Efficient Blood Pressure Prediction from Photoplethysmography Signals

Project Category: Athletics and Sensing Devices

Name: Edward Kim
SUNet ID: edkim36
Department of Computer Science
Stanford University
edkim36@stanford.edu

Name: Rohan Sanda
SUNet ID: rsanda
Department of Electrical Engineering
Stanford University
rsanda@stanford.edu

1 Mentors and External Collaborators

Our mentor for this project is Parker Ruth, who is a graduate student in the Department of Computer Science at Stanford.

2 Introduction

Blood pressure (BP) has been shown to be a robust indicator of cardiovascular health and consequently forms the basis for the diagnosis of many diseases [1]. For this reason, continuous and accurate blood pressure monitoring is crucial to bringing about further advancements in personalized medicine and wearable biosensors. Currently, methods for measuring blood pressure have various shortcomings. For instance, clinical blood-pressure cuffs can lead to misdiagnosis by not measuring circadian blood pressure fluctuations [2]. Intravenous methods used in surgical settings are invasive and costly. One potential application of blood pressure monitoring systems is in smart watches – which already provide basic biometric data.

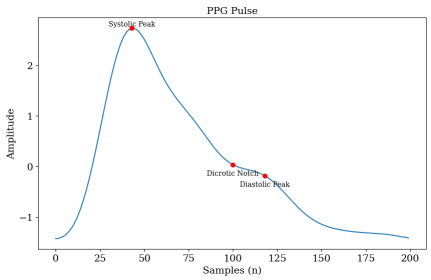
In this paper, we propose and implement a custom processing pipeline and signal regression models to predict systolic (SBP) and diastolic (DBP) blood pressure (in milligrams of Hg, represented as scalars) given a finger-tip photoplethysmography (PPG) pulse (represented as a vector). Finger-tip PPG pulses (Fig. 1a) are biosignals that correspond to volumetric variations in blood circulation. They are cheap and easy to measure – two necessary conditions for smart watches.

After preprocessing raw PPG signal data and computing the SBP and DBP from our ground truth (the arterial blood pressure waveform (ART)), we sought to understand the best way to represent each PPG pulse for regression. To answer this question, we compared the performance of three different methods of featurization using our baselines models: 1) hand-crafting physiological features, 2) embeddings from a linear or LSTM autoencoder, and 3) the PPG pulse itself. Our baseline models included Ridge Regression, SVR Regression, Adaboost Regression, and Random Forest models. We then fed the best performing featurization – the PPG pulse – into three different neural network architectures and compared performance: 1) a simple, two-layer fully connected model, 2) a 1-dimensional CNN, and 3) a CNN-ResNet model.

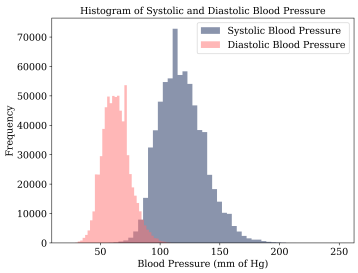
3 Related Work

Several prior studies have tried to use artificial intelligence methods to predict blood pressure using electrocardiogram (ECG) and finger-tip PPG data. Most of these papers use hand-crafted physiological features derived from PPG and ECG pulses for prediction [2] [3] [4] [5]. For instance, [2] used a large set of hand-crafted physiological features derived from single PPG and ECG pulses to estimate systolic SBP and DBP using Random Forest and multi-layer perceptron models. In [4], the authors centered their analysis on a feature called the Womersley number – a physiological feature derived from heart rate and blood viscosity – and used standard regression and Random Forest models to achieve mean absolute errors (MAE) under 9 mm of Hg. However, these methods all require additional patient data and ECG signals which are impractical for smart watches. For instance, accurately measuring ECG data requires an electrode array to be placed on the chest.

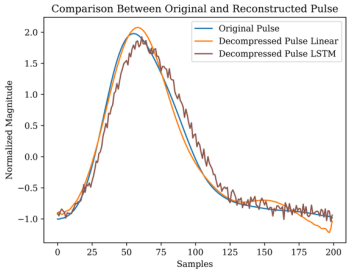
From our research, only [6] and [4] conduct experiments using only PPG-pulses to predict SBP and DBP. In [6], the authors use the first and second derivatives as well as the Fourier coefficients of each PPG-pulse as the input to their multimodal CNN-ResNet architecture to achieve moderate results – which we discuss later in this paper. We believe the ResNet architecture is a good choice for the blood-pressure prediction task due to its ability to capture local and global dependencies from skipped connections which is why we implement our own ResNet architecture in this paper. However, we disagree with the extensive processing and smoothing that [6] uses to create their dataset as this may remove relevant features for learning. Moreover, their multimodal model is quite large making it impractical for running on smart watches.



(a) A PPG Pulse with labeled critical points.



(b) Distributions of SBP and DBP labels.



(c) PPG Autoencoder reconstructions.

4 Dataset and Pre-processing Pipeline

Our raw PPG and ART signals come from the VitalDB dataset [7]. This dataset includes vitals data from 6,388 patients who underwent surgery at the Seoul National University Hospital in Seoul, South Korea. Because the time-domain biosignals in the VitalDB dataset are completely unprocessed, we have implemented a robust preprocessing pipeline from scratch to generate our dataset of PPG pulses and corresponding SBP/DBP measurements. Our pipeline differs from surveyed literature ([2], [6]) by minimally cleaning and smoothing the raw PPG and ART signal to preserve raw signal integrity.

Pipeline: Processing begins with Phase 1, where we run an initial pass through all patient recordings of ART (arterial blood pressure – our BP ground truth) and PPG to determine valid and invalid patients. We check for the percentage of NaN values, signal saturation, and unusual stochastic behavior to determine recording validity.

In Phase 2, we process the selected patient recordings (sampled at 125 Hz) from Phase 1. We first apply linear interpolation and a fourth-order Butterworth bandpass filter with a pass-band from 0.5 Hz to 8 Hz to remove baseline wander and high-frequency instrument noise from the PPG signal ([6]). The ART signal is not filtered to preserve the raw blood pressure values. We then remove chunks of the PPG and ART recordings that are highly stochastic or have physiologically impossible values. We then extract corresponding chunks of valid PPG and ART data that are at least 5 minutes long. We then apply the Bishop peak detection algorithm to locate the peaks the troughs of the PPG signal, and use this data to perform pulse segmentation. All signals are resampled or zero-padded to a length of 200 samples (which is the longest pulse length). For each pulse, the corresponding systolic and diastolic blood pressure is extracted using the peak and trough information from the ART signal. Pulses are then normalized using the mean and standard deviation of the dataset.

Dataset Summary Statistics: Our pipeline produced two datasets: one that used resampling and the other that zero-padded. We produced these two datasets as we were unsure which preprocessing choice would be better for our featurization methods. For instance, resampling signals would enable better physiological feature extraction by not altering the temporal representation of the signal. However, zero-padding would not distort the spectrum of the signal.

After processing the VitalDB data, we had a dataset where the examples were 200-point vectors corresponding to PPG pulses and the labels were a tuple of systolic and diastolic blood pressure. Both of the two datasets had 718,035 high-quality PPG pulses and corresponding SBP and DBP measurements from 300 patients (Fig. 1b). We employ a 70-20-10 training, validation, test split in our models. The summary statistics are as follows: the average age of patients is 59.98 years, average BMI is 22.91, and the gender split is 59.49% male and 40.51% female. We omit patients undergoing any surgeries related to the cardiovascular system from our processing pipeline so our results generalize to healthy people.

5 Methods/Experiments

We broke our project up into two main parts: 1) that considered the best way to encode/represent the PPG pulse for blood-pressure prediction, and 2) that compared various neural network architectures to optimize performance.

5.1 Feature Engineering

We began by first considering the best way to featurize the PPG pulses. We tested three possible techniques: 1) deriving a mix of physiological features from each pulse, 2) using lower-dimension embeddings found from an autoencoder, and 3) using the raw PPG signal. To our knowledge, much of the available literature considers only features extracted from ECG or PPG signals.

Mixed Features: We considered morphological features, PCA features, and frequency-dependent features - mostly inspired from [2] and [6] - on a resampled input dataset. The morphological features extracted were based on the highest (systolic)

peak of each segmented PPG pulse as well as the peak and valley intensities. Another type of morphological feature came from using the bandwidth of each pulse at different heights. The frequency-dependent features primarily used time measurements between morphological aspects of the signal such as the peak and the peak of the first derivative, well as the PSD (Power Spectral Density) and FFT (Fast Fourier Transform) of the signal. In total, there were 45 features extracted, with 5 PCA components and 40 morphological and frequency-dependent features. We used the SelectKBest algorithm to find that the 10 most relevant features were morphological, most being temporal bandwidth features.

Autoencoder Embeddings: Our second featurization approach involved using an autoencoder, which is a kind of artifical neural network often used for unsupervised learning tasks. An autoencoder uses an encoder to map an input to a lower-dimensional space of embeddings. The autoencoder then reconstructs the original input from the embeddings using a decoder, and minimizes the difference between the original and reconstructed data (Fig. 1c). In doing so, the neural network learns to extract and encode the most significant features of the data as the embeddings. We implemented a linear autoencoder and Long Short-Term Memory (LSTM) autoencoder and set the encoded dimension to 45 to match the number of features for the mixed-feature method. Due to convergence issues with the LSTM, we quickly abandoned this model and instead used the linear autoencoder (2-layer encoder and 2-layer decoder with ReLU activation).

Raw Signal: Our third approach was simply using the processed PPG pulse (which we call Raw Signal) as the input data without any feature extraction.

Baseline Models: We used Ridge Regression, Support Vector Regression (SVR), Adaboost Regression, and Random Forest (RF) Regression. Ridge regression was chosen to minimize overfitting and achieve better model generalization on the test set. Ridge Regression also generally works well with high-dimensional data, which matched our needs. SVR is similar to a Support Vector Machine but it instead tries to fit a hyperplane that has minimal error tolerance; it has been shown to be effective in performing regression for high-dimensional, non-linear tasks like our case. Adaboost regression was used following the recommendation of [4] and involves ensembling multiple weak, shallow, decision trees to achieve good performance. Finally, a Random Forest was used as a more complex regressor. Similar to Adaboost regression, Random Forest is an ensemble model and utilizes multiple decision trees for prediction.

Note that for our baselines we used a random subset of 100,000 samples from our processed datasets to improve runtime performance as Sklearn models cannot be optimized using a GPU. We also did not implement hyperparameter-tuning since though it may have been improved performance slightly, we believed that simpler models such as Ridge Regression and Adaboost Regression lacked the complexity to effectively express our non-linear problem. Also, the runtime for hyperparameter-tuning would have been too large.

5.2 Neural Networks

Unlike our baseline models, our neural networks are multi-output (predicting SBP and DBP simultaneously). Since the SBP and DBP are closely related, we believed the neural networks would be able to take advantage of this relationship. The multi-output neural networks used a combined loss function equal to the sum of the Mean Squared Error (MSE) of SBP and DBP. While we did not weight either loss term more than the other, due to the smaller magnitudes of the DBP values, there was an inherent weighting towards the SBP loss. This effect was desired since, as shown in 1b, the SBP distribution has a large variance and thus requires further tuning. Note, we used a training and validation batch size of 64 as this was empirically found to provide good results while also being efficient to run.

Fully-Connected Neural Network (ANN): We first developed a two-layer, fully-connected, multi-output neural network (which we call ANN) to see if we could improve upon the baseline models. We tuned several parameters, including the number of hidden layers (1, 2, 3), activation function (ReLU, TanH, LeakyReLU), optimizer (Adam, SGD, RMSprop from Pytorch), and learning rate. Later, we added hyperparameter tuning for the L1 regularization coefficient to counteract overfitting. We discuss the Grid Search hyperparameter tuning results later.

1D-Convolutional Neural Network (CNN): 1D-convolutional layers work by computing the element-wise multiplication between a sliding filter and the signal, generating an output feature map. By applying convolution operations, the CNN can learn to detect and capture different patterns and structures in the input data. We add batch normalization to normalize the mean and variance of mini-batches in each layer – thus stabilizing the model. Max-pooling helps to reduce the spatial dimensions of our model.

Building on the 1D-CNN employed by [8] for PPG-pulse based emotion recognition, we develop a multi-output CNN for SBP and DBP prediction (which we call CNN). Our model has two convolutional layers with kernels of size 3 and 10 – capturing both local and broader variations in the signal – as well as batch normalization and two max-pool layers. We use Grid Search to optimize the choice of learning rate, activation function, and L1 regularization parameter.

ResNet + 1D-CNN (ResNet): The ResNet-CNN architecture incorporates skipped connections into a normal CNN and has been shown to be effective in 1D-signal regression tasks ([9]). This is because skipped connections can allow the model to

learn deviations from an identity mapping. Thus, we augment our CNN with four residual blocks that each consist of two convolutional layers (kernel = 3) and batch-norms with one skipped connection. We call this model ResNet. Again, we tune the learning rate, activation function, and L1 regularization parameter.

6 Results/Discussion

Following the convention observed in the literature, we used Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 score for evaluating our signal regression, where:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \qquad MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \qquad R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

In the above equations, y_i represents the observed values, $\hat{y}_i$ represents the predicted values, and $\bar{y}_i$ represents the mean. Since MAE is the most intuitive metric, we use this as our primary evaluation metric.

Table 1: Baseline Results

Model	SBP Results						DBP Results					
	Featurized			Raw Signal			Featurized			Raw Signal		
	MAE	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$
Random Forest	13.957	17.940	0.156	12.178	16.054	0.327	7.816	10.097	0.214	7.282	9.551	0.300
SVR Regression	15.361	19.534	0	**11.802**	15.977	0.333	9.019	11.381	0.001	**7.019**	9.470	0.312
Adaboost Regression	14.532	18.436	0.109	13.898	17.461	0.204	8.399	10.500	0.150	8.402	10.508	0.153
Ridge Regression	15.052	19.310	0.022	15.077	19.245	0.033	8.624	10.949	0.076	8.875	11.277	0.024

Table 2: Raw Signal on Neural Network Results

Model	Train Results						Test Results					
	SBP			DBP			SBP			DBP		
	MAE	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$	**MAE**	**RMSE**	$\boldsymbol{R^2}$
ANN	10.908	14.332	0.440	6.922	8.948	0.365	11.124	14.668	0.411	7.029	9.07	0.348
CNN	10.040	13.341	0.514	6.310	8.270	0.457	**10.125**	**13.519**	**0.506**	**6.396**	**8.40**	**0.457**
ResNet	10.286	13.604	0.495	6.453	8.429	0.437	10.287	13.724	0.489	6.499	8.525	0.450

6.1 Feature Engineering Results

The results for our SBP and DBP predictions using the featurized and raw PPG signal are shown in the Table 1. Since we did not implement formal hyperparameter tuning for our baseline models, we had a 70-30 training-test split. Following the advice of [4], we use a Radial Basis Function kernel for SVR. We also used 100 estimators for our Random Forest Model, and 100 estimators of max depth 10 for our Adaboost Regressor. We find that the Raw Signal is the best way to represent the PPG pulse for our models, likely because it contains the most signal information.

We found that the resampled dataset worked better for the mixed features method, while the zero-padded dataset worked better for the raw signal. Thus the results in Table 1 use the resampled dataset for Featurized results and the zero-padded for the Raw Signal results. This is likely cause resampling can introduce a non-zero phase shift to the signals – which may affect a model's evaluation of the raw signal while not affecting the hand-crafted features of which many are phase independent.

The linear autoencoder embedding did not perform well on the baselines leading us to not continue with this line (we did not run DBP analysis after seeing the poor results for SBP). For instance, the SBP for the linear autoencoder embeddings had an MAE of 15.897. While more time and resources are needed to investigate exactly why this was the case, we hypothesize that because the reconstructed pulses are typically phase-shifted compared to the original signal (Fig. 1c), the embeddings are not representative enough of the original signal to be effective. Perhaps more training would improve the fit and reduce the phase shift – though we did notice our loss curve flat-lining after only 15 epochs.

It should be noted that our results match those in literature we surveyed. In [4] the authors achieve an MAE of 11.53 mm Hg and R^2 value of 0.292 for SBP prediction using a Random Forest with featurized PPG data. [6] achieves a SBP MAE of 18.34 mm Hg using a Random Forest with PPG features. Thus, while better MAE's can be achieved with ECG data, our baseline models seem to match those in the literature that only use PPG. Furthermore, the raw signal approach shows lots of promise. When the sample size was increased to 700,000 samples, we achieved an MAE of 9.943 mm Hg and R^2 value 0.495 using a Random Forest. Our DBP results also were similar to those in published literature [6], [4], [7].

6.2 Neural Networks

Based on our results from Section 6.1, the raw signal using the zero-padded dataset performed the best leading us to use this dataset for our neural networks.

Hyperparameter Tuning: For the ANN, after tuning its hyperparameters we found that a two layer model, TanH activation function, Adam optimizer, learning rate of $\alpha = 0.0005$, and regularization constant of $\lambda = 0.001$ performed best. Note that the L1 regularization was applied to successfully combat overfitting in all models – evidenced by a growing gap between the validation and training loss curves. For the CNN, we found that the TanH activation function, Adam optimizer, $\alpha = 0.0005$, and $\lambda = 0.01$ worked the best. For the ResNet, we found that TanH activation function, Adam optimizer, $\alpha = 0.001$, and $\lambda = 0.005$ worked the best. These findings make intuitive sense. The smooth, non-linear activation function of TanH may be better able to capture non-linearities patterns between PPG and blood pressure than ReLU – explaining why [2] used the TanH function in their fully-connected model. While the TanH activation function is prone to the exploding gradient problem, we manually inspected some random gradients in our model to ensure this was not the case.

Results: Our results for the neural networks are also shown in Table 2. We find that our best performing model is the CNN, which yields an SBP MAE of 10.1 and DBP MAE of 6.396 on our test set. However, the ResNet achieves the lowest combined loss while the CNN and ANN have similar combined losses Fig. 2. This performance is interesting, as it indicates that the skipped connections and additional convolution layers in the ResNet are not able to discern more information about the relationship between BP and PPG pulses - contradicting our hypothesis that this added complexity would improve results. This may because the model was too complex and overfitted too much, leading us to use large L1 regularization coefficients. Thus, while all models make significant improvements on our baseline results, further model optimization is necessary.

To localize the source of error in our models, we plotted the predicted versus actual SBP and DBP values in Fig. 3. We observed that all three models have similar point spreads that follow a trend line shallower than the ideal 45^o line – indicating that there are some common underlying sources of error across models. Additionally, all models seem to perform poorly in predicting high SBP and DBP values – perhaps because of our imbalanced dataset where the SBP and DBP distrbutions seem to follow a normal distribution (Fig. 1b).

7 Conclusions

In closing, this paper takes a comprehensive approach to analyzing the BP prediction problem for future smart-watch applications. We begin by comparing three featurization techniques proposed in the literature: 1) hand-crafted physiological features (used by [2], [3], [4]) linear autoencoder embeddings, and 3) the processed PPG pulse (used by [6]). We find that the processed PPG pulse (denoted "raw signal") performs the best on our baseline models, achieving a best SBP MAE of 11.8 and DBP MAE of 7.0 using SVR Regression. Next, we compare three different neural network architecture and find that all improve our results significantly. We find that out of our ANN, CNN, and ResNet models, the CNN performs the best – achieving a SBP MAE of 10.1 and DBP MAE of 6.3. Moreoever, our models are not excessively large. These results match existing literature on the topic.

While these results are not precise enough to be useful for medical diagnosis, they indicate that there is more work to be done. More model tuning and an investigation of error sources would be fruitful. Future work using traditional machine learning could investigate the use of models that take advantage of the normal distributions of DBP and SBP. Ultimately, our results indicate that there may be fundamental limits in the statistical relationship between BP and PPG. Thus, deep learning approaches (such as more advanced CNNs and LSTMs) that incorporate other biometric data will be useful. Additionally, exploring the development of patient-specific BP prediction models (not population models, like our approach) may yield better results.

8 Contributions

Rohan developed the processing pipelines, neural network models and associated infrastructure, and the LSTM autoencoder. He wrote sections 2, 3, 4, 5.2, 6.2, and 7 and made the poster and figures.

Edward worked on the feature extraction/selection, linear autoencoder, and baseline models. He wrote sections 6.1 and 7.1.

Parker provided qualitative feedback on our results and helped us find the VitalDB dataset. He also provided the *um()* plotting function and ideas for detecting physiologically impossible pulses that were used in the our *processor_phase1()* function in Stage 1 of the processing pipeline.

Github:
https://github.com/rohansanda/bp_prediction_cs229

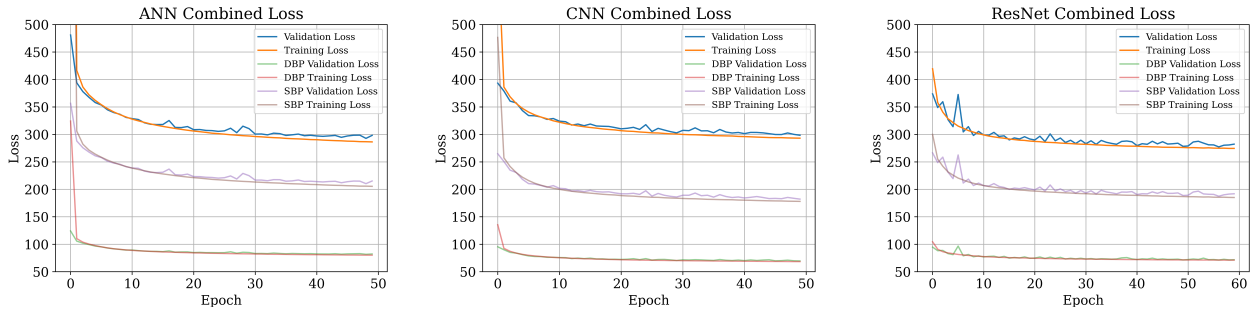


Figure 2: The loss curves for our three neural network architectures.

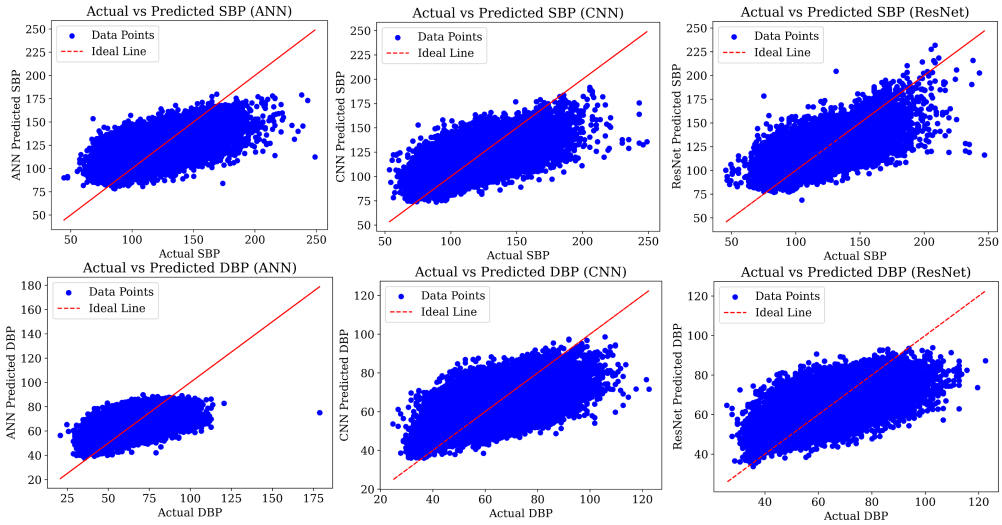


Figure 3: The predicted vs. actual curves for SBP (top row) and DBP (bottom row) for all three models.

References

[1] Seamus P. Whelton, John W. McEvoy, Leslee Shaw, Bruce M. Psaty, Joao A. C. Lima, Matthew Budoff, Khurram Nasir, Moyses Szklo, Roger S. Blumenthal, and Michael J. Blaha. Association of Normal Systolic Blood Pressure Level With Cardiovascular Disease in the Absence of Risk Factors. *JAMA Cardiology*, 5(9):1011–1018, 09 2020.

[2] Seungman Yang, Jangjay Sohn, Saram Lee, Joonnyong Lee, and Hee Chan Kim. Estimation and validation of arterial blood pressure using photoplethysmogram morphology features in conjunction with pulse arrival time in large open databases. *IEEE Journal of Biomedical and Health Informatics*, 25(4):1018–1030, 2021.

[3] Clémentine Aguet, Jérôme Van Zaen, João Jorge, Martin Proença, Guillaume Bonnier, Pascal Frossard, and Mathieu Lemay. Feature learning for blood pressure estimation from photoplethysmography. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 463–466, 2021.

[4] Umapathy Mangalanathan V. Jeya Maria Jose M. Anand Geerthy Thambiraj, Uma Gandhi. Investigation of the effect of womerseley number, ecg and ppg features for cuff-less blood pressure estimation using machine learning. *Biomedical Signal Processing and Control*, 60, 2020.

[5] Shiqi Tang Keke Qin, Tao Zhang. Machine learning and deep learning for blood pressure prediction: a methodological review from multiple perspectives. *Artificial Intelligence Review*, 2022.

[6] Nejc Mlakar Slapničar, Gašper and Mitja Luštrek. Blood pressure estimation from photoplethysmogram using a spectro-temporal deep neural network. *Sensors*, 9(15), 2019.

[7] HC. Lee, Y. Park, and S.B. Yoon. Vitaldb, a high-fidelity multi-parameter vital signs database in surgical patients. *Nature Scientific Data*, 9(279), 2022.

[8] Dong Sung Pae Myo Taeg Lim Dong Won Kim Tae Koo Kang Min Seop Lee, Yun Kyu Lee. Fast emotion recognition based on single ppg signal with convolutional neural network. *Applied Sciences*, 9, 2019.

[9] Ben Othman Soufiene Faris A. Almalki Hedi Sakli Obaid Ali Mustapha Najjari Nizar Sakli, Haifa Ghabri. Resnet-50 for 12-lead electrocardiogram automated diagnosis. *Computational Intelligence and Neuroscience*, 2022.