UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Pedrosa de Aguiar, Davi

Predicting Heart Rate During Physical Activities Using Artificial Neural Networks

Belo Horizonte
2021

Pedrosa de Aguiar, Davi

Predicting Heart Rate During Physical Activities Using Artificial Neural Networks

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Fabricio Murai Ferreira

Belo Horizonte
2021

Aguiar, Davi Pedrosa de.

A282p Predicting heart rate during physical activities using artificial neural networks [manuscrito] / Davi Pedrosa de Aguiar – 2021.
xxii, 67 f. il.

Orientador: Fabricio Murai Ferreira.
Dissertação (mestrado) – Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.63-67.

1. Computação – Teses. 2. Redes neurais (Computação) – Teses. 3. Frequência cardíaca – Teses. I. Ferreira, Fabricio Murai. . III Universidade Federal de Minas Gerais; Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82.10(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Predicting Heart Rate During Physical Activities Using Artificial Neural Networks

DAVI PEDROSA DE AGUIAR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

Prof. Fabrício Murai Ferreira - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Renato Martins Assunção
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Daniel Sadoc Menasche
Departamento de Ciência da Computação - UFRJ

Belo Horizonte, 15 de Abril de 2021.

Acknowledgments

In the last two years I've had some great experiences for which I'm very thankful. First and foremost, I thank my God for enabling me such great experience. His care has guided me not only the joyful, but also the hard times.

Many people have been part of this journey of mine, some of whom have helped me directly when needed, while others have offered their support. I want to thank them all. I will attempt to acknowledge all people who directly or indirectly contributed to the success of my masters in chronological order.

First, I'm thankful for my parents we were always there, supporting me in my decisions and encouraging me to always improve. I thank my friends, especially Andre Cardoso, Victor Oliveira and the Wrungs whom I could always count on for advice and support.

The beginning of my masters was a period when I felt insecure, as I endeavored into the world of computer science, having had little formal education in this area before. During this period, I was able to count of the advice of professor Dorgival Neto and the help from my colleagues in the e-Speed laboratory, especially Christiano Pimenta and Metheus Nunes. I'm thankful for their patience and assistance.

Also on the first semester, I was invited to join Treinus by Gutenberg Dias, then a classmate of mine in the Automated Machine Learning class. I'm thankful to him for being accommodating and trusting, even while I was a little more than a stranger to him.

Encouraged by prof. Dorgival and Gutenberg, I chose Machine Learning as the area of my research. To be my advisor, I invited professor Fabricio Murai , who promptly accepted. He has been very helpful in my research, offering insight, support, and even direct help, even staying up till late at night helping me meet deadlines for delivering articles to conferences. I very thankful to him.

I feel honored to have met all these special people – and other whom I undoubtedly forgot to mention. I want to tell them all that I deeply appreciate all their support during these years.

“*Praise the Lord, for He is good: for His mercy endures forever*”
(psalm 135:1)

Resumo

A frequência cardíaca (FC) é uma métrica amplamente utilizada por profissionais e amadores no treinamento de resistência por ser uma medida para esforço físico, uma vez que é através dos batimentos cardíacos que oxigênio, nutrientes e hormônios são distribuídos às células de todo o corpo. Essa métrica é importante para a prescrição de exercícios físicos, pois um treinamento efetivo deve provocar uma FC dentro de uma determinada faixa, de forma a treinar nem pouco, nem muito, um indivíduo.

Prever a dinâmica da frequência cardíaca, no entanto, é uma tarefa reconhecidamente difícil, devido a variedade de atributos que a influenciam, que vão desde a nutrição e humor até a genética de um indivíduo. Ainda assim, a atividade física é considerada um dos principais impulsionadores do frequência cardíaca. Alguns estudos modelam a frequência cardíaca usando diferentes medidas para representar atividades físicas específicas, como velocidade e aceleração para corrida ou torque para ciclismo. Embora essas métricas descrevam bem o esforço físico do indivíduo para essas atividades, elas não são gerais o suficiente para descrever a esforço físico em outras, como pular corda.

As medições de sensores de unidade de medida inercial (IMU), como acelerômetros e giroscópios presentes em smartphones e relógios esportivos, têm sido aplicadas com sucesso na previsão da atividade que está sendo realizada pelo indivíduo, uma tarefa amplamente conhecida como "reconhecimento de atividade humana" (HAR). Isso sugere que esses sensores poderiam, a princípio, fornecer representações mais gerais do esforço físico de uma pessoa, mesmo que seja por meio da previsão da atividade. Muito poucos estudos publicados, no entanto, usam sinais de IMU para prever a frequência cardíaca e, os que o fazem, apresentaram algumas limitações sérias, como prever apenas alguns segundos no futuro antes de requerer recalibração ou considerar apenas um único indivíduo em sua avaliação, levando a questionamentos sobre sua aplicabilidade geral.

Nesta dissertação, propomos um novo modelo para estimativa de FC utilizando dados IMU, baseado em Redes Neurais Recorrentes (RNN). A lógica por trás de nosso modelo é que uma mesma atividade provoca diferentes respostas de FC em diferentes indivíduos, dependendo do seu condicionamento físico. Portanto, nosso modelo tenta codificar o condicionamento físico de um indivíduo em um vetor, denominado PCE, usando um módulo especialmente projetado para isso. O PCE é então usado para inicializar os vetores ocultos iniciais de uma RNN, que usa de células do tipo LSTM. Reforçamos essa codificação treinando o modelo em conjunto uma rede que discrimina se dois PCEs pertencem ao mesmo indivíduo.

Avaliamos o modelo proposto ao prever a FC de 23 indivíduos realizando uma variedade de atividades físicas, a partir de dados IMU disponíveis em datasets públicos (PAMAP2, PPG-DaLiA). Para comparação, usamos o único modelo proposto especificamente para esta tarefa e dois modelos em estado da arte para a tarefa de HAR (pela semelhança entre as tarefas). Nosso método, denominado PCE-LSTM, resulta em erro absoluto médio mais de 10 % menor que os demais modelos avaliados. Demonstramos empiricamente que essa redução do erro se deve, em parte, ao uso do PCE. Por fim, usamos dois datasets (PPG-DaLiA, WESAD) para mostrar que o PCE-LSTM também pode ser aplicado com sucesso quando os sensores de fotopletismografia (PPG) estão disponíveis, superando o modelo baseado em redes neurais que é estado da arte em aproximadamente 30 %.

Palavras-chave: Séries Temporais, Aplicações Médicas, Freqûencia cardíaca, Redes Neurais Recursivas

Abstract

The heart rate (HR) is an important metric widely used by professionals and amateurs in endurance training as a proxy to physical strain, as it is through heartbeats that oxygen, nutrients and hormones are distributed to cells in the whole body. This metric is important for the prescription of physical exercises, as an effective training should elicit a HR within a certain range, so as to neither under-train, nor over-train an individual. Predicting the heart rate dynamics, nonetheless, is recognized as a hard task, due to a variety of influencing features, ranging from nutrition and mood to an individual's genetics. Yet, the physical activity is regarded as one of the main drivers of the heart rate. There has been a few studies attempting to model the heart rate using different proxies for physical activities, such as speed and acceleration for running, or torque for cycling. Although these metrics might be good descriptors of the physical strain on the individual, for these specific activities, they are not general enough for describing the strain incurred by other activities, such as rope jumping.

Measurements from Inertial Measurement Unit (IMU) sensor, such as accelerometers and gyroscopes present in smartphones and fitness watches, have been successfully applied in predicting the activity being performed by an individual, a task widely known as Human Activity Recognition (HAR). This suggests that these sensors could, in principle, provide more general representations of one's physical strain, even if it is through the prediction of the activity. Very few published studies attempted using IMU signals to predict the heart rate, and, the ones that did, had some serious limitations, such as predicting over only a few seconds into the future before requiring re-calibration or considering only a single individual, raising some serious questions over its general applicability.

In this dissertation, we propose a new model for HR estimation using IMU data, based on Recurrent Neural Networks (RNN). The rationale behind our model is that the same activity elicits different HR responses in different individuals, depending on the physical conditioning. Hence, our model attempts to encode the physical conditioning of an individual into a vector, dubbed PCE, using a specially designed subnetwork. The PCE is then used to initialize the initial hidden vectors of a RNN, that use long short-term memory (LSTM) units. We reinforce this encoding by jointly training a network which discriminates whether two PCEs belong to the same individual.

We evaluate the proposed model when predicting the HR of 23 subjects performing a variety of physical activities, from IMU data available in public datasets (PAMAP2, PPG-DaLiA). For comparison, we use as baselines the only model specifically proposed for this task and two adapted state-of-the-art models for the closely related task of HAR. Our method, named PCE-

LSTM, yields over 10% lower mean absolute error. We demonstrate empirically that this error reduction is in part due to the use of the PCE. Last, we use two datasets (PPG-DaLiA, WESAD) to show that PCE-LSTM can also be successfully applied when photoplethysmography (PPG) sensors are available, outperforming the state-of-the-art deep learning baselines by more than 30%.

Keywords: Time Series, Medical Application, Heart-Rate, Recursive Neural Networks

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The heartrate (HR), defined as the number of cardiac contraction cycles per unit of time, is an important metric for the cardiac function: these cycle promote blood circulation, which distributes nutrients, oxygen and hormones to cells throughout the body. This characteristic makes the HR a commonly used proxy for strain by both professionals and amateurs in endurance training, in addition to the fact that it is easier to measure in comparison to alternative metrics, such as stroke volume, oxygen uptake and hormone levels [17].

An effective fitness training is promoted by inducing an optimal cardiovascular reaction, making it essential to model and predict individual HR responses. This task is quite challenging as, besides the physical activity per se, the HR is influenced by a number of factors such as genetics, nutrition, environmental conditions, fitness and mood [17].

Attributes, such as speed and acceleration, which are very descriptive of running [6, 41], or the cadence which is a good representative of the cycling effort [18, 21], have been commonly used as attributes for HR estimation. Nonetheless, these attributes are not general enough to describe much of the wide gamut of activities practiced by people.

Human Activity Recognition (HAR) is a task which uses sensor data to predict activities being performed by people, such as climbing the stairs, running, cycling and rope jumping. The signals from accelerometers and gyroscopes – which are widely known as Inertial Measurement Units (IMU) – are commonly used in this task, with some methods achieving prediction accuracy in excess of 90% [26].

The high prediction accuracy using only IMU data suggests that, in principle, these same sensors could be successfully used to describe the physical effort of a large number of activities and, as a result, be a useful input for the HR prediction task. The IMU sensors have the additional advantage of being increasingly common in devices such as smartphones, smartwatches and fitness watches. Figure 1.1 presents an overview of this task, which we refer as IMU-based HR estimation. Like HAR, the time series are usually processed as sliding windows, called time snippets (TS).

The task of multi-step heart rate estimation from IMU sensor data remains little explored, despite recent advances in neural networks and the increasing effectiveness of HAR methods. In this work, we investigate neural architectures that could be used for this task, as this could (i) provide an alternative for users lacking specialized fitness watches which feature

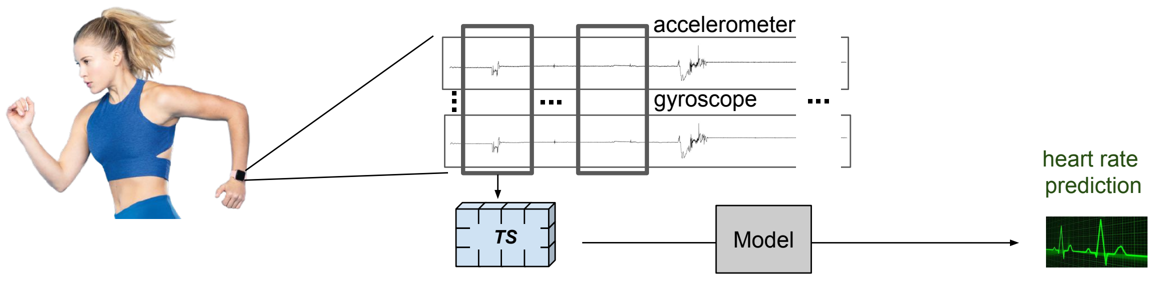


Figure 1.1: IMU-based HR estimation task

HR sensors or (ii) be used for further medical research modeling the impact of physical effort on the HR.

Several methods have been designed to predict the heart rate under the influence of physical activity. Most of them are based on differential equations [6, 11], Hammerstein and Wiener models [20] or Neural Networks [21, 38, 42, 40]. Among the latter, some predict many steps into the future [42, 38], and others use IMU signals as input [40, 38] but, to the best of our knowledge, only the model proposed in [38] does both.

1.1 Objectives

The model proposed by [38] achieved good estimation accuracy, but was limited to a single time-series, from one individual performing his/her usual daily activities. First, using the publicly available datasets PAMAP2 [24] and PPG-DALIA [25], which contain HR and IMU-sensor data from multiple individuals performing a wide range of activities, we aim to evaluate whether their results hold true for other datasets.

Our seconds objective is to devise a new neural network architecture for that task. In order to benchmark the model we propose against modern neural networks, given the similarities between estimating the HR from IMU sensor and HAR, we adapt recent, well-performing, deep learning-based models devised for the HAR task to the HR estimation task.

Third, based on the premise that different subjects will display different HR levels when performing the same exercises depending on their physical conditioning, we aim to investigate whether it is possible to encode the physical conditioning of a person using data from a previous, short-lived activity and use it to improve the performance of a HR prediction model.

A model is more useful when, besides performing well on a specific task, it is easily adaptable and effective in other tasks. Therefore, our last objective is to adapt the model we devised for IMU-based HR estimation to execute the photoplethysmography (PPG)-based HR estimation task, where IMU signals are used to remove noise from the HR measured by PPG sensors.

In summary, the overarching objective of this dissertation is to investigate whether current deep learning methods can accurately predict the HR when provided only with IMU signals and a small sample of past HR dynamics of as a function of IMU signals.

1.2 Contributions

This dissertation revisits the task of IMU-based multi-step HR estimation. This task, though not recently explored, demonstrates renewed research potential due to the newly available public datasets, the late deep learning advancements, and its practical applications, such as enabling an improvement in the prescription of physical activities.

Based on the premise that different subjects will display different HR levels when performing the same exercises depending on their physical conditioning, we propose a novel neural architecture that attempts to encode that attribute given IMU and HR data collected from a previous, short-lived activity performed by an individual. This attribute is extracted by a CNN network as a vector called the physical conditioning embedding (PCE). The PCE is used, in turn, to set the initial values of the hidden and cell states of an LSTM responsible for outputting the HR predictions. Henceforth, the proposed method is referred as PCE-LSTM.

In order to benchmark our model, in addition to using the model from [38], we adapt a network proposed for the closely related HAR task, where the goal is to identify the activity being performed by a person (e.g., running, walking, swimming) given some sensor data (mostly, IMU) as input. For HAR, Convolutional Neural Networks (CNNs) [26, 22] and Recursive Neural Networks (RNNs) [22] are amongst the best performing approaches.

The use of photoplethysmography (PPG) sensors to track heart rate (HR) is also becoming ubiquitous, especially in devices targeting fitness-conscious consumers, such as the Apple Watch, FitBit and Samsung SimBand [25]. Nevertheless, they are prone to measurement errors due to motion-related artifacts. In this case, data from IMU sensors can be used to correct the heart rate measurements. This task, called PPG-based heart rate estimation, has been explored by [27, 29] and [25]. To show that the proposed architecture can also be used for this task, we adapt PCE-LSTM to incorporate the PPG signal as an additional input and show that it can also outperform the state-of-the-art in this task.

In summary, our main contributions are:

1. We propose a new model to predict heart rate from IMU-sersor signals;
2. We compare our model to the only neural network devised for the same task [38] and to minimally adapted models shown to perform well on the related task of HAR [22, 26] and a slightly modified transformer-based model we designed;

3. We conduct an ablation study to assess the contribution of the PCE sub-network in the performance our PCE-LSTM model; and

4. We show our model also outperforms the state-of-the-art for heart rate estimation from PPG data.

1.3 Dissertation outline

This dissertation is organized in such way that its chapters can be read independently. In order to contextualize and help the reader decide the best way to read this document, this section summarizes the contents of next chapters of this dissertation.

Chapter 2 describes existing approaches for HR estimation, similar tasks using IMU-sensors, and current practices for preprocessing sensor signals in the context of HR predictions and HAR, as well as the existing methods for initializing an RNN's initial hidden vectors.

In order to provide a basic background on deep learning for those with limited expertise in that area, Chapter 3 presents the building blocks of the deep learning used in this work. It also describes in detail the original models used as baselines when evaluating our method.

Chapter 4 describes the datasets used, the preprocessing techniques applied to the data, the exploratory analysis we performed, including the two formulations of the HR estimation task: predicting the variation w.r.t. the initial value (delta formulation) and predicting the HR directly (direct formulation). In addition, we describe adaptations to baselines methods used, which were initially designed for the HAR task.

Chapter 5 details the novel aspects of our work, including the key hypothesis we investigate ("It is possible to encode information about physical conditioning from an individual's sensor data as a vector and use it as the initial state of a RNN to improve heart rate predictions?"). We also describe each component of the proposed model.

Chapter 6 describes how the experiments were conducted and their results, in the comparative and ablation studies, in both the delta and direct formulations. We also analyze the results, explaining the general behavior of each model, and their main differences.

In Chapter 7, we show that our method can be easily adapted to a related task: PPG-based HR estimation. We conduct a comparative study to the state-of-the-art using two important datasets.

Chapter 8 concludes this dissertation, wrapping up our contributions and present some suggestions for future, complementary works.

Chapter 2

Related Work

In this chapter we review the main studies related to the task of IMU-based multi-step HR estimation, including numerous approaches taken for estimating the HR, methods used in the related tasks of HAR and PPG-based HR estimation, techniques applied to extracting features from sensor signals, and various approaches to initialize RNN hidden vectors. We review these related tasks not only because they share characteristics with HR estimation, but also because we adapt the proposed model to these tasks in order to to evaluate whether it can be considered a fertile approach.

We find out that a wide variety of approaches have been proposed for predicting the HR. Classical methods used differential equations, while more recent techniques are based on Hammerstein-Wiener models, neural networks and ensembles. We also verify that the majority of studies used task-specific attributes and predicted only a few seconds into the future, though a single study ([38]) tackled both the multi-step prediction task and IMU data as attributes. In general, this studies lack of standardization: most of them demonstrate the strength of their proposed techniques on different datasets, not publicly available, thus hindering a fair performance comparison among them.

However, the HAR task using IMU sensors is much more standardized, with a few widely used datasets, providing a more systematic path for comparing the performance of different methods. Classical approaches have clearly been outperformed by deep learning methods, with LSTM-based and CNN-based methods showing the best performance. More recently, attention-transformer networks, have been shown to be on par with the best performing CNNs in some, but not all datasets [37, 26]. HAR studies also have shown that CNNs are effective feature extractors. However, we also review traditional alternatives available for features extraction in sensor data.

Unlike the IMU-based multi-step HR estimation, HAR does not involve predicting over a sequence of outputs. On the other hand, the task of PPG-based HR estimations, which makes use PPG signals as extra features, shares the important similarity to IMU based multi-step HR prediction of being modeled as a sequence to sequence task. PPG-based HR estimation, has been traditionally tackled by customly devised filtering methods, which have been now outperformed by NN on large datasets, indicating a convergence towards the use of the latter approach in dealing with IMU sensor data.

We also investigate the techniques proposed to initialize Recursive Neural Networks hidden vectors, finding out that techniques using both features and labels form a previous period have been used to encode a RNN's hidden state in the context of dynamics systems to improve the encoding of the dynamics systems initial state.

The following sections provide a detailed review of these topics.

2.1 Heart rate prediction

To model the HR dynamics under physical strain, many studies have limited their scope to specific activities and therefore chosen to use attributes closely related to the source of strain in those activities. The works of [21], [18] and [2] for example, involved the use of cadence – the rate at which a cyclist pedals – and power – the energy spent for pedaling per unit of time – as features for their HR prediction models under the load of cycling exercises. The models proposed in [6], [41], [12] and [42] on the other hand, made use of velocity and acceleration as predictor variables, since the scope of their work was limited to walking and running activities. In attempting to model the HR under general, daily activities, [40] and [38] used wrist-worn accelerometer sensor signals as features for their proposed models.

As various attributes were used for predicting the HR, a variety of modeling techniques have been applied for this task. The works of [6, 41, 18] used differential equations for modeling the HR dynamics, while [12] used 1st to 4th order polynomial regression as the prediction model. [2], on the other hand, predicted the heart rate using a Hammerstein-Wiener model.

Few studies used IMU sensors to predict the heart rate (HR). [40] used a simple Feed-Forward Neural Network to predict the HR one step ahead, given its current value and the average signal of each IMU sensor on the previous step. Quite similarly, [38] performed a multi-step HR prediction by repeatedly using the HR computed for step t to predict the HR for step $t+1$. Their experiments demonstrated promising results, but had some notable deficiencies, such as the use of data from a single individual in his daily activities, therefore without much variation in the HR values.

Also on multi-step HR estimation, but using speed and acceleration as inputs, [42] proposed a Bayesian combined predictor, where one of the estimators was a linear regression and the other a neural network architecture similar to that of [38]. In their experiments, data from multiple individuals performing running sessions was used, thus addressing some of the shortcomings in [38]. However, the proposed method required calibration with actual HR data every 90s, reducing its practical use.

Table 2.1 lists the main studies which proposed models for HR estimation. In the table methods are classified by their underlying model type as Neural Networks (NN), Differential

Equations (DE), or a combination of different types of models (Hybrid). It also classifies prediction span as short-term (predicting a few seconds into the future) or long term (when a model is used to predict at least tens of minutes into the future). Last, it also describes the types of attributes used by each model. We also include our proposed model in the table (PCE-LSTM). It differs from [38] in that we use more recent types of NN architectures, investigate RNN initialization and apply our model to publicly available datasets.

Study	method type	prediction span	attribute type
[6]	DE	short term	speed
[40]	NN	short term	IMU
[38]	NN	long term	IMU
[41]	DE	short term	speed
[18]	DE	short term	cadence and power
[21]	NN	short term	cadence
[42]	Hybrid	short term	speed
PCE-LSTM (ours)	NN	long term	IMU

Table 2.1: Classification of Existing HR Prediction Methods

2.2 Human Activity Recognition

Human Activity Recognition is a task which uses sensor data, commonly IMU, to predict the activity being performed by an individual. [9] studied the performance of Deep Feed-Forward (DNN) Networks, CNN and Long-Short Term Memory (LSTM) models, demonstrating that CNN and LSTM based models outperform DNNs and, among them, the best performing model was very dependent on the dataset used. In particular, for PAMAP2 [24], CNN was the best model.

A hybrid model based on both CNNs and LSTM was devised by [22]. This model transforms the sensor signals using a CNN-based module and the output is then used to feed the LSTM. Although it achieved good results, the CNN-based architectures introduced by [26] had better performance in all datasets which were used in both works.

More recently, some works proposed the use of self-attention based architectures for this task [33, 37]. The model proposed by [26] outperformed the one proposed by [33] in the datasets that were common to both works. [37] proposed a deep attention model that was able to surpass [26] in some, but not all datasets. A more extensive survey was conducted by [35], where the authors review and categorize the variety of neural network architectures proposed for this task as convolutional neural networks, deep fully-connected networks, recurrent neural networks, deep belief networks, stacked auto encoders and hybrid approaches.

2.3 PPG-based HR Estimation

The HR is precisely measured by Electrocardiography (ECG) devices, which use electrodes placed on cleaned areas of the skin, making it inconvenient for users to wear for continuous HR monitoring in their daily lives. An alternative for measuring the HR is Photoplethysmography (PPG) sensors. These are light-based, non-intrusive sensors, widely used in fitness watches, such as Apple Watch, Fitbit Charge and Samsung Simband [25]. User motion, nevertheless, interferes with PPG signals, reducing the sensors accuracy (this interference is referred to as *motion artifacts*). In order to make the PPG-based HR measurements more robust and accurate, a number of methods have been developed, which combine PPG and accelerometer signals to estimate the HR, attempting to remove the inaccuracies added by motion. This is known as PPG-based HR estimation.

Earlier work on the task tended to use filtering methods [23, 15] or techniques to separate the HR component from the noise component in the time domain. The release of two datasets for the IEEE Signal Processing Cup in 2015 [43] stimulated further research on PPG-based HR estimation, and many techniques, mostly based on the frequency domain, were developed [31, 39, 27, 4].

Further datasets have been released: PPG-Motion [13], PPG-Bruce [16], WESAD [28], PPG-DaLiA [25]. Deep learning methods outperformed the classical methods on the largest datasets, namely -WESAD and PPG-DaLiA [25]. This is expected as deep networks' prediction accuracy often continues to improve with the amount of training data due to their large capacity. Table 2.2 shows the mean absolute error of two classical methods [27, 29], a CNN method [25], and our proposed method, in four datasets (these values were transcribed from [25]).

	Shaek2017 [29]	SpaMaPlus [27]	CNN Ensemble [25]	PCE-LSTM Ensemble (ours)
IEEE Train [43]	2.91	4.25	4	6.96
IEEE Test [43]	24.65	12.31	16.51	26.21
WESAD [28]	19.97	9.45	7.47	4.97
PPG-DaLiA [25]	20.45	11.06	7.65	5.22

Table 2.2: Mean Absolute Error (MAE) of various methods on different datasets for PPG-based HR estimation task

2.4 Dealing with Sensor Data

Sensor data, such as those of accelerometers and gyroscopes (IMU), commonly present in smartwatches in the wrist of a person, are hard to be interpreted, especially in 6 degrees of freedom systems (3 for translation, 3 for rotation). Many approaches for extracting useful features from those signals have been proposed in the literature. In this section, we present some of the them, particularly those introduced in studies related to HR estimation and HAR.

[38] and [40] used averages of the accelerometer signals over sliding windows as features of their HR prediction models. Although very simple, this approach was enough for their model to achieve reasonably good results on their experiments.

A more elaborate approach was adopted by [34]. In that study, they extracted 561 statistical features from sliding windows of accelerometer and gyroscope data, and then performed dimensionality reduction, using Principal Component Analysis (PCA), down to 70 features. It showed that the reduction on the number of features slightly lowered the performance of their HAR model, which was compensated by a drastic reduction in training time in comparison to that required by the complete set of features.

Making use of the fact that motion and HR is periodic in nature, [25] transformed the raw signals from the time domain to the frequency domain using Fast Fourier Transforms (FFT), becoming the state of the art in PPG-based HR-estimation.

Building on the success of convolutional neural networks (CNN) on learning the good features for image classification, [22] and [26] used convolutional networks consisting of many layers as part of their models to extract deep features from the raw sensor data. Besides becoming the state of the art at their respective times of publication, this end-to-end neural network approach avoided the need of using human expertise to design adequate features.

Working with raw IMU sensor data is challenging and many approaches have been proposed to solve this task. Starting from simple statistical attributes extraction, approaches have evolved, tending to converge to the use of NN to perform feature extraction, even if custom human-designed transformations, such as the transformations to the frequency domain, still prove to be useful [25].

2.5 Recursive Neural Network Initialization

Recurrent Neural Networks (RNNs) are widely used for multi-step predictions because they can carry information regarding the “hidden state” of a sequence through time as vectors.

Although the performance of RNNs can be highly dependent on the initialization of these vectors, most often, they are simply set to zero. Sometimes, the RNNs are set up so that the state vectors go through some iterations (washout period) before the first prediction can be returned.

A few studies have proposed methods for initializing state vectors of RNNs. The early research of [44] proposed initializing the RNN with a vector proportional to the backpropagated errors to regularize and improve the training stability of a RNN.

[36] proposed a general framework for auto-regressive tasks, where contextual information to the task, such as the embedding of the fist word in a sentence in the Natural Language Processing (NLP) domain, is used by another NN to compute the initial hidden state of RNN. In the constructed tasks of associative retrieval – where, given a sequence of symbols and a key symbol, the model should return the symbol after the key in the original sequence – and the auto-regressive task of predicting damped oscillations ($x(t) = Ae^{-bt}\text{cos}(ct)$), they demonstrated their framework is able to significantly speed up the convergence of the model, but not improve the final performance of the model, when virtually unlimited training iterations is available.

Knowing that, in dynamical systems, the initial state is a determining factor in the future states, [19] sought to improve on the the typical use of a washout period to encode the state of the system. In modeling an aerial vehicle dynamics, they proposed a method where data from before the prediction, both features and labels, using a specialized NN, is used to compute the initial hidden state of the RNN. Using some simplifications, they formulated a loss function for the optimal hidden state, and jointly trained both networks, with the loss function defined as the sum of the loss in main task and that of the RNN initialization. In their ablation study, they showed that the improvement brought by their method, in comparison to using a washout period, was greatest in the first iterations with gains reducing over time, while still remaining better in the latter iterations.

A few studies have attempted to improve on the usual zero initialization of RNN hidden states. The more recent methods seem to converge on the use of specialized neural networks to compute the hidden state.

Chapter 3

Technical Background

As this dissertation presents a study about deep learning models to predict the heart rate, we dedicate this chapter to explain the building blocks of deep artificial neural networks. We also present the specific architectures that we adapt to create baselines for the methods we developed in this dissertation.

We start from the fundamentals of deep learning and one of its most basic building blocks: the multi-layer perceptron (MLP). Then we shift our attention to architectures which take advantage of properties of the task at hand. First we explain convolutional neural networks (CNN), which leverage on the spatial relationship between features, and then we look at Recursive Neural Networks (RNN), which take advantage of the sequential nature of some tasks. Finally we discuss attention transformers, a newer architecture which generalizes the way features are aggregated, as opposed to the first two, which work by imposing certain constraints.

In the second part of this chapter, we explore some details of two well performing architectures for HAR: the "Deep Convolutional LSTM" [22] and the "CNN IMU 2" [26]. We chose the former for its good performance and for being recursive, which facilitates the adaptation to multi-step predictions tasks, and the latter for being the current state-of-the art in HAR.

3.1 Multi-layer Perceptron

At a high level, Neural Networks consists of layers of non-linear transformations on the input features. A feature j of layer $\ell + 1$ (with N_ℓ features), denotedf_j^ℓ, is computed from a non-linear transformation (F^ℓ), named activation function, of a weighted sum of the features in layer ℓ.

$$f_j^{\ell+1} = F^\ell(\sum_{k=1}^{N_\ell} w_{j,k}^\ell f_k^\ell)$$

The idea behind using multiple layers in succession is that, with each new layer, a higher level representation is built into the latent space (features of that layer). This progresses to the last layer, where a prediction in computed from a linear transformation of the last layer's latent space. For binary classification, the output is commonly normalized using a sigmoid function, making the interpretation of the predicted value as the the probability of belonging to a certain class, and for multi-class classification, the normalization is usually performed by soft-max.

In supervised settings, the weights ($W = [w^{\ell}_{j,k}]$) of the model are computed by an iterative process called training, where the weights are fit to a set of data containing both the input features and the labels using optimization methods, aiming to reduce difference between what is predicted and the actual values. Nowadays, the most commonly used optimization methods for deep learning are based on gradients propagated along the layers in respect to a loss function, that is, a metric for the (lack of) quality of predictions.

In order to prevent the network from becoming too dependent on a subset of features, a method called dropout was developed. During the training phase of a network, this method randomly disables a fraction of the neurons in a layer, with a certain probability: a hyperparameter called dropout rate. This technique helps the optimization algorithm avoid local minima.

In early research of artificial neural networks, the most common activation function used was the sigmoid ($\sigma(u) = \frac{1}{1+e^{-u}}$), for being differentiable in the entire real domain, and for being interpretable in statistics, as this function maps any real number into $[0, 1]$ (a single neuron using this activation function is named perceptron, leading to the name of this architecuture: “multi-layer perceptron”). Another commonly used function, with similar attributes to the sigmoid, but mapping real numbers into $[-1, 1]$, is the hyperbolic tangent ($\tanh(u) = \frac{1-e^{-u}}{1+e^{-u}}$). More recently it has been noticed that using the Rectified Linear Unit ($ReLU(u) = \max(0, u)$), and variations thereof– like the Leaky ReLU ($LeakyReLU(u) = \max(au, u)$, with 0.01 as a common value for a) –usually allow for better training of the NN.

3.2 Convolutional Neural Networks

Convolutional Neural Networks were designed to take advantage of the spatial relationship between features, as those closer together tend to be more correlated than those further apart. Another key characteristic of CNNs is their invariant to the position in the input, as it performs the same computations, regardless of the location.

More specifically, in this architecture, a feature from layer ℓ, f^{ℓ}_{j}, is not computed from all features of layers $\ell - 1$, but only from those within a defined distance (kernel size) from it, that is, those features inside a certain a window (called kernel).

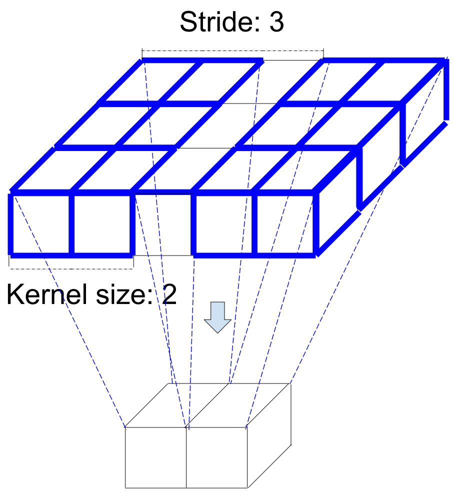
Another difference is that the weights are shared among windows, that is, the weights used to compute f^{ℓ}_{j}, are the same used to compute f^{ℓ}_{j+1}. Therefore, the weights are applied as a sliding window, with a specific step (called stride).

A kernel (which contains the weights) spans over all features spatially within that window, that is, in the case where the input is a picture with (3) RGB channels, a filter with kernel of size (2×2), has $12 = 2 \cdot 2 \cdot 3$ weights.

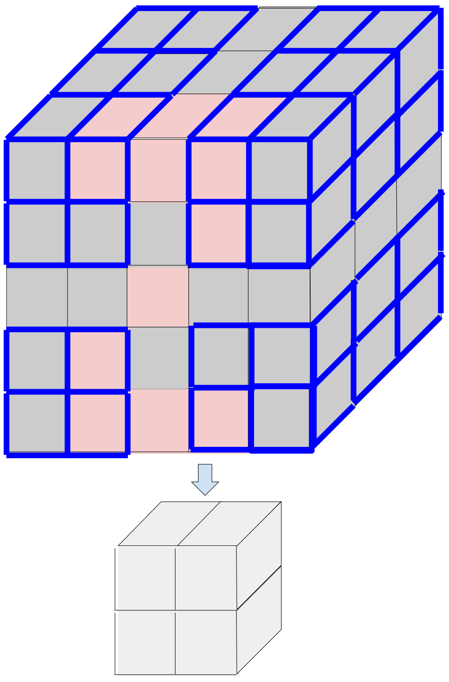
It is important to note that the definition of what is "spatially close" is chosen by the designer of the network. For example, a picture is represented as a tensor, where the width, height and channel dimensions are seen just as the dimensions of the tensor. There is nothing about the channel which would distinguish it from the other two dimensions. The designer of a network, with domain specific knowledge, defines what has "spatial correlation".

Figure 3.1 represents two types of convolutions, where the filter is represented in blue, bold edges and the feature data is represented in black edges. The first represents the application a 1D convolution filter, with kernel of size 2 and stride 3, applied over feature data of length 5 and three channels. The second represents the application of a 2D convolution with kernel of size 2×2 and stride 3×3 over feature data of width 5, height 5 and 3 channels, representing a 5x5 picture depicting the number "2" in red. We use strides larger than the kernel sizes only for illustration purposes, as this would cause some data loss.

In order to avoid the loss of data from edges or enable the retention of dimension size of the data, a technique called padding, is applied. With padding, the length of the feature input is increased on the extremes by adding 0 value points.



(a) 1D Convolution



(b) 2D Convolution

Figure 3.1: Examples of Convolutions

Another common operation used in CNNs is "Max-Pooling". In this operation, instead of the linear combination of the values in a kernel as done in the convolutions, it is the maximum value inside a kernel for each channel that is passed onto the next layer. In this way, dimensionality reduction is achieved without any additional parameters.

As a further source of information about the CNN, we refer the reader to [1].

3.3 Recursive Neural Network

Recursive Neural Networks (RNN) embody the idea that information may me acquired sequentially, so that a new representation of the data (i.e., hidden state) is created, while new inputs are added to the model, so that information about the order of the data is encoded by the order of the updates to the hidden state.

An RNN stores an internal representation h_t of the information seen up to step t in a latent space, which is updated as new data is fed to the network. A key characteristic of this type of network is that the same computations are applied to every new input (x_t) and the previous latent state (h_{t-1}), by a network called RNN Cell. An important characteristic of this type of network is that is does not constrain the number of inputs fed into the network, even allowing for sequences of different lengths. Figure 4.2 represents a general recursive neural network.

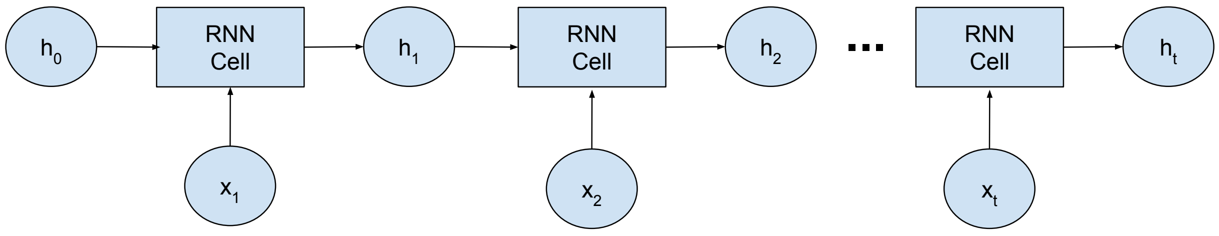


Figure 3.2: Representation of Recursive Neural Networks

A basic RNN represents the latent state as a single vector (h), and combines the values of the latent space with the features x_t of iteration t, using a sum of the linearly transformed vectors (with weights W_x and W_h), using a non-linear activation function (σ).

$$c_t = \sigma(W_x x_t + W_h h_{t-1})$$

This architecture has notable issues, known as the vanishing and exploding gradients. As the gradient between the loss function (E), with respect to the weights ($W : W_x, W_h$), at iteration t, can be approximated as proportional to the product of the weight matrices up to that iteration $\frac{\partial E}{\partial W} \sim W^t$, as the sequence grows, this gradient tends to 0, if the largest absolute eigenvalue λ of W, is less than 1 (“vanishing gradients”), and to infinity, if λ is greater than 1 (“exploding gradients”). This leads to instability during training using the usual gradient-based optimization methods.

To mitigate this problem, more elaborate architectures have been designed that use “skip-connections” to carry the hidden states without change when no relevant information is added in the input . The most well-known variants of the “vanilla” RNN cell are the Gated Recurrent Unit (GRU) and the Long Short-Term Memory (LSTM). We will delve into details of the LSTM, as it is used throughout this dissertation.

The LSTM is based on the idea that, when new information comes in, some of the old information may lose its relevance, in which case the new data must be used to update the hidden state. This architecture has components that represents this idea explicitly: a forget gate and an input gate.

LSTMs use a latent state composed of two vectors that carry information, namely the Hidden State h_t and the Cell State c_t. For each time step t, the input x_t is concatenated with the previous hidden state h_{t-1}, forming a new column vector I_t. From vector I_t, three other vectors are computed, namely the forget, update and output vectors. The forget vector define how much of c_{t-1} is passed onto c_t. The update vector define how much of I is added to c_t, and the output vector define how much of a scaled c_t makes up h_t. Figure 3.3 represents the components of a LSTM. For more in-depth information about RNNs, we refer the reader to [30].

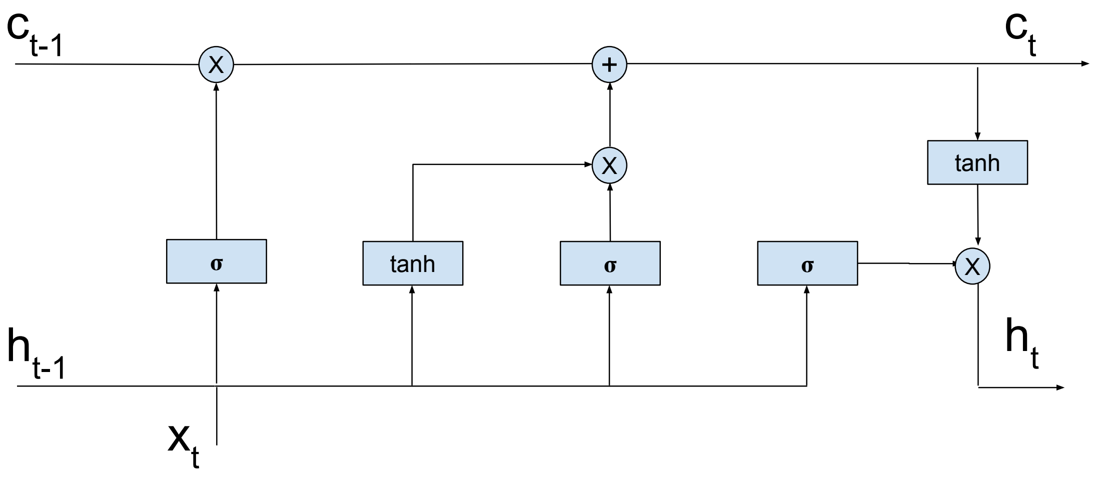


Figure 3.3: Components of a LSTM Cell

3.4 Attention Transformers

Attention transformer networks generalize the traditional neural networks, as the weights used to compute a neuron are not constant, but a function of neurons connected to it.

The attention mechanism is composed of three main parts, namely a “query” vector ($\mathbf{q}$) and key-value pairs of vectors $(\mathbf{k}, \mathbf{v})$. Pairs $(\mathbf{k}, \mathbf{v})$ are computed from the source neuron, whereas the $\mathbf{q}$ vector is related to the destination neuron. Considering the key and value vectors from the source neurons packed as rows into the matrices $\mathbf{K}$ and $\mathbf{V}$, the value of the destination neuron is computed by the dot-product attention as:

$$Attention(\mathbf{K}, \mathbf{V}, \mathbf{q}) = softmax(\mathbf{q}\mathbf{K}^T)\mathbf{V}$$

The case where $\mathbf{q}$ is also computed from the neurons of the previous layer is called *self-attention*.

A variation of the attention mechanism, called multi-head attention, occurs when, the $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ vectors are linearly transformed into h other vectors each, so that the h heads of the attention mechanism compute the results from these projected vectors in parallel. The output vectors are concatenated and linearly transformed to yield the final transformed output.

The attention-transformer is an encoder-decoder architecture, where a set of $\mathbf{k}$,$\mathbf{v}$ pairs are computed from a source sequence of inputs using self-attention layers, which make up the encoder, and the inputs from a target sequence are used to compute the $\mathbf{q}$ vectors in the decoder. The predictions calculated in the decoder using attention sublayers that attend to the $(\mathbf{k}$,$\mathbf{v})$ pairs coming from the encoder's last layer and self attention sublayers that attend to the decoder's previous layer. Figure 3.4 illustrates an attention-transformer architecture.

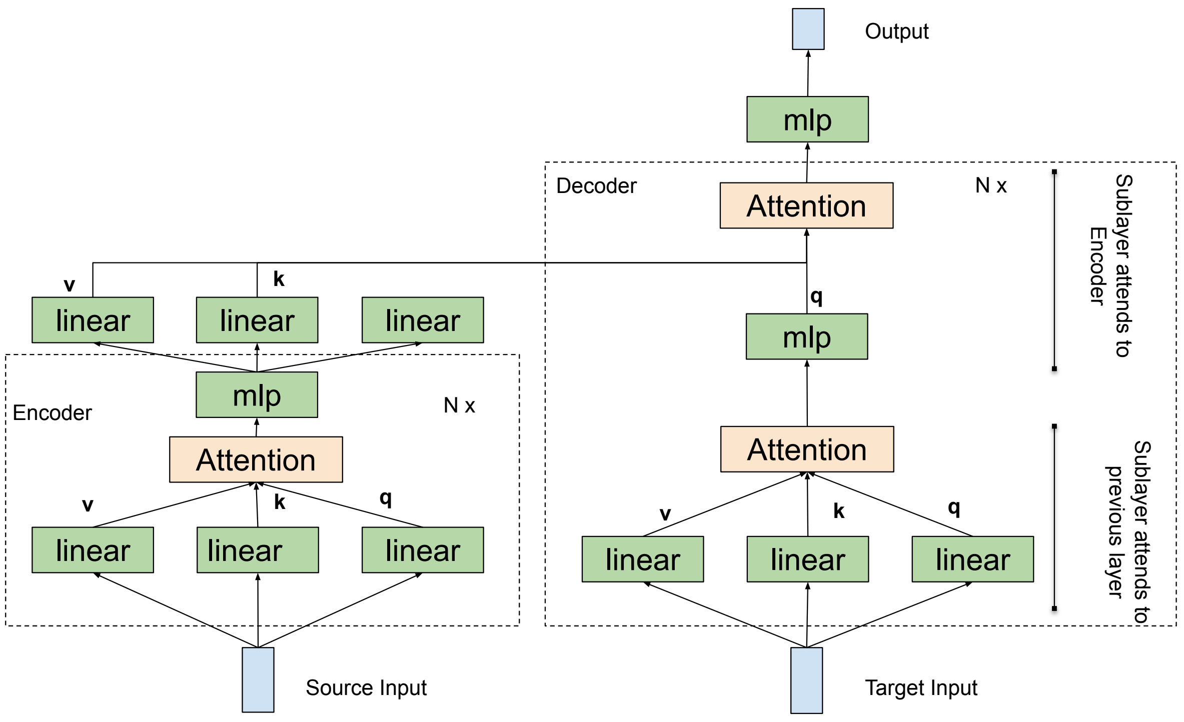


Figure 3.4: Attention Transformer

Differently from RNNs, Attention Transformers attend to all inputs simultaneously, therefore solving the bottleneck of RNNs, where information from previous inputs should be kept into the hidden state for long-term dependencies. One shortcoming is that this causes transformers to be position-invariant. Therefore a positional encoding is usually added to the input vectors.

Nowadays, most state-of-the-art language models are based on attention transformers. BERT [7], which is regarded as one of the best language models, for example, uses only the encoder part of the transformer, and innovates especially on how the models are trained.

For further information regarding to the attention mechanism, we refer the reader to [33] which introduced attentions transformers. In the next section we turn our focus to well-performing architectures designed for the HAR task, which we adapt to benchmark our model.

3.5 Deep Convolutional LSTM

The Deep Convolutional LSTM [22] is a neural network architecture proposed for the IMU-based HAR task. It is composed of two key parts, a Convolutional Neural Network to extract patterns in the raw sensor data, and a 2-layer LSTM network, to account for temporal dependency in the data.

This study segmented the raw sensor data into 500ms windows, from which the predictions are computed. The sensor data in the time window is arranged in a 2-D tensor, where time is represented along the length dimension, and the sensors along the height dimension. The CNN subnetwork, processes data from each of the sensors independently, while sharing the filters by using a 1×5 kernel in all convolutional layers. There are 4 layers of convolutional transformations, each made of 64 filters and followed by ReLU activation functions. It does not use padding, therefore, the length of the input tensor is reduced by 16 units after passing through the four convolutions.

To account for temporal relations between the attributes, this architecture uses a 2-layer LSTM architecture where a vector containing the probabilities of belonging to each class is computed using a softmax normalization from the last hidden state from the second layer. Figure 3.5 represents the entire architecture, with the convolutional components represented in green, the LSTM components in blue, the softmax in red and the tensors in white.

3.6 CNN-IMU2

More recently, an architecture called CNN-IMU2 was proposed for HAR by [26] and it is composed of CNN and MLP subnetworks. It has some striking similarities to the CNN component of the Deep Convolutional LSTM [22], but contains some key differences, especially in

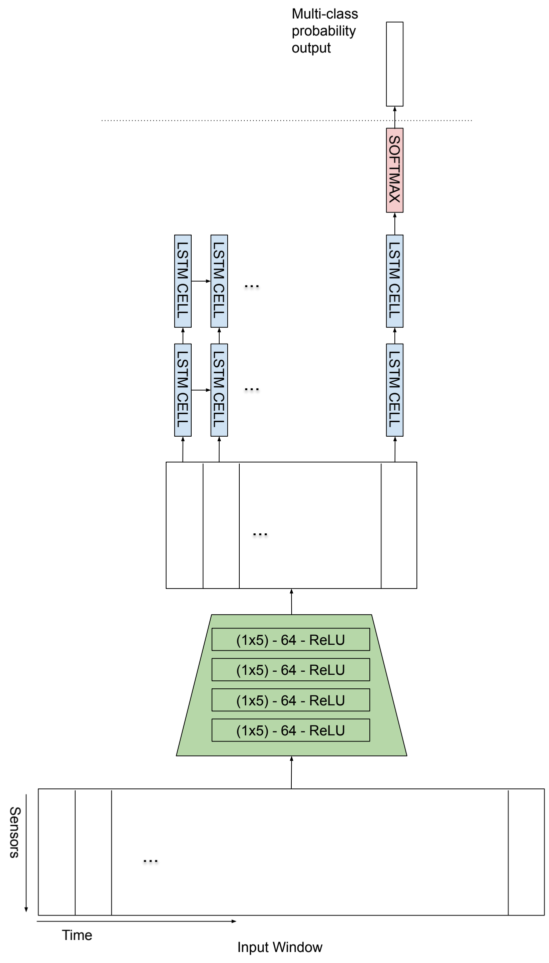


Figure 3.5: Deep Convolutional LSTM Architecture

the way it splits the sensors into separate sets, and by using max-pooling components.

Like Deep Convolutional LSTM, CNN-IMU2 also predicts over time windows, but it regards the window size a hyperparameter and uses different window sizes for different datasets. The convolutional subnetwork is made of the same four layers, as Deep Convolutional LSTM, but has two max-pooling layers of size 1x2, added as layer 3 and 6 of the architecture.

Given that most datasets have sensors from a number of devices, each attached to a different part of the human body, this work points that the set of sensor from each device should have its features extracted by a different set of filters. Therefore, this architecture uses parallel convolutional networks for the sensors in each of (K) devices.

After going through the CNN subnetworks, each tensor is passed onto a fully connected layer that outputs a 512-dimensional vector . Then, after being concatenated, it goes through another MLP composed of two layers, the first with 512 neurons and the second, without activation function, and a number of neurons equal to the number of classes of dataset. Finally a softmax is computed, outputting the probabilities of each of the classes. Figure 3.6 represents the architecture, where the convolutional parts are shown in green, the MLP in orange and the softmax in red.

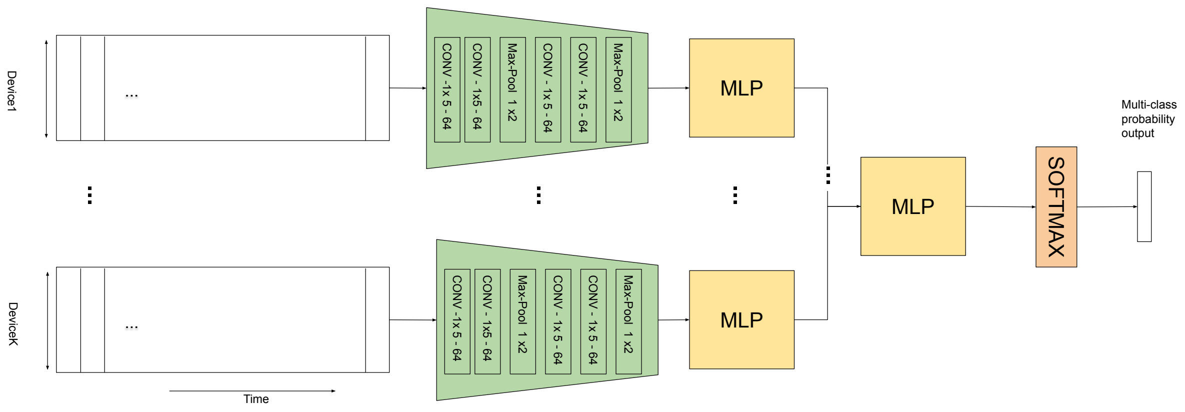


Figure 3.6: CNN-IMU2 Architecture

Chapter 4

Methodology

This dissertation addresses the problem of predicting the heart rate H_t of an individual at time t, given IMU sensor data gathered up to time t and HR values from an initial, short lived period, $H_1, \ldots, H_I$. We refer to this task as **IMU-based multi-step heart rate estimation**. For this study, we choose the PAMAP2 [24] and PPG-DaLiA [25] datasets, which are among the very few publicly available sets containing both IMU and HR signals from individuals performing a variety of activities. PPG-DaLiA also has data from PPG sensors, which is disregarded in this first task.

The case where PPG sensors are available (in addition to IMU) is then addressed in this dissertation as a secondary prediction task, referred as **PPG-based multi-step heart rate estimation**. Since PPG data are heart rate measurements that can be perturbed by movement (*motion artifacts*), this task consists of correcting such measurements based on IMU data. For this task, we use the WESAD [28] and the (complete) PPG-DaLiA datasets. We show that it is possible to adapt the architecture proposed for the first to the second task with minor changes.

We consider two alternative formulations to tackle the problem of IMU-based multi-step heart rate estimation. The first, which we call **"delta formulation"**, defines the difference from the initial heart rate H_I as the response variable. This formulation was defined to enable the use of a larger number of baseline methods, for it is easier to adapt non-recursive methods to this formulation. The second, named **"direct formulation"**, has the heart rate itself as the target. In some sense, the delta formulation was considered for preliminary experiments. The direct formulation, arguably the most natural among the two, is considered for a much more thorough set of experiments.

We begin this chapter by detailing the three publicly available datasets we used and how they may or may not contribute to the task of predicting the heart rate for individuals undergoing physical activity.

Next, we describe the preprocessing used in this work, including how we deal with the missing data-points, how we join data from different sampling frequencies and how we deal with the time series data using sliding windows.

We then present the "delta formulation" and the "direct formulation" used in the experiments for the IMU-based HR estimation task.

Last, the baseline models are presented. They include: NN architectures proposed for HAR, the only model proposed for IMU-based multi-step HR estimation and a standard attention-transformer developed by us. We include this architecture because transformers are being increasingly regarded as the best model for sequential data.

In summary, this chapter we describe how we approach the task of estimation the HR multiple-steps into the future, and the methods used as baselines.

4.1 Datasets

Because this dissertation models the impact of physical activity in the HR using IMU sensors in different individuals, it is important that the datasets used in the evaluation contain IMU data from a large number of subjects, performing varied physical activities. Yet, these characteristics have not been found simultaneously in any single publicly available dataset.

The PAMAP2 dataset comprises data from many different physical activities, which are both intense and significantly distinct from one another, such as soccer playing, rope jumping, running and cycling. It also contains data from lower-intensity activities, such as laying down, sitting and watching the TV. Its greatest disadvantage is having a smaller number of subjects (9) and a shorter of time series (about one hour per subject) in relation to the other datasets.

In contrast, PPG-Dalia has a larger number of subjects (15) and far longer time series (most last longer than 2 hours). It has, nevertheless, a smaller number of activities, from which cycling and walking are the most intense. This dataset also contains PPG data, enabling it to be used in PPG-based HR estimation.

The WESAD dataset, despite containing an equal number of subjects and sensors as PPG-DaLiA, is not adequate for evaluating the impact of physical activities on the HR, as the subjects remain seated/standing during the entire recorded duration. This dataset is useful, though, for PPG-based HR estimation.

Below we describe each dataset in more detail.

The PAMAP2 Dataset [24] consists of data from 40 sensors (accelerometers, gyroscopes, magnetometers, thermometers and heart rate sensor) sampled at 100Hz of 9 individuals performing 18 different activities, ranging from rope jumping, cycling and running to laying down, sitting and standing. There is a single time series of sensor signals per individual, each performing a sequence of activities. Later on we explain that the time series for 1 of the 9 the individuals is too short for training the models.

The PPG-DaLiA Dataset [25] is composed of signals from two devices: a chest-worn device which provides accelerometer and ECG data, and a wrist-worn device measuring the photoplethysmography (PPG) and triaxial acceleration, sampled at 32 Hz. The heart rate series is computed from the ECG signals. This dataset contains a contiguous time series of sensor signals from 15 individuals performing 8 activities. We create a variant of this dataset, hereby referred simply as **the DaLiA dataset**, which does not include the PPG signals to use in the IMU-based multi-step HR estimation task.

The WESAD Dataset [28] consists of data from 15 subjects wearing the same sensors available in PPG-DaLiA, but differently from that dataset, the individuals remain seated/standing during the whole study while going through different affective states (neutral, stress, amusement), which are elicited by watching funny video clips to bring about amusement, and by compelling the subjects to deliver speeches, in order to trigger stress. Unlike PPG-DaLiA, WESAD does not provide precomputed heart rate series, therefore we used the *heartpy* library [32] to extract heart rate measurements from the ECG signals. Although subjects are indexed up to number 17, the dataset does not include subject identifiers 1 and 12.

In total, we use data from 23 (resp. 30) individuals for the IMU-based (resp. PPG-based) multi-step heart rate estimation task.

4.2 Preprocessing

In this section, we describe the data transformation operations performed in order to prepare the data to be used by the model. We start by explaining the handling of missing IMU-signal data points and the normalization performed. Then we describe the aggregation of the time series into time snippets. Finally, as we aim to predict the impact of physical exercises for specific individuals through the encoding of physical conditioning using a small, previous interval containing both HR and IMU data, we detail the distinctions between this interval and the interval we actually use for prediction.

Basic Preprocessing: We upsample the heart rate signal using linear interpolation in all datasets to make its sampling rate consistent with the other signals. Only PAMAP2 contains a few missing sensor signal data points, which we handle by local averaging the data around the missing point using a 0.4s window [8]. To make the use of the PAMAP2 and PPG-DaLiA datasets more consistent, we use only the accelerometer signals of the chest and wrist, downsampled to 32Hz, under the direct formulation. Conversely, under the delta approach we use accelerometers, gyroscopes, and temperature sensors, on the hand, chest and ankle as features, since we conduct experiments only on PAMAP2. All signals s are z-normalized, i.e., $\hat{\mathrm{s}} = (s - \mu(\mathrm{s}))/\sigma(\mathrm{s})$, where μ stands for the mean operator and σ for the standard deviation operator.

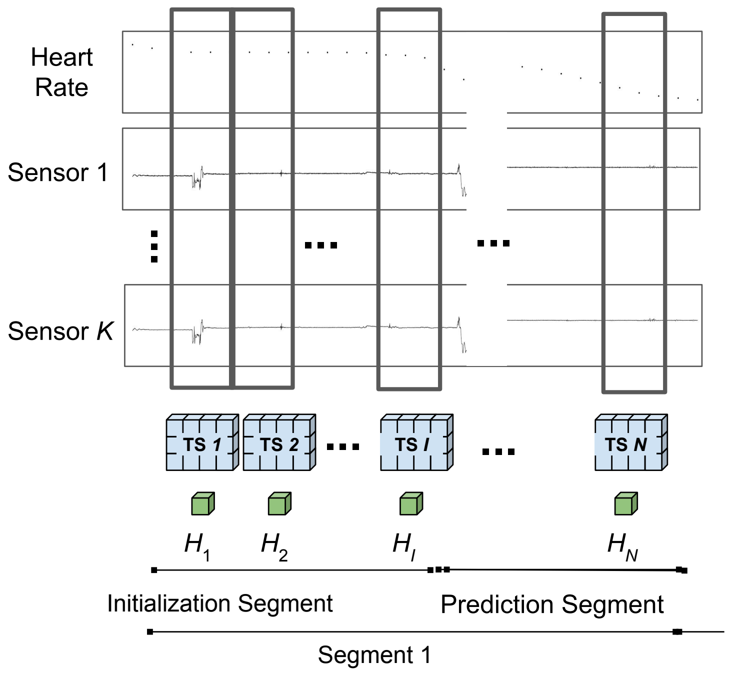


Figure 4.1: Time Snippet Representation

Time Snippet Discretization. Like many techniques based on high frequency sensor signals, we split the time series signals into **time snippets** (TS), i.e., partially overlapping windows of fixed duration τ_{TS} and overlap ratio r_{TS}. Figure 4.1 illustrates this procedure for the case when $r_{\text{TS}} = 0$. Each time snippet TS_t is a matrix where each row represents a sensor. The heart rate signals are also included in the time snippet, but in the *Predictions Segment* (which will be discussed shortly), when no heart rate signal should be available to the models, their values are replaced by zeros. As the models make one prediction per time snippet, this determines the granularity of the predictions. Accordingly, we define the average heart rate H_t for each time snippet TS_t as the response to be predicted.

Time Series Segmentation. A contiguous segment of sensor signals of a subject is partitioned into two smaller segments. The first, called **Initialization Segment**, contains the IMU and HR signals for the first I time snippets, and can be used by a neural network (NN) to encode a "state" specific to that time series and individual. The second, called **Prediction Segment**, contains the IMU, but does not contain HR, as it is used by a (possibly different) NN to output predictions for each time snippet. Figure 4.1 illustrates the subdivision of a segment, in the case where time snippets do not overlap in time ($r_{TS} = 0$).

During the training phase, in order to create a fixed-length training set, we segment the time series of each individual in a sequence of N contiguous time snippets. Hence, a longer series can yield more segments and, in turn, more initialization segments to train the physical conditioning embedding subnetwork.

4.3 Problem Formulations

Prediction models may be classified as either recursive or non-recursive. In addition to input features, recursive models use the values computed in the previous step, such as its latent state or the last prediction, to compute the prediction in the current step. On the other hand, non-recursive models only use the input features for their predictions.

Multi-step prediction problems are more naturally represented by recursive models. Yet, the state-of-the-art model for the HAR task, used to create one of the baselines for evaluate the proposed model, is non-recursive.

To enable the use of non-recursive models for IMU-based heart rate estimation we reformulated the prediction task so that the models must predict the difference in heart rate between the current step and the previous step. This prediction is fairer to non-recursive models, as in this way, the description of the heart rate as a continuous function over time is intrinsic to this modeling. We call this approach “delta formulation”.

On the other hand, in the “direct formulation’, the models must predict the actual HR values. Some baselines from the literature are recursive and hence, suitable for this type of prediction.

In the remainder of the section we formally define both formulations.

Delta Formulation: In principle, non-recursive models could be used for multi-step heart rate prediction, such as the state-of-the-art model for HAR [26], by replacing the response variable y_i with the heart rate H_i. However, this is unlikely to yield good results as there is no constraint that prevents subsequent predictions $\hat{H}_i$, and $\hat{H}_{i+1}$ to be very different. To address this issue, this formulations considers the problem of predicting the change $\Delta H_i = H_i - H_{i-1}$, between two measurements. Since we have the last heart rate measurement H_i from the initialization segment, we can reconstruct a HR prediction $\hat{H}_i$, at time i, from the predictions $\Delta\hat{H}_i$, as

$$\hat{H}_i = H_I + \sum_{j=I+1}^{i} \Delta\hat{H}_j, \text{ for } i \geq I+1.$$

We take this approach mostly to perform comparative studies, though we also conduct ablation studies in this formulation.

Direct formulation: The direct formulation is more natural and, hence, the main approach in this dissertation. We consider it when experimenting with the use of ensembles and with a method for better encoding the physical conditioning of the subjects.

In order to adapt non-recursive methods to multi-step predictions in this formulation, more changes to the original architectures would be required, compromising a fair comparison.

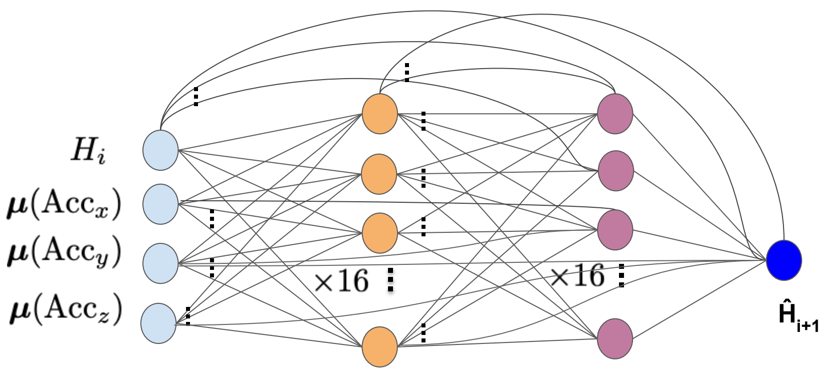


Figure 4.2: FFNN architecture

Therefore we chose to benchmark our model only against recursive baselines, in this formulation.

4.4 Baseline Models

To the best of our knowledge, the task of predicting multiple steps of heart rate given IMU signals has only been attempted by [38], which reported their experiments only in datasets not publicly available. In addition to using our implementation of their model as a baseline, we also adapt two models designed for a similar task (HAR), as well as a standard attention-transformer-based model to benchmark our method. In this section, we describe each of these models, the task they were originally designed to address and minor changes required to adapt them for the task at hand.

FFNN [38] is a feedforward neural network by [38], and the only model in the literature proposed for multi-step HR prediction from IMU sensor data. Although some promising results were reported, this model was applied only to a dataset that is not publicly available, which comprised a single time series of one individual performing normal daily activities. FFNN is a recursive architecture with skip connections using data from a wrist-worn triaxial accelerometer. The model used the average measurement of each sensor in a non-overlapping window of 30s. As the architecture details were not reported, we used ReLU as the activation function, set the layer size to 16 (we also experimented with sizes 8, 32, 64, obtaining worse results). We adapted their architecture, using a time window of $\tau_{\mathrm{TS}} = 4$s to make it more comparable to our method (testing with $\tau_{\mathrm{TS}} = 30$s as in the original work yielded worse results). We also used Adam for network weight optimization, instead of the genetic algorithms used in that study.

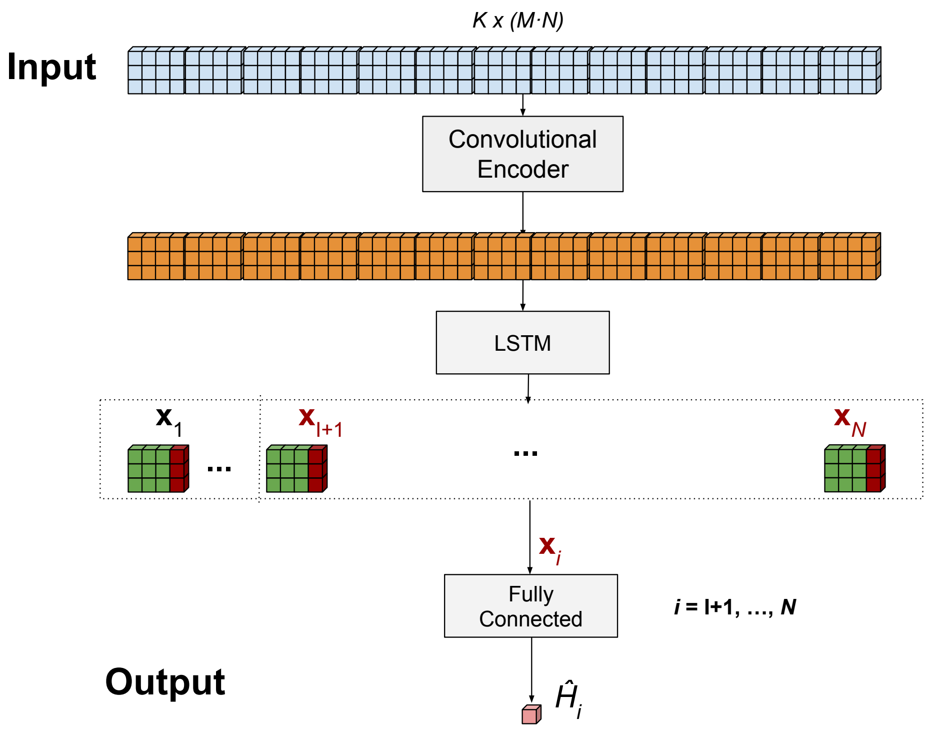


Figure 4.3: DeepConvLSTM architecture

Figure 4.2 illustrates the FFNN architecture.

DeepConvLSTM [22] is an architecture designed for the HAR task. It performs a sequence of convolutions on the input series, with padding adjusted so as to keep the dimensions of the tensor the same. The deep features from each time entry feed a LSTM. From each of the LSTM's hidden vectors, a prediction is computed using a single linear layer. In our adapted version, we select only the outputs corresponding to the last time input of each time snippet, as we have only one label per time snippet. As in the original article, we set the length of time snippets to $\tau_{\text{TS}} = 3s$. Figure 4.3 depicts DeepConvLSTM adapted architecture.

CNN-IMU-2 [26] was proposed for the related task of HAR. This architecture was designed for single step prediction. It consists of a sequence of convolutions on the input series, followed by a sequence of fully connected layers leading to the prediction. In our adaptation, we used padding in the convolutions, in order to keep the tensor dimensions the same and we split the series into time snippets after the convolutions, so that the deep features are extracted from a larger period than otherwise. Figure 4.4 shows CNN-IMU-2 adapted architecture.

ConvAttention (ours) is an Attention Transformer created by us as an additional baseline to PCE-LSTM, for a thorough comparison. It uses an Encoder-Decoder architecture, where the encoder is responsible for transforming a sequence of input vectors into pairs of key and value vectors using multiple layers of transformer cells. Each cell is made of attention heads and feed forward layers. The decoder also receives a sequence of vectors as input, each transformed into

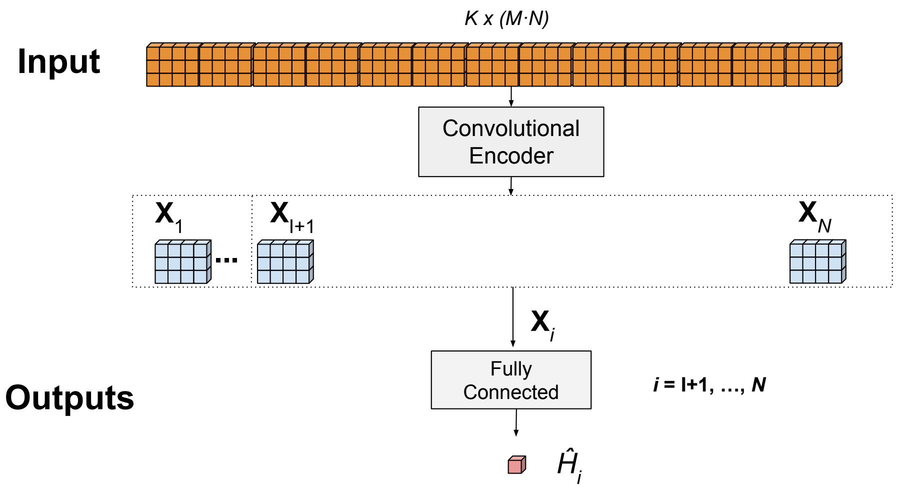


Figure 4.4: CNN-IMU-2 architecture

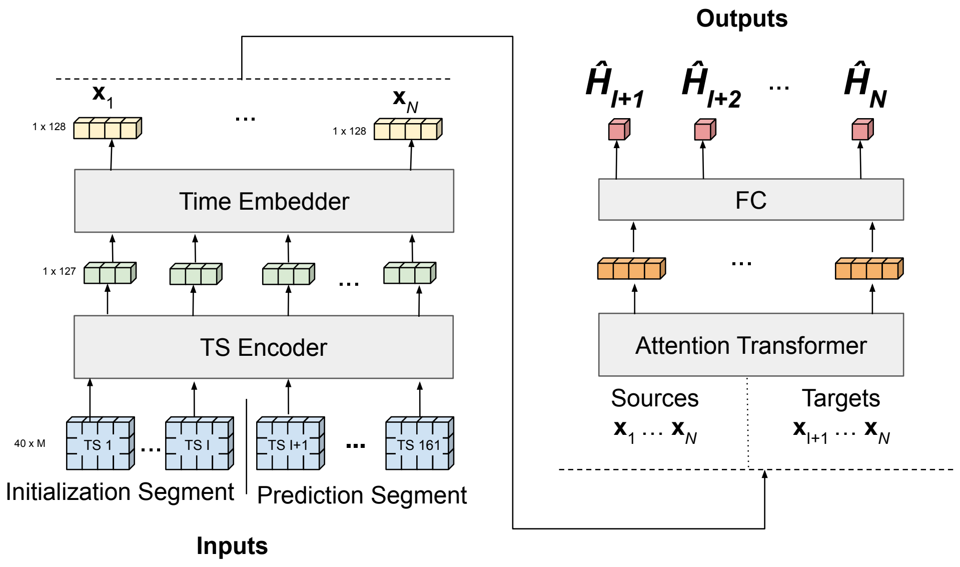


Figure 4.5: ConvAttention architecture

query vectors of the same size as the input using multiple layers of transformer cells. For each decoder vector, the output is computed by the weighted sum of the encoded values where the weights are computed from the normalized softmax of dot product between the encoded keys and the decoder queries.

ConvAttention is made of three components: the **Time Embedder**, which concatenates to the deep features a value proportional to their position in the prediction vector; a standard **Attention Transformer** with 16 heads, 6 layers and feed forward expansion of 512, used for encoding the time snippets into a vector; and the **FC Prediction Decoder**, used for computing the heart rate prediction for each vector outputted by the attention transformer. This module is a 2-layer fully connected network, using ReLU activation in the first layer. The architecture is shown in Figure 4.5.

Chapter 5

Proposed Model: PCE-LSTM

In this chapter we describe PCE-LSTM, our proposed neural network architecture for heart rate prediction. We start by describing the differences of our approach with respect to the existing ones, including the hypothesis and assumptions relevant to the modeling. We then delve into the details of the proposed architecture, including the structure of the multiple components of the model.

PCE-LSTM is composed of convolutional and LSTM layers, similarly to the state-of-the-art techniques for the closely related task of HAR. However, PCE-LSTM presents a novel approach for encoding a subject's physical conditioning using a previous short-lived activity. We also approach the processing IMU-sensor data differently than related works.

Our model is an end-to-end deep learning, hybrid architecture, taking advantage advantage of the qualities of different types of NN, including FCNN, CNN and LSTM in the multiple components which comprises PCE-LSTM. We also take advantage of multi-task training, using partially Siamese networks to enable better training of the network.

5.1 Differences to previous approaches

PCE-LSTM is composed of convolutional and LSTM layers, similarly to the state-of-the-art techniques for the closely related task of HAR. The novel aspects of this model are discussed below.

Recurrent Neural Networks (RNNs) are a natural choice to model HR as they have been especially designed to work with sequential data. They are composed of cells with shared parameters, which process units of the input sequentially, using one or more vectors to carry state information through time. In particular, the unidirectional LSTM (long short-term memory) cell uses two vectors – the hidden state and the cell state – which are received from the previous iteration, updated based on the input for that time and passed onto the next iteration. The first iteration, however, receives these vectors as they were initialized, typically as zero vectors. The implicit assumption is that the network will gradually be able to encode the correct state from the inputs as the vectors are passed through the cells.

For HR prediction, we argue that if some data on the relationship between the input and output signals is available prior to the prediction, it can be beneficial to use a specialized network to encode this relationship as the RNN initial state. This initial state should contain information about physical conditioning: a more fit individual is able to sustain similar movement levels with smaller increase in HR. Thus, the main hypothesis investigated here can be stated as

> ***Hypothesis:*** *It is possible to encode information about physical conditioning from an individual's sensor data as a vector and use it as the initial state of a RNN to improve heart rate predictions?*

This is the rationale behind the main difference between PCE-LSTM and what distinguishes our approach from existing ones, namely the initialization of the RNN hidden vectors using a specialized network, which we call the Physical Conditioning Encoder.

We also employed convolutional sub-network differently than in other studies that process IMU-sensor data. The most well-known architectures in the literature [26, 22, 25] keep the transformations on each signal separate during the convolutional section of their architectures by the use convolutional kernels spanning a single signal, only combining them later on. However, we argue that the role of each sensor in the description of the intensity of an activity is equivalent to role of each RGB channel in the description of a picture. Hence, it is reasonable to combine them early on in the NN. To do so, we use 1D convolutions along the time dimension, with the sensors stacked along the channel dimension, as in Figure 3.1, presented in Chapter 3.

5.2 Model Architecture

Many studies which use IMU-sensor data as input for a prediction task aggregate the signals into sliding windows, called time snippets. In this dissertation we take this same approach, which is explained in detail in Section 4.2.

PCE-LSTM uses different subnetworks to (i) extract a latent representation of sensor-data inside a window – time snippet (TS) –, (ii) embed the physical conditioning, (iii) maintain a temporal consistency between predictions and to (iv) compute the prediction itself. Figure 5.1 shows the high level structure of our architecture, which is made of five components:

- the **Time Snippet Encoder** (TS Encoder), a convolutional network which encodes the 2-dimensional TS into a vector;

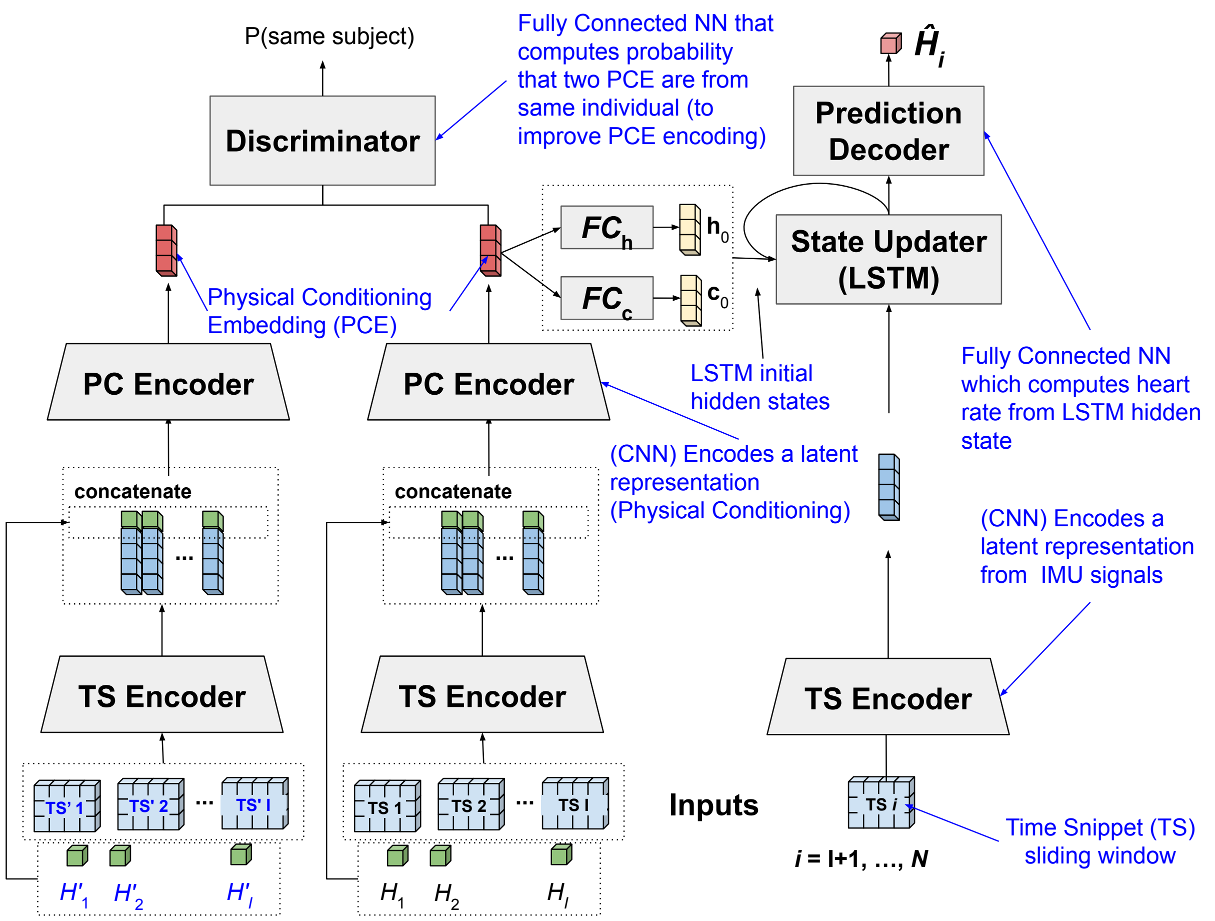

P(same subject)
Discriminator
Fully Connected NN that computes probability that two PCE are from same individual (to improve PCE encoding)
$\hat{H}_i$
Prediction Decoder
FC_h
h_0
FC_c
c_0
State Updater (LSTM)
Physical Conditioning Embedding (PCE)
PC Encoder
PC Encoder
LSTM initial hidden states
Fully Connected NN which computes heart rate from LSTM hidden state
(CNN) Encodes a latent representation (Physical Conditioning)
concatenate
concatenate
(CNN) Encodes a latent representation from IMU signals
TS Encoder
TS Encoder
TS Encoder
TS' 1
TS' 2
TS' l
TS 1
TS 2
TS l
Inputs
TS i
H'_1 H'_2 H'_l
H_1 H_2 H_l
i = l+1, …, N
Time Snippet (TS) sliding window

Figure 5.1: PCE-LSTM (proposed architecture).

- the **Physical Conditioning Encoder** (PC Encoder), a convolutional network which extracts, from signals of the Initialization Segment (including heart rate), a Physical Conditioning Embedding (PCE) used for initializing the LSTM's hidden vectors;
- the **Discriminator**, used for forcing the PCE of a given subject to be similar across time segments and different from other subject's PCE.
- the **State Updater**, a LSTM which maintains and updates the subject's state, encoded in its hidden vectors; and
- the **Prediction Decoder**, a Fully-Connected network which decodes the heart rate prediction from the state tracked by the State Updater.

Each component is described in detail below.

TS Encoder: It is the CNN used to extract a latent representation from one time snippet, which we call TSE. The TS Encoder comprises multiple layers, each made of a 1D convolution followed by a Leaky Rectified Linear Unit activation and a dropout layer of rate $\text{TSE}_{\text{dropout}}$. After each layer, the tensor length ℓ is reduced to $\lfloor \ell/2 \rfloor$ by filters of size 3 and stride 2. When ℓ is even, we use padding $= 1$, except when the the tensor length is 2, in which case we use a

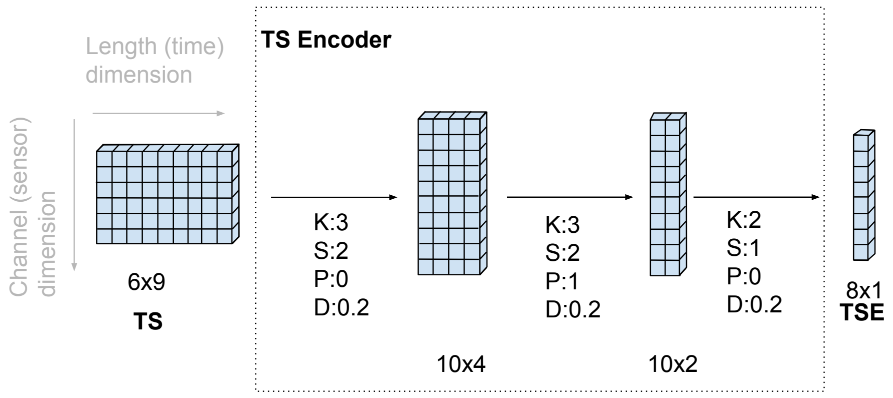


Figure 5.2: TS Encoder illustration for a time snippet of 6 sensors and length 9 ($\text{TS}_\text{N} = 9$), with TS Encoder parameters: $\text{TSE}_\text{F} = 10$, $\text{TSE}_\text{out} = 8$, $\text{TSE}_\text{dropout} = 0.2$
. Legends: Kernel size K, Stride S, Padding P, Dropout D

filter of size 2 without padding. All layers have TSE_F filters, except the last layer, which has TSE_out filters. The number of layers is $\text{TSE}_N = \lfloor \log_2(\text{TS}_L) \rfloor$, where TS_L is the length of the time dimension of the TS, so as to transform the size of the time dimension to one. Figure 5.2 illustrates the TS Encoder architecture for a time snippet of 6 sensors, length 9 (TS period of 90ms for a 100Hz sampling rate, for example) with the following TS Encoder parameters: $TSE_F = 10$, $TSE_{out} = 8$, $TSE_{dropout} = 0.2$ (These hyperparameters are used for illustration only).

PC Encoder: The TSE vectors extracted from each time snippet of the Initialization Segment are stacked along the time dimension (1), and the mean HR of each TS is concatenated along the channel dimension (0). More formally, let $\textbf{TSE}_i$ be the ith columon vector outputted by the TS Encoder (of length TSE_{out}), and **Hs** = $[H_1,H_2, ..H_I]$ be a row vector containing the heart rates measured during the Initialization Segment. The input tensor PCI to the "PC Encoder" is defined by:

$$\mathbf{TSEs} = concatenate([\mathbf{TSE}_1..\mathbf{TSE_I}], axis = 1)$$
$$\mathbf{PCI} = concatenate(\mathbf{Hs}, \mathbf{TSEs}, axis = 0)$$

The PC Encoder takes the $\mathbf{PCI}$ as input and computes a latent representation, dubbed **physical conditioning embedding** (PCE). The PC Encoder is a multi-layer convolutional network that transforms the 2D-input ($(\text{TSE}_\text{out} + 1) \times I$) into a single vector of length PCE_out. PC Encoder is a convolutional architecture designed with the same principles as the TS Encoder. Since it has access to both the HR and the TSE embedding of the activity for a few time snippets, it should be capable of construct some representation of the physical conditioning. Figure 5.1 illustrates how the the PC Encoder fits in the architecture.

From the PCE, the LSTM's hidden state vector and cell state vector are computed using a single linear layer each, represented in Figure 5.1 by FC_h and FC_c.

State Updater: It is a standard LSTM with both state vectors (cell and hidden state) of size LSTM$_H$, and input of size TSE$_{\text{out}}$. These state vectors are initialized by the PC Encoder using only signals from the Initialization Segment. The LSTM is then fed with the deep attributes extracted by the TS Encoder from each time snippet of the Prediction Segment.

Prediction Decoder: It takes the hidden state representation for each time snippet from the State Updater and computes the prediction from them. It is made of three fully connected layers, where the first two layers have 32 neurons, each followed by a ReLU activation function, and the last layer have LSTM$_H$ neurons, without activation function, outputting the predicted HR. We use the mean absolute error (ℓ_1 loss) as the cost function L_{HR} associated with this output because using the ℓ_2 loss hampers training, as larger differences between the predicted and actual heart rate have an outsized impact on the loss, according to our preliminary experiments.

Discriminator: Regarding a subject's physical conditioning as constant in the short term, we reason that employing a Siamese network [5] to discriminate whether two Initialization Segments in the embedding space (PCE) belong to same person will foster a better embedding, when trained jointly with PCE-LSTM.For the discrimination, two Initialization Segments are independently transformed into PCEs using the TS Encoder and the PC Encoder (the Siamese components), concatenated, and fed into the Discriminator module, which outputs the probability that they are from same subject. To train the discriminator, for each segment in the training set, we sample another segment from the same individual and a different individual with equal probability (50%). For each pair, we measure the cross-entropy as the loss function (L_D).The total loss L_{total} is given by a convex combination between the HR estimation loss L_{HR} and the Discriminator loss L_D, where the weight γ is a hyperparameter.

$$L_{\text{total}} = \gamma L_{\text{HR}} + (1 - \gamma) L_D. \tag{5.1}$$

The Discriminator's chosen architecture is comprised of 5 fully connected linear layers, each with 64 neurons and followed by ReLU activation function and a dropout rate of 0.15, except for the last layer, which uses a sigmoid activation function. Figure 5.1 shows the representation of the discriminator in the upper left corner.

Chapter 6

Experimental settings and Results

In this chapter we describe the experiments conducted to evaluate PCE-LSTM, present and discuss their results. We begin by explaining the experimental setup, including how the data was split into training, validation and test sets and how the hyperparameters were chosen. Then we report the prediction experiments regarding the delta and direct formulations. We compare the results achieved by the baseline models and by the variations of our proposed model.

As explained in Chapter 4, we consider the delta formulation to enable the comparison of our model to some non-recursive models from the literature that are known to perform well. Yet, we argue that the direct formulation is a more natural modeling of the task we investigate. We conduct a more through set of experiments with the latter formulation, including the use of ensembles and the use of the discriminator Siamese network, to assist in the training of the model.

6.1 Experimental setup

To evaluate the IMU-based multi-step HR estimation results, we use both the Mean Absolute Error (ℓ_1) and the Maximum Absolute Error (ℓ_∞) as evaluation metrics. The ℓ_1 metric provides the average error across the prediction segment, indicating the overall performance of the model. On the other hand, a large ℓ_∞ error could be dangerous if we are trying to determine the safety of a physical activity for a subject.

6.1.1 Train-test split

There are many different ways to generate a train-test split for time series data from different subjects. [14] investigated the main advantages and drawbacks of different strategies, from which we used two:

- **Full-Non-Overlapping Window (FNOW)**: a non-overlapping sliding window is used to generate the samples, where each sample is randomly assigned to the training or test sets;
- **Leave-One-Subject-Out (LOSO)**: the entire data from one subject is used for test, while data from the remaining subjects is used for training;

These strategies are not mutually exclusive. We chose to use the LOSO strategy to separate test data from the rest (training and validation). We argue that this strategy best mimics a realistic setting where the model would be applied to individuals not present in the training set, as in the case for most models trained offline [14]. For train/validation splits, we used different strategies for each of the problem formulations.

Delta Formulation: In the delta formulation, we consider only the PAMAP2 dataset. We use subject 5 for validation, with each of the other subjects iteratively used as test and the remaining ones used for generating training samples, that is, we use the LOSO approach also for training/validation splits. All methods were trained on time series segments of length $N = 162$.

Direct Formulation: Using this formulation, we make more complex experiments using both the PAMAP2 and DaLiA datasets. In a dataset with S subjects, each of the subjects is used once as the test subject and the remaining $S-1$ subjects' time series are split into training segments of $N = 50$ time snippets each and then randomly assigned to Train/Validation sets using an 80/20 split, that is, we use the FNOW approach for training/validation splits. For evaluation, the entire series of the test set is used, as none of the models evaluated under this formulation requires a fixed size input. For each of the S subjects as the test subject, we perform 7 executions, with different Train/Validation splits and different neural network weight initialization, as done by [25].

6.1.2 Hyperparameter Tuning

In order to set PCM-LSTM's and the optimizer's (Adam) hyperparameters, we applied a random search, as it has been shown to be more efficient than grid-based optimization methods [3]. We used PAMAP2's subject 5 as reference, following [26] and [22], which used the same subject for validation.

Hyperparameter	value
PCE-LSTM hyperparameters	
TSE_F	16, **32**, 64, 128
TSE_{out}	16, 32, 64, **128**
$\text{TSE}_{\text{dropout}}$	0, **0.15**, 0.25, 0.5
PCE_F	16, 32, **64**, 128
PCE_{out}	16, **32**, 64, 128
LSTM_H	16, **32**, 64, 128
Adam's hyperparameters	
learning rate	$\mathbf{10^{-3}}$, 10^{-4},10^{-5}
weight decay	10^{-3}, $\mathbf{10^{-4}}$,10^{-5}

Table 6.1: Search space for random search in Delta formulation. The selected values are shown in bold

Delta Formulation: Under this formulation, we fixed the time snippet generation to a non-overlapping time windows of period 3 ($\tau_{TS} = 3$, $r_{TS} = 0$) and used an initialization segment of length equal to two time snippets ($I = 2$). We also fixed the size of the training segment to $N = 162$. The remaining hyperperameters were selected using random search on a discrete search space, using 60 iterations. Table 6.1 shows the search space of each hyperparameter and, in bold, the selected value.

Direct Formulation: Under this formulation, we also used random search on the time snippet generating parameters, initialization segment size (I), and training segment size (N). We also downsampled the signals in PAMAP2, from 100 to 32Hz (the sampling rate of DaLiA). The search was performed over 80 iterations and Table 6.2 shows the search space of each hyperparameter and, in bold, the selected value.

6.1.3 Training setup

The models are optimized with Adam using the ℓ_1 loss as the cost function associated with the HR predictions. In each epoch, we compute the validation loss. After training is complete, we load the model weights that yielded the lowest validation loss. Subject 9 of the PAMAP2 dataset was not included in the analysis because the corresponding time series is shorter than the length of the training segment.

Hyperparameter	value
Sample Generation Parameters	
I	2, 4, 8, **12**, 16
N	30, 40, **50**
τ_{TS}	2,3,**4**
r_{TS}	0, **0.5**
PCE-LSTM hyperparameters	
TSE_F	**16**, 32, 64, 128
TSE_{out}	16, 32, 64, **128**
$\text{TSE}_{\text{dropout}}$	0, **0.15**, 0.25, 0.5
PCE_F	16, 32, **64**, 128
PCE_{out}	16, 32, **64**, 128
LSTM_H	16, 32, **64** 128
Adam's hyperparameters	
learning rate	$\mathbf{5 \cdot 10^{-3}}$,10^{-3}, 10^{-4},10^{-5}
weight decay	10^{-3}, 10^{-4}, $\mathbf{5 \cdot 10^{-5}}$, 10^{-5}

Table 6.2: Search space for random search in Direct formulation, the selected values are shown bold

Delta Formulation: Training was done using a batch size of 4, as the computational resources we had available could not handle the CNN-IMU2 with a larger batch size. All models were trained over 100 epochs.

Direct Formulation: Training was done using a batch size of 64 over 100 epochs for PCE-LSTM and DeepConvLSTM. FFNN showed slower convergence and hence was trained for 200 epochs.

Computational resources. All experiments were run on free Google Colab instances whose specs were Intel(R) Xeon(R) CPU@2.20GHz, 2 cores, 2 threads per core, Nvidia(R) Tesla T4 GPU, 12G RAM.

6.2 Results

In this section we describe the results of the our experiments, under the delta and direct formulation.

The delta formulation was devised to enable a comparison of PCE-LSTM with non-

Model	**no. parameters**	**train. time** (s)
CNN-IMU-2	34,067,713	4,795
DeepConvLSTM	1,571,521	5,413
ConvAttention	2,810,368	228
PCE-LSTM	64,097	265
FFNN	369	275

Table 6.3: Number of parameters and average training time (all models were trained over 100 epochs)

recursive methods. Therefore, we focus on assessing the performance of our method in its simplest form against a few strong baselines, through some simple experiments on a single dataset, specifically, the PAMAP2 dataset.

The direct formulation is the most natural among the two, hence, we perform a more through set of experiments, including some that attempt to improve the quality of the PCE encoding by using the discriminator network. We also evaluate the performance of our model in an ensemble and a standalone fashion.

In both variation and direct formulations, we attempt measure the PCE's impact on the performance through ablation studies, where several variations of the PCE-LSTM model are compared.

6.2.1 Delta Formulation

The delta formulation, where the predictions of the models are not absolute HR values, but rather the difference from an initial HR, is used to enable comparisons to well-performing, non-recursive methods devised for the related HAR task.

In this formulation, we perform some experiments on the PAMAP2 dataset. As described in the previous section, we train the models, only once per test subject and in every case Subject 5 is used for validation. As the focus of this section is to assess the performance of our method in relation to that of baselines, the experiments we the PCE-LSTM on its simplest form: trained without the help of the discriminator.

Comparison to the reference methods

While we are mainly interested in predictions over the long-term, here we consider both short and long-term prediction scenarios. We include the number of parameters and training time required by each model in Table 6.3 so that prediction accuracy can be analyzed in the light of model complexity and training cost.

Model	**Test Subject (ℓ_1 [beats/min])**							**Average**	
	1	2	3	4	6	7	8	ℓ_1	ℓ_∞
	Short-term prediction (480s)								
FFNN [38]	16.51	11.34	**7.98**	13.36	**13.17**	20.45	33.74	16.65	69.50
DeepConvLSTM [22]	16.61	11.81	11.43	13.65	15.15	18.46	15.88	14.71	66.10
CNN-IMU-2 [26]	15.50	10.67	11.33	13.88	13.94	19.34	13.69	14.05	66.50
ConvAttention (our)	14.95	10.44	12.82	15.38	16.14	17.99	14.88	14.66	64.69
PCE-LSTM (our)	**14.87**	**9.41**	12.75	**12.40**	14.23	**17.14**	**13.67**	**13.50**	**58.76**
	Long-term prediction (entire series)								
FFNN [38]	23.38	14.09	**10.08**	21.04	**12.88**	19.33	47.82	21.23	68.03
DeepConvLSTM [22]	18.27	18.50	20.05	14.38	21.17	21.67	**35.99**	21.43	78.08
CNN-IMU-2 [26]	21.20	18.19	17.75	**14.25**	18.80	21.67	36.29	21.17	80.94
PCE-LSTM (our)	**14.33**	**12.99**	14.37	17.95	18.35	**17.42**	37.21	**18.95**	**63.69**

Table 6.4: Mean Absolute Error (beats/min) of the baselines and our designed models

Table 6.4 shows the ℓ_1 error yielded by the methods for each of the train-test splits (i.e., when each of the subjects' series is used for testing) and the average across subjects. Since the observations below also apply to the ℓ_∞ errors, we report only the average value for this metric. Results for short- (resp. long-) term predictions are shown at the top (resp. bottom). We observe that CNN-IMU-2 model performs slightly better than DeepConvLSTM, in agreement with the results obtained for HAR reported by [26]. In the short-term prediction (top), ConvAttention's average performance is slightly better than DeepConvLSTM's while having nearly 13 times less parameters, as shown in Table 6.8. Yet, PCE-LSTM outperforms all the reference methods, achieving an average ℓ_1 4% smaller and an average ℓ_∞ over 10% smaller than the closest reference method (CNN-IMU-2). Moreover, it has far less parameters than any other method, except for FFNN.

In the long-term prediction (bottom), PCE-LSTM outperformed its closest competitor (CNN-IMU-2) by more than 20% w.r.t. the average ℓ_1. ConvAttention could not be evaluated due to its restrictions on the input size, because it can only be applied to series of the same size it was trained on.

To understand whether the models are capturing oscillations in the heart rate series or just an average behavior, we inspect their predictions for some representative subjects. Figure 6.1 shows the predictions for the first 10min of three subjects' series illustrating cases where PCE-LSTM performance is on par, superior and inferior to some of the baselines, respectively. In the top plot (Subject 3), we observe that all methods capture the valley in the 3-6 min interval but either underestimate or overestimate the HR by up to 10bpm. FCNN incorrectly predicts a peak in 8-10 min. In the middle plot (Subject 2), FCNN and PCE-LSTM achieve the closest predictions to the actual HR, but the proposed method clearly captures the oscillations better and achieves smaller error. In the bottom plot (Subject 0), FCNN underestimates the HR, especially during the 5-10 min interval. Both CNN-IMU-2 and the Attention model incorrectly predict a rise followed by a plateau in the 2 - 3 min interval. PCE-LSTM outperforms them in the 1-5 min interval. After 5 minutes, there is a peak taking the HR to 110 bpm followed by

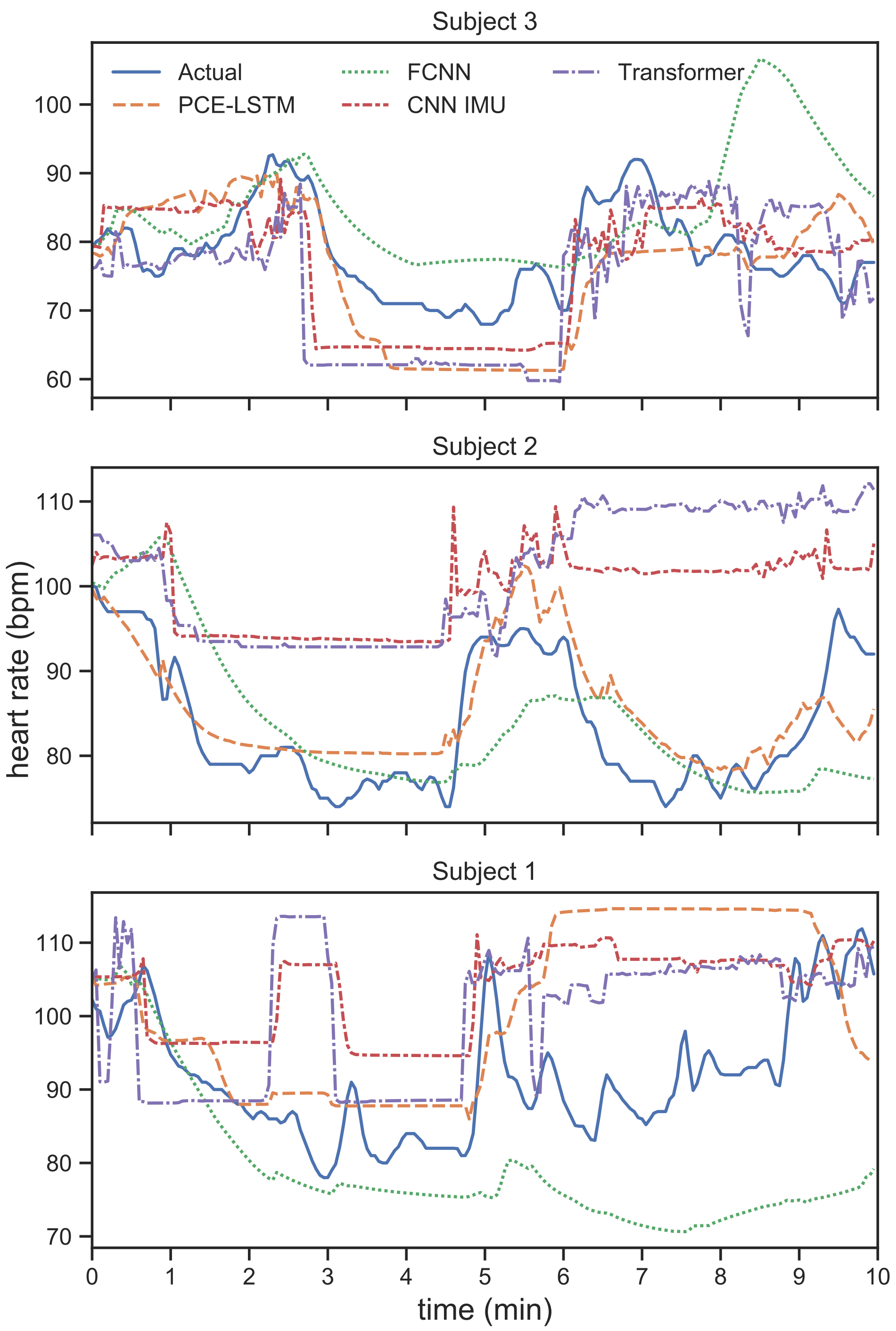


Figure 6.1: Model's predictions for the series' first 10min.

oscillations at around 90 bpm. CNN-IMU, the Attention Model and PCE-LSTM capture the rise in HR, but incorrectly keep the predictions roughly constant above 105 bpm. Among the three models, PCE-LSTM overestimates the HR the most, predicting it to be around 115bpm during that interval.

Model	ℓ_1 by Test Subject [beats/min]								ℓ_∞
	1	2	3	4	6	7	8	Avg	Avg
	Short-term prediction (480s)								
Suppressed PCE	15.24	10.20	11.17	14.43	14.91	18.82	16.94	14.53	59.94
LSTM self-encode	15.25	10.43	**9.76**	12.93	14.31	**16.87**	18.36	13.99	**58.31**
PCE on Previous Segment	**14.87**	**9.41**	12.75	**12.40**	**14.23**	17.14	**13.67**	**13.50**	58.76
	Long-term prediction (entire series)								
Suppressed PCE	22.26	14.92	18.11	**13.02**	20.34	20.01	47.12	22.26	73.57
LSTM self-encode	18.47	16.51	19.86	13.20	18.94	19.48	48.56	22.15	68.97
PCE on Previous Segment	**14.33**	**12.99**	**14.37**	17.95	**18.35**	**17.42**	**37.21**	**18.95**	**63.69**

Table 6.5: Mean Absolute Error (ℓ_1) and Maximum Absolute Error (ℓ_∞) for variations of the PCE-LSTM model (Ablation Study)

Performance Impact of PCE-LSTM's Hidden State Initialization

Here we demonstrate that PCE-LSTM's strategy to initialize the hidden state vectors is key to boost the model's performance. In our model, the LSTM's hidden state vectors are initialized by passing the sensor and heart rate data corresponding to the first two time snippets of the series through the Physical Conditioning Encoder (PC Encoder). Alternatively, one could consider using the LSTM itself to encode the relationship between sensor data and heart rate for a specific individual.

To quantify the performance impact of the proposed initialization, we conduct additional experiments with two alternative strategies of initialization: in "Suppressed PCE", PCE is ignored and hidden state vectors are initialized with 0s; in "LSTM self-encode", hidden state vectors are initialized by feeding the heart rate to the network as an additional input channel during the initialization segment (replaced by zeros during the prediction segment).

Table 6.5 shows the Mean Absolute Error (ℓ_1) metric for the experiments using the short-term segments and the entire series of each subject in the test set. We observe that using the "LSTM self-encode" is almost as good as using PC Encoder for the short-term segments, but this changes significantly when we analyze the performance on the entire series. In this scenario, "LSTM self-encode" performs poorly, only better than the configuration with no access to HR data, i.e., "Supressed PCE". This demonstrates empirically that the state encoded by PCE-LSTM contains useful information.

6.2.2 Direct Formulation

Under the direct formulation, we can only compare our method with recursive models. We experiment with predictions based on ensembles, where several instances of each model are fit to different sets and their individual predictions are combined.

Model	MAE by Test Subject [beats/minute]																ℓ_∞
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*Avg.*	*Avg.*
Mean																	
FFNN	13.2	10.5	12.9	9.0	45.0	**32.1**	13.7	16.0	**12.3**	**11.9**	**22.2**	17.8	17.2	12.7	14.0	17.4	63.7
DeepConvLSTM	9.8	7.7	16.1	12.9	44.1	34.9	15.8	**9.2**	14.8	13.2	25.5	**11.3**	15.3	13.6	11.2	17.0	69.4
PCE-LSTM	**9.3**	**6.5**	**12.2**	**8.7**	**42.0**	34.7	**11.3**	12.2	13.6	12.2	22.4	15.1	**10.9**	**11.5**	**9.2**	**15.5**	**51.3**
Ensemble																	
FFNN	12.2	8.0	11.4	7.7	45.0	**31.4**	12.4	14.5	11.5	8.6	22.1	17.3	15.0	11.2	10.6	15.9	60.1
DeepConvLSTM	9.2	7.1	15.3	12.7	43.9	34.7	15.4	**6.9**	13.9	12.9	25.4	**10.7**	15.1	13.2	10.5	16.5	68.1
PCE-LSTM	**8.4**	**5.1**	**7.8**	**6.6**	**41.9**	34.4	**7.4**	8.9	**11.4**	**8.4**	**19.6**	14.9	**9.3**	**9.8**	**8.5**	**13.5**	**48.6**

Table 6.6: Error Metrics on the DaLiA dataset (best shown in bold)

Model	Test Subject [beats/minute]									ℓ_∞
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*Avg.*	*Avg.*
FFNN (mean)	24.9	18.3	10.5	**17.2**	18.6	14.7	21.6	33.7	19.9	71.88
DeepConvLSTM (mean)	**14.7**	11.4	**8.8**	18.7	11.9	13.9	20.0	14.9	14.3	51.81
PCE-LSTM (mean)	16.6	**10.7**	9.0	19.5	**9.6**	**9.9**	**12.6**	**14.8**	**12.9**	**48.61**
FFNN (ensemble)	23.2	14.2	9.5	**15.8**	17.8	13.4	20.3	23.9	17.3	63.9
DeepConvLSTM (ensemble)	**11.8**	10.6	**8.3**	18.4	9.3	13.1	19.5	13.3	13.0	45.8
PCE-LSTM (ensemble)	16.5	**9.5**	8.4	16.7	**8.4**	**8.8**	**10.8**	**13.3**	**11.5**	**45.0**

Table 6.7: Mean Absolute Error (beats/min) on PAMAP2 (best shown in bold)

Here we use the discriminator network to improve the encoding of the PCE and investigate its impact on performance by conducting an ablation study.

Mean vs. Ensemble performance Since every method is trained 7 times for each test subject (using different train-validation splits), we compute a "mean" performance by averaging the errors of individual models and an "ensemble" performance by averaging the models' predictions and then computing the resulting error, as done by [25].

IMU-based multi-step heart rate estimation experiments

We begin by analyzing the error yielded by each method on the entire series for each subject, since we are mostly interested in predictions over the long term. Table 6.6 and 6.7 show the Mean and the Ensemble performances w.r.t. ℓ_1 by test subject and the ℓ_1 and ℓ_∞ averages, for PCE-LSTM and the baselines, on DaLiA and PAMAP2, respectively. We note that the series of subjects 5 and 6 of DaLiA can be regarded as outliers as none of the methods performed well on them. As expected, ensembles tend to outperform their standalone counterparts. Considering the ensemble performances, out of 23 subjects, FFNN, DeepConvLSTM and PCE-LSTM achieve the lowest errors for 2, 4 and 17 subjects, respectively. The lowest average error is obtained by the PCE-LSTM ensemble, with over 11.5% lower ℓ_1 and 19.1% lower ℓ_∞ than the next best method.

For a qualitative evaluation, we plot the predictions of each model. Figures 6.2 and 6.3 show the ensemble predictions for the complete series of five representative test subjects in the DaLiA and PAMAP2 datasets, respectively. We observe that DeepConvLSTM fails to

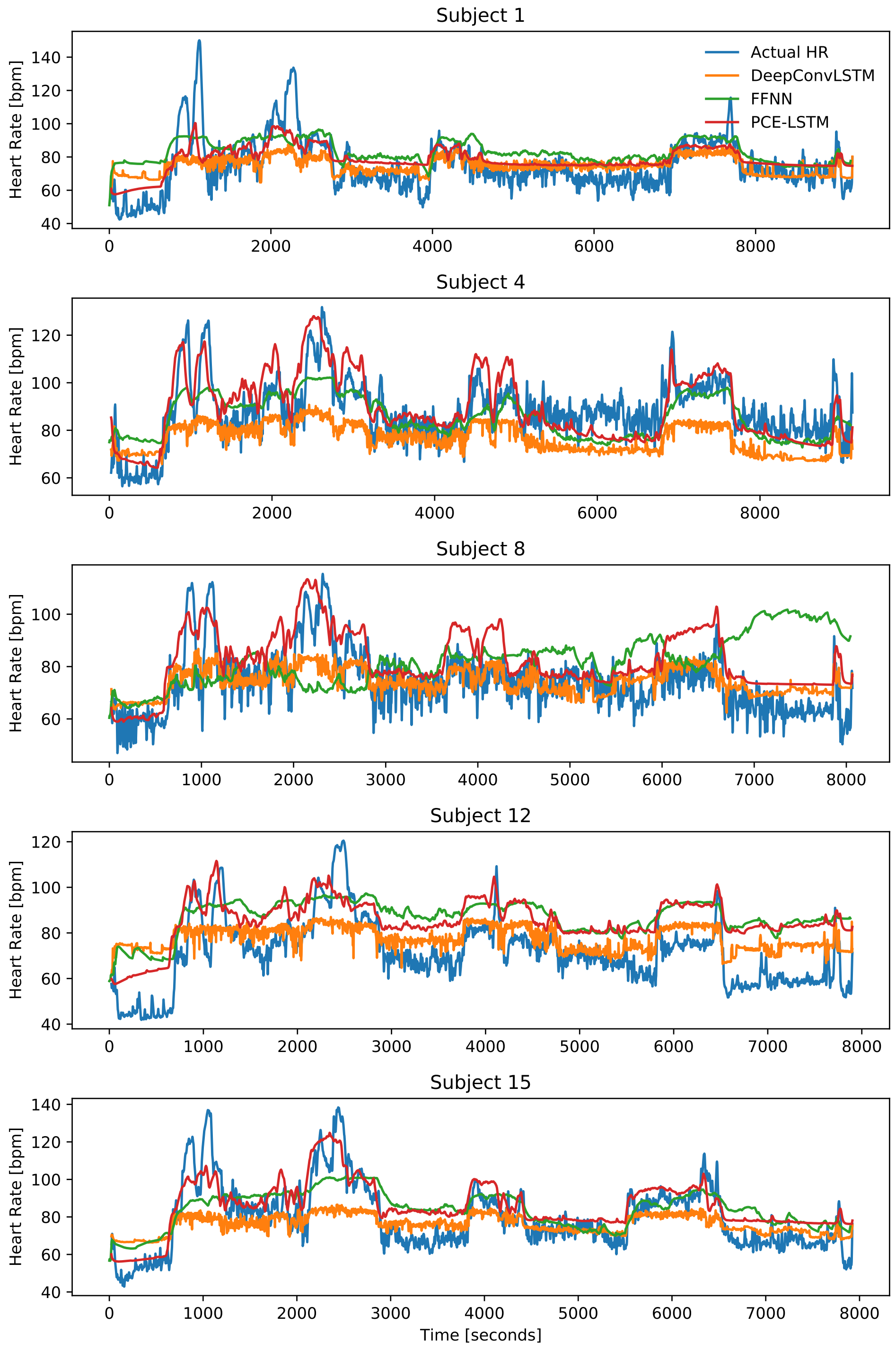


Figure 6.2: Long-term IMU-based HR estimation (DaLiA)

capture the variance of the HR series: although the predictions are correlated with changes in HR, they tend to remain close to an "average" HR. FFNN, in turn, exhibits more variance, but cannot accurately capture the amplitude of the peaks and sometimes overestimates the HR. In contrast, PCE-LSTM (our method) can capture both peaks and valleys more accurately than the baselines.

For a thorough comparison, we include some statistics related to the cost of each model

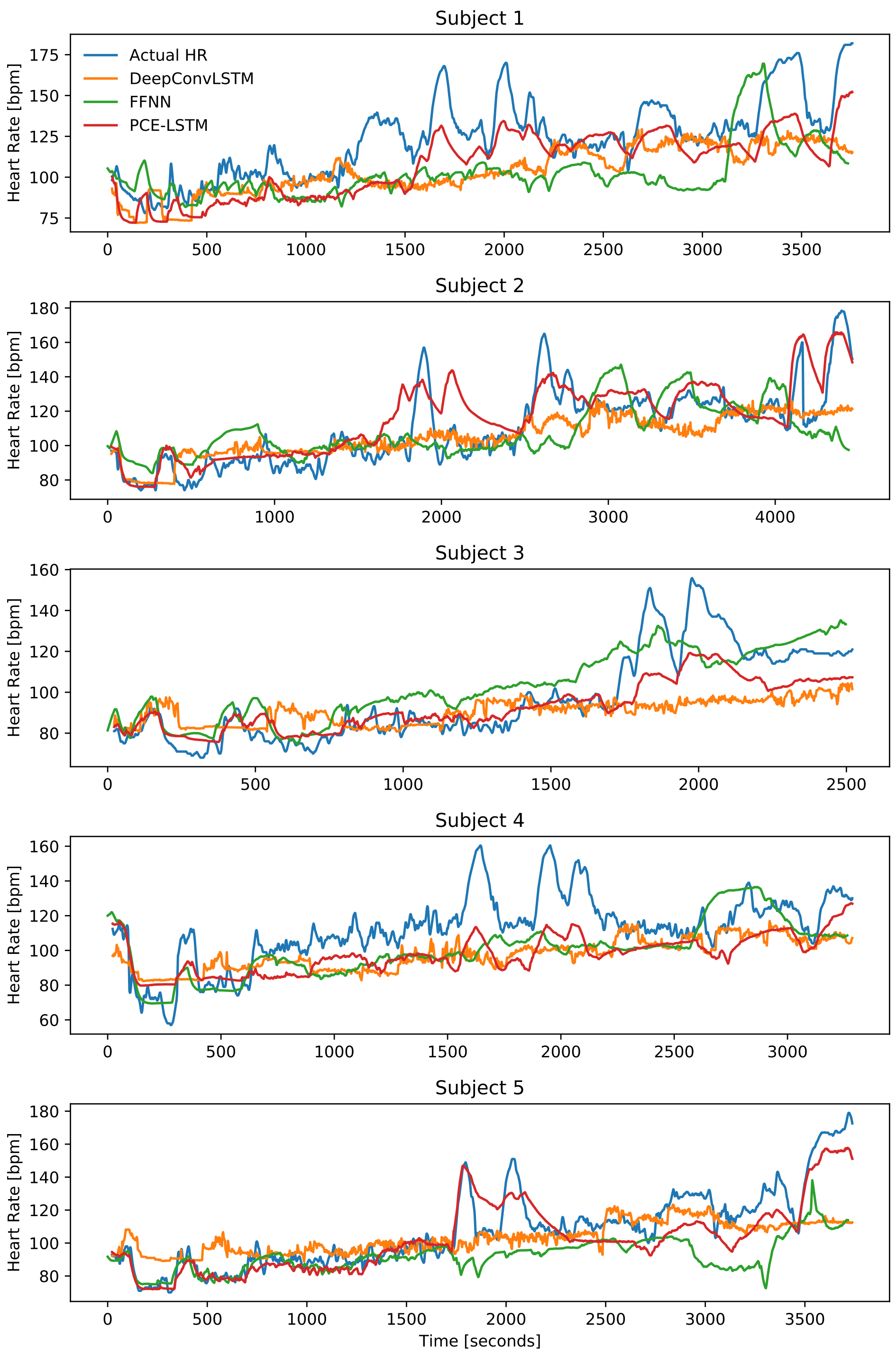


Figure 6.3: Long-term IMU-based HR estimation (PAMAP2)

in terms of memory and time resources. Table 6.8 shows the the number of parameters and training time required by each model, as well as the average time to compute a one-step prediction. Models sizes are larger for PAMAP2 because it has more accelerometers on the chest and wrist than DaLiA (and they are different from those under the delta formulation, because we downsampled the PAMAP2's signals from 100 to 32Hz and used less sensors under this formulation – only those on chest and wrist, in order to match those of DaLiA, which has no

Model	**number of parameters**	**train. time** [min]	**execution time** [ms]
	DaLiA		
DeepConvLSTM	490,177	77.1	5.23
PCE-LSTM	120,273	20.4	0.20
FFNN	726	4.1	0.50
	PAMAP2		
DeepConvLSTM	686,785	16.6	5.41
PCE-LSTM	120,561	5.1	0.14
FFNN	824	1.4	0.50

Table 6.8: Computational cost in number of parameters, training time and single-step prediction time

sensors on the ankle). Thus, we include separate statistics for each dataset. We highlight that the execution time for all models, on Google Colab's free servers, falls below the 2s interval required for online prediction, as this is the step period of the TS generating sliding window.

Performance impact of PCE-LSTM's hidden state initialization

Here we conduct an ablation study to demonstrate that (i) PCE-LSTM's strategy for initializing hidden state vectors is key to boosting the model's performance and that (ii) the use of a discriminator to distinguish whether two PCE come from the same individual or not contributes to the performance of the PCE-LSTM. A typical strategy to extract a hidden state is to use the LSTM itself to encode the relationship between sensor data and heart rate. In contrast, our model initializes the LSTM's hidden state vectors by passing the sensor and heart rate data corresponding to the first $I = 12$ time snippets of the series through the Physical Conditioning Encoder (PC Encoder) to extract the embedding called PCE. As we assume that the PCE encodes, at least in part, an individual's physical conditioning, we expect that the joint training of the PCE-LSTM regression and the discriminator will improve the regression predictions by promoting a better training of the PC Encoder subnetwork.

To quantify the performance impact of the proposed initialization and of using the discriminator, we conduct additional experiments with alternative initialization strategies. In what follows, "with discr." indicates the joint training with the discriminator, "without discr." indicates that the discriminator is not used; and "self-encode" indicates that hidden state vectors are initialized by feeding the heart rate to the network as an additional input channel in the TS of the Initialization Segment (replaced by zeros during the prediction segment), so that the PC Encoder subnetwork is not utilized and the LSTM initial hidden vectors are initialized with zero vectors, that is, the initialization segment is used as a washout period.

Tables 6.9 and 6.10 show the ℓ_1 and ℓ_∞ metrics for these experiments on the DaLiA and PAMAP2 datasets, respectively. On DaLiA, PCE-LSTM's outperforms an LSTM self-encode

	Test Subject [beats/minute]															
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*Avg.*
	Mean Absolute Error															
self-encode	11.0	7.6	**7.1**	7.4	48.6	33.6	10.7	15.6	11.5	9.3	21.2	15.2	10.7	10.7	10.4	15.4
without discr.	10.5	5.9	13.7	7.6	43.7	**29.5**	7.8	14.3	**9.4**	9.5	**18.8**	**13.6**	**9.1**	11.4	**7.7**	14.2
with discr.	**8.4**	**5.1**	7.8	**6.6**	**41.9**	34.4	**7.4**	**8.9**	11.4	**8.4**	19.6	14.9	9.3	**9.8**	8.5	**13.5**
	Maximum Absolute Error															
self-encode	52.6	**30.1**	39.4	29.6	102.6	79.1	49.2	51.4	60.1	52.7	69.0	44.4	51.5	56.2	**31.0**	53.2
without discr.	**50.3**	31.5	**35.5**	32.8	89.2	**71.5**	45.6	38.3	**49.2**	49.6	**61.5**	**28.3**	**43.7**	61.1	37.4	**48.4**
with discr.	52.7	37.2	36.6	**24.1**	**88.9**	77.9	**44.7**	**29.0**	50.4	**48.9**	66.4	29.0	55.0	**54.2**	34.2	48.6

Table 6.9: Ablation study: error results for variations of the PCE-LSTM model, in ensemble configuration, on DaLiA

	Test Subject [beats/minute]								
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*Avg.*
	Mean Absolute Error								
self-encode	**14.60**	11.27	**7.92**	**13.43**	**7.90**	9.95	13.83	14.17	11.63
without discr.	14.98	11.31	8.67	17.88	9.36	9.42	11.71	**13.10**	12.05
with discr.	16.46	**9.46**	8.34	16.70	8.40	**8.80**	**10.79**	13.25	**11.52**
	Maximum Absolute Error								
self-encode	44.65	51.71	**37.83**	**48.12**	31.87	45.66	**36.26**	55.73	**43.98**
without discr.	**40.98**	**46.76**	41.15	57.40	33.39	48.56	45.56	57.57	46.42
with discr.	42.69	52.28	42.68	50.83	**27.83**	**43.77**	47.46	**52.71**	45.03

Table 6.10: Ablation study: error results for variations of the PCE-LSTM model, in ensemble configuration, on PAMAP2

initialization strategy even without a discriminator. Nevertheless, the use of the discriminator reduces the error even further. On the other hand, on PAMAP2, PCE-LSTM without the discriminator yields larger error than the LSTM self-encode strategy. Even though the complete architecture – including the discriminator – achieves the lowest average error, the relative gains w.r.t. LSTM self-encode are marginal.

What can explain the difference between the results on the two datasets? To answer this question, we analyze the discrimination performance as a measure of the PCE's quality. Figure 6.4 shows the discriminator accuracy for DaLiA and PAMAP2, respectively, when the threshold is set to 0.5 (i.e., the discriminator returns "same person" when the predicted probability is above 50%). Although the overall accuracy is not very high, it is clear that it is higher on the DaLiA dataset, indicating that the resulting PCEs are more representative of the subjects' physical conditioning. We conjecture the higher difficulty of encoding the physical conditioning on the PAMAP2 dataset might be due to its smaller size and to its more varied set of physical activities.

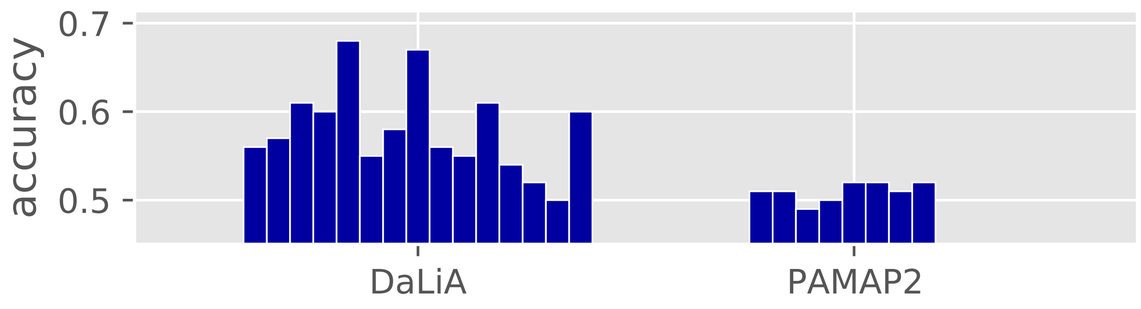


Figure 6.4: discriminator accuracy among test subjects in DaLiA and PAMAP2 datasets

Chapter 7

Secondary application: PPG-based HR estimation

We adapt PCE-LSTM for a secondary prediction task, namely, the PPG-based heart rate estimation. While being close to the IMU-based HR estimation, the former task has been better explored in the literature and has, consequently, well established baselines. We perform this analysis to show that the proposed model is fertile, in the sense that is has other potential applications.

Here we describe some adaptations to our model in order to make it better suited for PPG-based HR estimation and compare the performance of our model to the deep learning-based SOTA model [25].

7.1 Adaptations for PPG-based HR Estimation

The task of PPG-based HR estimation has three main distinctions from the IMU-only HR estimation and hence requires a few adaptations to our method. The differences are the following:

1. It is useful to represent the PPG signal in the frequency domain;
2. The PPG signal is more important than the other signals, since it is a rough estimation of the HR;
3. For PPG-based HR estimation, the heart rate predictions are done without a ground-truth HR at any point in time;

In order to deal with these differences, instead of using a single TS encoder module, we used two: one for the raw PPG and IMU signals (TSE_{raw}) and the other for the Fast Fourier Transformed-PPG signal ($\text{TSE}_{\text{PPG-FFT}}$). Their outputs are concatenated before being passed onto the next subnetwork (either the PC encoder or the LSTM). The $\text{TSE}_{\text{PPG-FFT}}$ has a smaller

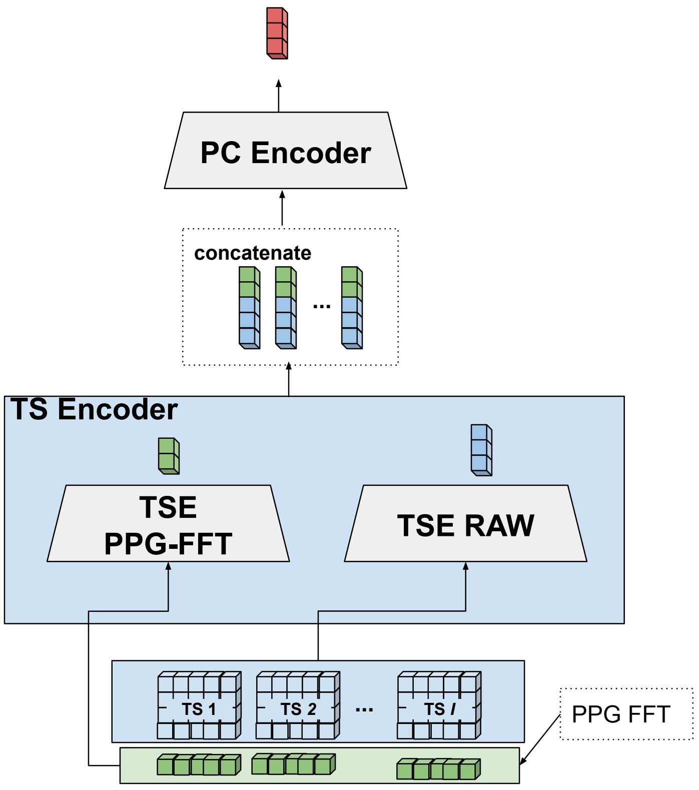


Figure 7.1: PPG-adapted PCE

output size: $[\text{TSE}_{\text{PPG-FFT}}]_{\text{out}} = 12$. This accounts for the first two differences, as we leverage the information contained in PPG signals by representing them in the frequency domain and by having a TS encoder exclusively for this input.

We have also slightly modified the PC Encoder, so that it only receives as input the concatenated outputs of the TSE (without the HR), hence addressesing the last difference. Figure 7.1 illustrates the changes to the PCE-LSTM architecture for the secondary task.

7.2 Results

The deep learning-based state of the art model for PPG-based HR estimation in the literature was proposed in [25]. This method is based on CNNs and will be referred simply as CNN. For this application we consider only MAE (ℓ_1), since we transcribe the results from the paper where the baseline method was proposed, which did not include other evaluation metrics.

In order to match the length and step size of each time snippet in [25], we set $\tau_{\text{TS}} = 8$ and $r_{\text{TS}} = 0.75$ for this task. Table 7.1 shows the ensemble performance for each method on PPG-DaLiA and WESAD, when the individual designated in the column is the test subject. The last column contains the row average. On both datasets, PCE-LSTM provides reductions in MAE of approximately 32% when compared to the SOTA method and has lower MAE for every subject, except for subject 8 in the WESAD dataset (even then, difference is under 10%). Another advantage of PCE-LSTM is that it has roughly two order of magnitude less parameters then the CNN, with approximately 120k parameters for PCE-LSTM vs. approximately 8500k parameters for the CNN model (according to [25]).

Model	**PPG-DaLiA**															
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*Avg.*
[25]	7.7	6.7	4.0	5.9	18.5	12.9	3.9	10.9	8.8	4.0	9.2	9.3	4.3	4.4	4.2	7.6
PCE-LSTM	**5.5**	**3.8**	**2.5**	**5.4**	**11.0**	**5.5**	**2.6**	**9.1**	**6.6**	**2.6**	**5.5**	**8.5**	**2.7**	**3.7**	**3.4**	**5.2**
Model	**WESAD**															
	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*13*	*14*	*15*	*16*	*17*	*Avg.*
[25]	5.1	14.5	7.8	7.7	3.9	6.8	**4.3**	4.0	8.9	11.1	6.5	5.3	4.2	12.8	9.4	7.5
PCE-LSTM	**3.6**	**9.8**	**3.5**	**4.7**	**2.6**	**4.6**	4.6	**3.0**	**4.9**	**7.1**	**4.8**	**4.7**	**3.5**	**4.9**	**8.4**	**5.0**

Test Subject (MAE [beats/min])

Table 7.1: PPG-based heart rate estimation experiments

Chapter 8

Conclusions and Future Work

In this chapter we describe how our research contributed to the body of knowledge on modeling heart rate using Neural Networks. We also present how the topics of this dissertation might be further investigated in future research.

8.1 Conclusion

In this dissertation we investigated the – much neglected – task of predicting heart rate from IMU sensor data. We achieved our goal of proposing a new model for this task, which outperforms the existing neural network devised for the same task [38] as well as models known to perform well on the related task of HAR [22, 26], which were minimally adapted for the task at hand,

We started from the premise that, depending on their physical conditioning, different people will display different heart rates when performing the same exercises. We proposed a neural architecture dubbed the Physical Conditioning Embedding LSTM (PCE-LSTM) that employs a convolutional network to extract vectors which carry information about the relationship between sensor measurements and the heart rate for a specific individual, thus representing his/her physical conditioning. These vectors are used as the initial state vectors for an LSTM network that outputs heart rate predictions from sensor data.

We then considered two formulations for the problem:a delta formulation, which enabled us to compare our neural architecture against non-recursive models; and a direct formulation, under which we performed studies on ensemble strategies and the use of a discriminator to improve the physical conditioning embedding.

We evaluated the prediction accuracy of PCE-LSTM w.r.t. the mean absolute error (ℓ_1) and maximum absolute error (ℓ_∞) using sensor data from public datasets (PAMAP2, PPG-DaLiA, WESAD). Under the delta formulation, PCE-LSTM yielded about 9% lower mean absolute error than the next best baseline; and, under the direct formulation, it yields over 10% lower mean absolute error in the IMU-based heart rate estimation task and 30% lower mean absolute error in PPG-based heart rate regression. Last, we conduct additional experiments to show that the performance gains achieved by our method are, in part, due to the strategy used to initialize the hidden vectors. Specifically, using the outputs of the PCE network applied to the data from the previous few seconds of the subject's time series is helpful and works better than using the LSTM itself to output hidden vectors from the same data, especially when using physical conditioning embedding discriminator during training.

8.2 Future Work

Our research could be complemented by the investigation of other architectures for learning a physical conditioning embedding. From a more general perspective, new architectures can be proposed to improve HR estimation, such as those based on Attention Transformers, which are increasingly regarded as the best architecture for sequential data, not only for language modeling, but also for other tasks, such as the protein folding problem. For these respective tasks, BERT [7] and AlphaFold [10] are notable examples of groundbreaking models.

In this section, however, we propose two research directions derived from the study in this dissertation, which are not just improvements on our research. The first relates to leveraging our approach for initializing an RNN's hidden state in other domains. The second is related to bringing the results of this research into practical use.

8.2.1 RNN Initialization For Dynamical Systems

A dynamical system represents the evolution of a state x as a function of time $t, t > t_0$ and input $u(t)$, given the systems internal parameters ϕ and an initial state $x_0 = x(t_0)$. Recursive Neural Networks, like dynamical systems, carry information through time and allow for an infinite unfolding over time. For this reason, it is a commonly used type of NN for modeling such systems.

In an RNN model, a system parameter – such as the drag coefficient of an aerial vehicle or a person's genetics for the human heart rate – is usually represented in the RNN's weights, and the system's initial state – such as the speed of a aerial vehicle, or the accumulated physical strain for human heart rate – is normally represented in the RNN hidden state vector, which is built up with new iterations, where a washout period is commonly defined, before which the predictions are to be discarded.

[19] proposed a way to improve the system's initial state representation in the RNN initial state vector, in the context of aerial vehicle dynamics. Although initially unaware of that work, we proposed a method, in the domain of heart rate dynamics, that promotes the encoding of the system's initial state, combined with the system parameters (which we named "physical conditioning" in the case of estimating the HR in as a function of physical activities) into the RNN state vectors.

Our method is based on the rationale that the system parameters (physical conditioning) are approximately constant in the short term, and, if they are contained in the RNN hidden state, a NN (discriminator) could distinguish whether the system parameters of two systems (subjects) are the same using the RNN's hidden vectors. We conjecture that the joint training of the discriminator network with the main model should improve the encoding of the system parameters into the hidden state, promoting a better training of the model. Our experiments, limited to the HR estimation domain, corroborate this line of thinking.

An important future work would be to check if our proposed method generalizes to other types of dynamic systems and to propose new components to the ablation study that could provide better insight on how the system parameters are being encoded in the RNN hidden state. A challenge in this study would be obtaining suitable datasets which would contain data from multiple dynamical systems. [19] exemplifies this challenge as each of the datasets used contained a single aerial vehicle, therefore being inadequate for one such comparative study. An alternative would be to simulate the systems, as [36] did in their study, where they used constructed problems, such as the auto-regressive task of predicting damped oscillations ($x(t) = Ae^{-bt}\cos(ct)$), to test their hypothesis.

8.2.2 Designing Custom Exercise Prescriptions

A motivational application for the study in this dissertation is enabling the design of a prescription of a custom exercise sequence which would elicit a desirable HR response. Yet, the work of this dissertation alone cannot reach this desirable outcome. In this section, we attempt to direct further studies based on our work to turn this research into a practical tool.

The problem could be more formally stated as "given an intended HR response $H(t)$ during a period $p = t_0, ..., T$, and an initial Physical Conditioning Embedding PCE, which exercises, represented by an activity class $A_C(t)$ and an intensity $A_I(t)$, should a person perform to elicit $H(t)$?"

Our model (PCE-LSTM) converts PCE into the LSTM state vectors (h_0, c_0) and maps the IMU signals from a Time Snippet i (IMU$_i$) into a HR value H:

$$h_{i-1}, c_{i-1}, \text{IMU}_i \Rightarrow H_i, h_i, c_i.$$

Our model also converts the IMU signals from a time snippet, into an embedding (TSE$_i$). Therefore, we have:

$$h_{i-1}, c_{i-1}, \text{TSE}_i \Rightarrow H_i, h_i, c_i.$$

While public datasets have annotations of the class of the activity, they do not quantify the intensity of the activities. We could estimate the activity intensity as a measure proportional to the increase in HR over the period of a time snippet (TS), creating one activity intensity label per TS. As the variation in HR also depends on the individual's physical conditioning and the activity being performed just before that, this estimation could be very inaccurate. However, we could simulate this variations of HR under different previous conditions and subjects using our model, potentially improving the activity intensity estimation.

In order to map ($A_C(t_i)$, $A_I(t_i)$) to a HR, given the previous state h_{i-1}, c_{i-1}, we could sample a time snippet from the training set, that has the same class A_C and has the closest intensity value to the intended A_I, and use our model to predict the HR. Therefore, we can now map:

$$h(t_{i-1}), c(t_{i-1}), A_C(t_i), A_I(t_i) \Rightarrow H_i, h_i, c_i.$$

That implies that we can now simulate the entire HR dynamics during the period as a function of the exercises $(A_C(t), A_I(t))$ and PCE. Finally, optimization algorithms can be used to find out a series of exercises to elicit a HR that closely matches the intended HR.

What if the intended data for activity class and intensity is not present in the training set and the closest value is significantly different from the intended one? We propose the use of a conditional Generative Adversarial Networks (Conditional GAN) to interpolate the exercises in the training set. In this type of NN, the generator would create a series of times snippets (TS), given A_C, A_I and a random vector and produce a plausible series of TS. The discriminator receives a series of TS, either real or generated, A_C and A_I and classifies as being real data or simulated and whether it corresponds to the said exercise.We understand that other approaches, such as the use of variational auto-encoders might also be successfully used for this task

This still leaves as an open question how to convert the estimation of activity intensity into a value that people can understand and realize it. It seems that a dataset with annotations of activity intensity would be required and, with that in hand, this would boil down to a standard regression problem.

Bibliography

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[2] Alexander Artiga Gonzalez, Raphael Bertschinger, and Dietmar Saupe. Modeling vo2 and vco2 with hammerstein-wiener models. In Pedro Pezarat Correia, editor, *Proceedings of the 4th International Congress on Sport Sciences Research and Technology Support : volume 1 icSports*, pages 134–140, Setubal, 2016. Scitepress.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] M. Boloursaz Mashhadi, E. Asadi, M. Eskandari, S. Kiani, and F. Marvasti. Heart rate tracking using wrist-type photoplethysmographic (ppg) signals during physical exercise with simultaneous accelerometry. *IEEE Signal Processing Letters*, 23(2):227–231, 2016.

[5] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[6] T. M. Cheng, A. V. Savkin, B. G. Celler, Lu Wang, and S. W. Su. A nonlinear dynamic model for heart rate response to treadmill walking exercise. In *IEEE IEMBS*, pages 2988–2991, 2007.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Odongo Steven Eyobu and Dong Seog Han. Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network. *Sensors*, 18(9), 2018.

[9] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *IJCAI*, pages 1533–1540, 2016.

[10] Lim Heo and Michael Feig. High-accuracy protein structures by combining machine-learning with physics-based refinement. *Proteins: Structure, Function, and Bioinformatics*, 88(5):637–642, 2020.

[11] Kenneth J Hunt and Andrzej JR Hunt. Feedback control of heart rate during outdoor running: A smartphone implementation. *Biomedical Signal Processing and Control*, 26:90–97, 2016.

[12] Dae-Geun Jang, Byung-Hoon Ko, Sub Sunoo, Sang-Seok Nam, Hun-Young Park, and Sang-Kon Bae. A preliminary study of a running speed based heart rate prediction during an incremental treadmill exercise. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5323–5326. IEEE, 2016.

[13] Delaram Jarchi and Alexander J. Casson. Description of a database containing wrist ppg signals recorded during physical exercise with both accelerometer and gyroscope measures of motion. *Data*, 2(1), 2017.

[14] Artur Jordao, Jr. Nazare, Antonio C., Jessica Sena, and William Robson Schwartz. Human Activity Recognition Based on Wearable Sensor Data: A Standardization of the State-of-the-Art. *arXiv e-prints*, page arXiv:1806.05226, June 2018.

[15] Boreom Lee, Jonghee Han, Hyun Jae Baek, Jae Hyuk Shin, Kwang Suk Park, and Won Jin Yi. Improved elimination of motion artifacts from a photoplethysmographic signal using a kalman smoother with simultaneous accelerometry. *Physiological Measurement*, 31(12):1585–1603, oct 2010.

[16] H. Lee, H. Chung, and J. Lee. Motion artifact cancellation in wearable photoplethysmography using gyroscope. *IEEE Sensors Journal*, 19(3):1166–1175, 2019.

[17] Melanie Ludwig, Katrin Hoffmann, Stefan Endler, Alexander Asteroth, and Josef Wiemeyer. Measurement, Prediction, and Control of Individual Heart Rate Responses to Exercise—Basics and Options for Wearable Devices. *Frontiers in Physiology*, 9:778, 2018.

[18] Michael Mazzoleni, Claudio Battaglini, Kerry Martin, Erin Coffman, and Brian Mann. Modeling and predicting heart rate dynamics across a broad range of transient exercise intensities during cycling. *Sports Engineering*, 19:117–127, 01 2016.

[19] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3370–3383, 2019.

[20] Sami Mohammad, Thierry Marie Guerra, Jean Marie Grobois, and Bernard Hecquet. Heart rate control during cycling exercise using Takagi-Sugeno models. *IFAC Proceedings Volumes*, 44(1):12783–12788, 2011.

[21] Kusprasapta Mutijarsa, Muhammad Ichwan, and Dina Budhi Utami. Heart rate prediction based on cycling cadence using feedforward neural network. In *IEEE IC3INA*, pages 72–76, 2016.

[22] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[23] M. R. Ram, K. V. Madhav, E. H. Krishna, N. R. Komalla, and K. A. Reddy. A novel approach for motion artifact reduction in ppg signals based on as-lms adaptive filter. *IEEE Transactions on Instrumentation and Measurement*, 61(5):1445–1457, 2012.

[24] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *IEEE ISWC*, pages 108–109, June 2012.

[25] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep PPG: large-scale heart rate estimation with convolutional neural networks. *Sensors*, 19(14):3079, 2019.

[26] Fernando Moya Rueda, René Grzeszick, Gernot A Fink, Sascha Feldhorst, and Michael Ten Hompel. Convolutional neural networks for human activity recognition using body-worn sensors. *Informatics*, 5(2), 2018.

[27] Seyed Salehizadeh, Duy Dao, Jeffrey Bolkhovsky, Chae Cho, Yitzhak Mendelson, and Ki Chon. A novel time-varying spectral filtering algorithm for reconstruction of motion artifact corrupted heart rate signals during intense physical activities using a wearable photoplethysmogram sensor. *Sensors*, 16(1):10, Dec 2016.

[28] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger, and Kristof Van Laerhoven. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ICMI '18, page 400–408, New York, NY, USA, 2018. Association for Computing Machinery.

[29] T. Schäck, M. Muma, and A. M. Zoubir. Computationally efficient heart rate estimation during physical exercise using photoplethysmographic signals. In *EUSIPCO*, pages 2478–2481, 2017.

[30] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR*, abs/1909.09586, 2019.

[31] A. Temko. Estimation of heart rate from photoplethysmography during physical exercise using wiener filtering and the phase vocoder. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1500–1503, 2015.

[32] Paul van Gent, Haneen Farah, Nicole van Nes, and Bart van Arem. Analysing noisy driver physiology real-time using off-the-shelf sensors: heart rate analysis software from the taking the fast lane project. *Journal of Open Research Software*, 7(1), 2019.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[34] Kishor H Walse, Rajiv V Dharaskar, and Vilas M Thakare. Pca based optimal ann classifiers for human activity recognition using mobile sensors data. In *ICTIS: Volume 1*, pages 429–436. Springer, 2016.

[35] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[36] Sam Wenke and Jim Fleming. Contextual recurrent neural networks, 2019.

[37] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *arXiv e-prints*, page arXiv:2001.08317, January 2020.

[38] Feng Xiao, Ming Yuchi, Mingyue Ding, and Jun Jo. A research of heart rate prediction model based on evolutionary neural network. In *IEEE ICBMI*, pages 304–307, 2011.

[39] J. Xiong, L. Cai, D. Jiang, H. Song, and X. He. Spectral matrix decomposition-based motion artifacts removal in multi-channel ppg sensor signals. *IEEE Access*, 4:3076–3086, 2016.

[40] Ming Yuchi and Jun Jo. Heart rate prediction based on physical activity using feedforwad neural network. In *IEEE ICHIT*, pages 344–350, 2008.

[41] Maria S Zakynthinaki. Modelling heart rate kinetics. *PloS one*, 10(4):e0118263, 2015.

[42] Haibin Zhang, Bo Wen, and Jiajia Liu. The prediction of heart rate during running using bayesian combined predictor. In *IEEE IWCMC*, pages 981–986, 2018.

[43] Z. Zhang, Z. Pi, and B. 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):522–531, 2015.

[44] Hans Zimmermann, Ralph Grothmann, Anton Schaefer, and Christoph Tietz. 8 modeling large dynamical systems with dynamical consistent neural networks. In Simon Haykin, Jose C. Principe, Terrence J. Sejnowski, and John McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brains*, chapter 8. The MIT Press, 2006.