time for signal extraction and only 4.32 % of the number of trainable parameters, implying better overall efficiency in the test setup. However, total RMSE values achieved by these two models are only close due to the better performance of SelfONN-shallow on LGI-PPGI, which is most likely due to its lesser capacity to overfit the train data. The results achieved using SelfONN-shallow on COHFACE and especially the OBF test set fall short of CNN-reduced, which in turn suggests a significant gap in performance on data represented in the train set. As such, CNN-reduced could potentially benefit more from increases in the variety of the train data. Still, SelfONN-shallow can achieve respectable performance given the significantly lesser computational complexity. Further research with more diverse data should be conducted to test the potential increases in efficiency achievable with Self-ONN models.

Comparing SelfONN-deep against CNN-deep, it can be seen that SelfONN-deep can produce better results on both COHFACE and LGI-PPGI, also achieving a significantly lower total RMSE, with a lower required extraction time. As both models have approximately the same number of trainable parameters, SelfONN-deep could be considered to provide the more efficient solution of the two in this case. However, the performance of SelfONN-deep on OBF test data falls short of CNN-deep for reasons discussed earlier in Section 6.1.

A significant point worth noting about the Self-ONN models is that as each of them achieves a lower train loss compared to its CNN counterpart in addition to requiring less time for producing its output signal, every Self-ONN model can be considered significantly more efficient in achieving the training target as defined by the train data and target NMCC loss function. As such, if the issues related to the training setup mentioned in earlier in Section 6.1 are addressed, making for a more appropriate train setup for the end task of HR estimation on outside data, Self-ONN models could eventually lead to a much more efficient solution to the problem at hand than can be achieved with conventional CNN models. Ways in which the train setup can be improved and other potential future research directions will be discussed further in Section 7.3.

7.2. Recognized Problems and Limitations

During the experiments, some problems were observed with both of the methods used for calculating heart rates from the extracted rPPG signals, i.e., HR calculation using the PPG peak detection function from the neurokit library and the implementation of Welch's method based on the one provided by the authors of [4]. In some cases, the methods failed to produce accurate or even reasonable heart rates from certain rPPG signals even if the rPPG extraction preceding the HR calculation could have been considered reasonably successful. Such shortcomings in the HR calculations had a significant effect on the results of the experiments and should taken into consideration when analysing the presented results. This section discusses these problems in more detail and demonstrates their effects with examples from the experiments.

7.2.1. Neurokit Peak Detection

The problems with calculating heart rates using the PPG peak detection function in the neurokit library are associated with missing or false detections of systolic peaks. As some of the rPPG signals produced by the tested methods are shaped irregularly and very different from typical PPG signals recorded from the finger, they can represent an unexpected type of input for this peak detection function.

The neurokit peak detection function is often able to give excellent results on OBF for the signals extracted using the neural networks. In this setting, even the peak locations are generally accurate, and the peak detections could even be applied to the calculation of HRV features. However, on other data, the signals produced by the tested methods are much more irregular in shape, and the peak detection function often misses peaks or gives false detections. In fact, there are some cases in which the locations of the peak detections in the rPPG signals and the intervals between them deviate significantly from the ground truth even though the average IBI, and as such the resulting HR estimate, are approximately correct. This means that although the estimated heart rates calculated with neurokit may be correct, the peak detection locations may not be meaningful, which not only makes it impossible to make any estimations of HRV features but also brings under question the basis on which these heart rates were calculated. One such example can be seen in the peak detection result on the signal given by CNN-reduced in Figure 12. However, achieving an accurate HR estimation despite failed peak detection could simply be coincidence, and as such these results should be counterbalanced by the inaccurate results resulting from the missed and false peak detections in the final results presented in Chapter 6.

Although the neurokit peak detection function generally performed well with the rPPG signals extracted from the OBF test set by the neural networks, certain problems sometimes occurred in detecting the peaks from the signals extracted using the Self-ONN models. The problems were characterised by falsely detecting the particularly prominent reconstructed diastolic peaks produced by the Self-ONN models as systolic peaks. The strong reconstructed diastolic peaks in turn result from the characteristics of the train data and the training setup. An example of this phenomenon can be seen in the example results presented in Figure 11, and the underlying reasons behind this phenomenon were discussed in detail in Section 6.1.

In addition, neurokit peak detection sometimes falsely detected diastolic peaks as systolic peaks even in the ground truth PPG signals in the OBF database as mentioned in Section 6.1, although such cases were rare and the ground truth HR values calculated using neurokit were mostly correct. An example of such a case can be seen in the PPG signal on the right in Figure 16, displaying a 10 second segment of a PPG signal from the OBF database with two diastolic peaks erroneously detected as systolic peaks. This issue should also be taken into consideration when interpreting the results on the OBF test set.

The peak detection function also does not perform well on signals extracted using the Face2PPG pipeline as the signals produced by this unsupervised method deviate significantly from typical PPG signals recorded from the finger. The peak detection function often made mistakes when presented with signals extracted using this pipeline, which can be seen as false peak detections in Figure 10 or missed peak detections in Figure 14, for example. As an unsupervised method, Face2PPG only attempts to extract the underlying blood volume information present in the face and has no reason to make its output look more like a PPG signal recorded from the finger. As such, the neurokit peak detection function, which is specifically designed for analysing PPG signals measured from the finger, is inherently unsuitable for processing the signals produced by Face2PPG, which should be taken into consideration when assessing the results produced with Face2PPG combined with neurokit peak detection.

As the neurokit peak detection function performs well on signals extracted by the neural networks from data which is similar to their train data, the peak detection function could perform better in a scenario in which the neural networks are trained on more diverse data and as such learn to produce regularly shaped signals from a wider variety of facial videos. In addition, addressing other problems related to the train setup as discussed in Section 6.1 could also lead to better peak detection results. In the case of Face2PPG, however, peak detection would most likely not be the best approach for HR calculation in any scenario. As an unsupervised method, the pipeline cannot learn to produce signals that resemble the PPG signals recorded from the finger, which are the type of input expected by the peak detection function. If the pipeline were somehow tweaked to produce signals with more pronounced peaks, however, a custom peak detection approach tweaked for use with such signals could potentially produce good results.

7.2.2. Welch's Method

A frequently observed problem with the implementation of Welch's method used for calculating heart rates during the experiments is that for certain signals, its output deviates very significantly from the ground truth or the result achieved using peak detection, often outputting approximately double the desired HR value. The problem is most prevalent with rPPG signals extracted using the neural network models, whereas for signals extracted using Face2PPG, the Welch's method implementation generally performed as intended. Such large deviations in the HR estimates result in very large increases in the values of the used error metrics, especially in the case of RMSE due to the square operation.

The phenomenon can be seen very clearly in the example from OBF presented in Figure 11 in which Welch's method provides highly erroneous HR calculations for the signals extracted using neural networks in spite of the HR estimates calculated with neurokit peak detection being mostly accurate. The examples in Figures 12, 13 and 15 also exhibit the same phenomenon to some extent. The HR envelopes produced with the Welch's method implementation contain drastic instantaneous changes in the calculated HR value, which show as jagged shapes in the envelopes.

As mentioned earlier in Sections 5.3 and 6.1, the highly erroneous HR values produced by the Welch's method implementation were associated with the signals in the OBF database, and specifically the strong second harmonic components in the ground truth PPG signals in this set of data. The second harmonic was observed to often be even more prominent than the desired first harmonic, which usually corresponds to the real heart rate, which led to Welch's method providing the second harmonic as its output, resulting in erroneous HR calculations. As the data used in training the neural networks also consisted of samples from OBF, the rPPG signals extracted using the neural networks also inherit these properties, leading to erroneous HR values given by Welch's method.

To test whether the properties of the PPG signals contained in the OBF database are the underlying cause of the problem, the Welch's method implementation was used to calculate the heart rates from the ground truth PPG signals from each set of test data, which were then compared against the ground truth HR values calculated with neurokit peak detection. The resulting RMSE values of the heart rates calculated from the PPG signals with Welch's method were 48.3 BPM for the OBF test data but only 5.64 for COHFACE and 4.72 for LGI-PPGI, which suggests that the highly erroneous HR values given by Welch's method are associated with the nature of the data in the OBF database in particular.

As briefly mentioned in Section 5.3, the Welch's method implementation provided by the authors of [4] was altered so that in the presence of two or more strong harmonic frequency components in the signal, only the first of them is considered instead of simply outputting the frequency associated with the component with the most power in order to address these problems associated with applying the method to rPPG signals given by the neural networks. With this altered implementation of Welch's method, the RMSE of the heart rates calculated from OBF PPG signals against the corresponding ground truth HR values given by neurokit reduced to 36.7 BPM, which further indicates that the strong higher harmonic components were causing erroneous HR values to be produced by Welch's method. Adopting this altered implementation was also found to improve the accuracy of the heart rates calculated from the signals given by the neural network models compared to the original Welch's method implementation given by the authors of [4].

However, these changes did not improve HR calculation results for signals extracted using the Face2PPG pipeline, for which reason the original Welch's method implementation provided by the authors of the pipeline was employed as-is for these signals. The problem does not occur when using the method with the signals extracted using Face2PPG as these signals do not resemble the PPG signals from OBF, and as such do not contain the strong second harmonic components.

Although altering the implementation of Welch's method as described improved the HR calculations for OBF PPG signals as well as neural network rPPG signals, the problems associated with applying Welch's method to these particular types of signals could not be fully solved by these measures. When evaluating the PSD given by Welch's method for the PPG signals from the OBF database, it was found that for certain signals, only the peak associated with the second or higher harmonic of the desired heart rate was present, representing a scenario which cannot be addressed by the measures employed.

Figure 16 shows an example of a PSD in which such a phenomenon can be observed, corresponding to a 10 second PPG segment from the OBF database, which is displayed on the right. As seen in the figure, there are 10 systolic peaks present in the 10 second segment, and the HR calculated from the average interval between these peaks is approximately 65 BPM. However, this desired heart rate is in no way observable from the PSD to the left as the only peak in the spectrum occurs at approximately 173 BPM due to the higher frequency components in the signal, such as strong diastolic peaks or artifacts, which can also be seen by inspecting the waveform on the right. In fact, the diastolic peaks in this particular signal are so prominent that even the neurokit peak detection function mistakenly detects some of the diastolic peaks as systolic peaks, resulting in an erroneous HR value of 76 given by taking the reciprocal of the average IBI if these erroneous peaks are not removed. As mentioned above, the employed changes to the Welch's method implementation do not improve performance in this case, and the RMSE against the HR values calculated using neurokit peak detection is 109.3 for the entire 5-minute PPG signal with both the original and altered implementations. It is worth noting, however, that even the ground truth HR values calculated using neurokit peak detection contain errors as demonstrated in Figure 16. It is also worth noting that this particular signal represents one of the most problematic examples that could be observed in the OBF database, and most other PPG signals were not as pathological for either HR calculation approach.

Another problem observed when employing Welch's method for HR calculation was that in some rare cases, no peak could be detected in the PSD at all by the applied peak detection algorithm from the scipy library. In such cases the frequency corresponding to the maximum value of the PSD was selected instead as mentioned in Section 5.3. However, the absence of a clear peak in the PSD which could be interpreted as the desired HR value brings the viability of such HR estimates under question.

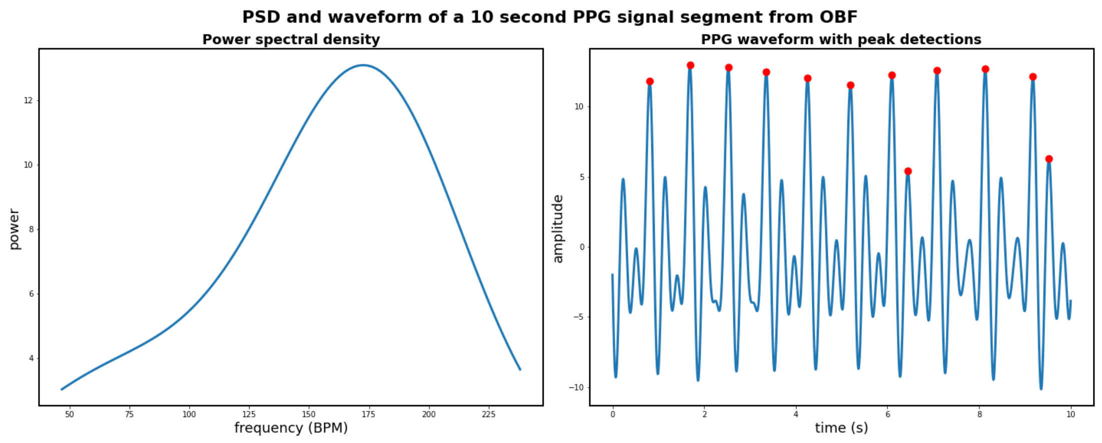

Figure 16. A figure showing the power spectral density and waveform corresponding to a 10 second segment of a PPG signal from the OBF database along with peaks detected using neurokit.

As Welch's method performed better of the two HR calculation methods in certain test scenarios, working around these issues could significantly improve the quality of the results that can be achieved with the neural network models as frequency-based methods provide a more robust approach for calculating heart rates than relying on achieving accurate peak detections. However, some of the problematic PSD calculations mentioned in the previous paragraph suggest that such a frequency based method may not be a suitable solution for all rPPG or even PPG signals. A possible way of mitigating the problem could be to make the signals more compatible with Welch's method by reducing the prominence of the second harmonic component in the PPG labels used in training the neural networks, e.g., by applying filters with cutoff frequencies based on the ground truth HR value. In addition, as the issues are mostly related to the nature of the PPG signals in the OBF database as established, training the neural networks using more varied data or different data altogether could also mitigate these problems.

7.3. Future Works

While the results presented in this thesis show mixed results for the applicability of Self-ONN models for remote PPG extraction from facial videos, it is important to note that these results are only a preliminary proof-of-concept, and there are numerous ways to research the topic further and potentially achieve much better results. The previous section already proposed some steps for improving the pipeline by suggesting solutions to problems recognized during the experiments. This section will provide further suggestions for improving the performance of the proposed approach and give insight into potential future research directions.

The most glaring issue with the experimental setup presented in this thesis is the formulation of the training setup of the neural network models described in Section 5.1. As briefly covered in Section 6.1, the Self-ONN models fail to achieve better HR estimation performance compared to their CNN counterparts despite fitting the train data more closely. In addition, the Self-ONN models outperformed their CNN counterparts on the validation set and OBF test set based on the NMCC loss function which was used in training the models. As the Self-ONN models were able to exhibit superior learning performance in the proposed setup, it is likely that reformulating the train setup to incorporate only the features which are useful for the desired end goal of HR and HRV calculation could result in the Self-ONN models producing better results compared to CNN alternatives in this end task. The primary problem in the training setup which should be addressed in future research is the choice of target ground truth used in training the models. In the experiments presented in this thesis, the neural networks were trained to replicate the full PPG waveform, which contains unnecessary features for the tasks of HR and HRV calculation, such as diastolic peaks. During the experiments, these features were found to cause problems for both HR calculation methods used, as discussed in the previous section.

A solution to this problem would be to adopt a similar approach to training the models as the one proposed in [11] in which the proposed neural network model was trained to only predict the systolic peak locations in the reference PPG signal based on the input facial videos, ignoring the overall shape of the PPG signal in question. The authors also list multiple alternatives for a suitable loss function for this task in addition to proposing their own novel loss function, the Wasserstein distance loss, which they demonstrate to perform better than the other tested alternatives found in literature. This kind of approach to training the supervised models makes sense as the peak locations contain all the relevant information for calculating both HR and HRV features.

In fact, the overall shape of the ground truth PPG signal is not only irrelevant for the task of HR and HRV calculation but also misleading as the shape of a PPG signal recorded from the finger does not correspond exactly to the blood volume changes that can be observed on the face. As such, training the models to output entire PPG signals requires the models to have more capacity in order to produce feasible estimates of what a corresponding PPG signal recorded from the finger may have looked like based on the facial video, even though this information cannot be reliably inferred from the face alone, and in reality, the peak locations are all that is needed for HR and HRV calculation. As the models require more capacity and only learn the characteristic shapes of the PPG labels in the train data with which they are presented, a training setup based on signal shape as a target can also lead to overfitting in itself as the models attempt to output similar signals to the train data labels regardless of the type of data with which they are presented.

The results presented in [11] show much promise for using the Wasserstein distance loss in training models for systolic peak detection as the authors report accurate detection of the systolic peaks and successfully apply the extracted HRV features in a classification scheme for atrial fibrillation. Adopting a training approach based on peak locations can also directly address some of the problems observed with the PPG signals in the OBF database, such as the strong diastolic peaks and second harmonic components, as in this kind of training setup the networks do not learn to replicate these undesired features. In addition, it simultaneously addresses other problems associated with PPG signals recorded from the finger, such as varying amplitude resulting from inconsistencies in skin contact, for example.

In addition to changing the train target and loss function, applying a different optimization approach could potentially lead to better results. Such approaches could include employing a different variant of the Adam optimizer or a different type of optimizer altogether, or adopting a learning rate scheduling scheme, for example.

Another very significant issue with the experimental setup presented in this thesis is the homogeneity of the data used for training the neural network models. Like all supervised methods, Self-ONNs require the train data to be sufficiently varied in order to avoid overfitting and allow the models to generalize to the wide range of input data associated with real use cases. In the experimental results presented in this thesis, the homogeneity of the train data was apparent from the poor generalization performance achieved by all the tested supervised models. As such, an important direction in the future research of applying Self-ONNs to rPPG extraction would be to train, validate and test the models on a much wider variety of relevant data which represents the situations under which the models would realistically be applied.

However, as relevant data is rather scarce, with only few public data sets available, data augmentation could also be applied as an alternative method to increase the variety of the train data. Augmentation steps such as rotation or changes in luminance, saturation or contrast represent variations that can occur in real use cases in which videos are recorded with different cameras, under different lighting conditions, and at different angles. Another potential method for overcoming the limited availability of relevant train data could be the application of self-supervised generative models for generating synthetic data samples based on real data.

It is also worth noting that in the experiments presented in this thesis, the supervised models were both trained and validated on data from OBF. Validating the models on outside data can also reduce the potential of overfitting to a certain distribution of data, improving generalization performance.

In addition to focusing on the training setup and data used, testing the performance of different Self-ONN architectures should be a direction of future research. The Self-ONN models proposed in this thesis are directly based on the CNN architectures described in [3] and [11] which provide some of the best performance achieved by supervised approaches in relevant literature. However, while the performance of different CNN architectures for different types of problems has been researched extensively, Self-ONNs have only seen a small fraction of the attention given to other neural network approaches by the research community as of yet. As such, no clear design paradigms or reference architectures for Self-ONN models have been established, and the architectures achieving the best performance for the task at hand could differ significantly from the architectures proposed in this thesis or from any typical CNN architectures seen in literature. Possible ways of altering the proposed architectures could be changing the number of layers in the network, changing the values of the nonlinearity parameters Q, removing some or all of the pooling operations, instead increasing the stride of the neural layers to reduce spatial dimensionality, or changing the kernel size or number of channels of the layers. Whether or not these changes would yield better results for the task at hand remains a question to be researched.

8. CONCLUSIONS

This thesis presented a novel approach for remote photoplethysmograph signal extraction based on self-organized operational neural networks. A literature review of existing remote PPG extraction approaches and their applications as well as a description of the recently proposed operational neural networks and self-organized operational neural networks was also presented. The efficacy of the proposed approach was evaluated on varied data, based on the accuracy of heart rate estimates calculated from the extracted remote PPG signals, and Self-ONN models with varying depth were compared against convolutional neural network models with corresponding architectures as well as a typical unsupervised rPPG pipeline.

Based on the experimental results, the Self-ONN models were not found to improve performance over their CNN counterparts based on the accuracy of HR estimates calculated from the remotely extracted PPG signals. However, the Self-ONN models were able to achieve a better fit of the train data compared to their parameter-equivalent CNN counterparts based on the learning criterion, which indicates higher learning ability. As such, the poorer results achieved by Self-ONN models on test data based on HR estimation accuracy likely stem from problems observed in the train setup as opposed to any inherent lack of suitability to the problem at hand.

The main issues with the training setup observed during the experiments stem from the problematic features in the PPG signals used in training, such as strong diastolic peaks and higher harmonic components, which the neural network models learned to replicate due to being trained on the entire PPG signal waveform. The Self-ONN models were more affected by these problems compared to CNN alternatives due to their higher learning ability. In addition, the results also demonstrate the overfitting phenomenon typical of supervised approaches, resulting from the homogeneous nature of the set of data used in training the neural network models. The phenomenon was apparent from the supervised Self-ONN and CNN models being outperformed by the tested unsupervised pipeline when evaluated on data which differs significantly from the data used in training.

Despite being unable to outperform CNN-based approaches as measured by heart rate estimation accuracy, the proposed Self-ONN approach for rPPG extraction is nonetheless a promising proof-of-concept which could be improved upon significantly — primarily by altering the training setup. If the training setup were defined so that the features which are unnecessary for the end goal of HR estimation are ignored, a Self-ONN approach could even hold potential for state-of-the-art performance due to the superior learning performance of Self-ONNs over the current state-of-the-art in CNN models, which was also demonstrated in these experiments. In future research, the performance of the proposed approach could also be improved further by employing some of the strategies for combatting overfitting discussed in the discussion chapter and exploring alternative training strategies and model architectures.

9. REFERENCES

[1] Kiranyaz S., Ince T., Iosifidis A. & Gabbouj M. (2020) Operational neural networks. Neural Computing and Applications 32, pp. 6645–6668.

[2] Kiranyaz S., Malik J., Abdallah H.B., Ince T., Iosifidis A. & Gabbouj M. (2021) Self-organized operational neural networks with generative neurons. Neural Networks 140, pp. 294–308.

[3] Yu Z., Li X. & Zhao G. (2019) Remote photoplethysmograph signal measurement from facial videos using spatio-temporal networks. arXiv preprint arXiv:1905.02419 .

[4] Álvarez Casado C. & Bordallo López M. (2022), Face2ppg: An unsupervised pipeline for blood volume pulse extraction from faces. URL: `https://arxiv.org/abs/2202.04101`.

[5] Allen J. (2007) Photoplethysmography and its application in clinical physiological measurement. Physiological measurement 28, p. R1.

[6] Elgendi M. (2012) On the analysis of fingertip photoplethysmogram signals. Current cardiology reviews 8, pp. 14–25.

[7] Tamura T. (2019) Current progress of photoplethysmography and spo2 for health monitoring. Biomedical engineering letters 9, pp. 21–36.

[8] Tamura T., Maeda Y., Sekine M. & Yoshida M. (2014) Wearable photoplethysmographic sensors—past and present. Electronics 3, pp. 282–302.

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

[10] Verkruysse W., Svaasand L.O. & Nelson J.S. (2008) Remote plethysmographic imaging using ambient light. Optics express 16, pp. 21434–21445.

[11] Sun Z., Junttila J., Tulppo M., Seppänen T. & Li X. (2022) Non-contact atrial fibrillation detection from face videos by learning systolic peaks. IEEE Journal of Biomedical and Health Informatics .

[12] Yu Z., Li X. & Zhao G. (2021) Facial-video-based physiological signal measurement: Recent advances and affective applications. IEEE Signal Processing Magazine 38, pp. 50–58.

[13] Wang W., Stuijk S. & De Haan G. (2015) A novel algorithm for remote photoplethysmography: Spatial subspace rotation. IEEE transactions on biomedical engineering 63, pp. 1974–1984.

[14] Li X., Chen J., Zhao G. & Pietikainen M. (2014) 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.

[15] Asthana A., Zafeiriou S., Cheng S. & Pantic M. (2013) Robust discriminative response map fitting with constrained local models. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3444–3451.

[16] Tomasi C. & Kanade T. (1991) Detection and tracking of point. Int J Comput Vis 9, pp. 137–154.

[17] Tarvainen M.P., Ranta-Aho P.O. & Karjalainen P.A. (2002) An advanced detrending method with application to hrv analysis. IEEE transactions on biomedical engineering 49, pp. 172–175.

[18] Welch P. (1967) The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on audio and electroacoustics 15, pp. 70–73.

[19] Soleymani M., Lichtenauer J., Pun T. & Pantic M. (2011) A multimodal database for affect recognition and implicit tagging. IEEE transactions on affective computing 3, pp. 42–55.

[20] Poh M.Z., McDuff D.J. & Picard R.W. (2010) Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. Optics express 18, pp. 10762–10774.

[21] Kwon S., Kim H. & Park K.S. (2012) 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, IEEE, pp. 2174–2177.

[22] Poh M.Z., McDuff D.J. & Picard R.W. (2010) Advancements in noncontact, multiparameter physiological measurements using a webcam. IEEE transactions on biomedical engineering 58, pp. 7–11.

[23] Balakrishnan G., Durand F. & Guttag J. (2013) Detecting pulse from head motions in video. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3430–3437.

[24] Boccignone G., Conte D., Cuculo V., d'Amelio A., Grossi G. & Lanzarotti R. (2020) An open framework for remote-ppg methods and their assessment. IEEE Access 8, pp. 216083–216103.

[25] Wang W., den Brinker A.C., Stuijk S. & de Haan G. (2017) Algorithmic principles of remote ppg. IEEE Transactions on Biomedical Engineering 64, pp. 1479–1491.

[26] De Haan G. & Jeanne V. (2013) Robust pulse rate from chrominance-based rppg. IEEE Transactions on Biomedical Engineering 60, pp. 2878–2886.

[27] Lewandowska M., Rumiński J., Kocejko T. & Nowak J. (2011) Measuring pulse rate with a webcam—a non-contact method for evaluating cardiac activity. In: 2011 federated conference on computer science and information systems (FedCSIS), IEEE, pp. 405–410.

[28] Pilz C.S., Zaunseder S., Krajewski J. & Blazek V. (2018) Local group invariance for heart rate estimation from face videos in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 1254–1262.

[29] De Haan G. & Van Leest A. (2014) Improved motion robustness of remote-ppg by using the blood volume pulse signature. Physiological measurement 35, p. 1913.

[30] Yang Y., Liu C., Yu H., Shao D., Tsow F. & Tao N. (2016) Motion robust remote photoplethysmography in cielab color space. Journal of biomedical optics 21, pp. 117001–117001.

[31] Stricker R., Müller S. & Gross H.M. (2014) Non-contact video-based pulse rate measurement on a mobile service robot. In: The 23rd IEEE International Symposium on Robot and Human Interactive Communication, IEEE, pp. 1056–1062.

[32] Heusch G., Anjos A. & Marcel S. (2017) A reproducible study on remote heart rate measurement. arXiv preprint arXiv:1709.00962 .

[33] Bobbia S., Macwan R., Benezeth Y., Mansouri A. & Dubois J. (2019) Unsupervised skin tissue segmentation for remote photoplethysmography. Pattern Recognition Letters 124, pp. 82–90.

[34] Shi J., Alikhani I., Li X., Yu Z., Seppänen T. & Zhao G. (2019) Atrial fibrillation detection from face videos by fusing subtle variations. IEEE Transactions on Circuits and Systems for Video Technology 30, pp. 2781–2795.

[35] Li X., Alikhani I., Shi J., Seppanen T., Junttila J., Majamaa-Voltti K., Tulppo M. & Zhao G. (2018) The obf database: A large face video database for remote physiological signal measurement and atrial fibrillation detection. In: 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018), IEEE, pp. 242–249.

[36] Kiranyaz S., Ince T., Iosifidis A. & Gabbouj M. (2017) Generalized model of biological neural networks: progressive operational perceptrons. In: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 2477–2485.

[37] Kiranyaz S., Ince T., Iosifidis A. & Gabbouj M. (2017) Progressive operational perceptrons. Neurocomputing 224, pp. 142–154.

[38] Everingham M., Van Gool L., Williams C.K.I., Winn J. & Zisserman A. (2010) The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88, pp. 303–338.

[39] Jain V. & Learned-Miller E. (2010) Fddb: A benchmark for face detection in unconstrained settings. Tech. rep., UMass Amherst technical report.

[40] Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.Y. & Berg A.C. (2016) SSD: Single shot MultiBox detector. In: Computer Vision – ECCV 2016, Springer International Publishing, pp. 21–37. URL: https://doi.org/10.1007%2F978-3-319-46448-0_2.

[41] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L. et al. (2019) Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32.

[42] Malik J., Kiranyaz S. & Gabbouj M. (2020) Fastonn–python based open-source gpu implementation for operational neural networks. arXiv preprint arXiv:2006.02267 .

[43] Malik J., Kiranyaz S. & Gabbouj M. (2021) Self-organized operational neural networks for severe image restoration problems. Neural Networks 135, pp. 201–211.

[44] scipy.signal.correlate — scipy v1.11.3 manual. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate.html.

[45] Loshchilov I. & Hutter F. (2017), Decoupled weight decay regularization. URL: https://arxiv.org/abs/1711.05101.

[46] Makowski D., Pham T., Lau Z.J., Brammer J.C., Lespinasse F., Pham H., Schölzel C. & Chen S. (2021) Neurokit2: A python toolbox for neurophysiological signal processing. Behavior research methods 53, pp. 1689–1696.

[47] Van Gent P., Farah H., Van Nes N. & Van Arem B. (2019) Heartpy: A novel heart rate algorithm for the analysis of noisy signals. Transportation research part F: traffic psychology and behaviour 66, pp. 368–378.

[48] Legrand N. & Allen M. (2022) Systole: A python package for cardiac signal synchrony and analysis. Journal of Open Source Software 7, p. 3832.