POLITECNICO DI TORINO

Master degree course in Computer Engineering
Artificial Intelligence and Data Analytics

Master Degree Thesis

Enhancing PPG-Based Heart Rate Estimation combining Data Augmentation and Model Pre-training

Supervisors
Prof. Daniele Jahier Pagliari[1]
Dr. Alessio Burello[1,3]
PhD. student Luca Benfenati[1]
Dr. Xiaying Wang[2]
Prof. Luca Benini[2,3]

Candidate
Sofia Belloni

1. Politecnico di Torino, 2. ETH Zurich, 3. Università di Bologna

Academic Year 2023-2024

Abstract

Accurate heart rate (HR) estimation from photoplethysmographic (PPG) signals is critical for health monitoring applications, such as cardiovascular disease management, fitness and activity tracking, and detecting arrhythmia or other heart-related abnormalities. Without subjects' movements and with personalized subject data, nowadays HR- estimation can be considered a "solved problem", even with low-cost commercial smartwatches. However, in a more general and realistic scenario of daily life, this task remains challenging due to motion artefacts (MA) and inter-subject variability. This thesis addresses these challenges by proposing a series of improvements to existing neural network architectures for more robust and accurate HR estimation.

We first explored different Deep Learning approaches, starting with a Masked Autoencoder for unsupervised feature learning. While this model provided an initial reference, it did not meet the desired performance in HR estimation. Subsequently, we focused on improving the state-of-the-art PULSE (Ppg and imU signal fusion for heart rate Estimation) architecture, a model designed specifically for HR estimation from PPG signals. To better capture localized features, we replaced the dilated convolutions in the original model with standard convolutions, increasing the filter size and eliminating the need for dilation.

Building on our new architecture, we introduced a pre-training step where the model task is the reconstruction of PPG signals. To implement this, we restructured the PULSE model into an encoder-decoder architecture. Specifically, we replaced the final two linear layers with a decoder that mirrors the encoder architecture, utilizing both the multi-headed attention mechanism and transposed convolutional layers to reconstruct the input signal from the latent space. Inspired by U-Net, we also introduced two skip connections between the corresponding encoder and decoder layers to retain spatial information during the up-sampling process.

As a last step, to further improve model robustness and generalization, we applied a variety of data augmentation (DA) techniques to expand the training data, particularly targeting heart rate variability. These augmentations, inspired by a recent paper on improving PPG-based HR monitoring with synthetically generated data, proved highly effective in enhancing model performance, especially for subjects with atypical beats-per-minute (BPM) ranges. In our experiments, we applied the pre-training on the PULSE model using data from the public WESAD dataset and from a new un-labelled dataset coming from West Attica University, to form a comprehensive dataset of

490,000 samples. The model has been finally evaluated on the PPG-Dalia dataset, the widest labelled dataset to compare with state-of-the-art models, using the Leave-One-Subject-Out (LOSO) protocol.

In conclusion, our proposed enhancements to the PULSE model led to a 12.4% reduction in MAE on the PPG-DaLiA dataset, achieving a final MAE of 3.53 compared to best state-of-the-art competitor, PULSE, that achieves 4.03 BPM of MAE with similar model complexity. Note that most of our enhancements are applied during training, therefore not modifying significantly the complexity of the model inference, usually executed on a low-power edge device such as a smartwatch. This work sets a new benchmark in HR estimation accuracy from PPG signals, demonstrating the power of pre-training and data augmentation techniques in improving the performance of an already state-of-the-art architecture.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Heart rate (HR) is a key indicator of cardiovascular health and can provide important information about an individual's physical and mental state. Continuous HR monitoring has become essential not only for patients with medical conditions, but also for healthy individuals interested in fitness and general well-being [1]. Early wrist-worn HR monitoring devices used a separate chest band, equipped with Electrocardiogram (ECG) sensor. However, this solution is very expensive and uncomfortable to wear on a daily lives, even though it provides accurate results. In recent years, the use of photoplethysmography (PPG) for heart rate estimation has grown significantly due to the popularity of wearable devices such as Apple Watch [2] and Fitbit [3]. In fact, they have become increasingly popular due to their convenience, portability, and ability to collect and process various types of data. Wrist-worn wearable devices have made continuous access to this information possible, improving users' quality of life through non-invasive HR monitoring. PPG sensors use light-emitting diodes (LEDs) and photodiodes to measure blood volume changes in the microvascular bed of tissue, estimating HR in real time [4]. While emitting light onto the skin, the photodiode detects changes in light intensity caused by blood flow: the greater the change in blood volume, the greater the attenuation of light picked up by the photodiode [5]. This relationship allows the peaks of the PPG signal to be associated with the user's heart rate, making this method a reliable option for HR tracking. HR estimation from PPG signals, however, still presents challenges. Under ideal conditions, without subjects' movements and personalised subject data, HR estimation can be considered a 'solved' problem, even using low-cost commercial devices. However, in realistic scenario of daily life, body movements

generate motion artefacts (MA) that alter the signal and compromise the accuracy of HR estimation. These artefacts are caused by variations in sensor pressure on the skin or the infiltration of ambient light between the photodiode and the wrist, which introduce variability and noise into the PPG signals. Common practices utilize filtering approaches that correlate the motion data coming from acelerometers and the PPG signal using classical signal processing techniques, such as adaptive filtering or peak detection to cancel out the noise and eventually remove the MAs [24].

In recent years, various deep learning approaches have been explored to improve HR estimation from PPG signals. For example, models based on convolutional networks (CNNs), which exploit their ability to extract spatial features from signals, and recurrent networks (RNNs) have been implemented [6]. Furthermore, the publication of large datasets for monitoring heart rate in the presence of motion artefacts has accelerated the spread of new HR estimation model. The most common datasets in this field are PPG-DaLiA [6] and WESAD [7]. PPG-DaLiA is a large dataset that collects PPG and accelerometer data during the daily activities of 15 subjects, with the aim of compensating possible MA and accurately estimating HR in real-life contexts. WESAD, on the other hand, is a multimodal dataset designed for the study of stress and HR monitoring, containing data collected from PPG, ECG and accelerometer sensors on different subjects during specific activities. However, one of the biggest obstacles in this field is the difficulty of finding labelled data. Labelling is a long and onerous process that requires considerable time and resources. This problem is particularly evident in the context of PPG data where, on the other hand, the collection of unlabelled data is relatively straightforward. In fact, unlabelled PPG signals can be easily collected in large quantities because they do not require complex human supervision during collection. The opportunity to have a huge amount of unlabelled data available is a very important resource that can be exploited, for example, using self-supervised learning (SSL) techniques. The main idea of SSL is to define an unsupervised task on a large unlabelled dataset, with the aim of capturing the underlying structure and patterns of that specific type of data, in order to learn a data representations as generally as possible. Once this *pre-training* phase is completed, the general representations obtained can be used to *fine-tuned* the model in specific downstream tasks, which are usually tasks for which a relatively small amount of labelled data is available. Although the SSL paradigm is widely used in other fields such

as computer vision and natural language processing [28], the use of SSL applied to PPG data is still a relatively unexplored area. This is one of the main strengths of our research: we are among the first to implement SSL techniques for PPG data analysis, reducing the need for HR labelling.

In this context, a significant contribution of this thesis is the use of a new dataset provided by West Attica University, containing a large amount of unlabelled data collected from subjects without heart disease and mainly at rest. The integration of this data is an opportunity to improve the performance of HR estimation models through an effective pre-training. For this reason, we propose an innovative approach that integrates self-supervised learning and data augmentation techniques, inspired by [20] to enhance HR estimation performance of deep learning models. While our method is mostly orthogonal to the specific deep neural network selected, we assess its performance using PULSE (**P**pg and im**U** signa**L** fu**S**ion for heart rate **E**stimation pulse), the state-of-the-art deep learning model for HR estimation from PPG signals on PPG-DaLiA. PULSE integrates temporal convolutions and a feature-level Multi-Head Cross-Attention (MHCA) module to improve accuracy and ensure greater interpretability of results. By incorporating a pre-training phase alongside augmentation techniques, we enhance the robustness and accuracy of PULSE without adding to the inference complexity, making it well-suited for deployment on low-power edge devices such as smartwatches. We are the first to introduce a self-supervised pre-training step by restructuring the selected deep learning model into an autoencoder-like framework, inspired by U-Net [27]. During this first training phase, the autoencoder-like architecture is used to learn a compressed and meaningful representation of PPG signals, improving the model's ability to recognise relevant patterns even in the presence of noise or signal variability. After pre-training the model is fine-tuned on PPG-dalia reaching state-of-the-art performance for HR estimation. Additionally, we explored different data augmentation (DA) techniques to further increase the variability and representativeness of the data. These techniques were applied to create a dataset that better reflects the possible patterns that may be encountered in real-life contexts. Although the available datasets contain a large amount of data, they were collected from a relatively small number of subjects, limiting their ability to represent a heterogeneous population in terms of age, gender, physical condition etc. [6]. The use of DA allowed a greater variety of conditions to be simulated, thus improving the robustness of the model in real-life scenarios.

To conclude, the contributions of this thesis can be summarised as follows:

- **Modification of the PULSE architecture**: we replaced dilated convolutions with standard convolutions, increasing the filter size and removing dilation, enabling the model to focus on more localized features and improving efficiency.

- **Introduction of a new unlabelled dataset:** this dataset has been collected by West Attica University and it includes the PPG and tri-axial accelerometer data.

- **Introduction of a pre-training step**: we restructured the PULSE model into an autoencoder-like framework, inspired by U-Net [27]. This allowed us to pre-train the model on WESAD and the newly collected dataset from West Attica University, mentioned above. The pre-training focuses on PPG signal reconstruction, improving the model's ability to capture critical signal characteristics, thereby enhancing HR estimation accuracy.

- **Integration of data augmentation**: we applied data augmentation techniques, inspired by [20], to expand the pre-training dataset, significantly improving the model's generalization and robustness to noisy inputs.

It is important to underline that, since we worked only on the training phase, our modifications do not affect significantly the inference time. In fact, the complexity of inference remains mostly unchanged, making it suitable for deployment on low-power edge devices like smartwatches.

This thesis is structured as follows:

- Chapter 2 explains the background, providing an overview of the fundamental theoretical concepts, including PPG signals, HR estimation and deep learning techniques used in the context of this thesis.

- Chapter 3 presents a review of the existing literature, exploring the main approaches and models developed for HR estimation from PPG signals.

- Chapter 4 discusses the different architectures and methods tested, including the pretraining and fine-tuning protocols and the data augmentation techniques implemented.

- Chapter 5 presents and analyse all the experiments and the results obtained during the work.
- Chapter 6 addresses some final considerations on the results and on the overall project, as well as some comments on possible future works.

Chapter 2

Background

2.1 HR estimation

Heart rate (HR) represents the number of heart beats per minute (BPM) and it is an essential physiological measurement as it is one of the most common and significant indicators of an individual's cardiovascular health, indicating how efficiently the heart pumps blood throughout the body. Usually, adults have a resting HR ranging from 60 to 100 BPM [8]. However, it can vary in response to several factors, including physical activity, emotional state, posture and general health. HR estimation is crucial in many applications, from fitness to health monitoring, to assess physical condition and detect any anomalies. HR estimation commonly involves calculating the time gap between two consecutive heartbeats. One of the most accurate and common methods of measuring heart rate is the Electrocardiogram (ECG). The ECG measures the electrical activity of the heart through electrodes placed on the skin at specific points on the body. A notable limitation of ECG is its invasiveness and the necessity of electrode placement, which may not be practical for continuous daily monitoring [5]. As a result, wrist-worn heart rate monitors have gained popularity in recent times. These devices typically use PPG to monitor heart rate, which is now widely used in most commercial wearable devices. Nevertheless, estimating heart rate from PPG signals presents more challenges compared to traditional ECG data [6].

2.1.1 PPG signals

A photoplethysmogram is an optical measurement that uses Light-Emitting Diodes (LEDs) and a photodetector at the surface of the skin to detect variations in blood volume [24]. The two most common LEDs are red and infrared (IR), which exhibit distinct absorption properties in the bloodstream.

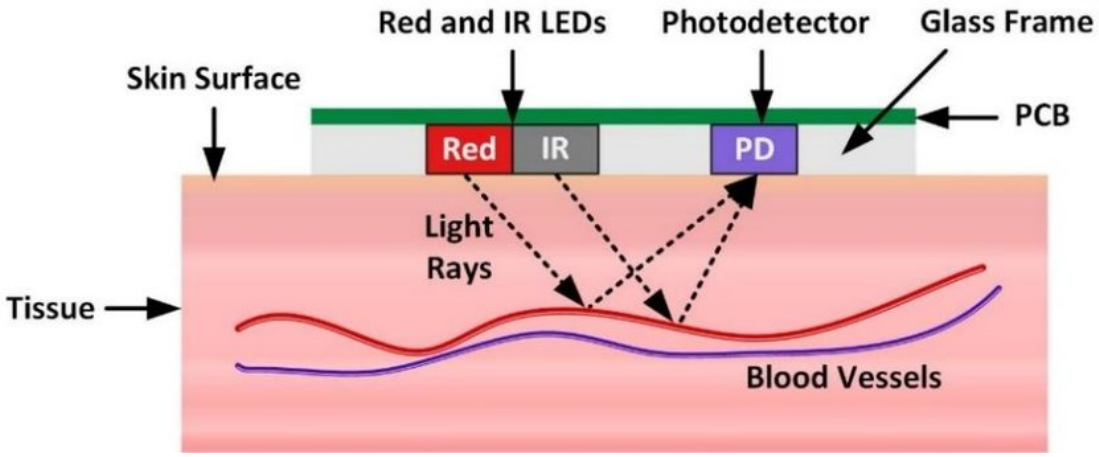


Figure 2.1: Working principle of a reflection-type PPG sensor [21].

Figure 2.1 illustrates the working principle of a PPG sensor. The photodetector captures light and it is used to estimate blood volume changes. The PPG signal is widely used for HR estimation due to its non-invasive nature, ease of acquisition and integration into wearable devices such as smartwatches and fitness trackers. The signal is composed of a pulsatile component ("AC"), which reflects changes in blood volume synchronized with each heartbeat, and a slowly varying basis ("DC"), which includes lower frequency components due to breathing, to the activity of the sympathetic nervous system and the regulation of body temperature. The first phase is primarily concerned with systole, and the second phase with diastole and wave reflections from the periphery [4]. HR can be estimated by analyzing the repetitive spikes of the AC signal, which correspond to heartbeats. The PPG signal is typically represented as a cyclic waveform in which each cardiac cycle appears as a peak, as shown in Figure 2.2.

However, PPG signals are often contaminated by noise and artifacts, such as body motion and changes in sensor pressure. They are known as MA and they complicate signal analysis, making HR estimation a complex task. In fact, MA can cause irregularities in the PPG waveform, including fluctuations

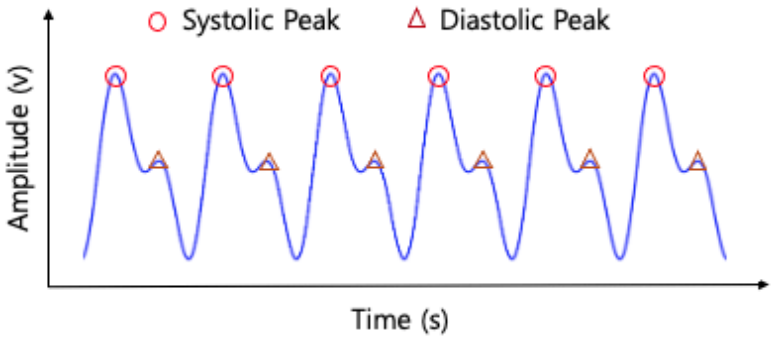


Figure 2.2: Typical PPG signal form [9].

in the signal amplitude and shape, as represented in Figure 2.3. As a result, the use of PPG signals affected by artifacts may limit the accuracy of HR estimation. For this reason, they represent one of the main challenge in this field.

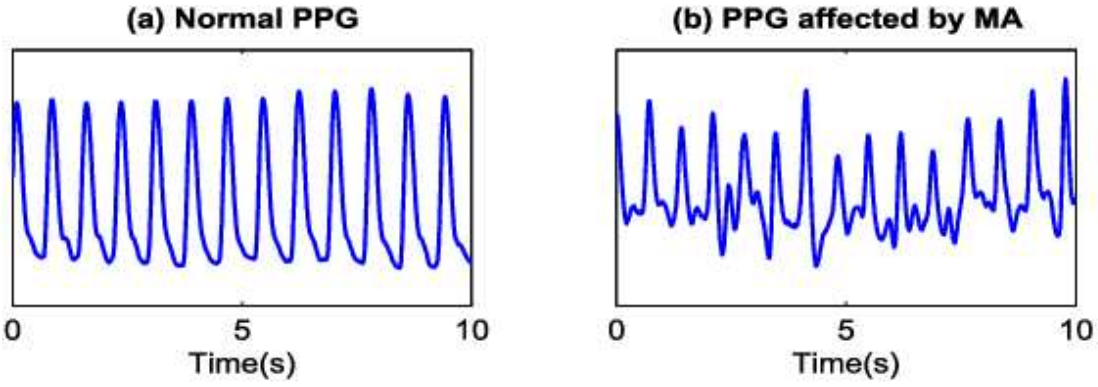


Figure 2.3: The Photoplethysmography signals. (a) is the normal PPG signals. (b) is the PPG signals affected by motion artifacts during movement [10].

2.1.2 Accelerometer data

Accelerometer data is used to detect body movements and provide additional information that helps improve heart rate estimation from PPG signals. An accelerometer measures acceleration along one or more axes and can identify intentional movements, tremors, or vibrations. In combination with PPG data, the accelerometer data helps to identify and correct segments of the

PPG signal that are altered by artefacts. By integrating this information, it is possible to mitigate the effect of motion noise, improving the accuracy of estimates. Initial approaches for MA removal use adaptive filters or techniques based on spectral analysis, which exploit the correlation between acceleration signals and noise in the PPG signal to clean them from artefacts.

2.2 Deep Learning approaches

From the past, several algorithms have been used to estimate HR from PPG signals in the presence of MA, including traditional machine learning models such as adaptive filters and signal processing techniques. More recently, deep learning has become a powerful technology, capable of achieving state-of-the-art performance in several fields. Deep learning-based approaches have shown promising results in improving the accuracy and robustness also of HR estimation by exploiting the ability of these models to learn and generalize from large amounts of data. This chapter explores some of the main models used in Deep Learning approaches for PPG-based HR estimation..

2.2.1 Temporal Convolutional Neural Network

Temporal Convolutional Neural Networks (TCNs) are a type of neural network designed specifically to handle sequential data. They consist of dilated, causal 1D convolutional layers with the same input and output lengths [11]. As we sad, *causal convolutions* are one of the fundamental characteristics of TCNs as they do not allow information from the future to influence predictions in the past. This ensures that the model respects the temporal sequence of events. On the other hand, a *dilated convolution* allows to expand the receptive field of the network without having to increase the number of layers. The concept is to introduce a space between the convoluted elements, allowing to "look" further into the past. A convolutional layer in a TCN is then formulated as:

$$y_m(t) = \sum_{i=0}^{K-1} \sum_{l=0}^{C_{in}-1} x_l(t - d \cdot i) \cdot W_{l,m}^{(i)}$$

where x and y are the input and output feature maps, t and m are the output time-step and channel, respectively, W represents the filter weights, C_{in} is the number of input channels, d denotes the dilation factor, and K is the filter size [24]. Thanks to these characteristics, TCNs offer many advantages in the analysis of sequential data with respect to other types

of models such as Recurrent NNs. First of all, convolutions can be run in parallel, improving efficiency compared to other models. Another important feature is the flexibility of the receptive fields: by increasing the number of layers, the size of the filters or the dilation factor, the length of the network memory can be adjusted. Finally, TCNs avoid the problem of explosive or vanishing gradients, as they do not use recurrent iterations.

2.2.2 Autoencoder

An autoencoder is a type of artificial neural network used primarily for unsupervised learning, designed to efficiently compress (*encode*) the input data so that the original input can be reconstructed (*decoded*) from this compressed representation [12]. The main goal is to minimize the difference between the input ($\mathbf{x}$) and the output ($\mathbf{x}'$). The difference between the original and reconstructed data is quantified using a loss function, which for an autoencoder is often defined as:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_i')^2.$$

So, minimizing the loss means minimizing the discrepancy between $\mathbf{x}$ and $\mathbf{x}'$. An autoencoder consists of 3 components: encoder, code and decoder (Figure 2.4). The encoder compresses the input into a more compact representation

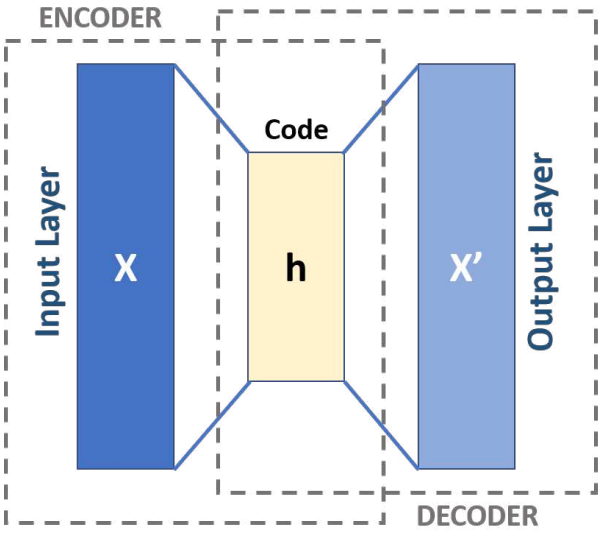


Figure 2.4: Schema of a basic autoencoder.

called *code* or *latent space*. The encoder's goal is to reduce the size of the

input while preserving the most important features. This can be expressed mathematically as:

$$\mathbf{z} = f_{\text{enc}}(\mathbf{x}) = \sigma(\mathbf{W}_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}})$$

where $\mathbf{W}_{\text{enc}}$ and $\mathbf{b}_{\text{enc}}$ are the weights and biases of the encoder, and σ is a non-linear activation function. The decoder attempts to reconstruct the original input from the compressed representation. It is symmetrical to the encoder in order to expands the latent space, trying to recreate the original data. This reconstruction process is given by:

$$\mathbf{x}' = f_{\text{dec}}(\mathbf{z}) = \sigma(\mathbf{W}_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{dec}})$$

where $\mathbf{W}_{\text{dec}}$ and $\mathbf{b}_{\text{dec}}$ are the weights and biases of decoder. Autoencoders are often used as a pre-training task for supervised models, where latent representations $\mathbf{z}$ are used as relevant features that describe underlying pattern in the input. These learned features can be used to initialize the weights of a neural network to speed up training and improve model convergence. Beyond this, autoencoders are employed in several tasks such as denoising, where they learn to remove noise from input, or anomaly detection, where they are used to identify anomalies in new data that do not match the learned patterns.

2.2.3 The transformer

Transformers are a neural network architecture introduced in 2017 by Vaswani et al. [13] and revolutionized deep learning, especially in natural language processing (NLP), thanks to its ability to manage data sequences more efficiently than previous architectures. The main feature of Transformers is the "attention" mechanism, which allows the model to weigh the importance of different parts of an input during processing, eliminating the need for recurrent or convolutional networks, which were previously dominant in sequence modeling tasks. Transformers are composed of two main blocks: the encoder and the decoder, as shown in Figure 2.5. The **encoder** transforms the input into an internal representation and it consists of multiple identical layers, each of which is composed of:

- *Multi-Head Self-Attention (MHSA)* allows the model to consider all positions of the input sequence simultaneously, evaluating the relationships between embeddings regardless of their position in the sequence.

- *Feed-Forward Neural Network (FFNN)* through which input is passed after attention to apply nonlinear transformations.

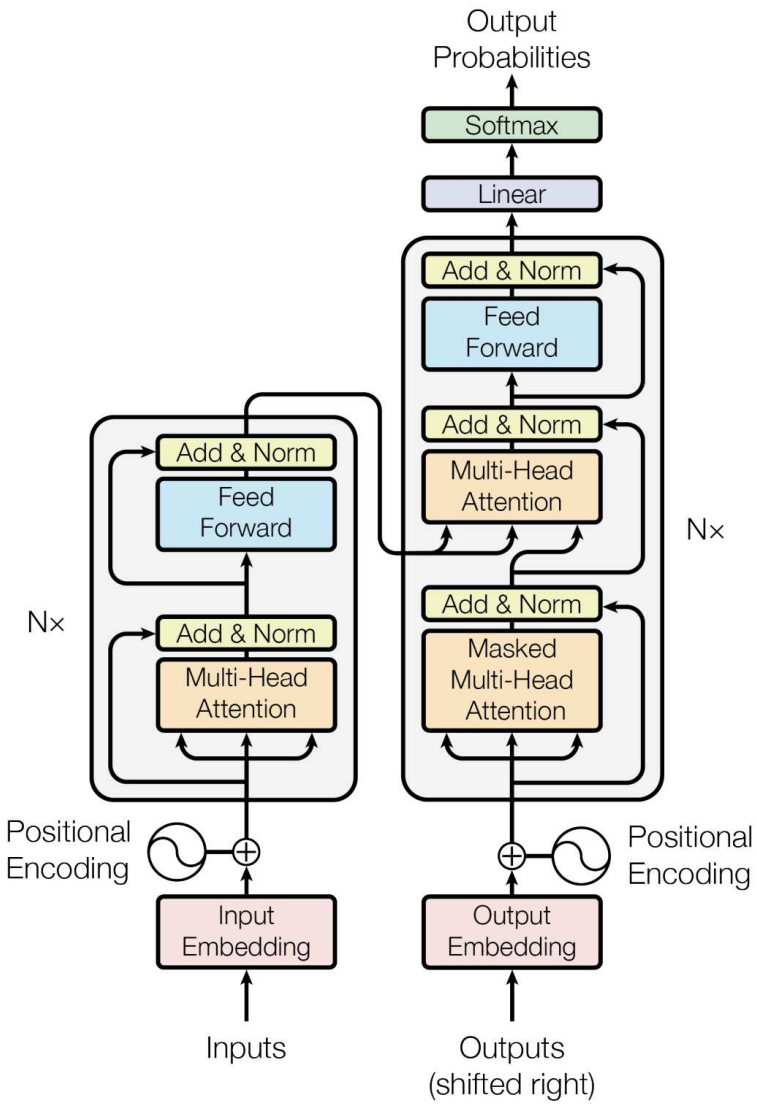


Figure 2.5: The Transformer- model architecture [13].

Each subcomponent is followed by a normalization and residual connection mechanism, which adds the subcomponent's original input to its output. The **decoder** uses the internal representation to generate the output. It is composed of multiple identical layers and it has an additional step compared to the encoder.

- *Masked Multi-Head Self-Attention*, similar to the encoder attention, but

with a "mask" that prevents the model from looking beyond the current position when generating output.

- *Multi-Head Attention* where the decoder receives input from the encoder, allowing the model to focus on specific parts of the input when generating the output.

- *Feed-Forward Neural Network*, same as the encoder.

At the end of the decoder, a final classification layer uses the softmax activation function to predict next-token probabilities.

As mentioned at the beginning of this section, the key element is the **attention mechanism**, which computes a representation of the input by considering all positions in the sequence. This mechanism is carried out in multiple so-called heads of both the encoders and the decoders: within these heads, multiple operations and steps are performed. A graphical representation of these operations is shown in Figure 2.6. The first step consist of creating

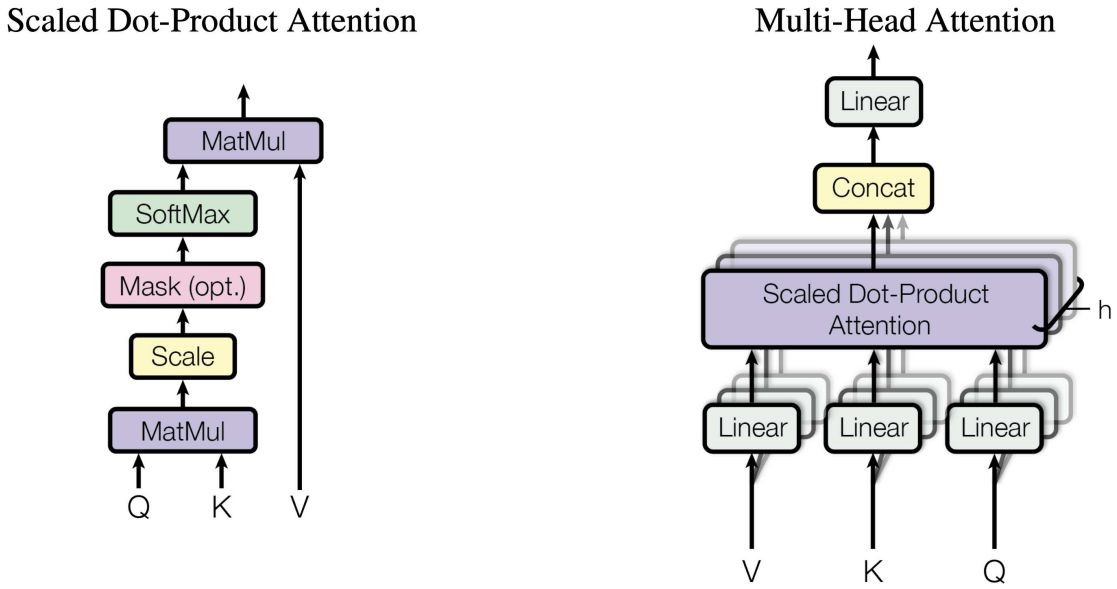


Figure 2.6: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel [13].

three vectors from each encoder's input vector by multiplying the embedding by three matrices that are trained during the training process. As a result, for each input a Query vector, a Key vector, and a Value vector are

created. The attention mechanism computes the relevance of each input word to every other word in the sequence by taking the dot product of the Query with the Key, followed by a scaling operation and the application of a softmax function to ensure that the weights sum to 1:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where Q represents the query matrix, K is the key matrix, V is the value matrix, and d is the dimension of K. To make the attention work effectively, the dot products are scaled by $\frac{1}{\sqrt{d}}$, which mitigates the issue of having large dot products when the dimension d increases, preventing small gradients in the softmax function. This produces the attention score. This score is calculated for each element of the input sequence and is used to generate a weighted sum of the value vectors, which is the new representation of the input word. This operation is done in parallel across multiple heads, allowing the model to attend to different aspects of the input sequence simultaneously. This is known as *Multi-Head Attention*:

$$E(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)W^0$$

where each attention head computes attention independently with different learned projections for Q, K, and V, and the results are concatenated and linearly transformed. The final output is a representation of the input sequence where each element is influenced by the most relevant parts of the input. One of the main advantages of this mechanism is its ability to focus on specific parts of the input sequence depending on their relevance, instead of relying on fixed-length context windows.

Chapter 3

Related works

In this chapter, the main works related to heart rate estimation based on PPG signals are presented. In recent years, research in this field has developed fast due to the advent of new wearable devices and the implementation of advanced machine learning and deep learning techniques. The accurate estimation of HR represents an important challenge, especially in real-life conditions, where user movements and subject variability can compromise the quality of the prediction. Several approaches have been proposed to address these difficulties, from traditional models based on signal processing to more recent deep learning techniques.

3.1 Classical Approaches

To address the challenges posed by MA, initial approaches in the literature proposed filtering techniques, where the correlation between acceleration data and PPG signals was leveraged to remove noise and mitigate MA before extracting HR from the cleaned signal [14], [15].

Among the model-driven approaches, TROIKA [16] paved the way to algorithm exploration, combining three main modules that lend their names to the system: decomposiTion, sparse signal Re-cOnstructIon, and spectral peaK trAcking. Signal decomposition allows partial mitigation of MA, while the sparse reconstruction module provides a high-resolution estimate of the PPG signal spectrum that is robust to noise. Finally, spectral tracking module identifies heart rate-related peaks and improves the spectral resolution of the signal. For validation, the authors gathered a new dataset with recordings from 12 subjects during fast running at the peak speed of 15 km/hour.

They collected a single-channel PPG signal, a three-axis acceleration signal, and an ECG signal, allowing HR annotations as ground truth. TROIKA succeeds in partially mitigating the impact of MA on PPG signals, achieving a MAE of 2.34 BPM on their dataset. However, these methods often suffer from poor generalization due to the high number of tunable hyperparameters.

Other studies such as Independent Component Analysis (ICA) [17] and Kalman filtering [18], explored only scenarios with minimal motion artifacts. To address this limitation and reach a satisfying accuracy in case of high MAs, Zhang et al. [19] introduced a novel approach called JOint Sparse Spectrum reconstruction (JOSS). This approach focuses on the fact that the spectra of simultaneous PPG and acceleration signals share common spectral structure characteristics, resulting in an alignment between the frequency locations of MA within PPG spectrum and corresponding locations in acceleration spectrum. By using the multiple measurement vector (MMV) model to estimate signals, JOSS allows easy identification and removal of MA spectral peaks present in the PPG spectra without the need for additional signal processing modules. Experimental results, obtained in the same 12 PPG datasets used in TROIKA, shows improvement in accuracy with a MAE of 1.28 BPM.

Afterwards, Huang et al. [14] introduced a novel approach, TAPIR, a lightweight algorithm based on four main steps: adaptive filtering, peak detection, interval tracking and refinement. TAPIR uses both PPG and acceleration data to effectively remove movement artefacts using least mean square (LMS) adaptive filtering. MA are eliminated by the adaptive filter, which takes accelerometer data with delays of up to 250 ms as input and the PPG signal as the desired output. The filtered signal provides an estimate of the noise caused by motion, which is then subtracted from the PPG signal. After MA removal, the signal is processed to identify local peaks, which are then tracked and corrected at different time scales to ensure accurate heart rate estimation. Unlike previous approaches, TAPIR is designed to be robust under intense motion conditions, demonstrating a MAE of 4.6 BPM on the DaLiA dataset and 4.2 BPM on the WESAD dataset, obtaining significantly lower error rates compared to most of the contemporary algorithms. In addition, TAPIR is computationally efficient, making it suitable for low-power portable devices.

Despite significant advances, these approaches still suffer from limitations in complex real-world environments where movement variability of physical

activity can introduce significant MA that reduce the accuracy of HR estimation.

3.2 Deep Learning Approaches

Only in recent years HR estimation has been investigated using Deep Learning techniques. The main difficulties lie in the limited available memory of the microcontrollers (MCUs) used in wrist-worn wearable devices, which cannot support DL models with millions of parameters. In addition, training effective DL models requires large amounts of labelled data, which are difficult to obtain because annotation of signals is an expensive and time-consuming process. This scenario has changed with the introduction of large datasets such as PPG-Dalia, which was specifically designed for HR tracking in the presence of motion artefacts and includes recordings from 15 subjects during different daily activities.

3.2.1 Deep-PPG

One of the first deep learning model was introduced by the authors of PPG-DaLiA dataset and demonstrated better performance than previous approaches. Reiss et al. [6] proposed a method based on convolutional neural networks (CNN) that takes as input PPG and accelerometer signals processed through the Fourier transform (FFT) to analyse them in the frequency domain. Following this approach, the time signal of PPG and accelerometers is segmented using a 8-seconds sliding window, with 2-seconds shift. As a second step, the FFT is applied to each time signal segment, resulting in N_{ch}=4 time-frequency spectra, one for each signal channel. Subsequently, these spectra are filtered by keeping only the range between 0 and 4 Hz, which corresponds to a maximum heart rate of 240 BPM. The number of FFT points per segment and channel is 1,025. Z-normalisation is applied to each channel to ensure consistency of the input data. The final time-frequency spectra, represented as a matrix $N_{\text{ch}} \times N_{\text{FFT}}$ for each 8-second segment are the input for the deep learning model, as shown in Figure 3.1.
For the architecture of the model, several hyperparameters were tested, including the number of filters and filter size in each convolutional layer, the activation function, the convolutional and pooling layer steps, the size of the fully connected layer, the dropout rate, the loss function and the optimisation method. The first convolutional layer performs the fusion of PPG

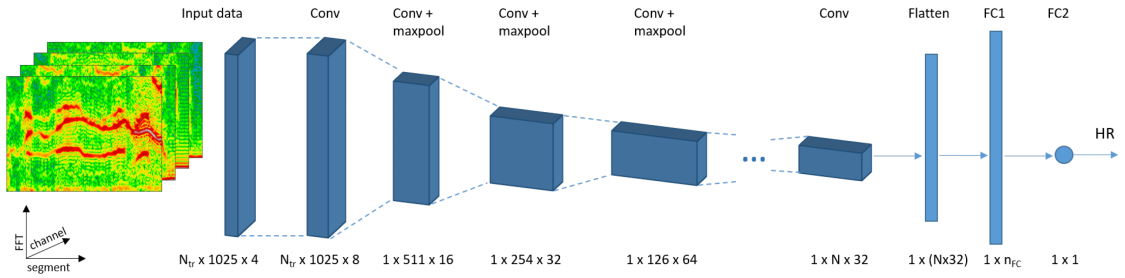


Figure 3.1: Proposed CNN-architecture with $NL = 1 \ldots 8$ convolution-maxpool layers. N_{tr} refers to the number of segments used together for heart rate tracking. N depends on NL. Input: $N_{tr} \times N_{ch} \times N_{FFT}$ matrix [6].

and accelerometers channels, while the second level fuses the time segments involved in heart rate tracking. Subsequent convolution and pooling layers improve the model's ability to learn increasingly complex patterns in the data, thereby improving the accuracy of heart rate estimation. Finally, the last fully connected layer outputs the estimated heart rate. CNN has proven to be more effective than classical approaches, reducing the mean absolute error by 31% on the PPG-DaLiA dataset (reducing MAE from 11.06 BPM to 7.65 BPM) and by 21% on WESAD dataset (reducing MAE from 9.45 BPM to 7.47 BPM).

3.2.2 Q-PPG

In [22], Burrello et al. focused their research to reduce the complexity of models used for estimating heart rate based on PPG. Quantized-PPG (Q-PPG) uses Neural Architecture Search (NAS) and quantisation techniques to generate a set of deep Temporal Convolutional Networks (TCNs) from a single basic architecture. The exploration process is divided into two main phases, as represented in Figure 3.2:

1. Architecture Optimization: Starting from a seed TCN, they vary its structure to trade-off computational cost and HR tracking error. To achive that, they use a cascade of two NAS tools, *MorphNet*, used to optimize the number of output channels (or features), and *Pruning-In-Time (PIT)*, used to search for the optimal dilation parameter d.

2. Precision Optimization: To trim down the model size, enhancing the Pareto frontier, hardware-friendly quantization is introduced. It allows

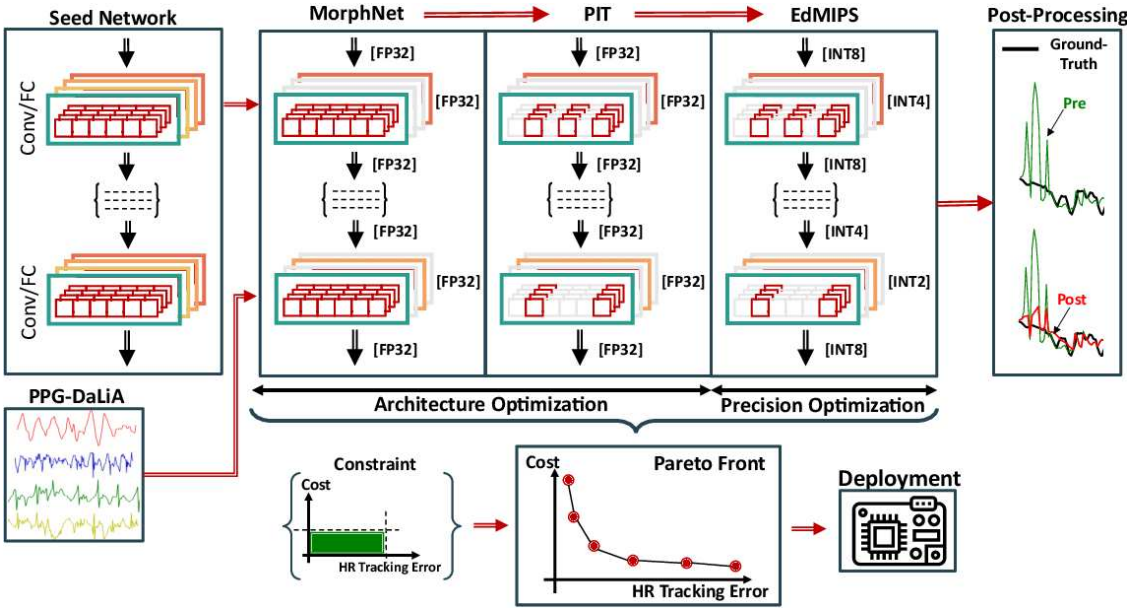


Figure 3.2: Proposed Q-PPG design space exploration flow [22].

to replace all floating point multiply-and-accumulate (MAC) operations required for inference with integer MACs.

One benefit of this approach is that Q-PPG exploration has to be performed only once for a given dataset and seed model. The researchers deployed their results on a real embedded smartwatch device powered by an STM32WB55 MCU from ST Microelectronics. The quantization optimization results in networks spanning a size spectrum, ranging from the largest at 1MB (in floating-point representation) to the smallest at less than 1kB. The most extensive model that can accommodate the target embedded device, STM32WB55, requires approximately 412kB, yielding a Mean Absolute Error (MAE) of 4.41 beats per minute (BPM).

3.2.3 Attention-PPG

In [24] Kasnesis et al. introduced PULSE, a novel and lightweight DL architecture that utilises TCNs and a multi-headed cross-attention module (MHCA) at the feature level for the fusion of sensory data. This model represents the current state-of-the-art and the starting point for our project. In this work, authors also have implemented a knowledge distillation mechanism to transfer acquired skills from a more complex network (Teacher Network) to a lighter network (Student Network). This architecture aims to improve

HR estimation based on PPG sensors, addressing the problem of MA affecting the measurements.

The *Teacher Network*, similar to Q-PPG [22], consists of:

- 3 consecutive convolutional blocks, each with dilated 1D convolutions, ReLU activations, average pooling, and dropout (0.5).
- A MHCA module that integrates PPG data features with accelerometer features. In this process, the PPG signal features serve as queries (Q), while the accelerometer features form the keys (K) and values (V), enabling effective fusion of sensory modalities.
- Layer normalization operation .
- Two dense layers, with the last one producing the estimated HR.

The *Student Network* is a lighter version of the Teacher Network, where the MHCA module is replaced with a modality-wise convolution. This substitution reduces computational complexity while maintaining good performance.

The *Knowledge Distillation* process transfers the Teacher Network's knowledge to the Student Network using three types of losses:

- L_{Hard}: calculates the difference between the true HR values and those predicted by the Student Network:

$$L_{\text{Hard}} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i^s|$$

  where y_i represents the true values and $\hat{y}_i^s$ represents the values predicted by the Student Network.

- L_{Soft}: measures the difference between the predictions of the Teacher and Student Networks:

$$L_{\text{Soft}} = \frac{1}{N} \sum_{i=1}^{N} \left|\hat{y}_i^t - \hat{y}_i^s\right|$$

  where $\hat{y}_i^t$ are the predictions of the Teacher Network and $\hat{y}_i^s$ those of the Student Network.

- L_{Rel}: evaluates the distance between the pre-final dense layer outputs of the Teacher and Student Networks, using the softmax function to ensure numerical stability:

$$L_{\text{Rel}} = \frac{1}{N} \sum_{i=1}^{N} \left(\text{softmax}(R_i) - \text{softmax}(\hat{R}_i) \right)^2$$

  where R_i and $\hat{R}_i$ represent the pre-final activations of the Teacher and Student Networks, respectively.

The overall loss for knowledge distillation is given by:

$$L_{\text{KD}} = \gamma \left(\beta L_{\text{Hard}} + (1 - \beta) L_{\text{Soft}} \right) + (1 - \gamma) L_{\text{Rel}}$$

where β and γ are hyperparameters that balance the contributions of the different loss components.

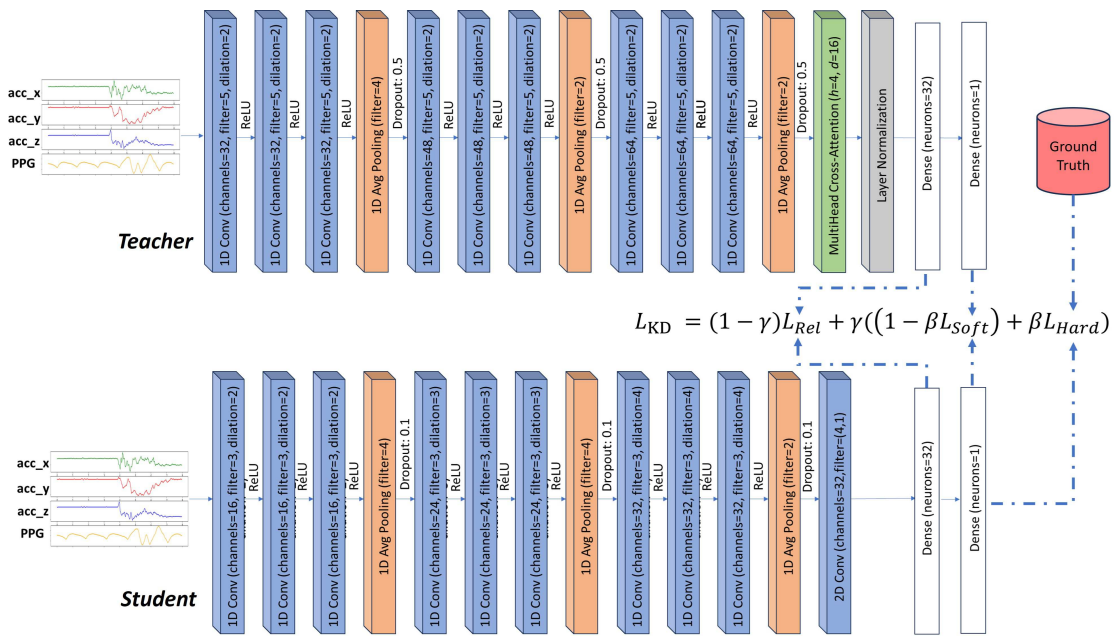


Figure 3.3: Detailed illustration of the teacher-student knowledge distillation process. The Teacher Network is shown in the upper section, and the Student Network in the lower section. The losses L_{Rel}, L_{Soft}, and L_{Hard} are combined to compute L_{KD}, which is used to update the Student Network's weights [24].

The entire process is summarised in Figure 3.3. After the distillation process, quantization is applied to further reduce the model size and facilitate execution on low-power microcontrollers. The models have been deployed on two

microcontrollers, demonstrating their ability to perform real-time inference, with a MAE of 4.81 BPM and memory usage of 37.9 kB, 10.9 times lower than current state-of-the-art models.

Chapter 4

Methods

In this chapter, we describe in detail our innovative approach that integrates self-supervised learning and data augmentation techniques to enhance HR estimation performance of deep learning models. Our method can be divided into several steps including data preparation and processing, pre-training, fine-tuning with transfer learning, and post-processing to improve the reliability of the results. The first step involves data pre-processing, where the PPG and accelerometer signals are segmented into time windows with overlap. Subsequently, data are normalised using z-score normalisation, ensuring that data are on the same scale. To further improve the model's performance and enlarge the amount of available data for training, Data Augmentation (DA) techniques are applied, creating synthetic data.

As our approach is orthogonal to the deep learning model selected, we tested two different architecture. In the first part of the this work, we evaluated the performance of a Masked Autoencoder to establish a benchmark that would serve as a reference model. During this phase, the autoencoder was trained to test its ability to extract relevant features from data by reconstructing the input signals. Subsequently, during the fine-tuning phase, the last part of the network was slightly modified to adapt it to the final HR estimation task, using the knowledge gained in the previous phase.

Next, we focused on the PULSE model, which currently represents the state of the art for HR estimation. To evaluate the effectiveness of our approach, we re-implemented the PULSE model as a baseline, introducing some changes to the hyperparameters to improve its robustness and to reduce computational complexity. In particular, the dilated convolutions used in the original

version are replaced by standard convolutions. This changes allow the model to maintain high performance by reducing the computational load, without compromising the model's ability to extract relevant features from PPG signals. In this case, during pre-training phase, the model is trained for the input signal reconstruction task on a large combined dataset including WESAD and West Attica University data, using an autoencoder-like framework inspired by the U-Net model [27]. The autoencoder is based on the PULSE model architecture, without the classification head, and employs multi-head attention mechanisms to improve feature extraction. We explore DA techniques to significantly increase also the pre-training data, enhancing model's robustness to different and noisy inputs. After pre-training, the model is fine-tuned for HR estimation. During this phase, the autoencoder decoder is removed and replaced by a classification head consisting of two linear layers. The model is then fine-tuned using the PPG-DaLiA dataset. Finally, a post-processing step is introduced to smooth out the prediction avoiding random peaks.

4.1 Data Preprocessing

To prepare the PPG signal and acceleration data, we segment the raw data into contiguous time windows. Specifically, raw data is divided into sliding windows of 8-seconds each and 2-seconds shift and, based on the training phase, the label is calculated as the average of the signal peaks in each time window. In addition to time domain analysis, where signal characteristics are extracted directly from sampled values, data can also be analysed in the frequency domain. Working in the frequency domain, PPG signals are divided into time windows, and Fast Fourier Transform (FFT) is applied to convert the data from the time domain to the frequency domain, generating a spectrogram showing the energy distributed between the different frequencies over time. This approach allows the heart rate to be estimated by identifying peaks in the frequencies associated with the heart rate, generally between 0 and 4 Hz. Figure 4.1 represents the process described with a visual example. To improve the representation of frequencies in relation to human perception, the mel scale is used, which transforms frequencies so that differences between low frequencies are more perceptible than high ones. We perform the following mathematical operation to convert frequencies to the mel scale:

$$\text{mel}(f) = \begin{cases} f, & \text{if } f \leq 1000\,\text{Hz} \\ 2595 \times \log\left(1 + \frac{f}{700}\right), & \text{if } f > 1000\,\text{Hz} \end{cases}$$

For the analysis, the MelSpectrogram from the TorchAudio library was used, configured to optimise the extraction of frequency characteristics from PPG signals in the following way:

```
transform = torchaudio.transforms.MelSpectrogram(
    sample_rate=32,
    n_fft=510,
    win_length=32,
    hop_length=1,
    center=True,
    pad_mode="reflect",
    power=2.0,
    normalized=True,
    f_min=0,
    f_max=4,
    n_mels=64
)
```

4.1.1 Z-Score Normalization

Normalization is a crucial step in pre-processing as it helps to mitigate the influence of outliers and ensures that the data is on a the same scale. We normalize the windows using per-channel z-score:

$$z_i = \frac{x_i - \mu_{\text{train},i}}{\sigma_{\text{train},i}},$$

where x_i denotes the samples of channel i, while $\mu_{\text{train},i}$ and $\sigma_{\text{train},i}$ represent the mean and standard deviation values, respectively, estimated on the training set. In particular, in cases where a pre-training phase is included, the mean and standard deviation are computed considering all the datasets that constitute the pre-training dataset. These values are then used to normalise the entire fine-tuning dataset. In the absence of pre-training, the mean and standard deviation are computed on the target dataset only. These values are then used to normalise the validation and test datasets, ensuring consistency across all phases of the model's development.

4.1.2 Data Augmentation Techniques

Data augmentation (DA) has become an essential technique in the field of biomedical signal processing, especially to enhance deep learning models

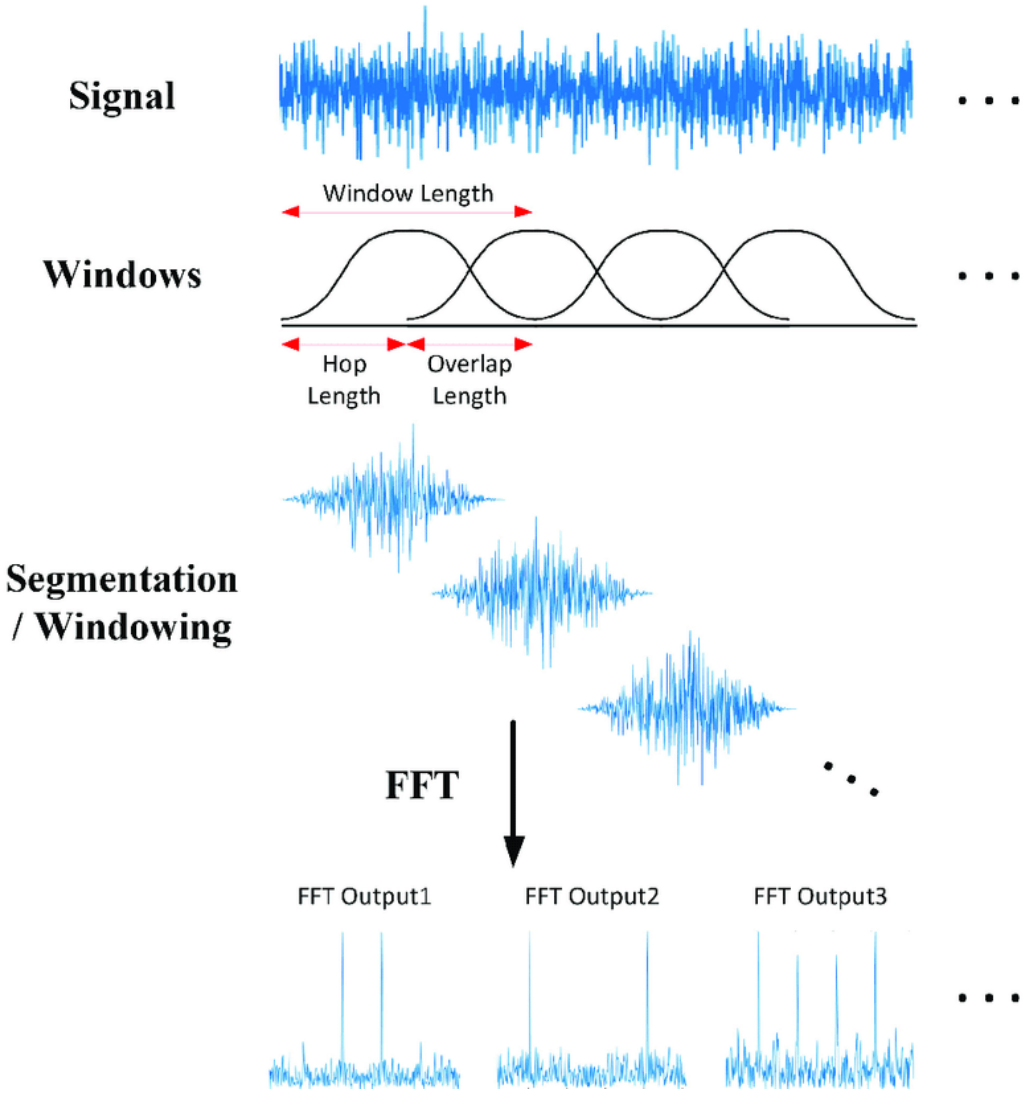


Figure 4.1: Spectogram from a raw signal using FFT on overlapping windowed segments [26].

without increasing algorithm's complexity. In this thesis, we investigate the use of data augmentation strategies to improve the performance and robustness of models trained on biomedical PPG signals for heart rate monitoring. The main goal of DA is to reduce the risk of over-fitting and improve the generalization capabilities of deep learning models. In fact, the performance of deep neural networks tends to improve as the amount of data increases. However, collecting "real-world" data for PPG-based heart rate monitoring is expensive, primarily because it requires the use of a medical-grade ECG

device to accurately measure the ground truth heart rate.

Data augmentation represents a widely employed data-driven strategy that involves creating altered versions of the original dataset to increase the size of the training set. Specifically, new data points, denoted as $\hat{x}_i$, are generated from the original data points x_i by introducing noise or implementing scale, shape, or temporal transformations. Burrello et al. [20] propose four classical DA techniques designed for time-series data and two novel and customize transformations, on both PPG and 3D-acceleration data. Figure 4.2 shows an example of the 6 augmentations considered, each of which will be described in detail below.

The classical DA transformations are jittering, scaling, magnitude warping and time warping. These are applied to the single 8-second window and they keep the HR label unchanged.

- **Jittering** involves introducing noise to the signal by adding a value sampled from a Gaussian distribution with mean $\mu = 0$ at each time step t. This is mathematically represented as:

$$X_t = X_t + N(0, \sigma) \quad \forall t$$

- **Scaling** allows to enlarge or reduce the amplitude of the original signal. A single value is drawn from a Gaussian distribution with mean $\mu = 1$, and it is used to multiply the entire input window:

$$X = X \times N(1, \sigma)$$

- **Magnitude Warping** modifies the magnitude of the input signal by scaling the signal's envelope with a cubic spline that interpolate points randomly chosen from a Gaussian distribution with mean $\mu = 1$:

$$X = X \times \text{CubicSpline}(0...T, N(1, \sigma))$$

- **Time Warping** alters the signal on its time axis. As a result, the signal appears stretched in some areas and compressed in others. In fact, it changes the time intervals between successive samples from the original fixed interval $\frac{1}{f_s}$ (where f_s represents the sampling frequency) to the differences between points on a randomly generated cubic spline, denoted as $C_{St} - C_{St-1}$:

$$X = \text{Interp}\left(\text{Cumulative}\left(\text{CubicSpline}(0 \dots T, N(1, \sigma))\right), X\right)$$

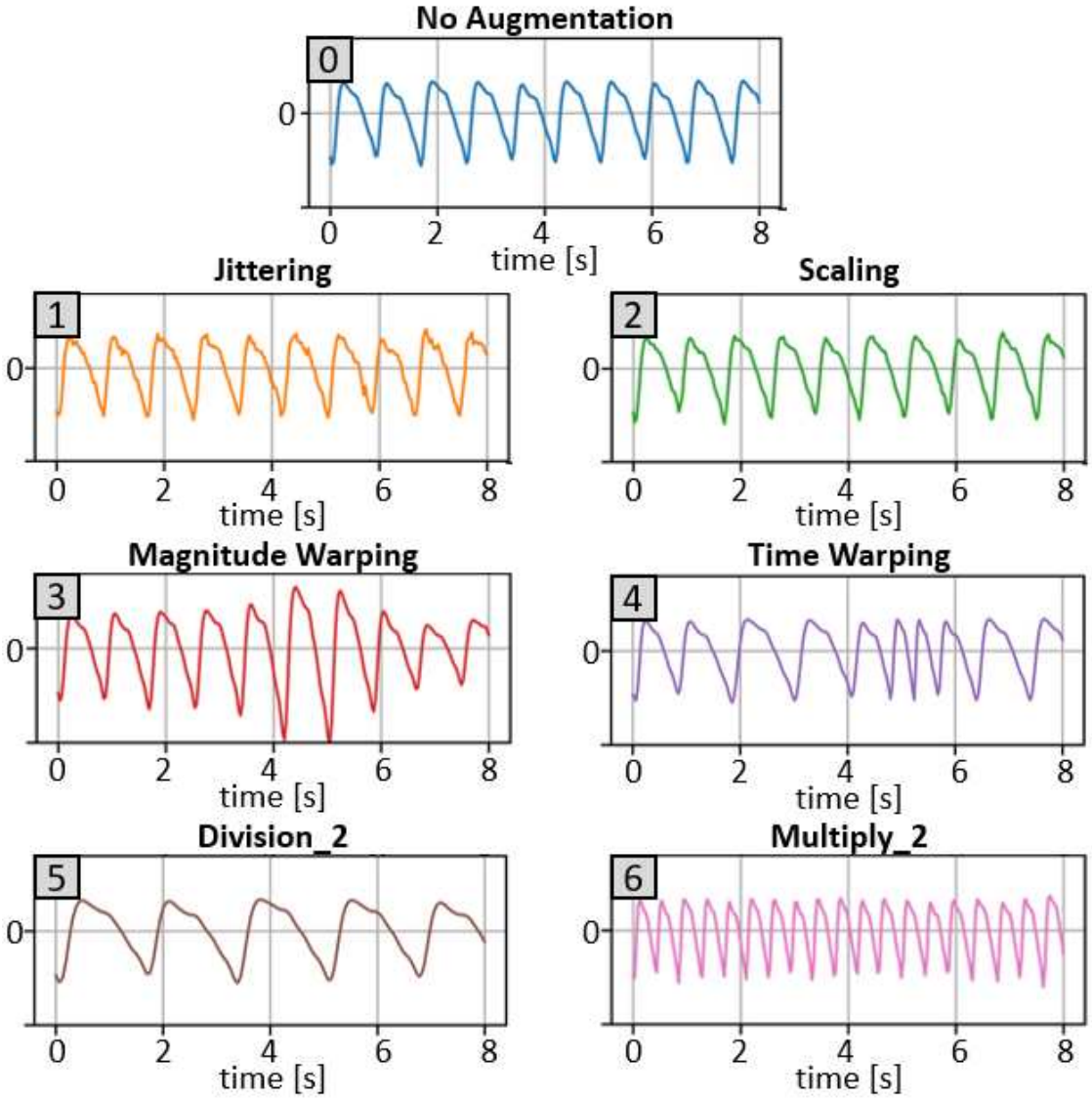


Figure 4.2: Synthetic signals obtained from the application of the six DA technique to a 8 seconds window of the PPG-DaLiA dataset [20].

Burrello et al. [20] observed also that individuals at the extremes of the Beats Per Minute (BPM) spectrum, with very low or very high average BPM, were often poorly tracked. This issue arises because the majority of the training data tends to fall within a more typical BPM range. To address this challenge, they introduce two novel data augmentation techniques which not only help to reduce over-fitting but also expand the BPM range within the training dataset.

- **Divide_2** extend ranges the lower bound of BPM. For each input window of duration T, a continuous segment of length $T/2$ is randomly selected and is then time-stretched back to the original window size (0 to T) with resampling. The BPM label corresponding to this window is reduced by half.

- **Multiply_N** is designed to expand the upper limit of the BPM range. This approach uses various up-scaling factors ranging from 1.1× to 2×, creating a temporary augmented dataset consisting of windows of length $2T$ formed by concatenating two consecutive original windows. The BPM label for these windows is calculated over the entire $2T$ duration. According to the selected multiplication factor, a segment of length $1.1T$ to $2T$ is randomly sampled from the $2T$window. Finally, this signal is rescaled to fit the original T-length window using linear interpolation, with the BPM label adjusted to reflect the up-scaling.

To further improve the performance of our models, we apply DA before feeding the data to the neural network. In particular, we implemented the same DA transformations described above. We explore two configurations of DA to cover different sizes of training datasets, and see how they affect the results. Table 4.1 shows the configurations, indicating the increase in the training set size compared to the original non-augmented dataset, the type of data augmentation applied, and the percentage of new samples generated by each transformation relative to the original dataset size. The notation Multiply_[x-y, z] specifies the minimum (x), maximum (y), and step (z) of the multiplication factors used. For instance, Multiply_[1.4-2.0, 0.6], 200% refers to a 200% augmentation of the original dataset with synthetic data generated by multiplying the heart rate frequency by 1.4 and 2.0.

As the table shows, the first configuration tested **8x Data Augmentation** includes all the techniques presented by Burrello et al. [20] while the second one **11x Data Augmentation** focuses only on those approaches that expand the BPM range. These configurations were tested during both pre-tuning and fine-tuning. All signals have been normalized with z-score normalization before applying the transformations. Finally, the models were trained with the augmented dataset, including both synthetically generated data and the original data.

Augmentation [x]	Augmentation Type	Percentage (%)
8x	Jittering	100%
	Scaling	100%
	Time Warping	100%
	Magnitude Warping	100%
	Divide_2	100%
	Multiply_[1.4 - 2, 0.6]	200%
11x	Divide_2	100%
	Multiply_[1.2 - 2, 0.1]	900%

Table 4.1: Data Augmentation configurations explored

4.2 Model Architecture

After the pre-processing step, in which the raw input signal is segmented, normalized and various DA techniques are applied, our approach involves the integration of SSL techniques to improve accuracy in the final HR estimation task. A key aspect of our approach is that it is orthogonal to the specific deep neural network selected. This means that the principles on which it is based can be applied regardless of the type of network used. We test its performance by applying this approach to two different models, a masked autoencoder and PULSE, both described in detail below.

4.2.1 Masked Autoencoder

The first architecture considered in this thesis is based on Masked Autoencoders and allows us to establish a benchmark that would serve as a reference model. The key concept behind the Masked Autoencoder architecture is to mask part of the input data and train the model to reconstruct the missing content. This is why they fall into the category of denoising autoencoders. This architecture is particularly effective for learning general representations of the data, as it forces the model to extract the key features and underlying patterns of the data itself, allowing large models to be trained by reducing redundancy in the input information.

First, the input is masked: the PPG signals are transformed into Mel-spectrograms and divided into a regular grid of non-overlapping patches. These patches are subsequently flattened and projected using a linear projection. A fixed sinusoidal positional embedding is added to each patch,

which keep information about relative position of the patches within the spectrogram. The positional embeddings are vectors that provide the model information about the order of the patches, thus preserving the structure of the input during processing in the Masked Autoencoder. Then a part of them is removed. The *masking ratio* determines the amount of data that is removed. In our work, this hyper-parameter is set to 15%. The visible part of the input is then sent to the model. Our Masked Autoencoder, like any autoencoder, consists of:

- *Encoder*: receives the masked input and processes it to extract the relevant features. Its goal is to create a latent representation of the signal. Our encoder consists of Transformer Encoder blocks and it processes the input through attention layers and multi-layer perceptron (MLP), as shown in Figure 4.3.

- *Decoder*: receives the full set of encoded masks, and a set of "random-noise" patches, i.e. patches composed by random numbers that fill the space left by the masked patches. The decoder also consists of Transformer Encoder blocks and it has the task of reconstructing the original input from the extracted features.

We utilized the following hyperparameters:

- **depth = 4:** The number of transformer encoder blocks in the encoder. A higher value allows the model to learn increasingly abstract representations, but may increase the computational complexity.

- **num_heads = 16:** The number of heads in the MHA mechanism. Each attention head focuses on a different part of the input, allowing the model to consider different aspects of the several in parallel.

- **embed_dim = 64:** This parameter defines the size of the latent space, in which each patch is encoded. Each patch is transformed into a vector of size 64, thus reducing the complexity of the original signal, but retaining relevant information.

- **decoder_depth = 8:** The number of transformer encoder blocks in the decoder.

- **decoder_embed_dim = 64:** Defines the size of the embedding in the decoder.

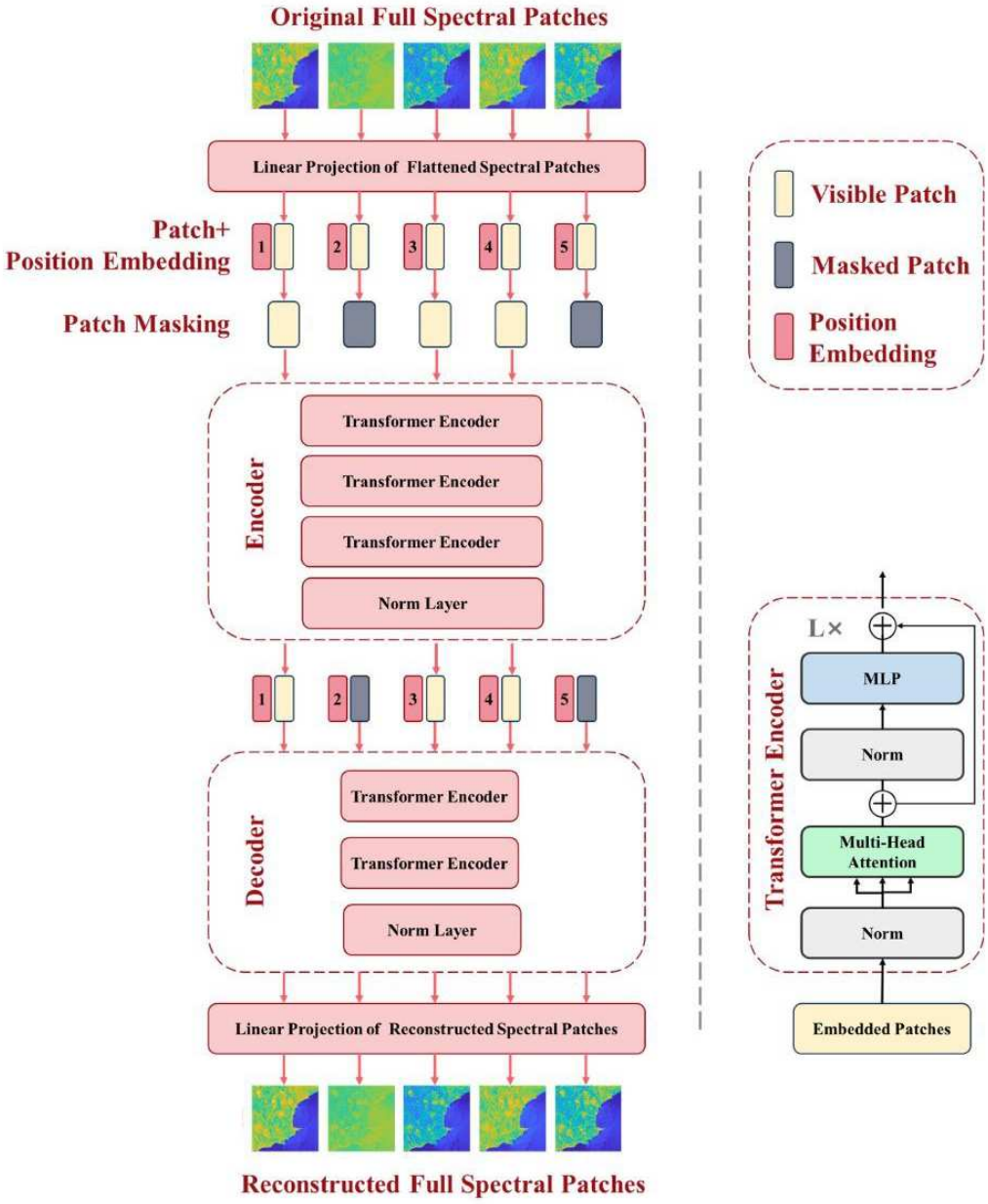


Figure 4.3: Detailed description of the architecture of our Masked Autoencoder for manage PPG signals in frequency domain [25].

- **decoder_num_heads = 16:** Similar to the *num_heads* parameter in the encoder, this value represents the number of MHA heads used in the decoder.

We used this model for the pre-training task on signal reconstruction. This

step allows us to train the model on large amounts of unlabelled data, exploiting the model's ability to learn general representations of the signal, which can then be used for the specific task of HR estimation. During the pre-training phase, the loss function is computed using the Mean Squared Error (MSE) metric, i.e. as the distance in pixel between the original and reconstructed spectrogram. The loss is computed only on masked patches. Figure 4.4 and Figure 4.5 show a graphic example of the PPG heatmap reconstruction via Masked Autoencoder.

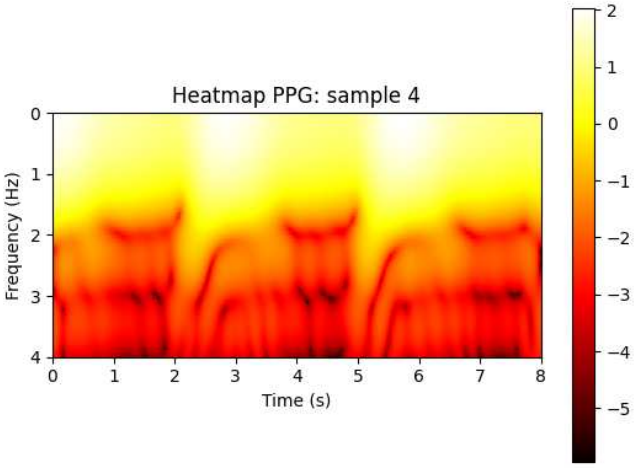


Figure 4.4: Original PPG heatmap.

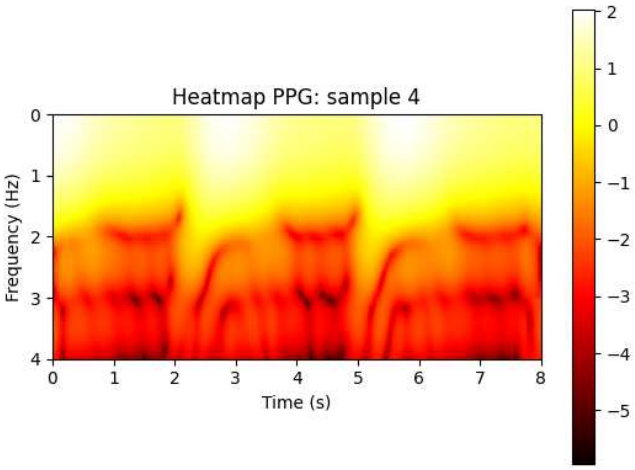


Figure 4.5: Reconstructed PPG heatmap, with 15% *mask ratio*.

Once the pre-training is over, we replace the decoder from the original Masked Autoencoder with a regression tail composed of two convolutional layers, a pooling layer, and a final linear layer. Starting from the hidden representation created by the encoder, the regression tail can perform the final HR estimation task. Pre-training and fine-tuning phases are described in detail in Section 4.3, while Figure 4.6 graphically illustrates the pipeline of these two stages. However, the results of this phase did not meet expectations

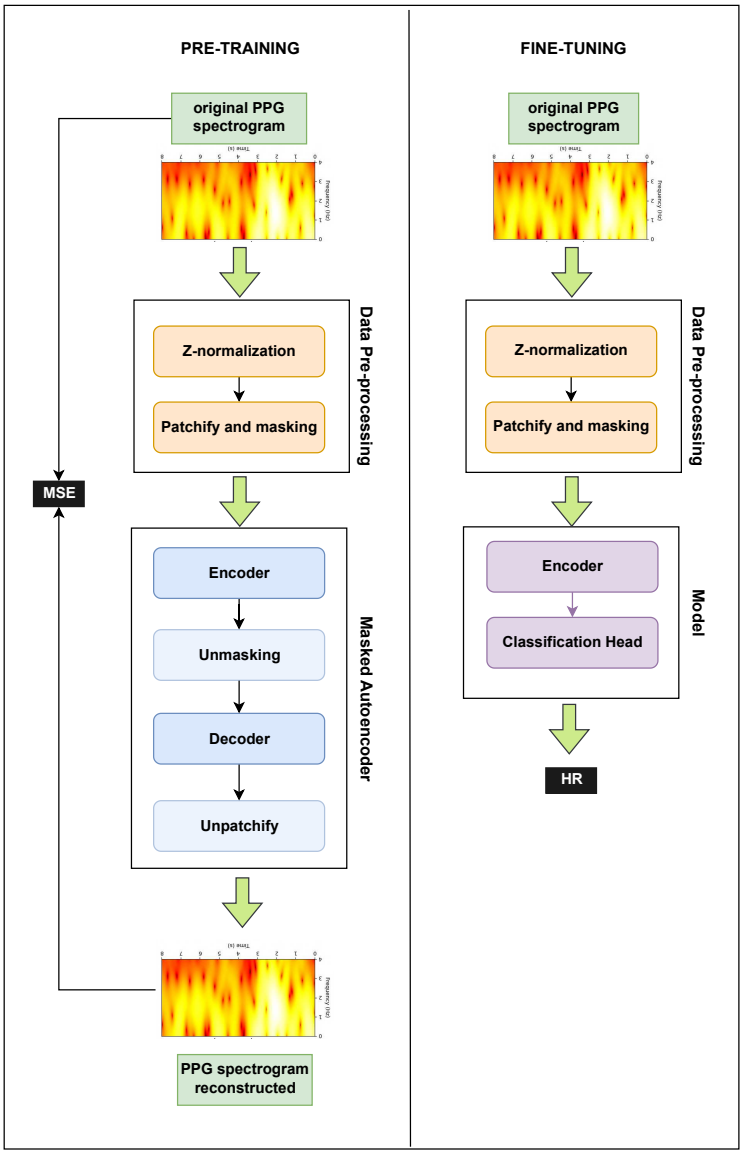


Figure 4.6: Masked Autoencoder pre-training and fine-tuning pipelines.

in terms of HR estimation accuracy, highlighting the need for a different architecture.

4.2.2 PULSE

The PULSE (Ppg and imU signaL fuSion for heart rate Estimation) model is a state-of-the-art deep neural network for heart rate estimation that uses PPG signals and tri-axial motion data from accelerometers, presented by Kasnesis et al. [23]. PULSE integrates temporal convolutions and a feature-level Multi-Head Cross-Attention (MHCA) module to improve accuracy and ensure greater interpretability of results. The network architecture (Figure 4.7) consists of three 1D convolutional blocks of increasing channel number (32, 48, 64). Each block contains three consecutive dilated convolutions with a dilation rate of 2, allowing the model to capture short-term temporal variations of both PPG and 3-axis acceleration changes. The outputs from each block are downsampled using average pooling layers with predefined pooling sizes. After the convolutional blocks, the generated feature maps are fed into the MHCA module, where the PPG signal acts as a query vector (Q) and the features extracted from the three-axis accelerometers act as key (K) and value (V) vectors, as shown in Figure 4.8. This approach allows the model to better capture dependencies between physiological and motion data and reduce over-fitting. Finally, the output of the attention module is normalized and passed through two dense layers, the last of which produces the estimated HR.

In our implementation of the PULSE model, we utilized the following hyper-parameters:

- **conv_blocks=3**: The model consists of three convolutional blocks.
- **conv_layers=3**: Each convolutional block consists of 3 layers.
- **in_channels=[1, 32, 48]**: The input channels for each convolutional block. The blocks increase the number of channels allowing more sophisticated feature extraction as the data progresses through the network.
- **out_channels=[32, 48, 64]**: The output channels for each convolutional block. These values indicate how the feature maps are expanded as they move through the network, with the last block producing 64 output channels.

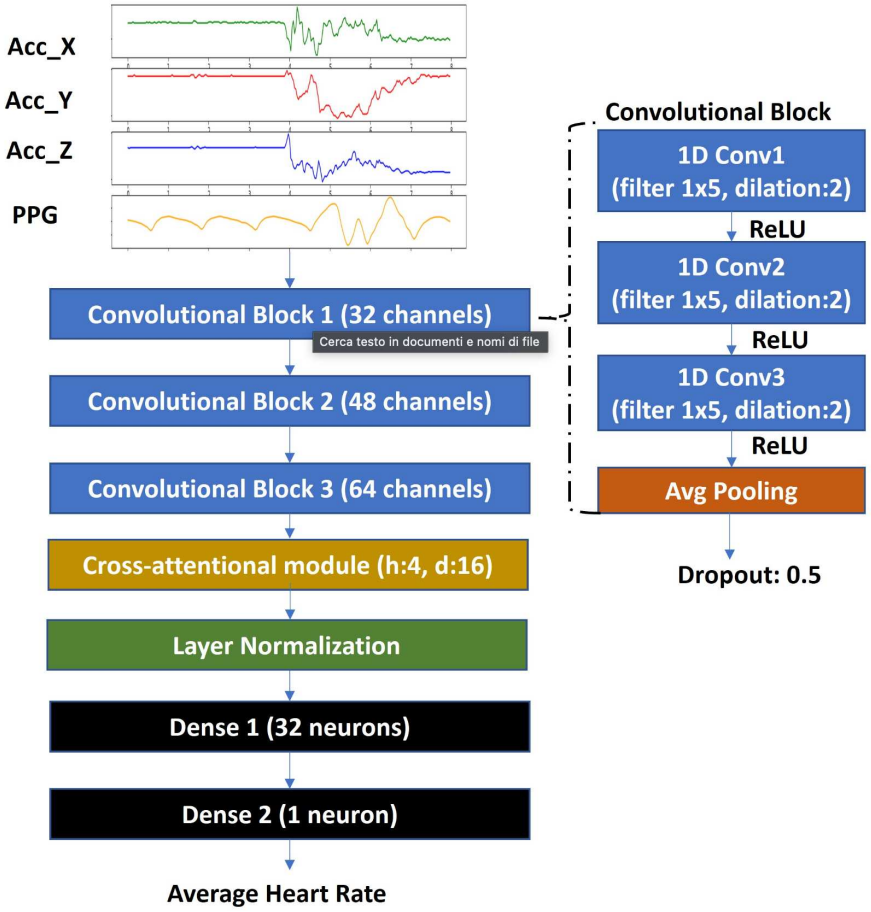


Figure 4.7: PULSE network architecture [23].

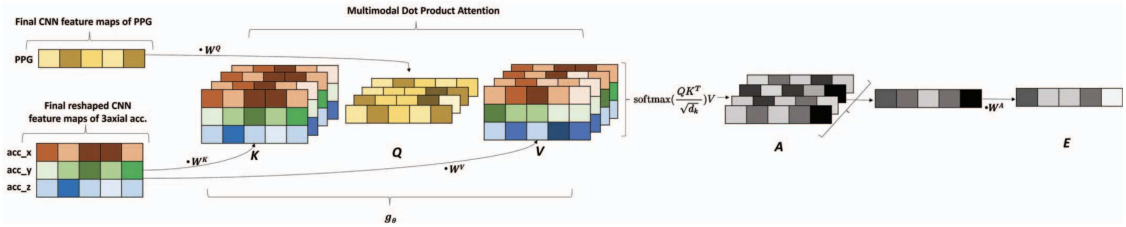


Figure 4.8: Multi-head cross-attention module applied to PPG and 3axial accelerometer feature maps. The PPG embedding acts as a Query tensor, while the 3-axial accelerometer embeddings are Key and Value tensors [23].

- **padding=(0,4)**: Padding of (0,4) is applied to maintain the spatial dimensions of the feature maps after convolution, ensuring that the output dimensions are consistent throughout the network.
- **dropout=[0.5, 0.5, 0.5]**: A dropout rate of 0.5 is applied across all convolutional blocks to help in preventing overfitting.
- **pooling_size=[(1,4), (1,2), (1,2)]**: Average-pooling is applied after each convolutional block with sizes (1,4), (1,2), and (1,2) respectively. This reduces the dimensionality of the feature maps, while preserving the most important features.
- **heads=4**: Number of parallel attention heads used in the attention mechanism.
- **dim=16**: The dimension of the attention mechanism.
- **dense_out=32**: The first fully connected layer has an output size of 32.
- **ppg_channels=1**: The model processes a one-channel PPG signal.

We modify the original architecture by changing:

- **kernel_size=(1,9)**: This represents the size of the filter applied during convolution.
- **dilation=1**: Convolutional filters are applied to adjacent elements without skipping any, resulting in a standard convolution.

The original model utilizes a kernel size of (1,5) with a dilation of 2. This choice is due to the fact that, since dilation is set to 1, our model is performing standard convolutions that allow to concentrate on more localized features maintaining a comparable inference latency. This modification forms the first key contribution of this project, as it leads to improve heart rate estimation performance while maintaining efficiency.

4.2.3 Pre-training architectures

In order to consider a self-supervised pre-training step, we need to modify the deep learning architecture that we chose to validate our approach, i.e. PULSE, and restructure it in a way that can enable its pre-training on a

huge amount of unlabelled data. We take inspiration from the autoencoder architecture, which compresses input data into a lower-dimensional representation (encoding), called latent space, and then reconstructs the input from this encoding. Two different autoencoder architectures were implemented, both inspired by the U-Net model [27], which presents a similar structure, consisting of an encoder and a decoder, both implemented using multiple convolutional layers, dropouts for regularisation, and MHA mechanisms to improve feature extraction. The main differences between the two architectures lie in the number of convolutional blocks used and their complexity, as described below.

Light Autoencoder Architecture

In the *Light* Autoencoder Architecture, the encoder consists of PULSE model without the classification head, as shown in Figure 4.9.

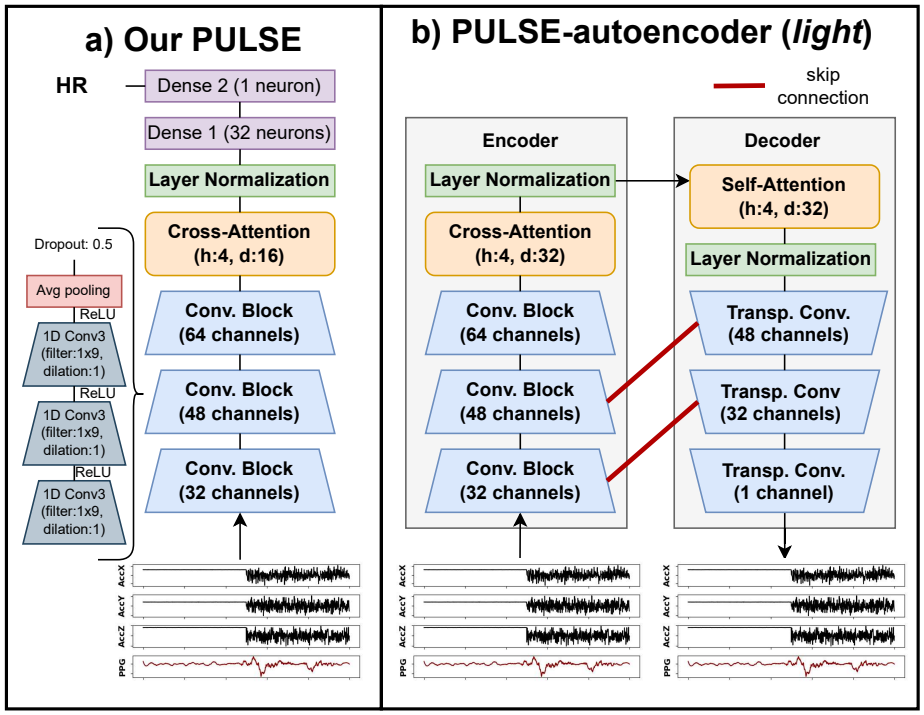


Figure 4.9: Architecture of *light* autoencoder based on PULSE for signal reconstruction pre-training task, compared to our PULSE architecture.

As in PULSE model, the input signal is processed using three convolutional

blocks with increasing channels (i.e. 32, 48 and 64). Afterward, the produced feature maps are fed to the MHCA module, which helps to catch dependencies between the PPG and accelerometer data. The attention output is then normalized with a normalization layer, obtaining our latent space. The decoder is designed to reconstruct the original signals from latent space and for this purpose it mirrors the structure of the encoder. Similar to the encoder, the decoder also employs multi-headed attention mechanisms. In this case, however, the input tensor is correlated with itself. Afterward, transposed convolutional blocks attempts to reconstruct the spatial dimensions of the input. Inspired by U-net [27] architecture, two skip-connections have been added: after each up-sampling step, features from the corresponding block of the encoder are concatenated. As a result, the first two convolutional steps of the decoder path consist of an upsampling of the feature map that doubles the number of feature channels, as they consider a concatenation with the corresponding feature map from the contracting path. This skip-connection approach helps in retaining the spatial information which might get lost during down-sampling.

Heavy **Autoencoder Architecture**

Next, we implemented a more complex and deeper version compared to the *Light* one. This architecture uses a higher number of convolutional blocks in the encoder, allowing a more detailed and richer representation of the signal features. Instead of three blocks, the *Heavy* architecture uses four convolutional blocks, with increasing number of channels (i.e. 32, 48, 64 and 80), as shown in Figure 4.10. This increase in complexity allows the encoder to capture a wider range of patterns in the data, improving the quality of the latent representation. Also in this case, the decoder mirrors the structure of the encoder to reconstruct the original signal. Skip-connections play an even more crucial role in this version, helping to maintain high spatial resolution in the reconstruction stages.

4.3 Self-supervised learning

Self-supervised learning (SSL) is a machine learning approach that helps models learn from the information contained in the data themselves, without the need for labels and it is useful when annotated data is scarce or difficult to obtain. SSL has emerged as a dominant paradigm because of its ability to learn from huge amounts of unlabelled data. Initially, a *pretext task* is

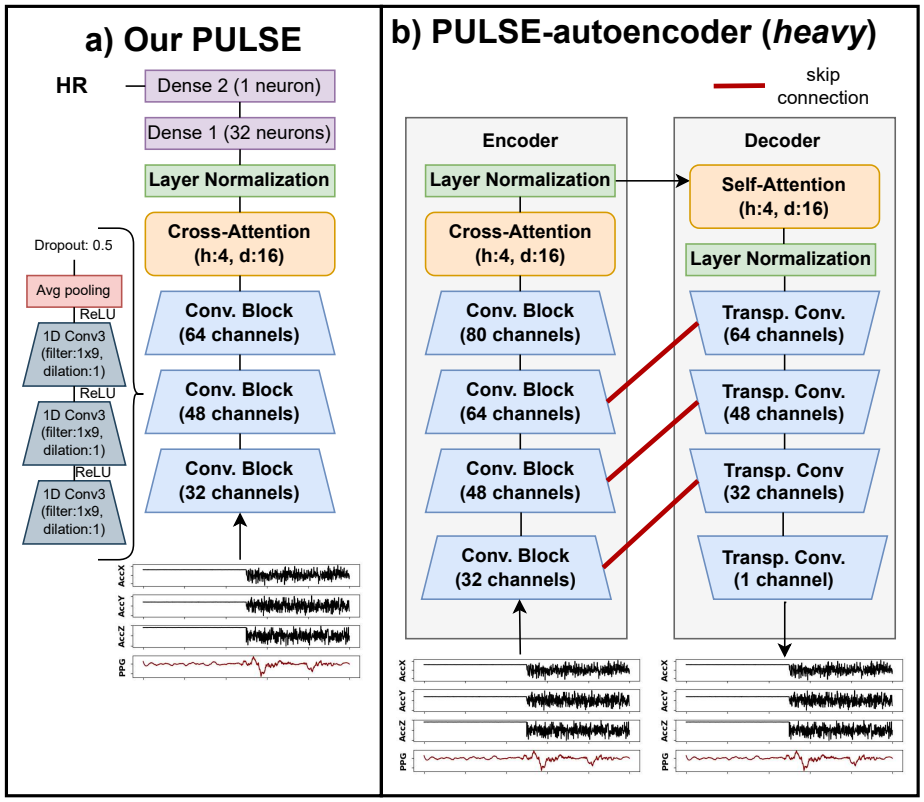


Figure 4.10: Architecture of *heavy* autoencoder based on PULSE for signal reconstruction pre-training task, compared to our PULSE architecture.

defined. This task involves generating pseudo-labels from specific properties or transformations of the unlabeled data, without human intervention. This phase is also called *pre-training*. After the model is trained on the pretext task, it has learned useful representations from the data. These representations capture meaningful information about the data's underlying structure, even though the model was not explicitly given any labels during the pretext task. The acquired information are then fine-tuned on a downstream task, which is the task we are truly interested in. This is why this phase is called *fine-tuning*. Usually, the dataset on which the model is fine-tuned is smaller than the one on which it has been trained, and the task is more specific. While SSL has seen most of its success in Computer Vision [29], Natural Language Processing [30][31], it has drawn a lot of attention also in the Medical AI fied [32][33]. To the best of our knowledge, this is the first effort to apply SSL to PPG-data to improve HR estimation. We use SSL to pre-train a generic model, specifically the PULSE model, and transform it into an autoencoder-like architecture, to compress and reconstruct input signals. This approach forces the model to capture meaningful latent representations. During the pre-training phase, the model learns to reconstruct input signals by extracting the most relevant features from unlabelled data, which allows it to recognise the underlying structures of PPG signals. Once the pre-training phase is complete, we go back to PULSE architecture and use the weights learnt during pre-training to initialise the encoder part of the model. This part represents the majority of the architecture, with the exception of the last classification layers. In this way, we exploit the representations already learned to improve the accuracy of the HR estimation during the fine-tuning phase. Figure 4.11 graphically illustrates the pipeline of the entire process.

4.3.1 Pre-training

The model undergoes a pre-training phase where it is optimized for the signal reconstruction task and a fine-tuning phase trough transfer learning for HR estimation, described in details in Section 4.3.2. During pre-training, we use the Mean Square Error (MSE) as the reference metric, defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i represents the true values, $\hat{y}_i$ represents the predicted values, and n is the number of data points. The goal of pre-training task is to minimise

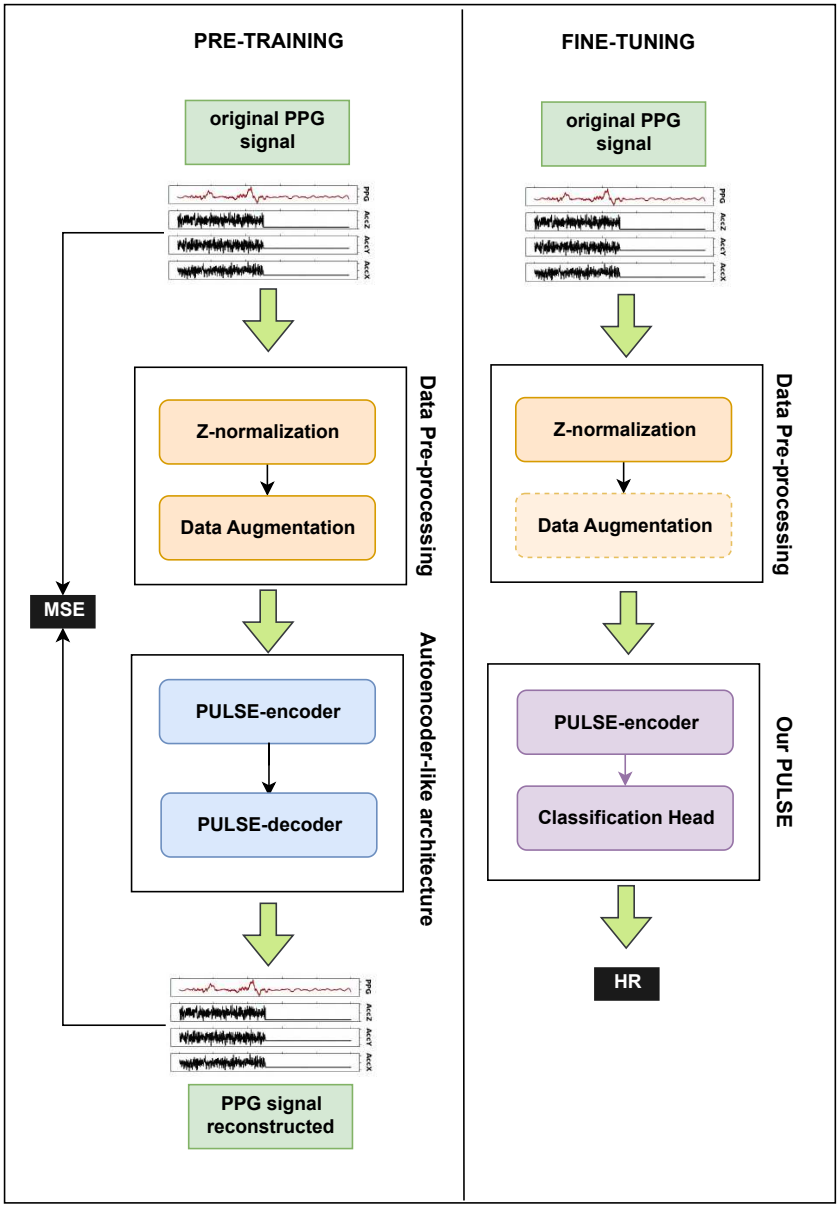
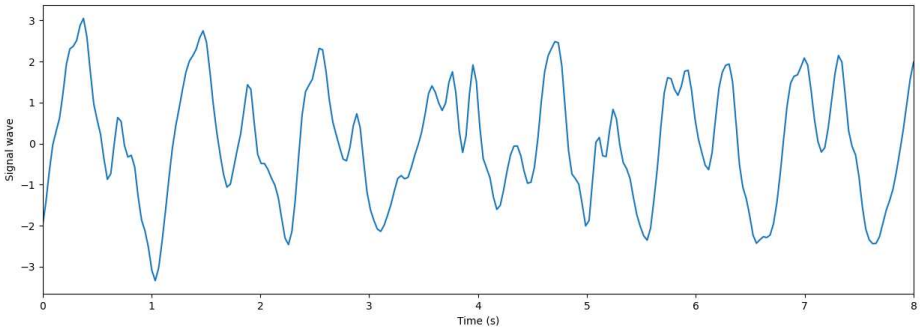


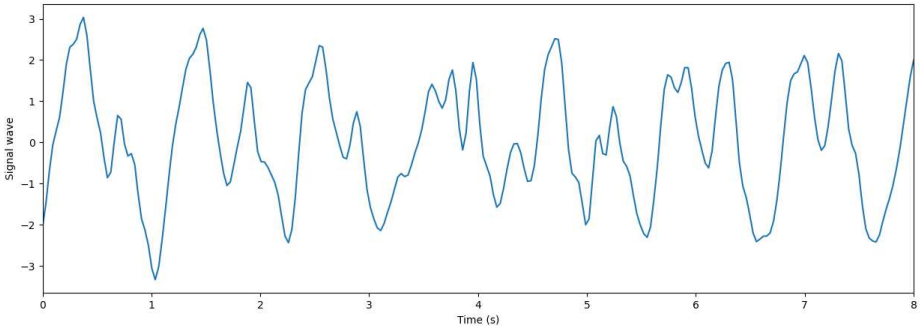
Figure 4.11: PULSE pre-training and fine-tuning pipelines.

MSE between the original and the reconstructed signals. Pre-training has multiple benefits. By training the model on a generic task such as signal reconstruction, it develops a more abstract understanding of the data, without being bound to a specific task, reaching a greater generalization. This

approach improves the robustness of the model to adapt to different types of signals. Pre-training also helps to learn the intrinsic patterns of the PPG signal, providing the model with a solid base as it captures key information that will be useful during fine-tuning. SSL reduces the need for large amounts of labelled data and prepares the model for fine-tuning, where it uses the learned representations to specialise in the heart rate estimation task. Pre-training, therefore, not only improves the overall performance of the model but also its ability to adapt to new data, making the model more efficient and accurate. Figure 4.12 shows a graphic example of the PPG signal reconstructed via autoencoder.



(a) Original PPG signal.



(b) Reconstructed PPG signal.

Figure 4.12: PPG signal reconstruction example.

4.3.2 Fine-tuning

Once the model has learned general representation of PPG data during pre-training, we exploit these during the fine-tuning, where the model learns to predict the hearth rate of each time window. In the fine-tuning phase, the decoder is removed and the classification head, composed of two linear

layers, for HR estimation is added to reconstruct the initial PULSE model. The pre-trained weights learned in the previous phase are used to initialize the layers shared between the two architectures, i.e. the layers belonging to the PULSE-autoencoder part of the architecture. We use the Mean Absolute Error as the reference metric, defined as the differences between the predicted HR and the true value from the dataset:

$$\text{MAE} = |HR_{true} - HR_{pred}|$$

A lower MAE indicates that the model is better at predicting targets. To ensure a robust and sufficiently generic model, we validated all models according to the Leave-One-Session-Out (LOSO) cross-validation protocol proposed in [23]. LOSO allows to test the model on sessions completely independent of those used for training, simulating real scenarios where the model must adapt to new subjects not previously seen. The LOSO protocol consists of dividing the dataset into distinct sessions, each representing a specific subject or condition. At each iteration, one session is dropped from the training set and used as the test set. This process is repeated until each session has been used once as a test set.

4.4 Post-processing

The last component of our methodology is post-processing, which we apply directly on the output of PULSE model. This step is independent from the others and is related to the need to deal with the unpredictable errors that data-driven models may generate, especially when the processed inputs differ significantly from those used during the training phase. As described in [22], during continuous heart rate monitoring, it is reasonable to implement a simple filtering mechanism on the outputs. Indeed, we can assume that a prediction that deviates significantly from previous ones represents a model error. The post-processing methods clips output values in case the prediction is more or less 10% of the averaged 10 last estimated values. Specifically, the latest PULSE prediction HR_n is compared with the average of the previous N, $E_{n,N}$ = E[HR_{n-1},..,HR_{n-N}]. If the difference between these two values is larger than a threshold P_{th}, the estimate is clipped to $HR_n = E_{n,N} \pm P_{th}$, with N = 10 and P_{th} =$E_{n,N}$ /10, identical for all patients.

Chapter 5

Results

5.1 Datasets

In this project, three different datasets have been used to train and test the models previously described: PPG-DaLiA [6], WESAD [7] and an additional private dataset provided by West Attica University. Every sample is comprised of one-channel PPG (sampling rate 64Hz) and tri-axial accelerometer (sampling rate 32Hz), used to compensate motion artefacts. It is common practice in heart rate tracking via PPG signals to use a sliding window approach characterized by 8 seconds length window with 2 seconds shift, as described in Section 4.1. This means that all data signal is segmented with this sliding window, and the goal is to determine the heart rate on each 8-second window segment. In this project only PPG-DaLiA has been used for HR estimation, since it is the only dataset that provides labels extracted from the ECG-signal that can be used as HR ground truth. On the other hand, WESAD dataset and the data provided by West Attica University dataset do not have labels that represent HR ground truth for the 8-second segments. This is why they were used for the pretraining task. All datasets are described in detail in the following paragraphs.

5.1.1 PPG-DaLiA

PPG-DaLiA is a large dataset with a wider range of activities performed under natural and close to real-life conditions used for motion compensation and HR estimation. The dataset includes 15 subjects, eight female and seven males, aged 21–55 years, who were asked to perform 8 everyday activities (walking, driving, walking upstairs/downstairs, playing table soccer, eating,

working, cycling). These activities are also included as extra labels for human activity recognition. The dataset was gathered using an E42 [34] wrist-worn device placed on the subjects' nondominant wrist, capturing one-channel PPG (at a 64Hz sampling rate), tri-axial accelerometer (at a 32Hz sampling rate), Electrodermal Activity (EDA) (at a 4Hz sampling rate), and body temperature (at a 4Hz sampling rate) signals. Simultaneously, the subjects wore a smart belt on their chest to collect ground truth ECG labels. The ground truth heart rate was calculated as the average instantaneous heart rate within each 8-second window. This dataset has approximately two-hour recording sessions per subject, that corresponds to 64,697 samples in total after segmentation (8-second window with 2-second shift).

5.1.2 WESAD

WESAD is a multimodal dataset for wearable stress and affect detection. Data collection was conducted using two wearable devices: a wrist-worn device equipped with sensors for PPG, accelerometer, electrodermal activity, and body temperature, and a chest-worn device equipped with sensors for ECG, accelerometer, respiration, and body temperature. The primary goal behind this dataset was to collect a wide amount of data to identify and differentiate various affective states, such as neutral, stress, and amusement. As a result, WESAD mainly consists of data recorded during sedentary activities. PPG and accelerometers are sampled at 32Hz while ECG signal at 700Hz. Unlike typical PPG datasets, the labels in WESAD do not directly correspond to heart rate values; instead, they are discrete numbers representing specific states (e.g., 0 = not defined / transient, 1 = baseline, 2 = stress, 3 = amusement, 4 = meditation...). It is possible to easily extract the HR value from the ECG signals, but this step is not necessary as this dataset will only be used for an unsupervised task. The data collection involved 15 participants, aged between 24 and 35 years, each contributing approximately 100 minutes of data, that corresponds to 43,385 samples in total after segmentation (8-second window with 2-second shift).

5.1.3 Dataset from West Attica University

This dataset has been collected by West Attica University. The data were acquired using the Empatica E4 device, a wearable wristband that allows monitoring physiological parameters, including the PPG and accelerometer signal. Data collection was carried out on a group of healthy subjects with

no diagnosed heart disease. During the collection process, the participants were mainly in a resting condition. This dataset consists of 449,544 samples in total after segmentation (8-second window with 2-second shift).

5.2 Experimental Set-up

We used Python 3.7 and PyTorch framework to design and train the neural network. All the datasets are downsapled to 32 Hz, using input windows of dimension 4×256, normalized with a per-channel z-score normalization, as described in Section 4.1. We used the pretraining and fine-tuning protocols described in the previous chapter, measuring accuracy of our models through the Mean Absolute Error (MAE). Finally, we applied the post-processing method outlined in Section 4.4, where output values are clipped if the prediction deviates by more than 10% from the average of the last 10 estimated values.

We selected Adam [35] as network optimizer, having the following hyperparameters:

- During pretraining: learning rat equal to 0.001, β_1:0.9, β_2: 0.95, and ϵ: 1×10^{-8}. The learning rate is adjusted using a half-cycle schedule after the warm-up phase, ensuring a smooth decay of the learning rate over the remaining epochs.
- During fine-tuning: learning rate equal to 0.0005,β_1: 0.9, β_2: 0.999, ϵ: 1×10^{-8}.

The pre-training is performed on a large combined dataset including WESAD and West Attica University data, with a total of over 490,000 samples. For the Masked autoencoder pre-training, the batch-size is set to 128 and the pre-training loop consists of 200 epochs but to prevent over-fitting this number represents an upper bound, as an early stop with 20 epochs patience occurs when the loss starts to increase or no longer decreases consistently. Instead, for PULSE-based autoencoder pre-training the batch-size is set to 512 and the pre-training loop consists of 500 epochs with 50 epochs patience. In both cases, we save model parameters as soon as a new minimum of the loss function is found. Once the pre-training phase is complete, the stored checkpoints are used as the starting point for the heart rate estimation task. The fine-tuning is performed on the PPG-DaLiA dataset. We validated all models according to the Leave-One-Session-Out (LOSO) cross-validation protocol

proposed in [23], where the 15 subjects are divided into four data folds; 3 are used as the training set, while the remaining one is divided to form the test set (1 subject) and the validation set. For HR estimation through Masked autoencoder architecture the batch-size is set to 128 and the training cycle for each patient consists of 200 fine-tuning epochs, with an early stop on the validation fix to 20 epochs. The batch-size for PULSE heart rate estimation is set to 256 and the training cycle consists of 500 fine-tuning epochs, with a patience of 150 epochs.

5.3 Masked Autoencoder results

As explained in Chapter 4, in the first part of this work we focused on a Masked Autoencoder model, to establish a benchmark that would serve as a reference model. According to the self-supervised learning approach, we first pre-train the Masked Autoencoder on WESAD and West Attica University datasets to be able to extract good features from data by reconstructing the input signals in frequency domain. Once the model has learned general representation of PPG data during pre-training, we exploit these during the fine-tuning phase, with the transfer-learning approach. The last part of the network was slightly modified to adapt it to the final HR estimation task where the models learns to predict the heart rates of the PPG-DaLiA subjects.

Table 5.1: MAE Post-processing Results on PPG-DaLiA.

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**Mean**
Masked Autoencoder	5.36	5.30	3.65	5.84	17.04	5.91	3.28	8.15	8.08	4.31	7.46	5.84	3.52	4.28	4.83	6.19
Masked Autoencoder with PT	5.21	5.38	3.50	5.93	13.63	6.82	3.22	8.42	7.92	4.05	6.94	5.91	3.45	4.17	4.76	5.95

Table 5.1 shows the results obtained using the Masked Autoencoder and its pre-trained version for heart rate estimation on PPG-DaLiA dataset subjects. The evaluation parameter used is the MAE, expressed in BPM, calculated for each subject, with a final average reported in the last column. The results show that the Masked Autoencoder without pre-training achieves an average MAE of 6.19 BPM. There are significant differences between subjects: for example, for S5 the error is very high (17.04 BPM), whereas for other subjects such as S7 it is much smaller (3.28 BPM). This could suggest that the model generalises better for some subjects than others, possibly due to

MA in the PPG signals or different physiological conditions. When pre-training is performed, there is an overall reduction in the average error to 5.95 BPM. This demonstrates the effectiveness of pre-training on different datasets, as it allows the model to learn more robust representations of the data, which are then refined during fine-tuning for the specific task of heart rate estimation. However, although pre-training generally reduces the error, there are still particular cases (such as S5 and S6) where the error remains relatively high, indicating the possibility of further improvements.

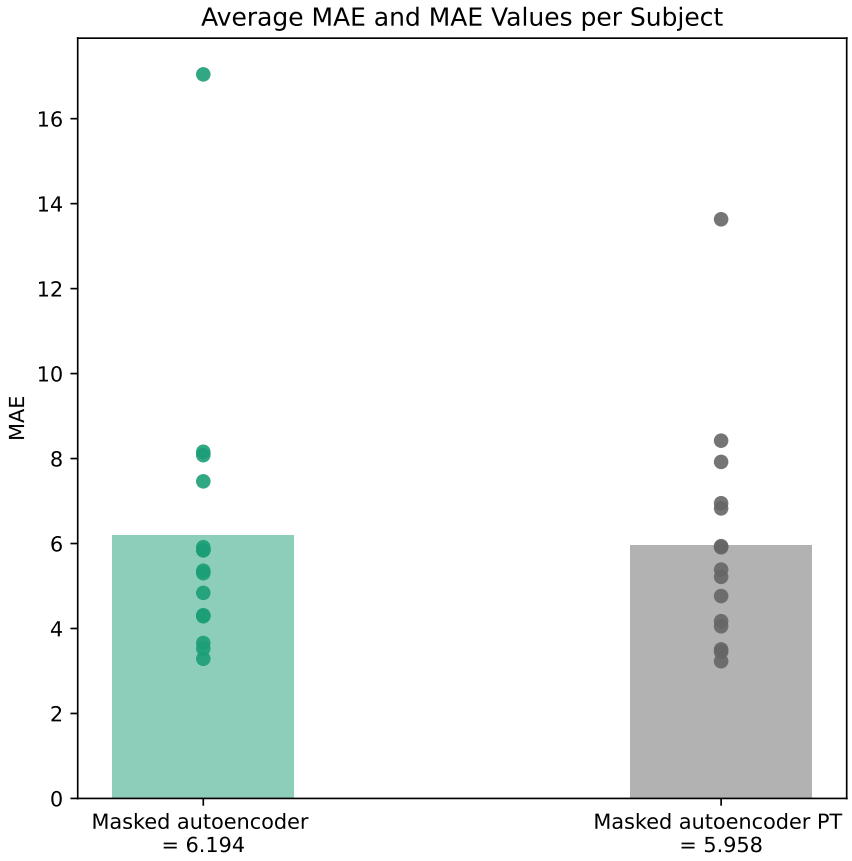


Figure 5.1: Results of HR estimation through Masked autoencoder.

Figure 5.1 shows the results of HR estimation with the Masked Autoencoder architecture through the use of a bar chart: bars represents the average MAE BPM across all patients, while dots represent the MAE BPM of each single patient. This visualization allows us to understand if there are critical patients, i.e. those patients that experience a MAE BPM which is too high to be considered for real-world deployment. It is evident that pre-training has a

positive impact on the performance of the Masked Autoencoder, improving the model's ability to generalise to a complex task such as heart rate estimation. However, even with pretraining, this model did not meet the desired performance in HR estimation.

5.4 PULSE results

In this section, we present experimental results from various configurations of our approach on the selected deep learning models, i.e. PULSE. We explore the impact of several techniques on the baseline PULSE model, including modifications to the original architecture, data augmentation, pre-training with an autoencoder, and combinations of these methods. The aim is to assess how each approach influences the accuracy of heart rate estimation. We begin by examining our version of PULSE model, with standard convolution instead of dilated ones, followed by the application of data augmentation techniques during training to improve model performance. Next, we investigate the effects of pretraining using an autoencoder combined with the PULSE model. Finally, we integrate data augmentation during both pre-training and fine-tuning. Each subsection provides a detailed analysis of each approach and the corresponding results, highlighting its strengths and possible limitations.

5.4.1 Baseline Model (Our PULSE)

First of all, the PULSE model described in Section 4.2.2 is replicated to obtain baseline results, which are then used as a reference for comparing the outcomes of subsequent approaches. We modify the original architecture, presented by Kasnesis et al. [23], by changing the convolutional kernel size from (1, 5) with dilation 2 to (1, 9) with dilation 1, switching from dilated convolution to standard one. In fact, the larger kernel size enhances the detection of subtle, short-term variations in PPG and accelerometer signals, while dilated convolutions with dilation 2 may miss critical local fluctuations. Additionally, standard convolutions reduce the over-smoothing effect of dilation and improve noise handling. As a result, thanks to this modification, we have improved heart rate estimation performance while maintaining efficiency. We use the LOSO training protocol described above. Table 5.2 shows the MAE after post-processing of the HR estimate on PPG-DaLiA, together with results reported in the original paper [23] and in Deep PPG [6] and

Table 5.2: MAE Post-processing Results on PPG-DaLiA.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Mean
Our PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	**3.73**
PULSE [23]	3.78	3.04	2.20	4.41	6.95	3.71	2.39	8.17	6.19	2.60	3.85	5.22	1.98	3.13	2.79	4.03
Deep PPG [6]	7.73	6.74	4.03	5.90	18.51	12.88	3.91	10.87	8.79	4.03	9.22	9.35	4.29	4.37	4.17	7.65
Q-PPG [22]	3.78	3.36	2.33	4.84	9.95	4.38	2.20	5.88	7.59	2.74	4.55	5.20	2.14	2.99	3.47	4.36

Q-PPG [22] approaches. The results demonstrate that the changes made to our version of the PULSE model allow to obtain results comparable to those of the original model, with an improvement in performance in some subjects. In fact,standard convolution let the model focus on more localized features and reduce the complexity associated with capturing temporal dependencies, thereby enhancing the model's performance in dynamic environments.

5.4.2 Data Augmentation results

To further improve the performance of the PULSE model compared to the baseline results, we applied the two DA configurations, described in Section 4.1.2, to the dataset used during the training phase. Also in this case, we use the LOSO training protocol described in 5.2. We act only on the input data, keeping the model unchanged. Table 5.3 shows the post-processed MAE for each subject in the PPG-DaLiA dataset using two DA configurations increasing the training set by 8 and 11 times respectively.

Table 5.3: PULSE MAE Post-processing Results on PPG-DaLiA with Different DA Configurations.

Augmentation	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Mean
Our PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	3.73
8x	3.18	3.18	1.97	4.23	5.69	2.36	2.18	7.23	7.55	2.65	3.40	4.88	2.01	2.49	2.56	**3.71**
11x	3.93	3.79	2.36	5.21	**4.33**	2.84	2.34	**6.78**	6.23	3.53	3.31	**3.93**	2.06	2.69	3.27	3.78

The PULSE model without augmentation achieves a mean MAE of 3.73. This is used as a benchmark to assess the effectiveness of the different DA techniques applied subsequently. The 8x augmentation, which includes all the techniques presented by Burrello et al. [20], shows an average MAE of 3.71, very similar to the configuration without augmentation. This suggests that the augmentation applied in this configuration does not lead to a significant improvement in the average model performance. However, there is a reduction in MAE for subject S5 (from 7.67 to 5.69), suggesting that augmentation may be effective for certain subjects. The 11x augmentation,

which focuses on techniques that extend the BPM range, achieves an average MAE of 3.78, slightly higher than the baseline model without augmentation. However, this configuration shows the most significant improvement for S5 (4.33 MAE), suggesting that DA focused on BPM range expansion may improve performance for patients with a different average BPM. On the other hand, higher MAE on some subjects indicates that, for subjects with a HR range already well-represented in the original data, DA is not always useful, since it can increase the noise in the training data.

5.4.3 Pre-training results

The main objective of this step is to test whether pretraining based on signal reconstruction can improve performance during subsequent fine-tuning by learning meaningful representations of the PPG signal. Next, we evaluate the impact of DA on model accuracy. Specifically, we analyse (1) the application of DA during pretraining only, in order to make the autoencoder more robust to different patterns of the PPG signal, and (2) the application of DA during both pretraining and fine-tuning with the PULSE model. This approach allows us to understand the effectiveness of augmentation at different stages of the learning process and to measure its impact on the final model accuracy. In following subsections, we present the results obtained using the *light* autoencoder, which has proven to be the most effective in terms of performance. The results of the *heavy* autoencoder model can be found in the appendix.

PULSE-based *Light* Autoencoder

First, we use the autoencoder described above in 4.2.3 to perform the pre-training on signal reconstruction task. Pre-training is performed using the signals from the WESAD dataset, containing 449,544 samples, and the dataset provided by West Attica University, with 43,385 samples, as a training set. Once the pre-training is completed, transfer-learning is performed as described in Section 4.3.2. Table 5.4 shows the post-processed MAE for each subject in the PPG-DaLiA dataset after fine-tuning with transfer-learning. The application of pre-training with the autoencoder followed by fine-tuning with the PULSE model leads to a slight reduction in the average MAE to 3.67. Although the improvement is not significant in all cases, the results obtained suggest that pre-training the model with an autoencoder can indeed improve the overall performance of the PULSE model. However, the

Table 5.4: PULSE MAE Post-processing Results on PPG-DaLiA with transfer-learning (TL).

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Mean
PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	3.73
PULSE with TL	3.08	3.67	1.91	4.18	7.28	2.61	1.83	8.23	6.27	2.30	3.22	4.94	1.76	2.29	2.56	**3.67**

inter-subject variability highlights the importance of further experiments to optimise pre-training and to explore data augmentation approaches during pre-training and fine-tuning to improve model accuracy.

PULSE-based *Light* Autoencoder with Augmentation on Pre-training

The pre-training dataset is then further expanded using the two data augmentation techniques described in Section 4.1.2. We act only on the pre-training input data, keeping the model and the training protocol unchanged. Table 5.5 shows the MAE results of the PULSE fine-tuned model on PPG-DaLiA dataset, with the application of the different DA configurations during pre-training.

Table 5.5: PULSE MAE Post-processing Results on PPG-DaLiA after transfer-learning with different DA configurations applied during pre-training (PT).

Augmentation	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Mean
Our PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	3.73
8x during PT	3.32	3.07	2.02	4.44	**5.40**	2.50	1.91	8.25	8.45	2.65	3.37	7.40	12.81	2.45	3.03	4.74
11x during PT	3.35	3.15	2.20	4.38	5.64	2.35	1.93	**5.16**	6.38	2.87	3.30	5.49	1.84	2.43	2.53	**3.53**

Expanding the pre-training dataset 8-times (*8x augmentation*), mean MAE increased to 4.74. The application of DA brings improvements for specific subjects such as S5 (5.40 vs. 7.67 without DA), but worsens results in other cases such as S8 (8.25 vs. 7.61) and S13 (12.81 vs. 1.84). This indicates that the addition of synthetic data introduced variability that made pre-training less effective for some data patterns. By expanding the pre-training dataset 11-times with techniques that expand the BPM range, the average MAE drops to 3.53. Significant improvements are shown in problematic subjects such as S5 and S8, demonstrating that more synthetic data can actually improve model learning by providing a more varied and robust pre-training set. However, we have to underline that this improvements are not uniform across all subjects, suggesting a sensitivity to DA effects.

PULSE-based Light Autoencoder with Data Augmentation on both Pre-training and Fine-tuning

In the last step, we evaluate the effect of applying data augmentation not only during pre-training but also in the fine-tuning of the PULSE model. The same two augmentation techniques described above are applied to further expand the fine-tuning data set. Table 5.6 shows the post-processed MAE for each subject in the PPG-DaLiA dataset using two DA configurations increasing the training set of both pre-training and fine-tuning by 8 and 11 times respectively.

Table 5.6: PULSE MAE Post-processing Results on PPG-DaLiA after transfer-learning with different DA configurations applied both during pre-training and fine-tuning.

Augmentation	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Mean
Our PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	**3.73**
8x	3.23	3.30	8.23	9.57	5.16	2.45	1.96	7.21	9.25	2.77	2.99	3.76	1.91	2.78	2.77	4.48
11x	3.58	3.38	2.17	4.52	**3.76**	2.53	2.17	7.60	7.66	2.89	3.82	5.36	2.18	2.70	3.11	3.83

The expansion of both pre-training and fine-tuning dataset 8-times (*8x augmentation*) resulted in an average MAE of 4.48, higher than baseline. Although in some subjects the MAE remains stable or slightly better than baseline, subjects such as S3, S4 and S9 show a significant worsening in performance. This may be due to the fact that the huge augmentation of the dataset introduced variability that the model was unable to handle effectively during fine-tuning. Additionally, augmentation applied to the fine-tuning dataset requires modifying the labels, which can lead to poor generalization to unseen data. In the configuration with an 11x expansion of the dataset, the average MAE drops to 3.83, an improvement over the 8x configuration but still slightly worse than the baseline. This suggests that while increasing the dataset size through augmentation can provide more diverse data for training, the process of label modification introduces challenges in maintaining the model's ability to generalize effectively to new, unseen samples.

5.5 Discussion

The overall results of the experiments conducted highlight several key aspects. Our approach prove to be effective regardless of the model used,

demonstrating the success of SSL pre-training and data augmentation techniques in improving performance in HR estimation from PPG signals. We started with the Masked Autoencoder architecture. However, we obtained unsatisfactory results with respect to HR estimation, especially in the presence of MA. Nevertheless, pre-training on this architecture proved to be effective, allowing the model to learn meaningful representations of the PPG signal. This had a positive impact in the fine-tuning phase, demonstrating that even if the Masked Autoencoder architecture is not the most suitable for heart rate estimation, pre-training still plays a key role in optimising performance. In fact, introducing pretraining allows the MAE to reduce from 6.19 to 5.96, improving performance by 3.8%.

Afterwards, we moved to the PULSE architecture, i.e. the state of the art for HR estimation on PPG-DaLiA. Switching to PULSE confirmed the effectiveness of our pre-training and data augmentation strategies, which further improved the performance of the model. We restructure PULSE into an autoencoder-like framework, testing two different architecture, the *light* and the *heavy* autoencoder. Results shows that the performance improvement of HR estimation from PPG signals depends on the architecture used and the stage at which the augmentation techniques are applied. The results of the *heavy* autoencoder model can be found in the appendix. As highlighted by the charts in Figure 5.2, the *light* autoencoder performs better than the baseline PULSE model, with a significant improvement in performance. The original PULSE model achieved a MAE of 4.03 BPM using dilated convolutions, representing a starting point for our work. By modifying the architecture and replacing the dilated convolutions with standard convolutions, we are able to reduce the MAE to 3.73 BPM. This corresponds to an average error reduction of approximately 7.4%. At the beginning, we only apply DA techniques during the fine-tuning phase, without pre-training. The average MAE do not improve significantly, remaining around 3.71 and 3.78 BPM, depending on the DA configuration applied. This result could be due to the fact that applying DA at this stage requires changing HR labels, which can lead to insufficient generalisation to unseen data, making the learning process more difficult. However, in the DA configuration that expands the BPM range (*PULSE_11x*), although the MAE is slightly higher than in the baseline model, the variance between individual subjects decreases. By introducing pre-training through the *light* autoencoder, the finetuned PULSE shows improved performance, achieving a MAE of 3.67 BPM. Pre-training ensures that the model can learn broader features without compromising its

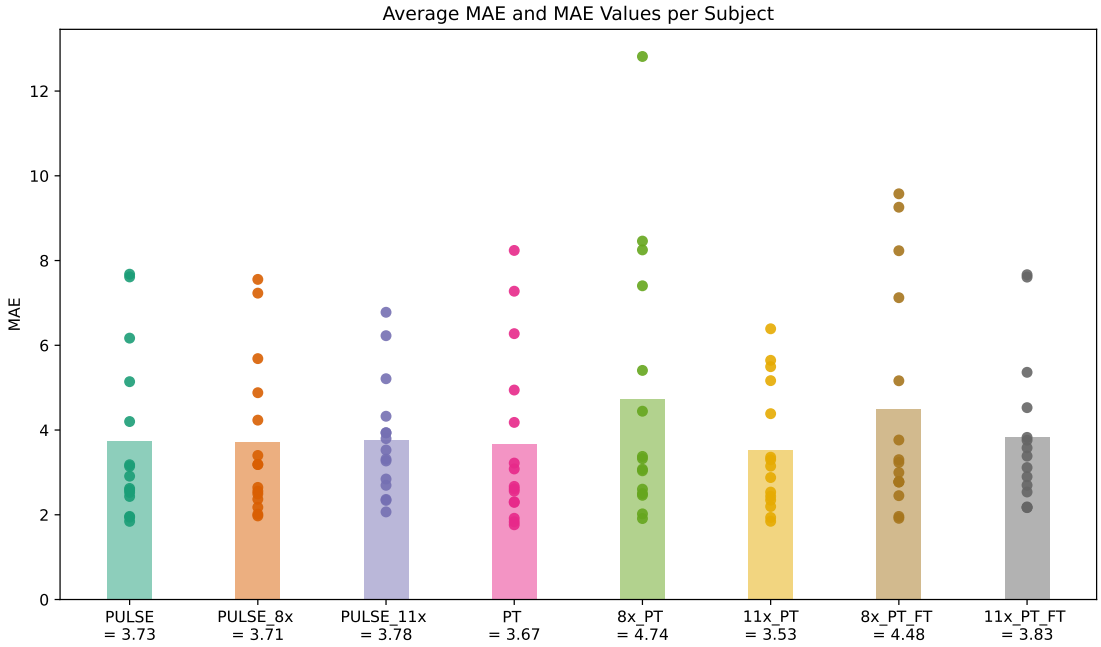


Figure 5.2: Results of *light* autoencoder based on PULSE.

ability to generalize to unseen data. While this improvement is modest, it underscores the potential of SSL, which could deliver even greater performance gains with access to more pre-training data. The application of DA techniques during pre-training further improves the results for some subjects, generating synthetic data that made the model more robust to variations not present in the original dataset. We must underline that DA improvements depend on the kind of DA configuration applied. When the pre-training dataset is increased by a factor of 8 (*8x_PT*), using all DA techniques presented in [20], the average MAE increases to 4.74. This suggests that the addition of synthetic data introduce variability that make pre-training less effective for some data patterns. On the other hand, by expanding the pre-training set only with techniques that increase the range of BPM, the average MAE decreases to 3.53 BPM, confirming our intuition that augmentation at this stage further improves generalisation especially for patients with a different average BPM. With this setup, we achieve a 12.4% MAE reduction compared to the original PULSE model and a 5.3% compared to the modified PULSE. This improvement is due to the larger pre-training dataset and the augmentation techniques that expand the BPM range, helping to reduce overfitting

and enhance model robustness. Using Data Augmentation only during pre-training leads to better results than applying it in both pre-training and fine-tuning. Expanding the fine-tuning set allows more synthetic data to be used, but this interferes with the learning of relevant features on target dataset, leading to performance degradation.

These results highlight the effectiveness of applying augmentation techniques to pre-training data in enhancing state-of-the-art performance, without adding complexity and without altering HR labels. Our approach avoids potential discrepancies between training and test data, as evidenced by the inferior results in applying augmentation to fine-tuning, when HR labels were modified alongside augmentation.

Chapter 6

Conclusion and future works

In this thesis, we addressed the problem of HR estimation from PPG signals, a crucial topic for health monitoring, fitness tracking, and the management of cardiovascular diseases. The main goal was to improve the accuracy and robustness of HR estimation in realistic scenarios, where the presence of MA and inter-subject variability represent significant challenges. To address these challenges, we propose a novel approach that integrates self-supervised pre-training and data augmentation techniques to improve the performance and robustness of deep learning models in estimating HR from PPG signals. A key aspect of our approach is that it is orthogonal to the chosen deep learning model: it can potentially be applied to different architectures, without introducing additional computational complexity. In particular, the combined use of pre-training and DA does not change significantly the inference latency, making it suitable for implementation on low-power edge devices such as smartwatches. To validate our approach, we chose to apply it to the PULSE model, which represents the state-of-the-art for HR estimation from PPG signals. Initially, we replaced the dilated convolutions in the original architecture with standard convolutions, increasing the size of the filters and eliminating dilation. While this architectural modification changed the computational complexity of our model, it allowed to focus on more localised features and improved accuracy. Next, we introduced a pre-training phase, restructuring the PULSE model into an autoencoder-like architecture, inspired by U-Net. This phase allowed the model to learn more general and robust representations of the PPG signal, improving its ability to extract

relevant patterns even in the presence of MA or signal variability. In addition, we applied DA techniques that proved to be effective in increasing the performance of the model, especially for subjects with atypical BPM ranges. Our experiments involved the use of data from the public WESAD dataset and a new unlabelled dataset provided by the West Attica University, creating a pre-training dataset comprising over 490,000 samples. The model was then evaluated on the PPG-DaLiA dataset, the largest labelled dataset for comparison with state-of-the-art models, using the LOSO protocol. These refinements led to a 12.4% reduction in MAE on the PPG-DaLiA dataset, achieving a final MAE of 3.53 BPM compared to the best state-of-the-art competitor, original PULSE, which achieves a MAE of 4.03 BPM. Notably, our approach significantly improved performance for critical patients, making it more applicable for real-world use in wearable health monitoring. In conclusion, our proposed approach, independent of the starting model, proved to be a key element in improving heart rate estimation, and the PULSE architecture proved to be the most suitable to fully exploit these techniques, representing a new benchmark in HR estimation task from PPG signals. Because it is orthogonal to the chosen model, our approach can potentially be applied to other architectures, paving the way for further research to make non-invasive HR monitoring increasingly accurate and reliable.

Despite the promising results, there are still some directions that can be explored for further improvements:

- Expansion of the pre-training dataset: The inclusion of more unlabelled data, e.g. from subjects with different demographic characteristics and physical conditions, could help the model to generalise even better.
- Research on the optimal DA strategies: Exploring new DA techniques, for example based on generative adversarial models (GANs) or the use of more sophisticated transformations, could lead to further improvements in performance.
- Refinement of self-supervised learning objectives: Developing more sophisticated self-supervised learning tasks, such as masking strategies, may help the model's ability to learn more abstract representations, further enhancing HR estimation performances.
- Application to other architectures: Since our approach is orthogonal to the model used, it might be interesting to apply it to different deep

learning architectures to assess how the observed benefits extend to other models.

Appendix

PULSE-based *Heavy* Autoencoder

In this section, we report the results obtained using the *heavy* autoencoder, a more complex architecture than the *light* autoencoder described in Section 4.2.3. All experiments conducted with the *light* model, presented in Section 5.4.3, were also replicated using the *heavy* autoencoder, although the results obtained were not as successful. The *heavy* autoencoder was tested under the same conditions, including the following scenarios:

- Pre-training of the *heavy* autoencoder, followed by PULSE fine-tuning on the heart rate estimation task.
- Pre-training of the *heavy* autoencoder with 8x and 11x Data Augmentation combinations, followed by fine-tuning on PULSE.
- Pre-training of the *heavy* autoencoder with DA, followed by fine-tuning with DA combinations applied to further expand the fine-tuning dataset.

The results show that despite the increase in model complexity and the use of Data Augmentation, the *heavy* autoencoder failed to achieve the same performance as the *light* autoencoder. The larger number of parameters and increased model depth seem to have introduced too much variability, leading to a deterioration in overall performance. Below, Table 6.1 shows the results in terms of MAE for the different configurations tested. The results confirm that the *heavy* autoencoder architecture fails to outperform the *light* autoencoder, despite the increased complexity and the attempt to improve the model through DA. Although there were some improvements in problematic subjects with the application of DA only during pre-training, the addition of DA during fine-tuning led to an overall deterioration in performance. This suggests that the increased complexity of the autoencoder *heavy* and the introduction of synthetic data during fine-tuning may overload the model, making it less efficient in handling variability in real data.

Table 6.1: PULSE MAE Post-processing Results on PPG-DaLiA pre-trained with *heavy* autoencoder.

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**Mean**
PULSE	3.14	2.91	1.96	4.20	7.67	2.53	1.94	7.61	6.17	2.58	3.18	5.14	1.84	2.43	2.62	3.73
PT	3.18	2.86	1.95	4.15	6.11	2.51	1.87	8.16	6.38	2.47	3.12	5.86	1.77	2.51	2.64	3.71
8x PT	3.11	3.05	2.00	4.05	6.44	2.63	1.89	7.62	6.73	2.53	3.21	5.14	1.78	2.42	2.86	3.70
11x PT	3.10	2.87	1.99	4.35	6.05	2.62	1.82	7.74	6.14	2.43	3.63	5.39	1.81	2.44	2.87	3.69
8x PT + 8x FT	3.32	3.43	1.94	4.52	6.84	2.67	1.84	7.12	8.23	2.90	2.23	4.28	1.92	2.38	2.93	3.84
11x PT + 11x FT	3.60	3.74	2.13	4.67	3.67	2.64	2.21	7.87	8.49	3.03	3.15	5.04	1.97	2.53	3.02	3.85

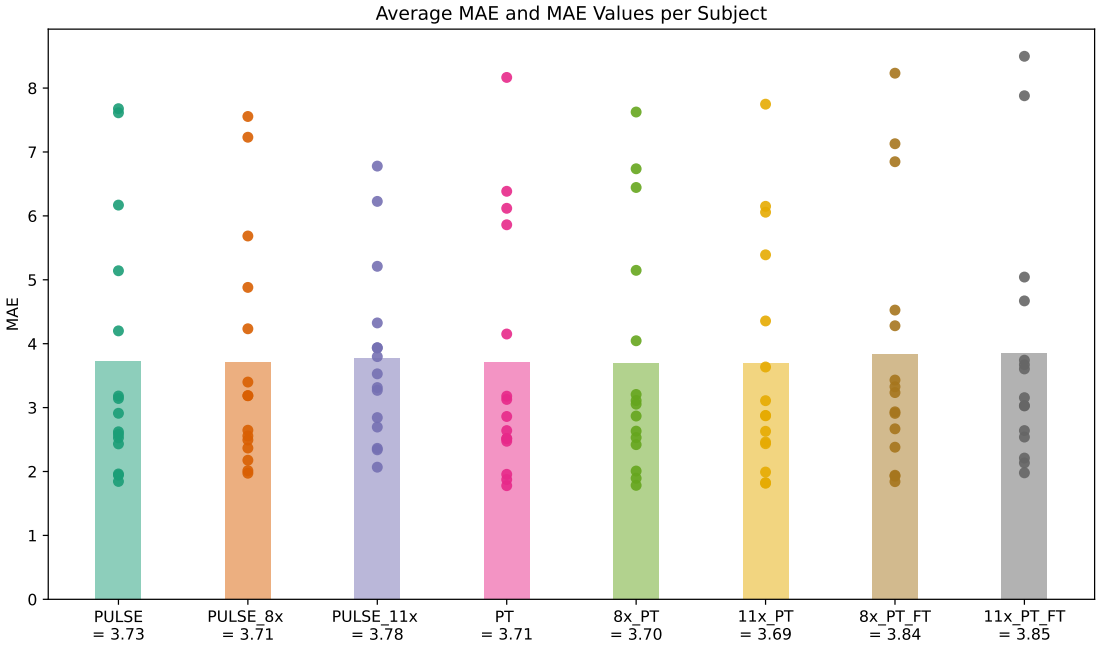


Figure 6.1: Results of *heavy* autoencoder based on PULSE.

Bibliography

[1] Avram, Robert, et al. "Real-world heart rate norms in the Health eHeart study." NPJ digital medicine 2.1 (2019): 58.

[2] Apple. Apple Watch Series. https://www.apple.com/lae/watch/.

[3] Fitbit. Fitbit Charge 4. https://www.fitbit.com/global/us/products/trackers/charge4.

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

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

[6] Reiss, Attila, et al. "Deep PPG: Large-scale heart rate estimation with convolutional neural networks." Sensors 19.14 (2019): 3079.

[7] Schmidt, Philip, et al. "Introducing wesad, a multimodal dataset for wearable stress and affect detection." Proceedings of the 20th ACM international conference on multimodal interaction. 2018.

[8] Avram, Robert, et al. "Real-world heart rate norms in the Health eHeart study." NPJ digital medicine 2.1 (2019): 58.

[9] Heo, Seongsil, Sunyoung Kwon, and Jaekoo Lee. "Stress detection with single PPG sensor by orchestrating multiple denoising and peak-detecting methods." IEEE Access 9 (2021): 47777-47785.

[10] Ye, Yalan, et al. "Combining nonlinear adaptive filtering and signal decomposition for motion artifact removal in wearable photoplethysmography." IEEE Sensors Journal 16.19 (2016): 7133-7141.

[11] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).

[12] Bank, Dor, Noam Koenigstein, and Raja Giryes. "Autoencoders." Machine learning for data science handbook: data mining and knowledge discovery handbook (2023): 353-374.

[13] Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).

[14] Huang, Nicholas, and Nandakumar Selvaraj. "Robust ppg-based ambulatory heart rate tracking algorithm." 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2020.

[15] Nabavi, Seyedfakhreddin, and Sharmistha Bhadra. "A robust fusion method for motion artifacts reduction in photoplethysmography signal." IEEE Transactions on Instrumentation and Measurement 69.12 (2020): 9599-9608.

[16] Zhang, Zhilin, Zhouyue Pi, and Benyuan Liu. "TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise." IEEE Transactions on biomedical engineering 62.2 (2014): 522-531.

[17] Zhang, Zhilin. "Heart rate monitoring from wrist-type photoplethysmographic (PPG) signals during intensive physical exercise." 2014 IEEE global conference on signal and information processing (GlobalSIP). IEEE, 2014.

[18] Lee, Boreom, et al. "Improved elimination of motion artifacts from a photoplethysmographic signal using a Kalman smoother with simultaneous accelerometry." Physiological measurement 31.12 (2010): 1585.

[19] Zhang, Zhilin. "Photoplethysmography-based heart rate monitoring in physical activities via joint sparse spectrum reconstruction." IEEE transactions on biomedical engineering 62.8 (2015): 1902-1910.

[20] Burrello, Alessio, et al. "Improving ppg-based heart-rate monitoring with synthetically generated data." 2022 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2022.

[21] Kumar, Sanjeev, et al. "A wristwatch-based wireless sensor platform for IoT health monitoring applications." Sensors 20.6 (2020): 1675.

[22] Burrello, Alessio, et al. "Q-ppg: Energy-efficient ppg-based heart rate monitoring on wearable devices." IEEE Transactions on Biomedical Circuits and Systems 15.6 (2021): 1196-1209.

[23] Kasnesis, Panagiotis, et al. "Feature-Level Cross-Attentional PPG and Motion Signal Fusion for Heart Rate Estimation." 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 2023.

[24] Kasnesis, Panagiotis, et al. "Replacing Attention with Modality-wise Convolution for Energy-Efficient PPG-based Heart Rate Estimation using Knowledge Distillation." Authorea Preprints (2023).

[25] Zhu, Lingxuan, et al. "Spectralmae: Spectral masked autoencoder for hyperspectral remote sensing image reconstruction." Sensors 23.7 (2023): 3728.

[26] Jeon, Hohyub, et al. "Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals." Applied Sciences 10.20 (2020): 7208.

[27] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer International Publishing, 2015.

[28] Del Pup, Federico, and Manfredo Atzori. "Applications of self-supervised learning to biomedical signals: A survey." IEEE Access (2023).

[29] Jaiswal, Ashish, et al. "A survey on contrastive self-supervised learning." Technologies 9.1 (2020): 2.

[30] Baevski, Alexei, et al. "Data2vec: A general framework for self-supervised learning in speech, vision and language." International Conference on Machine Learning. PMLR, 2022.

[31] Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in neural information processing systems 33 (2020): 12449-12460.

[32] Kostas, Demetres, Stephane Aroca-Ouellette, and Frank Rudzicz. "BENDR: Using transformers and a contrastive self-supervised learning task to learn from massive amounts of EEG data." Frontiers in Human Neuroscience 15 (2021): 653659.

[33] Benfenati, Luca, et al. "BISeizuRe: BERT-Inspired Seizure Data Representation to Improve Epilepsy Monitoring." arXiv preprint arXiv:2406.19189 (2024).

[34] Empatica. E4 wristband 2014. https://www.empatica.com/en-eu/research/e4/

[35] Kingma, Diederik P. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).