Deep learning to estimate power output from breathing

Erik Johannes Bjørnson Løvenskiold Grüner Husom

Thesis submitted for the degree of
Master in Computational Science: Physics
60 credits

Department of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2021

Deep learning to estimate power output from breathing

Erik Johannes Bjørnson Løvenskiold Grüner Husom

Deep learning to estimate power output from breathing

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

In collaboration with

Secure IoT systems group
SINTEF Digital
Oslo, Norway

Abstract

Activity tracking devices are widely used for monitoring and measuring different aspects of physical activity, such as heart rate, speed, energy expenditure and power output. Breathing variables like ventilation and oxygen uptake have also been used for estimating physical effort during exercise. The link between exercise intensity and the muscles' increased need for oxygen makes breathing a universally applicable metric across many activity forms. Breathing can be measured by standard equipment such as exercise spirometers, but they are impractical to use in normal exercise situations, because they cover the face or mouth and often require stationary recording equipment. Respiratory inductive plethysmography (RIP) is a method for measuring the movement of the rib cage and abdomen caused by breathing, and it enables us to have a portable, non-invasive way of recording breathing during exercise. Power output is a direct measurement of the work performed by a person doing physical exercise, and is a way of expressing the person's energy expenditure. This thesis studies whether we can use RIP signals to create predictive models for estimating power output during exercise.

An N-of-1 study was performed on a highly active adult male of age 25, who performed 21 workouts on a stationary bike. RIP signals from the rib cage and abdomen, in addition to heart rate and power output were recorded during the workouts. The data acquired was used to build predictive models, by using three different deep learning methods: Dense neural networks (DNN), convolutional neural networks (CNN) and long short-term memory (LSTM) networks. A person's breathing pattern may have characteristics that differ from that of another person, which is a reason for building personalized models.

A CNN trained on a combination of features derived from RIP signals and heart rate obtained a mean absolute percentage error (MAPE) of 0.20, which means a 20% error on average with respect to the ground truth. The model's ability to give precise predictions during workouts with large fluctuations in power output significantly outperformed the DNN and LSTM models tested in this study. We conclude that deep learning techniques can be used for creating personalized models that estimate power output from RIP signals.

Acknowledgements

Firstly, I would like to thank my main supervisor, senior research scientist Sagar Sen, who has provided continuous guidance and motivation throughout my work on this thesis. His experience and dedication has been invaluable when facing difficulties, and his supervision has enabled me to have confidence in my work. He always brings a positive energy and inspiring attitude, which has been crucial in making my work on this thesis a pleasant and enjoyable experience.

I am also thankful to my co-supervisor Pierre Bernabé, who has provided helpful advice and stimulating discussions regarding both theoretical and practical challenges during my work on this thesis.

My co-supervisor at the University of Oslo, Morten Hjorth-Jensen, has been a huge inspiration during my time as a master student at the Physics Department, and I would like to thank him for the care he shows his students, and for the way he motivates those around him through his passion for science and research.

I would also like to thank my parents, Hilde-Anette and Bjørn Olav, for their enthusiasm and mental support during challenging times. I am also grateful that they, together with my sister Erle-Andrea, volunteered for the data collection of this thesis, even though the data they contributed were in the end not used in the final results.

Lastly, I want to express my profound gratitude to my patient and loving wife, Ragnhild, who has given me constant support and encouragement, especially during the final months of completion of this thesis.

Contents

List of Figures

List of Tables

Acronyms

CNN	Convolutional neural network.
DNN	Dense neural network.
LSTM	Long short-term memory; a type of neural network.
MAPE	Mean absolute percentage error.
MSE	Mean squared error.
RIP	Respiratory inductive plethysmography.

Chapter 1

Introduction

1.1 Motivation

The motivation behind this thesis starts at the intersection of health and technology. Through science we gain increasing insights to how our bodies work, and which stimuli are needed to make it function optimally. We know that regular physical activity is needed to keep our bodies healthy and operative, and physical inactivity is a major contributor to non-communicable diseases on a global scale[1]. Physical inactivity was in fact listed as the fourth leading risk factor by the World Health Organization in 2009[2]. At the same time, many modern jobs are mainly sedentary, and the act of being physically active in daily life needs to be a conscious effort. In earlier parts of human history, we were forced to use our bodies frequently, because our bodies was our main means of transportation, and in general we were more dependent on physical labour. In modern society, the advancement of technology has enabled more and more things to be automated and handled by machines. We do not need to walk, because we have cars, trains and elevators. Jobs requiring hard physical labour are made more effective by using machines (gas, steam and electric engines) that do the heavy lifting for us. Although utilizing technological advancements to increase productivity and efficiency, it has a clear downside: We no longer get the natural physical stimuli through daily life that our bodies need to stay healthy.

Technology has not only caused a decrease in the need for general physical activity, but it has also enabled us to make a wide range of sensors that can monitor human activity[3]. These can be compact in size and easily embedded into portable devices, and make it possible to measure several variables while we are on the move. Examples include smartphones and smartwatches, which can use accelerometers[4], magnetometers[5] and gyroscopes[6] to track the physical movements of a person carrying such a device.

The use of activity tracking devices is not only commonly used through the ubiquitousness of smartphones, but it is especially prevalent in competitive sports. This goes for both amateurs and professionals. Athletes monitor their training in order to keep track of their performance, and it is a valuable tool for coaches that oversee the progress of an athlete. In modern times, the practice of quantified training monitoring started in the 1930s when interval workouts were introduced as a training methodology[7], which involves alternating between high intensity bouts of activity and active or passive rest periods. Both speed and heart rate were from the beginning commonly used as the guiding measure of intensity in these types of workouts, and these methods are still common today[7].

Heart rate monitors are particularly popular as activity tracking devices, and the first wireless ECG-based heart rate monitors produced for the general public came from the company Polar Electro Oy in 1983[8]. This type of sensor measures the frequency of the heart beat, and is one of the most common methods for tracking exercise intensity[9]. One of the challenges with heart rate as a measure of physical effort is *cardiovascular drift*. This is a phenomenon where the heart rate increases during prolonged exercise, even though the workload remains more or less unchanged[10]. Measurement of speed during exercise is also a common form of tracking physical activity, specifically in sports that involve moving across a distance. However, it fails to give a fair representation of the intensity when we have external factors such as varying terrain, surface and weather.

Power meters are another type of activity trackers that are used to monitor and control physical activity. In contrast to heart rate monitors, which measure the bodily response of physical activity, power meters measure actual force applied by a person on some sensor. These devices have been used for a long time in the sport of road cycling. Measuring power output may be regarded as one of the most direct methods of measuring physical effort during exercise[11]. Despite giving a very accurate representation of the work done while exercising, power meters have largely been exclusively used in cycling, in addition to activities that involve stationary ergometers. The dynamics of the bike makes it easy to place a sensor inside the system of propulsion, often in the pedals or the crank. This type of sensor usually consist of a set of strain gauges, and the deformation of these is proportional to the torque created by the cyclist[12]. The force applied by the cyclist can thus be measured close to the contact point between the rider and the system he or she applies the force on. In other common endurance sports, such as running, rowing, cross-country skiing or swimming, the contact points between a person and the environment do not provide an interface where it is trivial to embed a force sensor. An exception to this exists in the case of exercise ergometers, for example stationary bicycles (e.g. Concept2 BikeErg[1]), indoor rowing machines (e.g. Concept2 RowErg[2]), ellipticals (e.g. Technogym Syncro Forma[3]), or skiing machines (e.g. Concept2 RowErg[4]). These type of contraptions often contain an ergometer, which is equipment for measuring the work performed by the person using the machine. Ergometers usually consists of a flywheel that can be rotated by a mechanical drivetrain, connected to for example a set of bike pedals. The flywheel has a resistive load, which can be provided by a friction belt or air resistance, and the power output is calculated from the resistive load and the movement of the flywheel.[13][14]. While ergometers can be used to measure power output, they are restricted to stationary use, usually indoors.

The aforementioned methods of activity tracking each have their weaknesses. The monitoring of heart rate attempts to quantify the intensity of exercise, but suffers from cardiovascular drift. Speed is only applicable in contexts with stable external conditions. Direct measurements of power output requires an embedded sensor in the system of propulsion, or stationary equipment in the form of exercise ergometers.

Because of this, we turn to a largely unexplored variable in the context of activity tracking in fitness and sports: Breathing. The act of moving air in and out of the lungs is vital for supplying the body with oxygen, which it uses to produce energy. Breathing is thus fundamentally connected to physical activity, and it is a metric that is applicable in any form of physical exercise. Exercise spirometry is a typical method for measuring ventilation, and is based on measuring the flow of air through a tube held in the mouth[15][16]. However, this is unsuitable for use in the field, since the equipment involved is not easily portable. A non-invasive, portable method of recording breathing is respiratory inductive plethysmopgrahy (RIP). This method measures the expansion and contraction of the chest and abdomen, by the use of straps or wires worn around the upper body. Breathing variables have previously been suggested as a possible metric for activity tracking[17], and a review by Gastinger et. al. concluded that devices such as RIP sensors increases the potential of ventilation as a measure of energy expenditure[18].

The objective of this thesis is to estimate the physical effort of a person exercising based in the person's breathing, or more specifically: Estimate the power output during exercise based on RIP signals. We choose power output as our target because of its direct relationship to the actual physical effort. As men-

[1]Product webpage: `https://www.concept2.com/bikeerg/concept2-bikeerg`

[2]Product webpage: `https://www.concept2.com/indoor-rowers`

[3]Product webpage: `https://www.technogym.com/int/products/home-wellness/cross-trainers.html`

[4]Product webpage: `https://www.concept2.com/skierg`

tioned previously, it is not always trivial to measure the actual power output during exercise for an arbitrary activity form. Some exercise forms, especially team sports and ball sports, are complex and are often not characterized by continuous movement of the body. The sport might involve many different types of movements, and also interruptions or periods of inactivity. In such cases it will be challenging to obtain meaningful measurements of power. Because of this, we want to focus our efforts on activities that involve continuous repetitive movements, where the power output is uninterrupted throughout the duration of the activity or workout. Cycling is such an activity, and is a form of movement that is commonly used in studies of physical activity[11][19][20][21]. Because of this, we have chosen to focus our efforts on data collected during cycling.

In the following section we will expand and specify what our goal is and which research questions we aim to answer in this thesis.

1.2 Problem statement

The idea that sparked the inspiration of this thesis was: Can we use breathing to predict physical effort? From this idea, we formulated a series of research questions (RQs):

- **RQ1:** How can we acquire the data necessary to estimate power output from breathing?
- **RQ2:** How can we build a predictive model to estimate power output from breathing?
- **RQ3:** What machine learning architectures are effective in estimating power output from breathing?
- **RQ4:** What features of breathing data are useful for estimating power output from breathing?
- **RQ5:** What is the effectiveness of estimating power output from breathing?
- **RQ6:** What are the limitations of estimating power output from breathing?

To summarize, we want to create a system for estimating power output during exercise based on measurements during breathing, and evaluate the performance and useability of such a system. Although breathing, and more specifically respiratory inductive plethysmopgrahy, is our main input variable for obtaining an estimation of power output, we also investigate whether the use of heart rate as an additional input variable can improve the performance of our predictive models.

1.3 Thesis structure

Background

The background material necessary to understand the contents of this thesis is presented in Chapter 2. We will explore the existing methods for tracking and monitoring of physical effort, and explain the mechanisms behind our chosen procedures for this thesis. Furthermore, we introduce the theory behind predictive models, and specifically the techniques of various types of neural networks. We also discuss related work and research.

Methods

In Chapter 3 we present the methodology followed in our project. We give an account of which data we acquired and how they were acquired. The machine learning methodology will be explained in detail, together with the procedure of model selection and tuning.

Results

The results are presented in Chapter 4, where we show and discuss the final architectures of our neural networks in addition to performance scores for the various models.

Conclusion and future work

Finally, we summarize the thesis and give our perspective on future works in Chapter 5.

Chapter 2

Background

This chapter presents the theoretical background of the methods we have used to produce our results. Section 2.1 is devoted to the topic of tracking and monitoring physical effort during exercise. We will review the state-of-the-art methods used by both non-athletes and in professional sports. In Section 2.2 we discuss what breathing is and how it can be measured. We describe methods for measuring and recording various aspects of breathing. Section 2.3 contains an account of how collected data can be used to create models for prediction. We give an outline of the concept of machine learning, and go into detail on how deep learning and neural networks can be used to learn a representation from a data set. In Section 2.4 we review related work and research.

2.1 Tracking and monitoring physical effort

Physical activity may be broadly defined as "all bodily actions produced by the contraction of skeletal muscle that increase energy expenditure above a basal level"[22]. This definition includes also static activity[22], since skeletal muscular contractions can be both concentric, eccentric and isometric[23]. Within the scope of this thesis, we are mainly concerned with the activity taking place during *exercise*, which can be viewed as a subset of physical activity. The characteristics of exercise may be defined as "planned, structured, and repetitive bodily movement" where the objective is to "improve or maintain physical fitness components"[24]. With these definitions at hand, we are ready to discuss what we mean by *physical effort*. Simply put, we are talking about how hard a person is working during physical activity, which may overlap with the term *exercise intensity*. However, many aspects of physical activity can be monitored in order to give an indication of the effort or intensity, and we need to clearly define the term to be able to make measurements of it. Jeukendrup and Diemen suggest that exercise intensity should be determined as “the amount of ATP that is hydrolysed and converted into mechanical energy each minute”[11], and that the best definition would be “the amount of energy expended per minute to perform a certain task”[11], measured in kJ/min. This definition is close to core of what we mean by physical effort, and it ties into how we define physical activity as an increase of energy expenditure. However, it is in general very challenging and expensive to measure the energy expenditure directly[25]. An alternative to direct measurements is to do measurements of variables that are related to the energy expenditure. The ideal measure of energy expenditure or physical effort should closely represent how hard a person is exercising, and at the same time be easy to track in real time, also in field tests. This means that a portable solution, which can be worn and carried on the body, would be preferable.

Existing methods of measuring physical effort typically fall into one of two categories: 1) Movement sensors and 2) physiologic sensors[9]. Examples of movement sensors are pedometers (step counters), accelerometers and GPS-systems. Physiologic sensors include heart rate monitors and temperature monitors, as well as measurements of blood glucose, oxygen saturation and lactate[9][26]. From such sensors, several measures of intensity or effort have been derived and used for tracking activity and sports: Heart rate, percentage of maximal heart rate, percentage of heart rate reserve, speed of movement, percentage of $VO2_{max}$, percentage of lactate threshold, percentage of ventilatory threshold and power output[11]. We will in the following subsections review a few of the most popular methods for using technological sensors to track and monitor physical effort during exercise.

2.1.1 Heart rate as a method for tracking physical effort

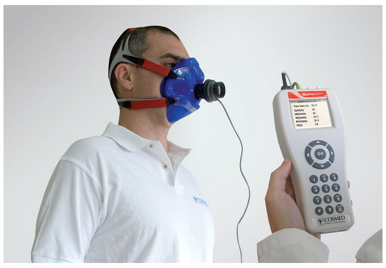
One of the most common type of sensors for activity tracking is heart rate monitors[29]. An example is shown in Figure 2.1a. This type of sensors measures the frequency of the heart’s contractions or beats, usually in the unit of beats per minute. The sensor itself usually consists of two main parts: A belt and an electronic chip. The belt has two ECG electrodes, which detect the electromagnetic impulses from the heart beat. The belt is placed just beneath the pectoral muscles. The chip can be integrated into the belt, or be a separate device which clips onto the belt. While some heart rate monitors stores the recorded data in the chip, it is more common that the chip wirelessly transmits the data to a receiver, often a wrist-worn sports watch.

The measuring of heart rate has been very common during exercise[29]. It can be used to prescribe and plan a certain training regimen, by assigning a target heart rate for the subject to aim at during exercise[30]. Furthermore, heart

(a) Heart rate monitor

(b) GPS-watch (speed measurements)

(c) Power meter[27]


(d) Spirometer[28]

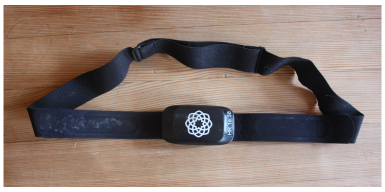

(e) RIP sensor

Figure 2.1: Sensors used for physical activity tracking.

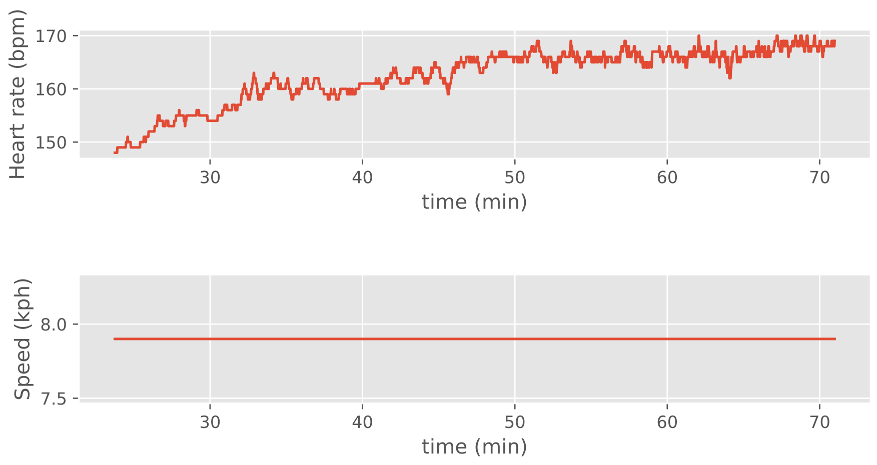


Figure 2.2: Speed and heart rate during a workout, where the heart rate exhibits cardiovascular drift. The workout was conducted on a treadmill with constant speed of 7.9 kph and an incline of 15%.

rate has been found to be moderately correlated to oxygen uptake (the rate of oxygen consumption per kilogram of body weight) $\dot{V}O_2$ ($mL \cdot kg^{-1} \cdot min^{-1}$) during exercise[19], and because of this, heart rate monitors are very popular to use as an indicator of the exercise intensity[9].

One of the most prominent problems with heart rate as a measure of physical effort is *cardiovascular drift*. This is a phenomenon where the heart rate increases with time during prolonged exercise where the workload remains constant[10]. This means that using the heart rate alone for estimating exercise effort will prove inaccurate for longer workouts, since the heart rate does not remain stable even if the workload does. An example of cardiovascular drift is displayed in Figure 2.2. The figure shows heart rate and speed from a selected portion of a running workout, performed at a constant speed of 7.9 kilometer per hour (kph). Even though the workload remains constant, the heart rate increases steadily from below 150 beats per minute (bpm) to a maximum of 170 bpm during the approximately 50 minutes of running.

2.1.2 Speed as a method for tracking physical effort

In activities where the main goal is to transport the body across a distance, it is possible to use speed as a measure of the physical effort. Examples of such activities include running, cycling, swimming, cross-country skiing and rowing. In some cases, measuring speed can be done by simply recording the time it takes to complete a certain distance. This is most common in swimming, which usually takes place in a fixed-size pool of 25 or 50 meters, and also in track and field running events, where race tracks are built after a standard of 400 meters of length. In other outdoor sports, the use of body-worn GPS-trackers makes it trivial to record speed during activity. GPS-trackers are often embedded in wrist-worn watches, as shown in Figure 2.1b.

One of the challenges with speed as a measure of exercise intensity appears when performing activities that take place on varying terrain or surfaces, or involving external factors such as wind resistance. An example is outdoor cycling. During uphills, the speed is low, even though the intensity might be high. In downhills the speed is much higher than in the uphills, even though the rider might not be working at all. During outdoor running, the speed can be very affected by whether the running surface is an asphalt road or a technical trail. Because of this, the speed itself will not represent the exercise intensity well.

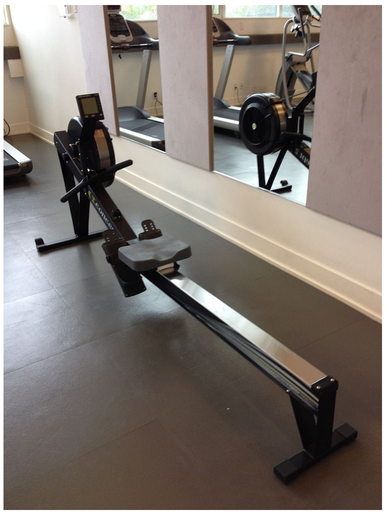
2.1.3 Power as a method for tracking physical effort

Power has been suggested as "the most direct indicator of exercise intensity"[11], and in some studies, power output is treated as the equivalent of exercise intensity[20]. Measuring power involves recording the actual effect of the work performed by the muscles. This is not trivial in many sports, because it is challenging to place sensors at the contact points between the body and the environment which the body acts on. An exception is cycling, where a power meter based on strain gauges can be placed inside the system of propulsion. The strain gauges are used to calculate the torque created by the cyclist. They can be placed in one of several places in the drivetrain of the bike, usually in the pedals or the crank[12]. An example is shown in Figure 2.1c. Power output can also easily be measured by indoor exercise machines, where an outside environment is replaced by mechanical resistance. These devices are often equipped with ergometers, which can measure the work performed by a user. When a person uses an exercise machine like this, they drive a flywheel with a certain resistive load, and the movement of the flywheel and the magnitude of the resistive load is used to calculate the power output. Examples of such machines include stationary bikes, rowing machines and elliptical trainers, which are shown in Figure 2.3.

2.2 Breathing

Breathing is the act where a person moves air in and out of the lungs. The process of inhalation and exhalation is essential for providing the body with the vital oxygen it needs to stay alive. The oxygen is used for breaking chemical bonds in the nutrients we digest, in order to create energy[32]. Breathing is

(a) Stationary bike


(b) Rowing machine[31]

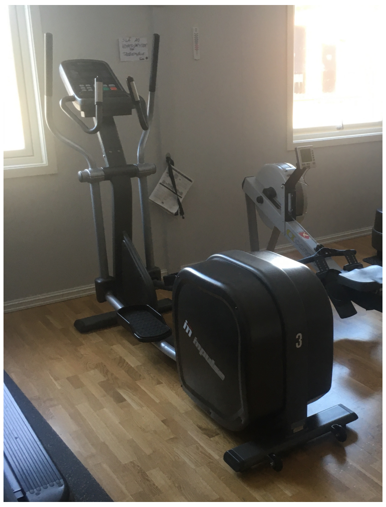

(c) Elliptical trainer

Figure 2.3: Examples of exercise machines.

intuitively linked to physical activity, as increased muscular work increases the need for oxygen, which in turn leads to heavier breathing. A strong correlation between energy expenditure and ventilation has been found, and suggests that measurements of breathing can be used for estimating physical effort[18]. In the following subsections we will review two methods of measuring breathing.

2.2.1 Spirometry

A common method for measuring pulmonary ventilation is exercise spirometry, which involves using a mouthpiece that measures the airflow passing through it[15][16]. An example is shown in Figure 2.1d. Various types of flowmeters are used to find the tidal volume, which is the volume of air moved in and out of the lungs during breathing. The measured signal during is normally flow or volume[15]. Spirometry is among other things used for diagnostics, monitoring of lung diseases and public health research[15]. Even though spirometry gives accurate recordings of the airflow, it introduces some challenges. The equipment used for measuring pulmonary airflow often increases the amount of dead space (which is the volume of the inhaled air that does not take part in the gas exchange of the lungs)[16][21]. During physical exercise, the additional dead space caused by wearing spirometry equipment has been found to increase the tidal volume and decrease the respiration rate[21]. Increased resistance for the ventilation has also been associated with decreased performance during exercise[33]. The use of spirometry and flowmeters also has the drawback of usually requiring a stationary setup.

2.2.2 Respiratory inductive plethysmography

A non-invasive way of measuring respiration is respiratory inductive plethysmography (RIP). This method involves measuring the expansion and contraction of the upper body as a result of breathing. This is done by using two individual straps or wires around the rib cage and abdomen, which record the variation in the cross-sections of these two areas. The RIP signal is found to have a strong relationship with the tidal volume measured by spirometry[34][35]. RIP has also been studied in the specific context of exercise, and Caretti et. al. found that the calibrated sum signal of dual RIP sensors "provided values statistically similar to flowmeter values at work rates below 180 W"[36]. The same study found that unintentional movement and slippage of the RIP bands could lead to greater variability in the results[36].

One example of a modern device capable of recording respiratory inductive plethysmography is a sensor called *Flow*[37], shown in Figure 2.1e. The Flow sensor is the one we have used in this thesis to record RIP signals. The setup and workings of this sensor is explained in Section 3.1.

2.3 Predictive models

Our aim for this thesis is to create a mapping between the RIP signals discussed in Section 2.2.2, and power output, discussed in Section 2.1.3. In order to build this mapping, we use predictive models, and in this section we describe the theory behind creating such models. By predictive modelling, we mean using collected data samples to create a model that can be used to predict or estimate unknown outcome from unseen data samples. The techniques we use to produce predictive models fall within a field of methods called *machine learning*. Machine learning is the process where a computer algorithm improves its performance automatically by adapting its parameters through experience. It is considered a subfield of artificial intelligence, and there exists a multitude of various algorithms that fall within the machine learning domain.

We will in this thesis deal exclusively with the branch of machine learning called *supervised learning*. In supervised learning, we provide a machine learning algorithm with two main components. The first of these is the input data, often called *predictors* or *features*. This is the data that will be used to make predictions or estimations. In our case, examples of predictors are heart rate and RIP signals, which we will use to estimate physical effort. The other component is the output data, often called *targets* or *labels*. These values are what we want to obtain when using the input data to make estimations, which in our case are the power output values. Several different supervised learning algorithms may be used for this type of *regression* problem, which involves estimating a numeric value of a continuous variable.

A potential challenge when working with machine learning is shortage of training data. The algorithms performance hinges in part on sufficient quantity of well-distributed data. A certain amount of examples is needed for the algorithm to adjust correctly, but it is also important that the training data contains the same distribution as the cases it will be tested for. Furthermore, if the training is of poor quality, for example containing noise and errors, it will be more challenging to uncover patterns in the data.

Several aspects need to be taken into account when choosing which algorithm to use for a specific problem. We need to consider what type of data we are studying, and what kind of structure it has. In this thesis we are dealing with time series data, which may contain temporal dependencies between multiple data samples and the target value. Because of this, we choose to focus on two types of neural networks that can handle data with local dependencies across multiple samples: Convolutional neural networks[38] and recurrent neural networks[39]. In addition, we use a simple fully-connected neural network for comparison.

In the following subsections we explain the basics of how neural networks function, and how they are used to learn representations from a data set. We look at three main types of neural networks: Dense neural networks (DNN), convolutional neural networks (CNN) and recurrent neural networks (RNN). DNNs get a more detailed account, as they lay the foundation for the other types of neural networks, while CNNs and RNNs are explained more concisely. We also discuss metrics that can be used to evaluate the models, and how the data is scaled before processed in the neural networks.

2.3.1 Basics of deep learning and neural networks

Artificial neural networks (ANN) is a type of machine learning technique that is inspired by neuroscience and the neural network of the biological brain. There exists several types of ANNs, and each is typically specialized to deal with a certain type or structure of input data. In order to explain the inner workings of ANNs, we will start by explaining how a fully-connected feedforward neural network works.

A fully-connected feedforward neural network is the most basic type of ANN, and sometimes referred to as a multilayer perceptron (MLP). The idea behind such a network is to make an approximation of a function f. The network is used to find a mapping between a target variable y and an input variable (or a set of variables) x:

$$y = f(x; \theta), \tag{2.1}$$

where the parameters θ are learned when training the network. The network consists of an input layer, one or several hidden layers, and an output layer. Each of the layers consists of a number of *nodes*[1] An illustration of an example is shown in Figure 2.4. The grey circles represent the input layer, the white circles are the hidden layers, and the black circle is the output. All nodes in one layer have a direct connection to all nodes in the next layer, which is why we label this type of network as *fully-connected.* The term *feedforward* means that the data is passed in only one direction through the network. In some type of the networks, we have a feedback loop where output is passed back into the network, in which case we call them *recurrent* neural networks.

[1]Because of the original inspiration from biological neural networks, these nodes are sometimes referred to as *neurons*, but we will stick to the term *node* in this thesis.

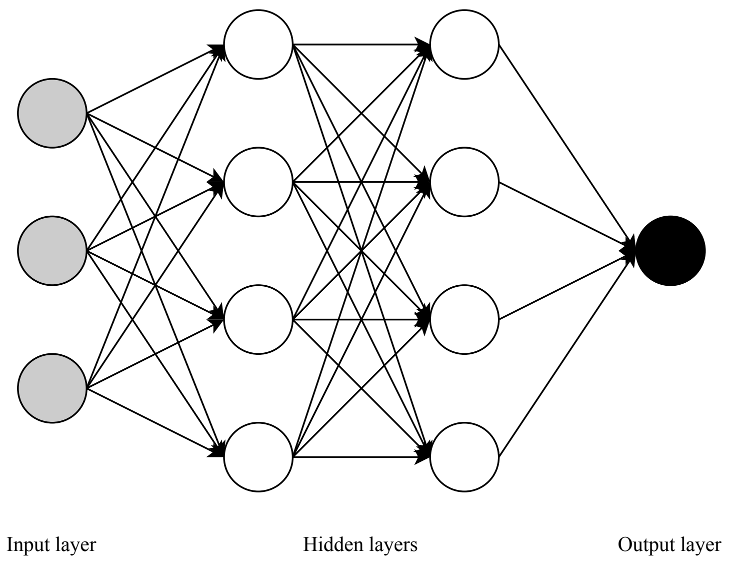


Figure 2.4: Schematic neural network. The circles represent nodes, and the colors indicate what layer they are a part of: Grey means input layer, white means hidden layer and black means output layer. The arrows show that all nodes of one layer is connected to each of the nodes in the next layer; i. e. we have only fully connected layers.

The number of layers in the network are sometimes referred to as the depth of the model, and this is related to the term *deep learning.* Although there is no strict definition of this term, we often use it when describing neural networks with multiple hidden layers, for example the network shown in Figure 2.4.

The input layer represents the data samples' entry point to the network. The number of nodes in the input layer corresponds to the number of variables per data sample. The last layer is called the output layer, and this is where we obtain the predicted values after the network has processed the input data. The layers between the input and output layer are called hidden layers. We will in the following subsection discuss how data is processed in a simple feed-forward, fully-connected neural network.

Feed-forward pass

The process of passing information from the input layer to the output layer is called a *feed-forward pass.* We will first take a look at how the data is processed from the first layer to the next. Let the first layer consist of n nodes, each corresponding to a single input variable. Each node i in the first layer, passes an input x_i to all nodes in the subsequent layer. The connection between a node in the first layer (subscript i) and a node in the second layer (subscript j)

also has a weight w_i and a bias b_i. The weights and biases are often called the *parameters* of the model. During a feed-forward pass, each input from the first layer is multiplied by the weight of this connection with the bias added. The result is summed for all the inputs x_i:

$$z_j = \sum_{i=0}^{n-1} w_i x_i + b_i \tag{2.2}$$

The result z_i is then evaluated by an activation function a:

$$a(z_j) = a_j, \tag{2.3}$$

where a_j becomes the output value from node j in the second layer. The value a_j is then used as input value to the next layer in the network.

When dealing with multiple input variables and multiple data samples, the input data are formatted as an input matrix X, where the columns represent the various predictors, and each row represents one data sample. Similarly, we assemble the weights into a matrix W and the biases as a vector b. The output a_l of a layer l the becomes

$$a_l = a\left(a_{l-1} W_l + b_l\right) \tag{2.4}$$

Here we assume that all layers have the same activation function a.

The initial values of the weights W between two layers are drawn from a uniform distribution dependent on the number of nodes in the two connected layers:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{2.5}$$

This initialization was suggested by Glorot & Bengio[40], and is called *normalized initialization* or simply *Glorot initialization.* The quantities n_j and n_{j+1} is the number of nodes in the two subsequent layers that are connected. The initial values of the biases are often set to a small non-zero value in order to ensure activation, but can also be set to zero initially.

Activation functions

The activation function has a great effect on the performance of the model, and many different options may be chosen. The choice depends on what data we want to process, what network architecture we are using, how the data is scaled and whether we are doing classification or regression. The activation function does not necessarily have to be the same for all neurons or all layers in the network.

Some of the common choices for activation functions include the sigmoid function, the tanh-function and the Rectified Linear Unit (ReLU). The latter is often a default choice for neural networks, since they are easy to optimize and does not saturate for large values[41]. ReLU is defined as:

$$\text{ReLU} = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.6}$$

This is the function we have used for all layers in the networks used in this thesis.

Backpropagation and gradient descent

When the feed-forward pass is done, the output layer contains the predicted values based on the input data. The next step in the process is to compare these predicted values to the true targets of the training data samples. Based on this comparison, the weights and biases will be adjusted in order to improve the performance. As mentioned previously, the weights and biases are randomly initialized, which means that the first feed-forward passes most likely will give a result with large errors. Each of the feed-forward passes is called an *epoch*, and the number of epochs is adjusted according to how much training the network need in order to give adequate results. The adjustment process of the weights and biases is called *backpropagation*, which was introduced in 1985 by Rumelhart et. al.[42]. The backpropagation algorithm is a method for automatically compute the error gradient of the network, and these gradients are used to adjust the weights and biases.

The output values of the network $\hat{y}$ are compared with the true targets y by using a *cost function* $\mathcal{C}$. This cost function is the formula we use to measure how well the network performs, and our aim is to minimize the overall cost for all the samples in the training set. When the gradients of the parameters are computed, the parameters are updated accordingly. A learning rate η specifies how large the update of weights and biases should be for each epoch. This type of technique is called *gradient descent*. Usually the weights and biases are updated based on only a subset of the training data, in contrast to computing the gradients for the complete data set at once. This method is called *stochastic gradient descent*[41]. In practice, a *minibatch* of samples from the training data is randomly selected for each epoch, and used for updating the weights and biases. This has an obvious advantage because it is much computationally cheaper to perform one epoch. It also has a regularizing effect by bringing stochastisity into the learning process[41]. The learning rate largely affects the performance of the network, and one way to further optimize the gradient descent is to use an adaptive learning rate. Kingma and Ba have suggested an optimization technique called *Adam*[43], which adapts the learning rate individually per parameter instead of using a fixed learning rate for the whole learning process.

Dropout layers

One technique for regularizing the training of the neural network is called dropout layers[41]. In the dropout layer, random input units are set to zero. The frequency of nodes set to zero is specified as a control parameter of the dropout layer. This method helps combat overfitting[44].

2.3.2 Convolutional neural networks

Convolutional neural networks (CNNs)[38] are another type of feed-forward neural network that consist of so-called *convolutional layers*. In contrast to fully-connected neural networks, the nodes of a layer in a CNN is only connected to a small number of nodes in the preceding layer. These types of networks are suited for processing data samples in a grid pattern, such as an image or a time-series with a fixed sequence size. A convolutional neural network often consists of three categories of layers: Convolutional layers, pooling layers, and fully connected layers. The latter has already been explained above, and the former two will be explained below.

Convolutional layers

A convolutional layer is made up by a set of *filters* (sometimes referred to as *kernels*). These filters are repeatedly applied across values of the input, and results in another matrix, which is often called a *feature map*. The values of the feature map are produced by applying a mathematical operation called *convolution* on the filter and the values of the input matrix that the filter cover.

Let us say our input matrix is called X and our filter K. Our feature map S can be calculated by[41]:

$$S[i,j] = X * K = \sum_m \sum_n X[i-m, j-n]K(m,n), \tag{2.7}$$

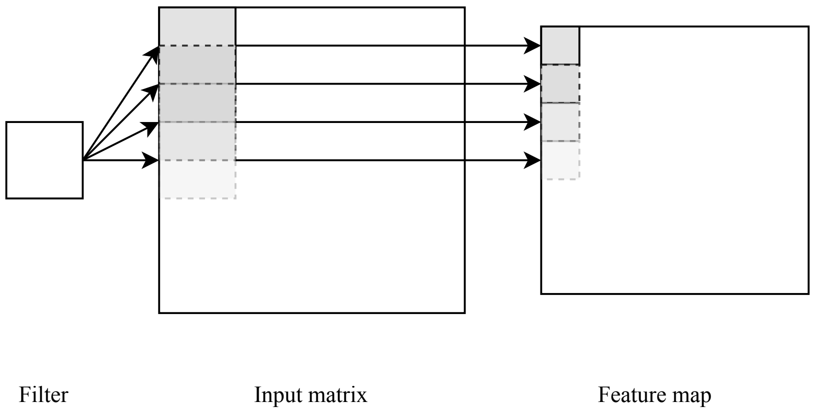


Figure 2.5: Diagram showing the principle of how filters are applied to an input matrix in a convolutional layer.

where i and j denotes the indices of our input matrix X, and m and n are the indices of the filter matrix K. The symbol $*$ is often used to indicate convolution. This formula is used to compute the complete feature map for a given filter, and a simplified diagram of the process is shown in Figure 2.5. In the case of CNN, it is the values in each filter which are the learnable parameters, and the learning process adapts the values of the filters in order to optimize the performance of the model.

The *receptive field* of a node is a term that describes the section of the input which affects a given node in the network. The filter size will affect this, and a larger filter size will result in a larger receptive field. Furthermore, values of the feature map are not necessarily computed for each location in the input matrix. A hyperparameter called *stride* controls how many steps the filter should move between each convolution. A greater stride will lead to fewer values in the feature map, i.e. a smaller feature map. Usually one applies multiple filters in each convolutional layer, which means that each layer produces multiple feature maps.

A key element to the convolutional layers is that all filters are used in multiple locations across the input, a property called *parameter sharing*[41]. While a fully connected neural network has separate parameters for each of the connections between all nodes, CNNs can detect a feature in any location in the input by using only one filter.

A typical example of a complete CNN architecture can be seen in Figure 2.6. The pooling layers will be explained in Section 2.3.2. The role of the fully connected layers is to map features extracted through convolutional and pooling layers to a given output.

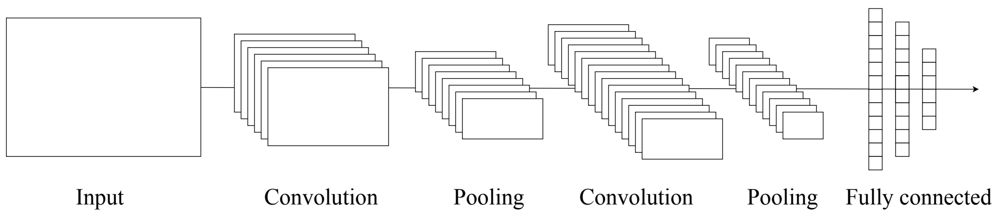


Figure 2.6: An example of a typical CNN architecture.

1D convolutional layers

The traditional way of using convolutional layers, as described in the previous section, can be adapted to work for time series data. In a normal convolutional layer, the filter is applied along two axis of the input data, and is therefore sometimes referred to as two-dimensional convolutional layers. This enables the network to learn spatial dependencies across both axis of the input matrix. In our case, where we deal with time series data, we are only interested in the dependencies along one axis, namely the temporal axis. Figure 2.7 shows a diagram of how a CNN with convolutional 1D layers looks like. In the example presented in Figure 2.7, there are three input features, and the filter covers three time steps. The convolutional filters produce 1D feature maps, which end up being processed through fully connected layers, and in the end give a single value as output. This figure is a close representation of the CNNs used to produce models in this thesis, with a singular output value.

Pooling

A common practice is to insert so-called pooling layers in-between convolutional layers. The pooling layers is meant to diminish the spatial size of the data. This is advantageous both because it reduces the amount of parameters needed to be processed, and it increases the invariance of the layer, in regards to minor translations in the input data[41]. The translational invariance means that the output from the pooling layers will not be affected by small changes in the input data. The pooling layers will help prevent overfitting to the training data.

To demonstrate how pooling works, take a look at Figure 2.8. In this case we use a filter of size of 2×2, and a stride of 2. The input matrix is on the left side of the figure, and has a size of 4×4. The filter is placed on four locations on the input matrix, shown in four different colors in the figure. There are several types of pooling operations, and in this case we use *max pooling*, which means that the output becomes the maximum value of all the values covered by the filter in the input matrix. The result of the pooling operation can be seen in the right matrix of the figure. An alternative to max pooling is average pooling, in

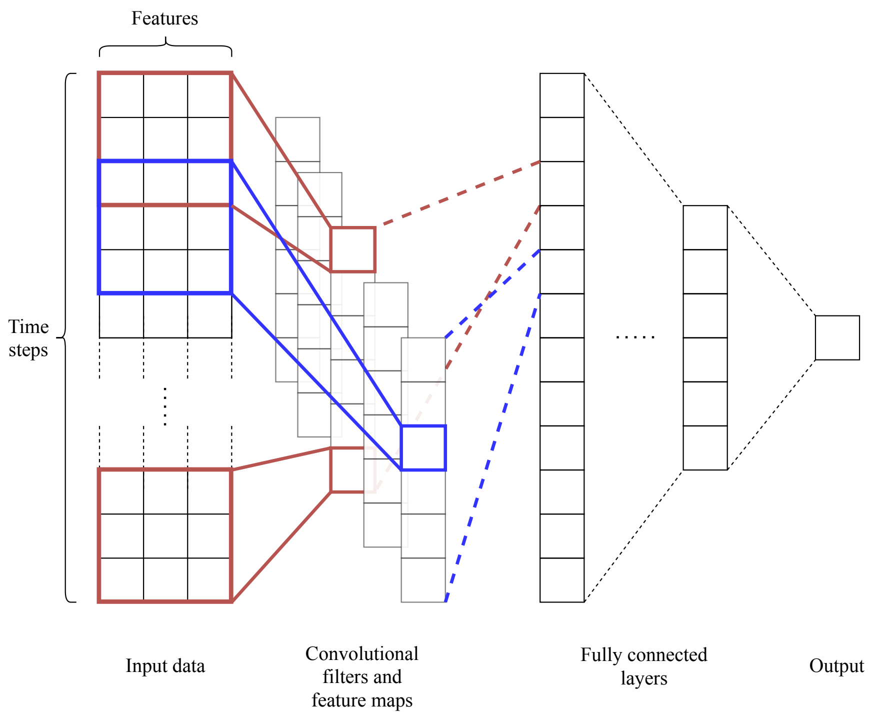


Figure 2.7: A convolutional network with 1D convolutional layers. The network also includes fully connected layers after the convolutional layer, and a single output value is returned from the network.

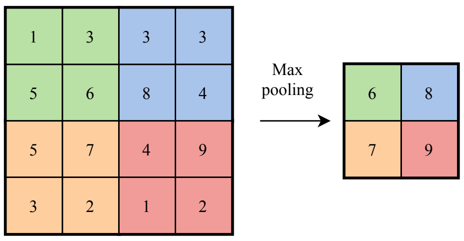


Figure 2.8: Max pooling with a 2×2 filter and a stride of 2.

which case the output becomes the average of all values covered by the filter.

In contrast to normal layers in a neural network, a pooling layer does not consist of any parameters, but is solely defined by the process it performs on the input data.

2.3.3 Recurrent neural networks

A third type of neural networks is recurrent neural networks (RNNs)[39]. They are designed to work with sequential data. The special characteristic of an RNN is that it processes the input data sequentially, and that the output of a previous step is used as input to the next step. This enables the network to "remember" information from previous steps, which can be used to identify temporally dependent features in an input sequence. An RNN consists of a series of *hidden units*, which is analogous to the nodes or neurons of other types of neural networks.

Let us define an input sequence $x_1, x_2, ...x_t, ..., x_\tau$, where t indicates the time step. The internal state, often called the hidden state, of the RNN is h_t. A general equation for computing a state in an RNN is[41]:

$$h_t = f(h_{t-1}, x_t), \tag{2.8}$$

where h_{t-1} represents the old state, h_t represents the new state, x_t is our input of the current time step, and f is a transition function. The input to a hidden unit is also multiplied by weights W and has added bias to it, as in a fully connected neural network. The transition function serves as an activation function, and are often chosen to be the function $\tanh(x)$ or the *sigmoid* function, given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.9}$$

If we use the sigmoid function, the computation of the hidden state h_t in a plain RNN then becomes

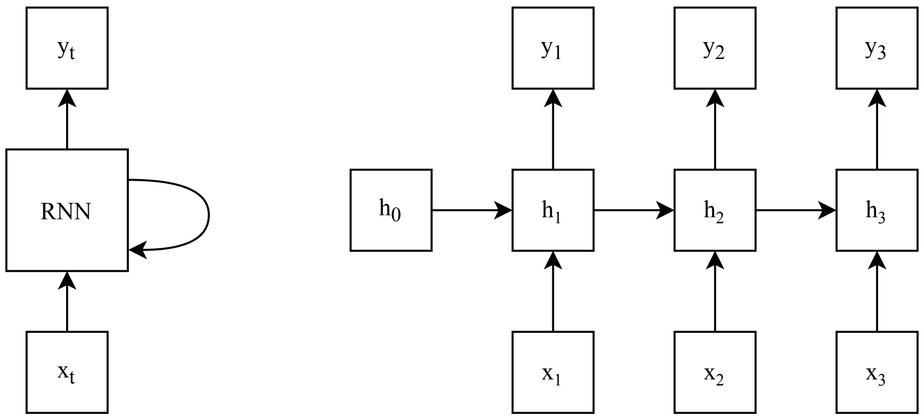


Figure 2.9: Conceptual diagram of an RNN. Left: Compact representation. Right: Unrolled representation.

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h), \tag{2.10}$$

where W_h is the weight for the hidden state, W_x is the weight for the input and b_h is the bias.

A simplified diagram of the forward pass through a recurrent neural network is shown in Figure 2.9. In this example, all hidden units produce an output y_t, but in cases where only a single output value is needed, the output is computed only for the last unit. The parameters of the model are trained using gradient descent.

Plain RNNs suffer from what is called the vanishing gradient problem, which was discovered in the early 1990s[45][46][47]. When training an RNN on sequences, the unrolled network can become very deep, similarly to a DNN with many layers. The problem arises when the gradients become increasingly small when propagating the parameter updates back through the network, and the updates become so small that, in effect, the network stops learning. One possible solution for dealing with the vanishing gradient problem is the Long short-term memory network, which will be discussed in the following section.

Long short-term memory (LSTM) network

A special type of RNN is the Long short-term memory (LSTM) network, which was proposed in 1997 by Hochreiter and Schmidhuber[48]. A hidden unit in an LSTM network is often called *cell.* This cell has four additional components compared to a hidden unit in a plain RNN: A *cell state c*, an *input gate*, an *output gate*, and a *forget gate.* The architecture of an LSTM cell is presented in

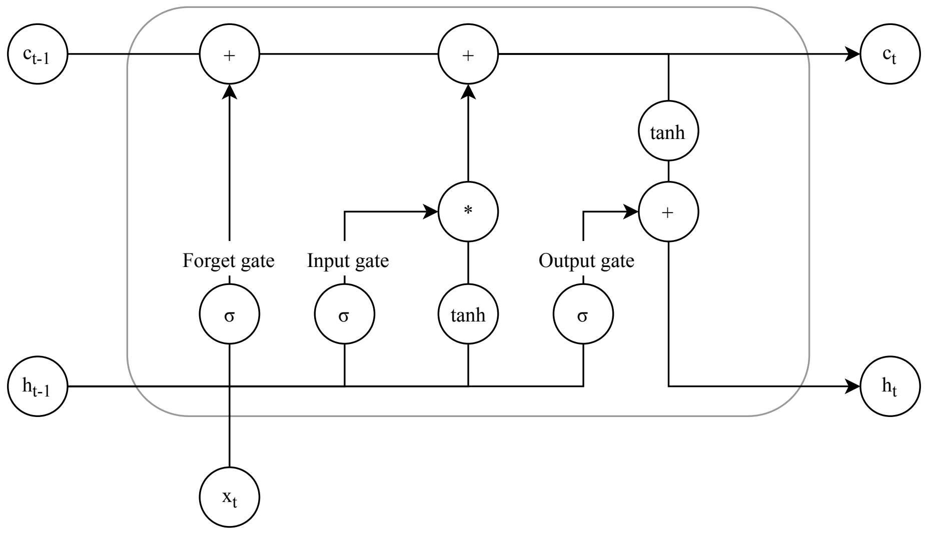


Figure 2.10: LSTM cell

Figure 2.10. The cell state enables the network to retain information over longer spans. The three gates are essentially learnable weights. The input gate controls which information should be added to the cell state. The forget gate controls which information discarded from the cell state. The output gate controls the output from the cell. Put simply, the gates of an LSTM unit enables the network to learn which information it should remember and what it can discard. The states of the LSTM cell are computed in a similar manner to Equation 2.10:

$$f_t = \sigma(W_f h_{t-1} + W_x x_t + b_f), \tag{2.11}$$

$$i_t = \sigma(W_i h_{t-1} + W_x x_t + b_i), \tag{2.12}$$

$$o_t = \sigma(W_o h_{t-1} + W_x x_t + b_o), \tag{2.13}$$

where f_t, i_t and o_t are the states of the forget, input and output gates respectively. The LSTM cell structure enables the gradients to be preserved for a longer sequences, and mitigates the problem of vanishing gradients[41].

2.3.4 Model selection and evaluation

In order to evaluate a trained model's performance, we test it on a set of data samples that was not used in training. Thus we divide the data set into a *training data set* and a *test data set*. After training is complete, the input data of the test set is passed through the trained model, and the output is compared against the true values. The true values are often called the *ground truth*. The comparison of the predicted and the true output is done according to the metrics described below.

Many learning algorithms involve the use of *hyperparameters*, which control how the learning process is performed. This could for example be the learning rate, number of nodes in each layer, dropout rate, batch size and number of epochs. These hyperparameters are not a part of the parameters that are adapted in the training process, but need to be tuned to obtain optimal model performance.

Metrics

In order to assess the quality of the predictive models, we need to define metrics that can quantify how well the models perform. For the models in this thesis, we use three different metrics to evaluate the performance of the model. The first metric is one of the most common evaluation criteria for regression models, the mean squared error (MSE)[41]:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2. \tag{2.14}$$

In this equation, $\hat{y}$ represents the predicted output values and y is the actual output values. The index i runs from 1 and up to n, which is the number of samples. The MSE will decrease as the difference between predictions and the ground truth becomes smaller, and we desire the resulting number to be as small as possible. Squaring the difference between predicted and true values means that we always will get a positive number. It also has the effect that large differences in y_i and $\hat{y}_i$ will have more significance than smaller differences.

The second evaluation metric we use is the *coefficient of determination*, often denoted R^2. This score is calculated by the following equation[49]:

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{2.15}$$

where $\bar{y}$ is

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \tag{2.16}$$

i.e. the mean of the true values. Explained in words, the R^2 score is the residual deviance divided by the total deviance, subtracted from 1. It gives us an indication of the ratio between the total variance explained by the model, and the total variance. We desire an R^2 score as close to 1 as possible.

The third metric is the *mean absolute percentage error* (MAPE), defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{2.17}$$

The resulting fraction can be multiplied by 100 to give an average percentage of the error in the model's predictions. We use this metric to give an intuitive measure of how well the models perform.

2.3.5 Scaling

When passing multiple input variables to a neural network, there is often a need for scaling the variables first. This is because the various input variables may have different orders of magnitude, and since the inputs will be combined through the connections of the network, certain variables will have a greater effect on the calculations of the gradients. Therefore it is a common heuristic to scale or transform the input data[50]. There are several methods of scaling, of which two were explored in this project.

Min-max scaling entails that the values are transformed such that the range is from 0 to 1. To do this we use the maximum and minimum value of a variable, denoted x_{max} and x_{min}, and the vector of scaled values x' can be computed from the original values x using the following equation:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2.18}$$

Another common method is **standardization**, where the mean μ is subtracted from all values, so the new mean becomes zero, and then the values are divided by the standard deviation (std), giving a variance of 1. The standardization equation is as follows:

$$x' = \frac{x - \mu}{x_{\text{max}} - \text{std}} \tag{2.19}$$

2.4 Relevant work

In Section 2.1, we reviewed some of the most common methods of measuring physical effort during exercise. Cardiovascular drift is a weakness of heart rate as an indicator of physical effort, and speed measurements suffers from the inability to account for external factors such as terrain. While power output measurement gives us a direct metric for physical effort, it is limited to activity forms where one uses specialized equipment with embedded power meters or ergometers. Breathing, as discussed in Section 2.2, is linked to physical effort, and the advent of portable RIP sensors enables us to easily measure breathing in a wide range of activities. As seen in section 2.3, deep learning has proven useful for building predictive models that can process time series data, and this is why we aim to explore the use of neural networks to predict physical effort from breathing. In this section, we review some of the previous research done regarding the relationship between breathing-related variables and physical effort during exercise.

Arts and Kuipers studied in 1994 the relationship between power output, oxygen uptake and heart rate[51]. They argue that even though maximal oxygen uptake ($\dot{V}O_2$max) is considered an important indicator of physical performance, it still has some limitations. One of these is that the performance of an athlete is not only governed by the maximal oxygen uptake, and further training may improve the performance in a specific sport despite no improvement in $\dot{V}O_2$max. Furthermore, athletes of similar performance can still have significantly different $\dot{V}O_2$max values. Because of this, maximal power output might be a preferable measure, since it is more true to the actual work the athlete is performing. During exercise, athletes do not work at maximum capacity, and Arts and Kuipers studied the relationship between power output and oxygen uptake at submaximal intensities, expressed as percentages of their respective maximum. They gathered data from 53 male athletes, who performed an incremental maximal test on an ergometer bike. The relationship between %Wmax (percentage of maximal workload in watts) and %$\dot{V}O_2$max (percentage of maximal oxygen uptake, measured in liters per minute) was found to be linear, with an correlation coefficient of $r = 0.98$. The relationship between %Wmax and %HRmax (percentage of maximal heart rate, measured in beats per minute) was also found to be linear, with a correlation coefficient of $r = 0.97$. This research relates to this thesis, since it shows that oxygen uptake is strongly related to power output during exercise, and supports the hypothesis that power output can be predicted from variables connected to breathing.

Muniz-Pumares et. al. studied the relationship between oxygen uptake ($\dot{V}O_2$) and power output during cycling. A total of 9 male cyclists performed two workouts on a cycle ergometer, while oxygen uptake ($\dot{V}O_2$) and power output were measured. One workout was a incremental test at submaximal intensities, while the other was a maximal test to exhaustion. A strong linear correlation ($r = 0.953$) was found between $\dot{V}O_2$ and power output.

Gastinger et. al. has done a thorough review of the case for using ventilation as a representative measure of energy expenditure[18]. The study states that while there exists a range of portable devices that estimate energy expenditure, such as pedometers, accelerometers and heart rate monitors, these show significant variability when it comes to accuracy of their estimations. The review states that sensors based on respiratory inductive plethysmography estimate tidal volume and ventilation reasonably well, which in turn can provide a portable, non-invasive way of estimating energy expenditure.

Indirect estimation of power output during exercise has been previously attempted for cycling. Costa et. al. studied in 2017 the validity of a device called PowerCal[52], which estimates the power output during cycling based on heart rate. Even though heart rate monitors give information about the exercise intensity, they can also be affected by other external factors, such as temperature and dehydration. The PowerCal device uses only heart rate to calculate the power output, based on a certain algorithm. The subjects were 21 well-trained male cyclists, who were put through an incremental test exercise and three time-trial workouts, where the actual power output was measured by a cycle ergometer, at the same time as the PowerCal device was used to estimate the power output based solely on recordings from a heart rate monitor. During the time trials, the mean power output calculated by the PowerCal was 242 ± 28 W, which was significantly lower than the power output measured by the ergometer, 282 ± 27 W. The mean power output over each kilometer was from 5.8% to 23.4% larger on the ergometer compared to the PowerCal. The researchers concluded that the power estimation based on heart rate calculated by the PowerCal device was not reliable and under-estimated the actual power output.

Hilmkil et. al. has previously applied deep learning to data from cyclists[53]. They used an LSTM neural network to train a model for predicting heart rate response of a cyclist during a workout. A wide range of input variables was used: Time (seconds), speed (km/h), distance (km), power (Watt), cadence (pedal strokes/minute), power/weight (Watt/kg), altitude (meters) and heart rate 30 seconds prior to the current time steps (beats per minute). A large data set of 7541 training sessions from 15 male professional cyclists was used for this study, where 5179 sessions from 10 of the cyclists where used for training. The study concluded that the heart rate predicted by the trained model was very close to the true heart rate, and that machine learning algorithms, especially deep learning algorithms, are promising methods for applications on sports performance data. Pfeiffer and Hohmann have also shown that neural networks can be succesfully applied on training data[54].

We have previously investigated how airflow can be estimated from RIP signals, in a project called DeepVentilation[55]. In that project, we applied deep learning to RIP signals in order to predict minute ventilation, which is how many liters of air a person moves in one minute. The work of DeepVentilation contributes to the foundation of this thesis, since we have been able to reuse some of the source code for processing the raw RIP signals. Additionally, we have made use of similar neural network architectures for building our predictive models. To the best of our knowledge, no research has attempted to estimate the power output from respiratory inductive plethysmography. That is the contribution of this thesis.

Chapter 3

Methods

The following chapter describes the experimental setup and procedures followed in this thesis, with the goal of estimating power output from RIP signals. In Section 3.1 we present the methods used for data acquisition, while in Section 3.2 we discuss how the preprocessing was done. Section 3.3 contains details on how we built our predictive models. We also give some details of the implementation and supporting software we used in Section 3.4. An overview of the main steps of our methodology is shown in Figure 3.1.

3.1 Data acquisition

The data set is based on measurements recorded while a subject was in one of two states: Either in activity or at rest. About 90% of the data samples were collected while the subject was in activity, and more specifically performing a cycling workout on a stationary bike. Cycling was chosen as the activity form for several reasons. Firstly, it is easy to measure accurate power output, because sensors can be placed inside the system on which the subject is exerting forces. In this case, a stationary bike from Concept2 was used, which has an built-in power meter. Further more, there are few or none external conditions that may affect the experiments while cycling indoors on a stationary bike.

Four different variables were recorded during the data collection:

- RIP from rib cage, unit: millivolt (mV). Sampled at 10 Hz.
- RIP from abdomen, unit: millivolt (mV). Sampled at 10 Hz.
- Heart rate, unit: beats per minute (bpm). Sampled at 1 Hz.

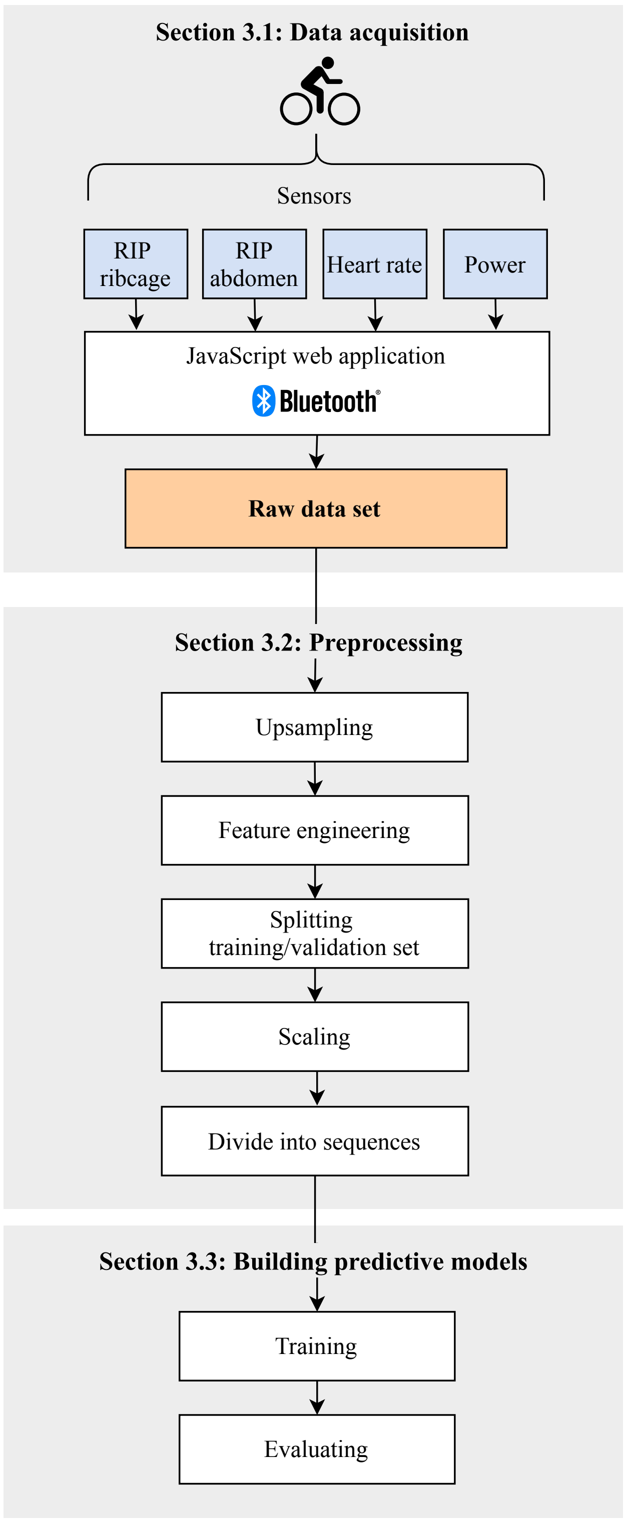

Section 3.1: Data acquisition
Sensors
RIP ribcage
RIP abdomen
Heart rate
Power
JavaScript web application
Bluetooth
Raw data set
Section 3.2: Preprocessing
Upsampling
Feature engineering
Splitting training/validation set
Scaling
Divide into sequences
Section 3.3: Building predictive models
Training
Evaluating

Figure 3.1: Overview of the complete data flow, from data acquisition to finished predictive model.

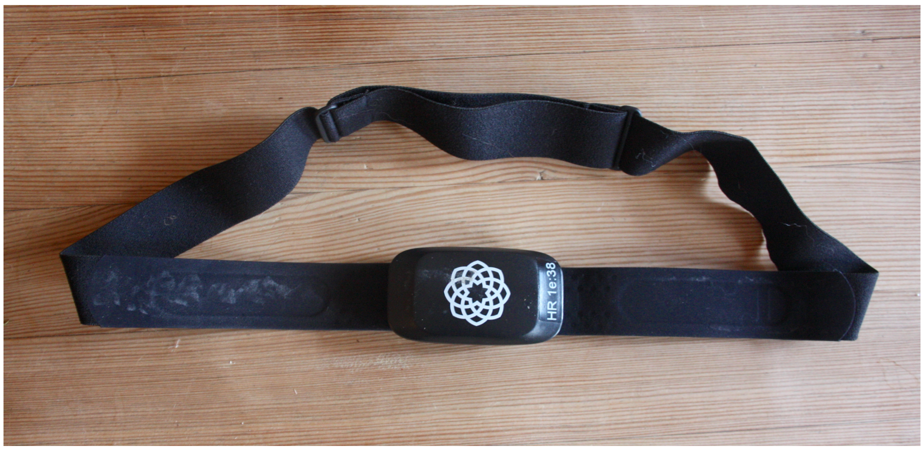


Figure 3.2: The FLOW sensor from SweetZpot.

- Power output, unit: Watt (W). Sampled at 1 Hz.

The three first variables are the basis for the predictors or input data, and the last variable is used as the target value.

The subject of the data acquisition was a highly active adult male of age 25. At the time of data collection the subject was an athlete competing in long-distance running, and had experience from competitive road cycling. The data collection sessions took place over a period of 7 months, between May 2020 and January 2021.

RIP sensors

The RIP data were collected using a product called Flow from the company SweetZpot. The sensor weighs 27 grams and measures 77x43x17mm. An image of the sensor is shown in Figure 3.2. It is powered by a Lithium battery of 3V. This sensor is fastened to a flexible belt that is worn around the chest or abdomen, as shown in Figure 3.3. Movements of the chest or abdomen will give increased strain on the belt, and this force is measured by a strain-gauge in the sensor itself. The strain-gauge is made up by a Wheatstone bridge, where variation in force will result in variation in resistance. The voltage over this resistance is the output of the sensor, and the unit of the output is millivolt (mV). The sensor samples the voltage at 10 Hz, and broadcasts the recorded values using Bluetooth Low Energy (BLE). During data collection the subject wore two of these sensors. One was placed such that the belt was lying directly below the pectoral muscles, while the other was placed over the navel, with the belt around the waist.

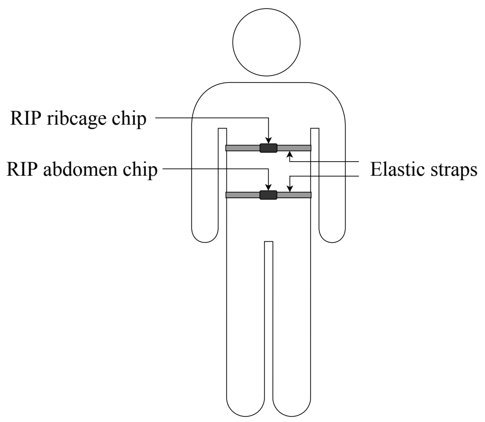


Figure 3.3: Placement of RIP *Flow* sensors.

Heart rate monitor

The heart rate data was collected using a Wahoo Tickr heart rate monitor, shown in Figure 3.4. This sensor also broadcasts its recorded values over Bluetooth. The Wahoo Tickr heart rate sensor weighs 17 grams and has dimensions of 65x37x10 mm. It is fastened to a flexible chest strap, which the subject wears around the chest below the pectoral muscles, just below the upper RIP sensor described above. The strap has two electrode pads which need to be moistured before use in order to register the electric signals from the heart beats. When the sensor itself is fastened to the belt, the heart rate can be transmitted over Bluetooth Low Energy (BLE). The SweetZpot Flow sensors have the option of recording both RIP data and heart rate, but when connecting to both Bluetooth services on a Flow sensor at the same time, we experienced that the two signals sometimes created a delay in delivering the data, which made the timestamps inaccurate. Because of this, we chose to use a separate sensor to record the heart rate.

Stationary bike and power meter

Power values were recorded directly from the stationary bike used in the experiments. The bike is called BikeErg, produced by Concept2, and it broadcasts all recorded values over Bluetooth. An image of the bike is shown in Figure 3.5. The pedals of the bike drives a chain connected to a flywheel. The power measured by the BikeErg is based on the work done by the flywheel, which will be different from the work done on the cranks of the bike because of drive transmission power loss. A study performed by Boyas et. al. has shown that the Concept2 ergometers underreport values by approximately 25 W[56]. However, for our purposes it is more important that the power output values are precise,

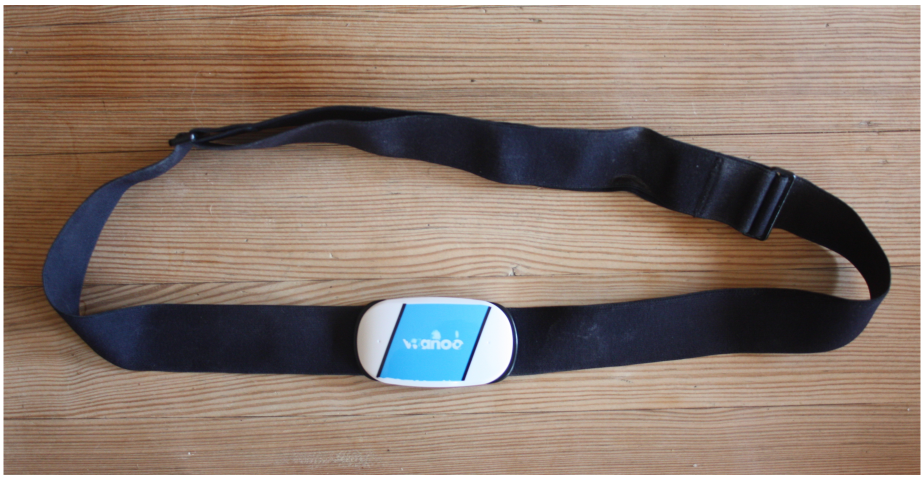


Figure 3.4: The Wahoo Tickr heart rate monitor.

by which we mean that any given input power exerted by the subject on the ergometer always gives the same power values calculated by the ergometer. A slight underreporting of power values is therefore not an issue when building predictive models for power output, since the relationships learned by a model are valid independent of a shift in the target values.

The experimental setup for the data collection is shown in Figure 3.6. A picture of a subject positioned on the bike and wearing all sensors can be seen in Figure 3.6.

Data collection during workouts

The workouts performed during data collection can be divided into three categories: Steady-state effort, high-intensity intervals and ramp structure. Table 3.1 contains the descriptions of the three different categories, and includes an example workout for each of them.

The main idea behind these workouts was to imitate a typical workout routine with a mixture of intensities and workout structures, in order to make a robust data set with a wide distribution of power values. One of the aims of this project was to create a model that can accurately estimate power output for arbitrary workout intensities, and that is why a mixture of workouts was chosen instead of some specific protocol.

Figure 3.5: The Concept2 BikeErg stationary bike.

Table 3.1: The three main categories of workouts performed during data collection.

Workout category	Description	Example
Steady-state effort	Cycling at the same intensity during the whole workout.	45 minutes easy cycling at 180 W
High-intensity intervals	Alternating between working at a high and low intensity for a fixed time or distance, and repeating the alternation a certain number of times.	4 × 4 minute on 350 W, with a break of 2 minutes
Ramp structure	Starting out at a low intensity, and then incrementing the intensity step-wise.	Starting at a power output of 100 W, increase by 100 W every 2. minute up to 500 W, and then decrease by 100 W every 2. minute, down to 100 W (pyramid workout).

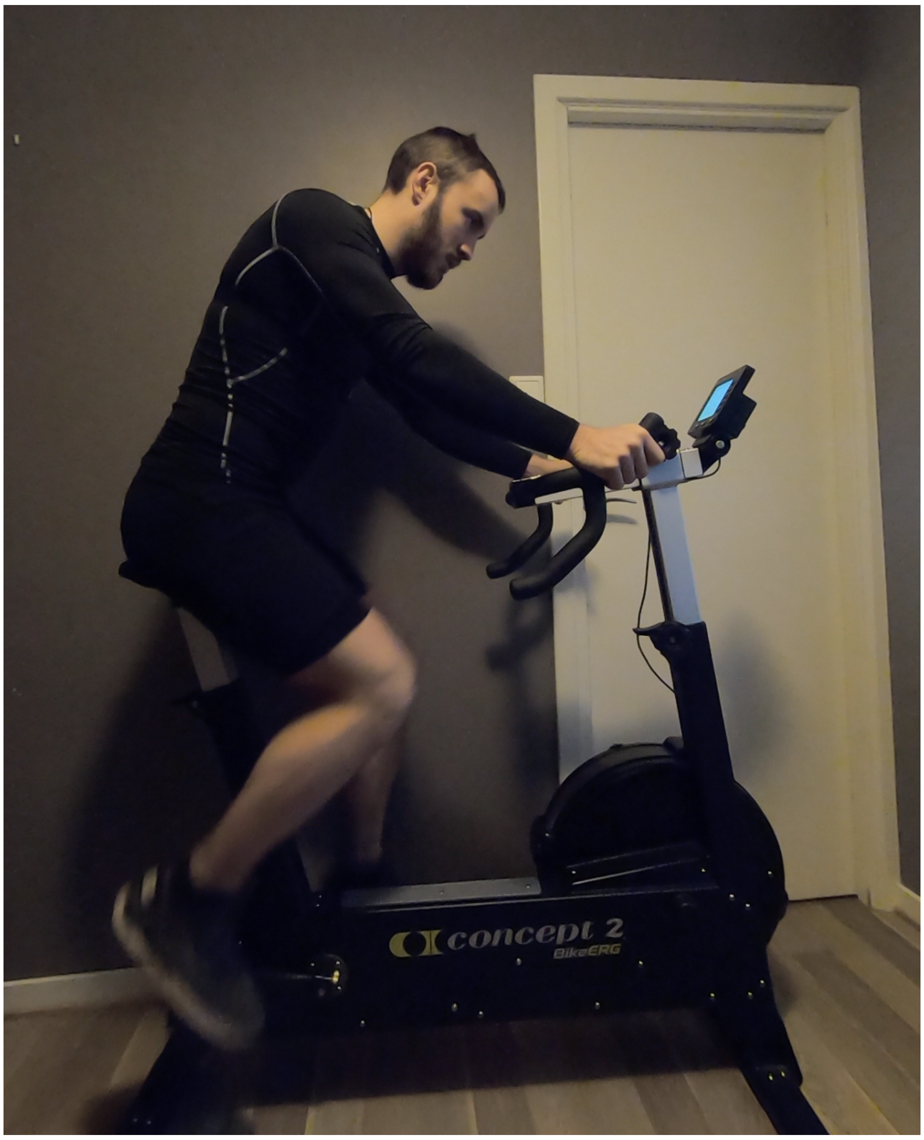


Figure 3.6: Experimental setup of the data collection, where the subject performs a workout on a stationary bike. The subject is wearing a heart rate monitor and two RIP sensors (not visible in the image).

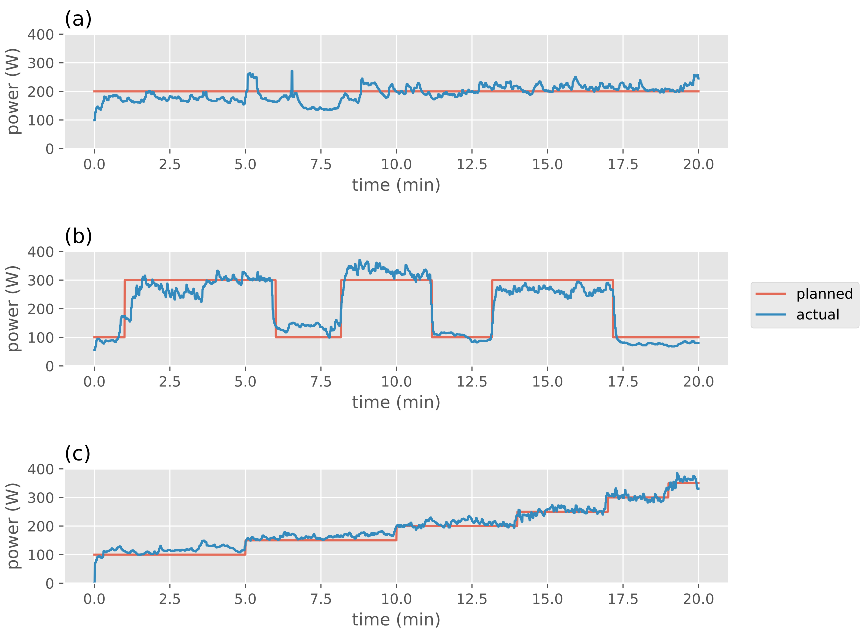


Figure 3.7: Examples from each of the three main categories of workouts performed in the data collection. Each of the plots has a red line representing the planned workout structure, to give a clear indication of what type of workout it is. The blue line represent the actual power output during the workout. **(a)** Steady-state effort at approximately 200 W. **(b)** High intensity intervals at 300 W. Interval length was 5 min, 3 min and 4 min, with 2 minute breaks in between. **(c)** Ramp workout. 5 min at 100 W, 5 min at 150 W, 4 min at 200 W, 3 min at 250 W, 2 min at 300 W and finally 1 min at 350 W.

Data collection during rest

Data was also collected when the subject was sedentary, and not performing any physical activity. The subject was in this case wearing two RIP sensors (around the rib cage and abdomen, respectively) and a heart rate monitor, but no power output was measured. The power values were manually set to zero for all time steps during this stage of the data collection, in order to provide data samples that represented no physical activity.

3.2 Preprocessing

The raw data went through several preprocessing stages before being fed into the machine learning algorithm. These stages are described below.

Upsampling

The heart and the power output were sampled at a lower rate than the RIP data, and had to be upsampled in order that each variable were represented at a time step of 0.1 second. The gaps were filled with the nearest measured value in time.

Feature engineering

A feature is in this context a characteristic of the data set that is present in all data samples, i.e. at each time step. During data collection we collected three such features: Heart rate, RIP signal from the rib cage and RIP signal from the abdomen. In order to improve the performance of the model, we also made other features derived from the raw input features. While neural networks are designed to learn features automatically, manually derived features tend to be very important to facilitate the learning ability of the neural network, and might reduce the need for a complex network architecture. Feature engineering is also done to reduce the dimensionality of the problem. If the engineered features prove to give valuable insight to the problem, the computational cost of training models might decrease, since the algorithm will need less epochs to reach a state were it can give accurate estimations.

A starting point for the feature engineering is what we intuitively know about how breathing behaves during physical activity. When increasing the intensity of a physical activity, we start to take deeper breaths and breathe at a higher rate. Based on these observations, we included respiratory rate and depth of breath into the features to be explored. The depth of breath was calculated as the difference between the maxima and minima of the RIP signal within a certain time window. We call this variable the **range** of the RIP signal. The **respiratory rate** was calculated by finding peaks in the signal, and looking at the distance between the peaks. The time gap t_{peaks} (measured in seconds) between one peak and the next was used to calculate the corresponding respiration rate R according to the following formula:

$$R = \frac{1}{t_{peaks}} \cdot 60 \tag{3.1}$$

We multiply by 60 to obtain the number of breaths per minute corresponding to the current distance between breaths. The peaks were found using the SciPy function `find_peaks()`[57], which determines the indices of peaks based on a certain distance and height from neighboring points. The accuracy of these calculations depends on whether the found peaks actually represent maxima in the breathing pattern, rather than noise from the RIP sensor. Breathing patterns may also be jagged, and the maximum value of one respiration cycle might not be easy to distinguish. Because of this, a smoothing was performed on the calculated respiration rate time series, in order to give a rolling mean of the computed values.

Another aspect of the RIP data we investigated was the **gradient** and **slope** of the signal. A mundane way of describing this would be: How fast do we breath in and out. There are several ways of representing this aspect of the data. One way is to simply calculate the gradient $\hat{f}$ of the signal f, which we did by using the following formula, implemented in the Python framework NumPy[58]:

$$\hat{f}_t = \frac{f_{t+1} - f_{t-1}}{2d}, \tag{3.2}$$

where t represents the index of the current time step, and d is the distance between each data sample of f.

Another way of representing how we draw our breath is to look at the slope of the signal, θ. To calculate the slope θ_t at time step t, we use the formula:

$$\theta_t = \arctan\left(\frac{f_t - f_{t-1}}{d}\right). \tag{3.3}$$

Furthermore, we used a sine- and cosine-encoding of the slope, which means that we actually computed two features from the slope. This gives us several advantages. Firstly, if we had used only the slope degree itself, then the steepest upward angle and steepest downward angle would be on opposite sides of the

Table 3.2: Features engineered from the raw data

Feature name	Unit
Range of RIP rib cage signal	mV
Range of RIP abdomen signal	mV
Respiratory rate calculated from the RIP rib cage signal	breaths/min
Respiratory rate calculated from the RIP abdomen signal	breaths/min
Gradient of RIP from rib cage	mV/s
Gradient of RIP from abdomen	mV/s
Sine of the slope of RIP from rib cage	radians
Cosine of the slope of RIP from rib cage	radians
Sine of the slope of RIP from abdomen	radians
Cosine of the slope of RIP from abdomen	radians

spectrum. By using $\cos(\theta)$, this value will keep a connection between the upward and downward angles that are of similar absolute value. By using $\sin(\theta)$, we capture the other phase of the slope angle. All the engineered features are summarized in table 3.2.

Splitting data into training, validation and test set

In our approach we divided the data into three categories: Training, validation and test data. These data sets served different purposes during the development of the machine learning models.

Out of the total 21 workouts in the data set, 4 were separated out as a test data set, on which the final models would be evaluated. This test set was only used once, after the final models were produced. This is important to note, because the final evaluation score should reflect how the model performs on unseen data. If a model is tuned further after having been evaluated on the test set, the model is essentially biased towards this specific test set.

After having partitioned the test set, the remaining workouts were divided into a training and a validation set, with a split of respectively 70% and 30%. The training set was used to do the actual model fitting, while the validation set was used to tune the architecture of the neural networks and all hyperparameters defining the models.

The training set was further split during the training process, where 25% of the training data was used to evaluate the model after each epoch. This provided information on the model's performance in relation to number of epochs, and let us know when the model starts to become overfitted to the training data. When the evaluation after each epoch starts to give deteriorating results, training can be stopped.

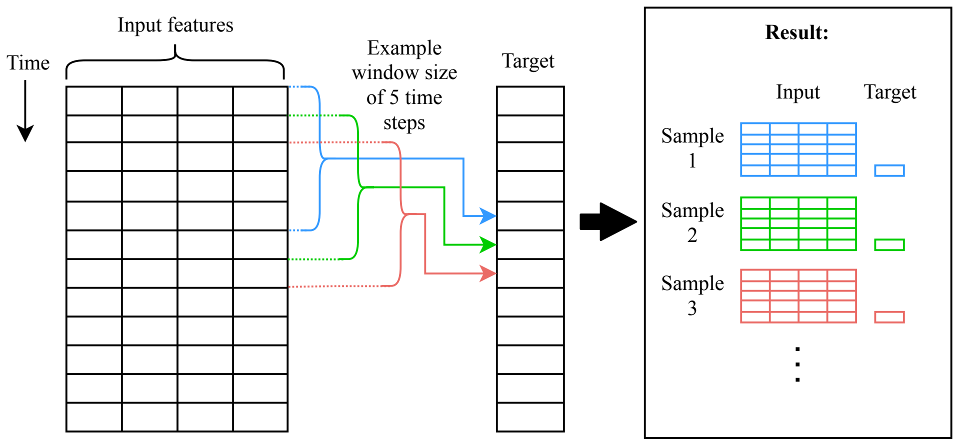


Figure 3.8: The process of matching an input sequence to a target value.

Scaling

The input features were scaled in accordance with either the min-max scaling or standardization outlined in Section 2.3.5. In general, these two methods of scaling gave similar results during experimentation.

Dividing data into sequences

The data was divided into overlapping sequences of a certain size, which we denote the *history size*, or s. Each sequence was matched to the power value recorded at the last time step of the sequence. In practice, this would mean that the last few seconds of recorded input data will be used to estimate the current power output. Consecutive sequences overlapped with $s-1$ samples (i.e. all but one sample), in order to get the maximum number of input sequences from the data set. Figure 3.8 shows a diagram of the process.

3.3 Building predictive models

Training and evaluating models

The neural networks were trained on the input sequences for a certain number of epochs. In order to evaluate the models created during each experiment, we ran the inputs of the validation set through the model, and compared the result with the expected output. We used the Mean Squared Error, as described in Equation 2.14, to get a concrete value of how the model performed. This MSE

Table 3.3: An overview of the various parameters that define the neural network architectures.

DNN	LSTM	CNN
No. of layers	No. of hidden units	No. of convolutional layers
No. of nodes in each layer		No. of nodes in each layer
Activation function		Activation function
		Kernel size
		Dropout layers
		Dropout rate
		Pooling layers
		No. of dense layers
		No. of nodes in dense layers

score where then used to tune all parameters related to the model, as described in the following sections.

Selecting features

The process of feature selection is based on continuous iteration over the different possible configuration of input variables along with the various hyperparameters and network architectures (described below). In order to optimize the performance of the model, one has to try out a lot of various combinations. We attempted to automate parts of this process using a tool called Keras Tuner[59], which searches for an optimal configuration based on a defined set of variable parameters. Even so, the search space is enormous, and we mainly had to rely on the common trial-and-error method. While this type of manual work can be slow, one can make use of domain knowledge and understanding of the methods to pick the features and parameters that are most likely to result in high performance.

Tuning hyperparameters and models

Regarding the architecture of the neural networks, we had to find the optimal configuration among several choices and control parameters.

The training process and result is also affected by how many epochs we run during training. To control this, we used functionality from TensorFlow that enables so-called *early stopping*[60]. The TensorFlow API allows us to set apart a certain fraction of the training set during the training process, to monitor the loss on unseen data for each epoch. This is called the validation loss, and is evaluated at the end of each epoch. When the validation loss has failed to

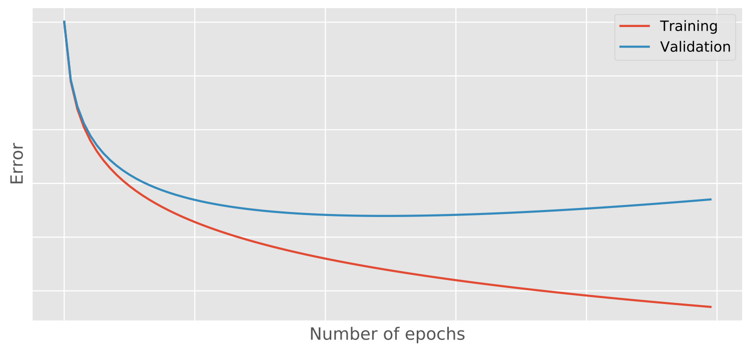


Figure 3.9: An example of how the error typically behaves during training. In the beginning, the error decreases as the epochs accumulate for both training and validation data. At a certain point, the model starts to become *overfitted* to the training data, and the error on unseen data from the validation set increases.

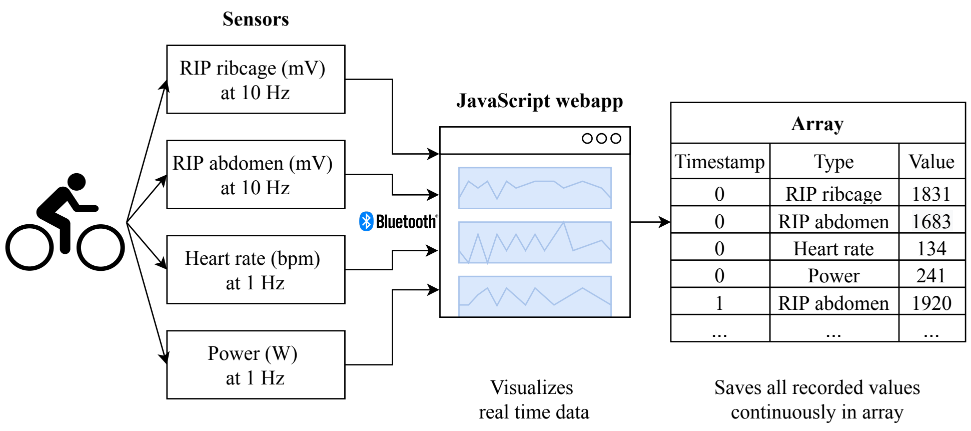
improve after a given number of epochs, training may be stopped, to reduce the danger of overfitting to the training data. Typical behavior of the training and validation loss is shown in Figure 3.9. One also has the option of saving checkpoints of the model during training, so after training has stopped, one can save the model which gave the lowest validation loss.

3.4 Implementation

In this section we present the implementation of the methodology discussed above.

3.4.1 Software for data acquisition

In order to collect data from the four sensors mentioned above (two RIP sensors, a heart rate monitor and the power meter on the bike), we developed our own, specialized web application. All three sensor types have built-in functionality for broadcasting the recorded values via Bluetooth, and so we connected the sensors to our application using Web Bluetooth[61]. The application is implemented in JavaScript, and runs entirely in the browser without a backend. The purpose of this tool was to have a unified interface for all four sensors, with reliable and synchronized timestamping of all samples. The part of the source code that enables communication with the Concept2 BikeErg was adapted from the open source web application Ergarcade[62].


Sensors
RIP ribcage (mV) at 10 Hz
RIP abdomen (mV) at 10 Hz
Heart rate (bpm) at 1 Hz
Power (W) at 1 Hz
Bluetooth
JavaScript webapp
Visualizes real time data
Array
Saves all recorded values continuously in array

Timestamp	Type	Value
0	RIP ribcage	1831
0	RIP abdomen	1683
0	Heart rate	134
0	Power	241
1	RIP abdomen	1920
...	...	...

Figure 3.10: Overview of the data flow in our data acquisition tool.

A screenshot of the web application can be seen in Figure 3.11. The source code for the application can be found at `https://github.com/ejhusom/flow`.

3.4.2 Supporting Software

In the following subsection we describe which supporting software and libraries were used to produce the results of this project.

Python: All computational experiments in this thesis has been done using Python[63] as a programming language. This choice was made since Python is a language that enables quick and flexible writing of code. Also, there exists several well-developed frameworks for machine learning that has a Python API, which makes Python a natural choice when doing machine learning experiments.

TensorFlow and Keras: There exists several frameworks for working with machine learning, and we wanted to choose one with the following features:

- Support for GPU usage.
- High-level Python API, to enable efficient and flexible development.
- JavaScript API, in order to enable a hypothetical web app with client-side prediction.

The machine learning framework TensorFlow[60] fulfills all of the demands listed above. It is developed by Google, available for a multitude of computing platforms, open-source and runs on both CPUs, GPUs and TPUs. When developing in TensorFlow, we used the Keras API. Keras is an API written in Python, providing a high-level interface for TensorFlow 2. The focus of this

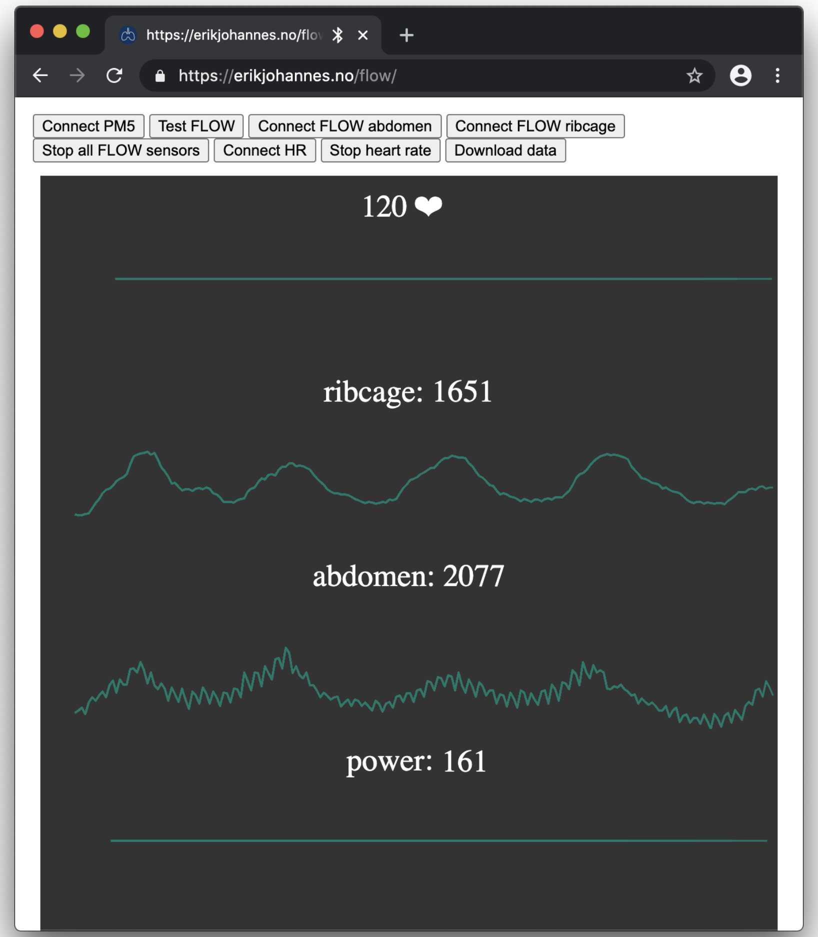


Figure 3.11: Screenshot from the web application used to collect data.

library is to reduce the number of actions and commands required to perform machine learning experiments. Keras was chosen in this project to enable efficient and flexible experimentation and prototyping of machine learning models. The machine learning experiments were mainly run on a cluster of four NVIDIA GeForce RTX 2080 Ti graphics cards, provided by the University of Oslo.

3.4.3 Tools for structuring experiments

An important aspect when analyzing the performance of machine learning algorithms is to keep track of how models perform in relation to the parameters that define the model. In the case of neural networks, the number of parameters are substantial, and it can be challenging to hold a thorough record of all details of a large amount of experiments and iterations.

In order to deal with this challenge, we used a tool called Data Version Control (DVC)[64], which among other things is designed to manage machine learning experiments. This software integrates with the version control system called Git, and tracks all parameters automatically during each of the stages of an experiment. DVC also has the ability to save the evaluation metric of every model that is made. When using this in combination with Git, one has complete and easy overview and control over all experiments, what version of the code with which they were run, which parameters were used, and the results of the experiments. When using DVC, all variable hyperparameters are kept separate from the code, and each stage of the machine learning pipeline is clearly defined. Each stage has its own dependencies, usually a set of input data, in addition to the hyperparameters that control the details of the stage execution.

A very handy feature is that DVC handles caching of data in-between stages of the pipeline, and does not run a stage if the stage has already been run with the same parameters before. Between each of the stages, the processed data set is saved to separate directories. DVC caches the data with a unique file hash, and fetches the correct output data for a stage when it sees that a stage is being run with the same parameters and the same input data. This speeds up the development cycle significantly, since rerunning identical stages are avoided automatically from the start to the end of the project.

Figure 3.12 shows how DVC is utilized in this thesis. The source code is kept in a Git repository, as shown in the column to the left. To run experiments, this source code is downloaded to a workspace, and the data set is added to the same workspace. The source code consists of three main parts: The DVC control file (`dvc.yaml`), scripts written in Python, and a file containing all control parameters that define any unique experiment (`params.yaml`). All dependencies between scripts and control parameters are defined in the DVC control file.

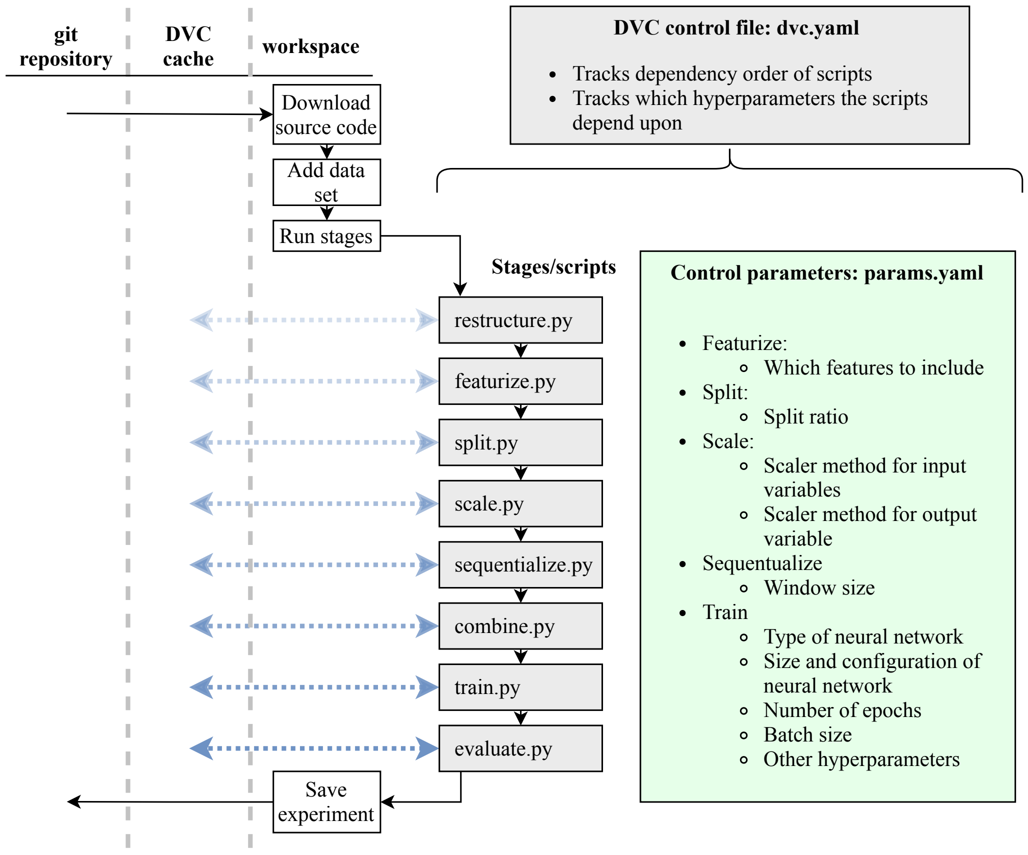


Figure 3.12: This diagram presents how experiments are executed with DVC. The gray background specify parts of the code that stay unchanged. The green background specify parts that are adjusted between experiments. Blue arrows indicate that data can be fetched from or put in cache, in order to save computation time.

When the stages of each experiment is run, DVC is used to check whether a stage has been run with identical control parameters during a previous experiment. The blue arrows show that for each stage, DVC can fetch data from cache if available, or put data in cache if necessary. The opacity of the blue arrows gives an indication on how often data need to be cached. The earliest stages are rerun more seldom, as they have fewer dependencies than subsequent stages.

Chapter 4

Results

In this chapter we present the results of our research, and discuss them in the context of our research questions.

4.1 Data acquisition and data quality

Our first research question, **RQ1**, was: **How can we acquire the data necessary to estimate power output from breathing?** We developed a web application for simultaneous recording of the data needed for our project, and this software is described in Section 3.4.1. This web application proved to be stable and reliable, and ensured that the recorded data from four sensors were timestamped correctly and stored securely. The results of our data acquisition was 21 workouts, totalling approximately 10 hours of recordings, where we sampled RIP signals from the rib cage and abdomen respectively, heart rate and power output. Figure 4.1 shows an example of the data collected from one of the workouts, which is an interval workout of approximately 20 minutes, with three bouts of intensity.

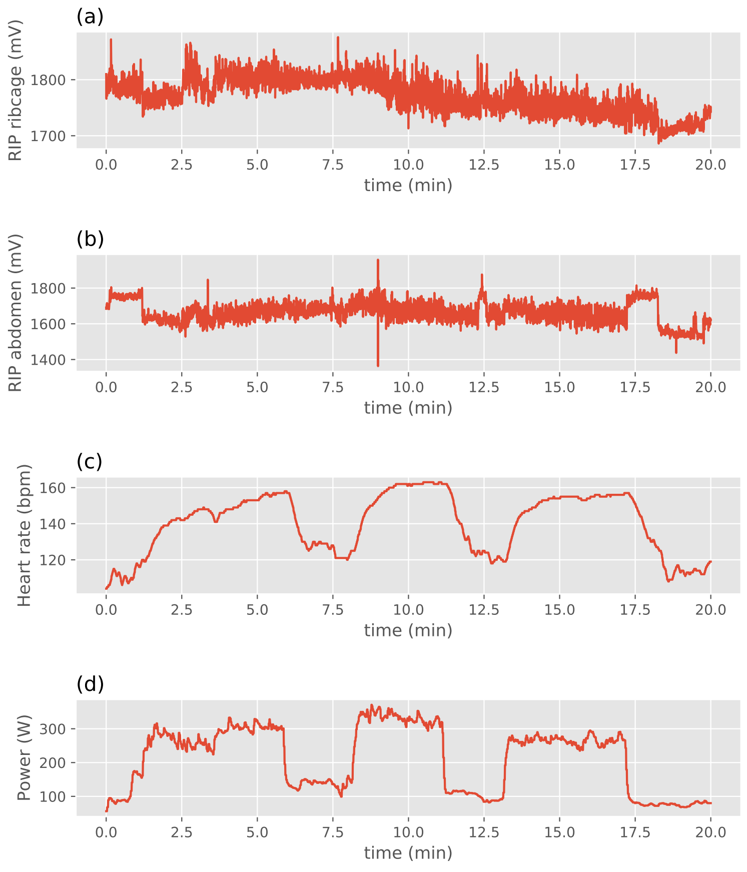


Figure 4.1: Sample workout, showing (a) RIP signal from the rib cage, (b) RIP signal from the abdomen, (c) heart rate and (d) power output.

In order to get a well grounded evaluation of our models, we need to make sure that the distribution of the values in our test data set is similar to that of our training data set. Because of this, we ensure that the workouts of the test data set was chosen from each of the three main workout categories, summarized in Table 3.1. Figure 4.2 shows the distribution of the training data set, and Figure 4.3 shows the distribution of the test data set. Overall, the distribution seems to be similar. Note that the training data set contains data collected while the subject was at rest, and therefore has a high presence of power values of zero and heart rate values below 100. We wanted to include rest data in order to give the predictive models a baseline for a power output of zero watts, which indicates that no sport-specific effort is undertaken. In reality, the power output is never zero, since we always perform some work in order to stay alive. During experimentation, the models improved significantly after including the data that was collected during rest. The test data do not contain data collected while at rest, since predicting power output while at rest was out of scope for this project. A summary of the descriptive statistics of the training and test set is shown in Table 4.1.

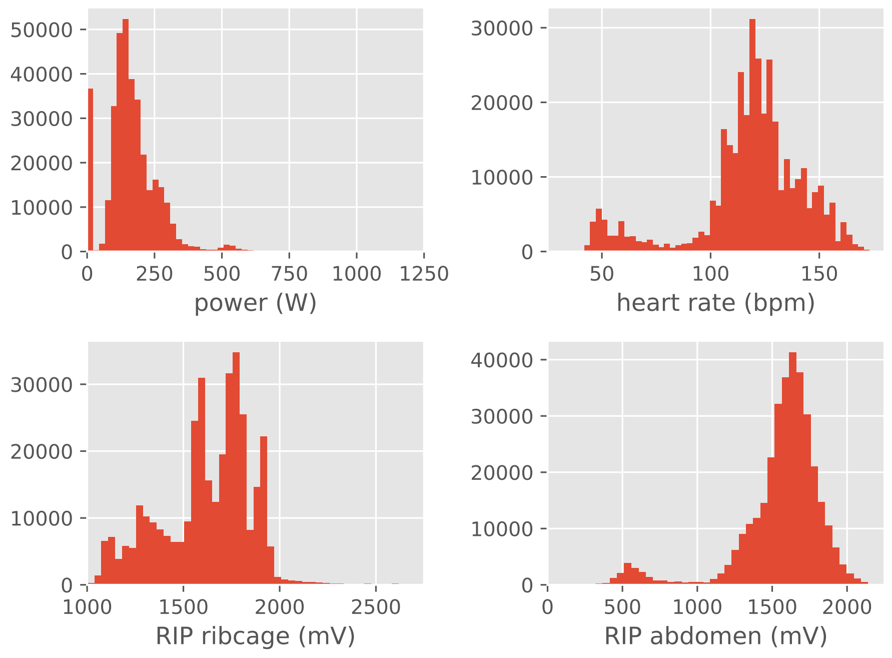


Figure 4.2: The distribution over power values in the training data set.

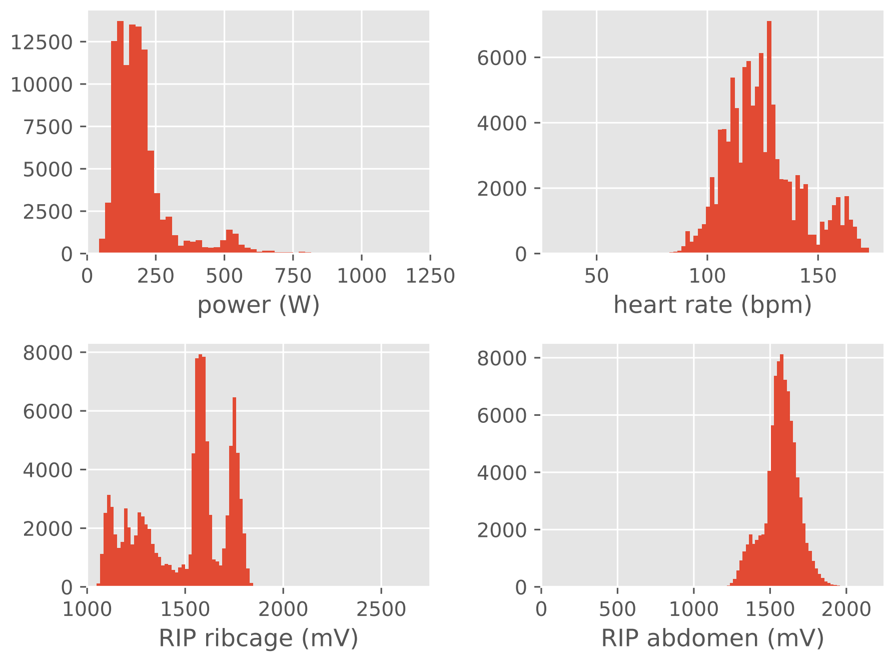


Figure 4.3: The distribution over power values in the test data set.

4.2 Network architectures

In this section we discuss our next research questions, **RQ2** and **RQ3**: **How can we build a predictive model to estimate power output from breathing, and what machine learning architectures are effective for doing this?** As described in Section 2.3, we chose to build our predictive models using neural networks, because of their potential when it comes to time-series data and input sequences with local dependencies. CNNs and RNNs are specifically designed for detecting patterns with such dependencies, but we also included the less complex DNNs to get a baseline for comparison. The process of choosing a fitting architecture for neural networks can be very challenging, primarily because the possible combinations and configurations are infinite. While there are some frameworks and tools that allows some automation of this process, for example Keras Tuner[59], it is easier to control and supervise the process by manually crafting the network architectures. Through the common trial-and-error process, we found three suitable architectures for examining the performance of DNN, CNN and LSTM networks. We started with networks with few layers and nodes/units, and expanded them until we reached architectures that yielded promising results. The control parameters that were experimented with

Table 4.1: Descriptive statistics for the training and test data set.

	Power (W)		RIP rib cage (mV)		RIP abdomen (mV)		Heart rate (bpm)	
Statistic	Train	Test	Train	Test	Train	Test	Train	Test
Mean	150.0	197.0	1677.8	1494.2	1554.6	1571.8	115.8	
Minimum	0	0	1002	1049	272	1177	42	81
Maximum	58	1101	2798	1915	2666	2203	169	173
Standard deviation	89.6	114.2	221.3	221.9	324.0	112.2	27.5	17.0
5th percentile	0	92	1274	1109	643	1358	50	102
95th percentile	301	498	1934	1778	1907	1748	152	161

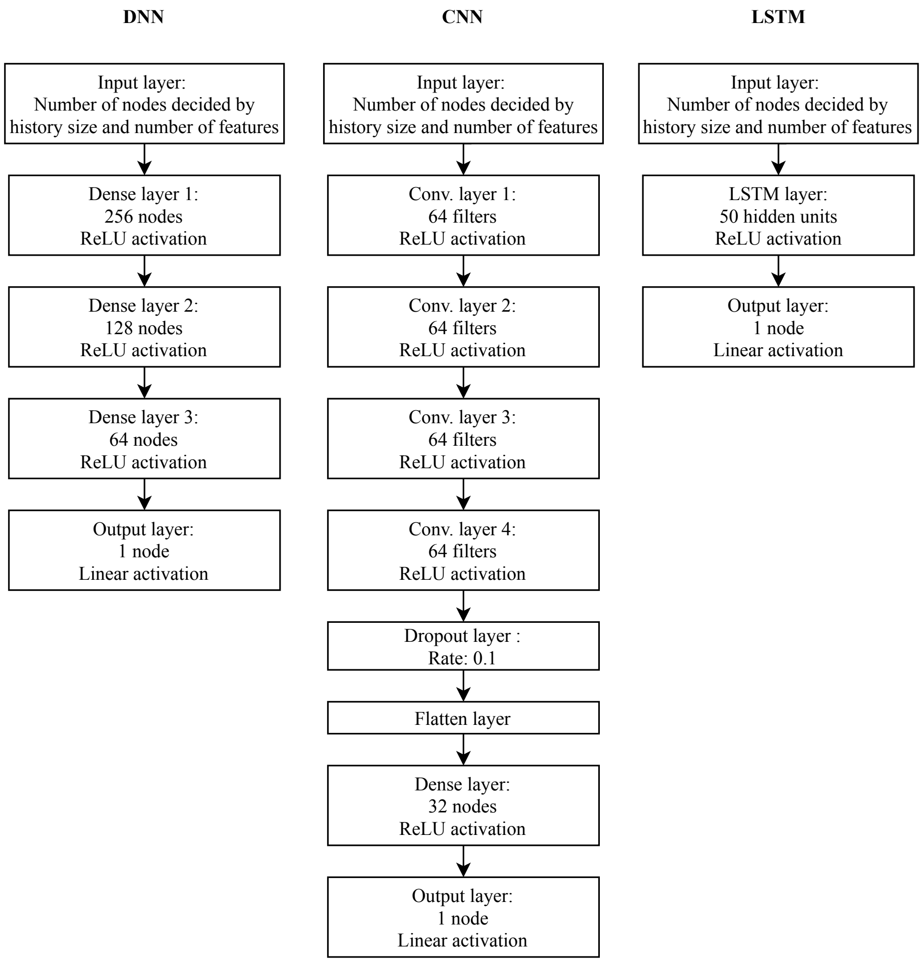

DNN
CNN
LSTM
Input layer: Number of nodes decided by history size and number of features
Dense layer 1: 256 nodes ReLU activation
Dense layer 2: 128 nodes ReLU activation
Dense layer 3: 64 nodes ReLU activation
Output layer: 1 node Linear activation
Conv. layer 1: 64 filters ReLU activation
Conv. layer 2: 64 filters ReLU activation
Conv. layer 3: 64 filters ReLU activation
Conv. layer 4: 64 filters ReLU activation
Dropout layer : Rate: 0.1
Flatten layer
Dense layer: 32 nodes ReLU activation
LSTM layer: 50 hidden units ReLU activation

Figure 4.4: The three network architectures used for producing our predictive models.

during this process is presented in Table 3.3. The three network architectures are presented in Figure 4.4. The LSTM network is here presented with 50 hidden units, which is what we used for choosing the optimal feature set. In section 4.5 we show the results for fine-tuning this number.

4.3 Feature selection

In Table 4.2 we have defined eleveen different sets of predictor variables. We want to compare these in order to answer research question RQ4: **What features of breathing data are useful for estimating power output from breathing?** The results are presented in table 4.3, where we have calculated the MSE and R^2 score. Feature set 11 only contain heart rate as an input

Table 4.2: Overview over the predictor variables used in the various models. The feature set number corresponds to the feature set number in table 4.3.

	RIP rib cage and abdomen					Heart rate	
Feature set no.	Raw	Range	Frequency	Gradient	Slope	Raw	Slope
1	x						
2	x					x	
3		x				x	
4			x			x	
5				x	x	x	
6				x	x	x	x
7		x	x	x	x		
8		x	x	x	x	x	
9				x	x		
10					x		
11						x	

feature, and this can be used as a baseline to indicate the usefulness of RIP signals compared to heart rate as a predictor for power output. In all of the models in Table 4.3 we used the network architectures presented in Figure 4.4, and a history size of 100 time steps, i.e. 10 seconds. For the CNN network, we used a filter size of 6. We employed *early stopping* during training, as explained in section 3.3. We used 25% of the training data for evaluating the validation loss for each epoch, and if the validation loss had not improved in the last 50 epochs, the best model checkpoint up until that point was saved and used for evaluation.

In table 4.3 we have emphasized the best performing models with bold typeface. Using DNN, the best models were obtained with feature set 3 and 6. Both feature sets gave an MSE of 0.007 and an R^2 score of 0.60. Since we prefer simpler models over more complex ones (i.e. ones that process more input features), we have emphasized feature set 3. With CNN, we obtained the best model using feature set 6, which gave an MSE of 0.006 and an R^2 score of 0.66. This was also the best model overall. LSTM gave best results with feature set 3, yielding an MSE of 0.006 and an R^2 score of 0.64. The benchmark models trained on only heart rate data (feature set 11) gave best result when using DNN, with an MSE of 0.006 and an R^2 score of 0.64. Only the best performing CNN model beats this score. The best performing model that did not include heart rate data as an input feature was a CNN model trained on feature set 10, giving an MSE of 0.009 and an R^2 score of 0.45.

The different sets of input variables presented in Table 4.2 where designed based on several hypotheses about what features might be used to estimate the power output. The range of the RIP values is a feature that can express the depth of

Table 4.3: Error metrics indicating the performance of the predictive models when testing various input features. History size was set to 100 time steps.

	DNN		CNN		LSTM	
Feature set no.	MSE	R2	MSE	R2	MSE	R2
1	0.021	-0.26	0.017	0.00	0.019	0.00
2	0.008	0.50	0.011	0.35	0.011	0.34
3	**0.007**	**0.60**	0.009	0.49	**0.006**	**0.64**
4	0.010	0.38	0.017	0.00	0.010	0.37
5	0.008	0.51	0.007	0.62	0.009	0.47
6	0.007	0.60	**0.006**	**0.66**	0.008	0.55
7	0.018	-0.08	0.017	0.00	0.016	0.06
8	0.011	0.35	0.017	0.00	0.007	0.56
9	0.026	0.28	0.010	0.43	0.016	0.07
10	0.016	0.07	0.009	0.45	0.014	0.20
11	0.006	0.64	0.007	0.60	0.007	0.61

our breath, while the frequency on the RIP signal might indicate our respiration rate. The slope and gradient of the RIP signal can express how fast we draw our breath. While we focused on engineering features that we can link to an intuitive understanding of how our breath works, these are not necessarily the best features for a machine learning algorithm. The trade-off between predictive and descriptive accuracy is explained in length by Murdoch et. al.[65]. The predictive accuracy is defined as how well a model performs in terms of the error metrics used to evaluate it. This is the performance of the model, or how well the model is able to estimate a desired function or mapping. The descriptive accuracy is how well a method of interpretation is able to describe the relationships that the model has learned. We have in our thesis focused on what Murdoch et. al. define as *model-based interpretability*, which is creating predictive models that enable understanding of the relationships learned by the machine-learning algorithm[65]. In practice, this means that we prioritize simpler models, which are easier to interpret, rather than complex models with a large number of input features. With fewer input features, we can more easily determine which features are important for making predictions. This is especially true for machine learning methods such as neural networks, where the model consists of a large number of parameters that are challenging to interpret (which is also why neural networks are referred to as "black boxes"). By using few and intuitively engineered features, we may more easily understand what information from the breathing dynamics is used to do power output estimation. A different approach, with a larger number of features and more complex models, may give a higher predictive accuracy, but the interpretability might suffer.

When looking at the error metrics in Table 4.3, we see that the models trained solely on heart rate (feature set 11) perform quite well compared to the feature sets which involve the RIP signals, with R^2 scores of 0.60 and above. The best model trained on feature set 11 was obtained with the DNN, and gave an MSE of 0.006 and an R^2 score of 0.64. This serves as a baseline for our other models, which are trained on features based either solely on RIP signals, or a combination of RIP signals and heart rate. For DNN and LSTM, we got the best models when using feature set 3, which consisted of RIP range and heart rate. However, neither of these models show improved error metrics compared to a DNN trained solely on heart rate. The only model that does so is the CNN model trained on feature set 6, which gave an MSE of 0.006 and an R^2 score of 0.66.

4.4 History size

After choosing the optimal feature set from Table 4.3, we used those results to fine-tune the size of the individual input sequences passed to the networks, which we call the history size. In Figure 4.5 we present the error metrics of the three different neural networks as a function of history size. For the CNN and the LSTM network, a history size of 100 gave the best results among the tested values, when we used respectively feature set 6 and 3. The DNN was tested with feature set 3, and in this case, a history size of 120 gave the best result.

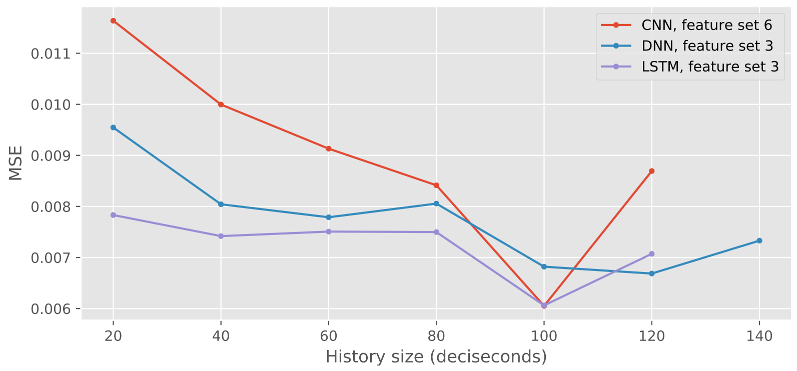


Figure 4.5: The MSE and R^2 score as a function of history size.

The analysis of history size shows that about 10 seconds of input samples give the best performing models when using the networks and features chosen by us. It makes sense that a sequence of values is needed to predict the power output, since the chain of values form a chronological context that might indicate at what intensity the subject is exercising. This is especially true for the RIP signals. The process of breathing has a cyclical nature, and single RIP values

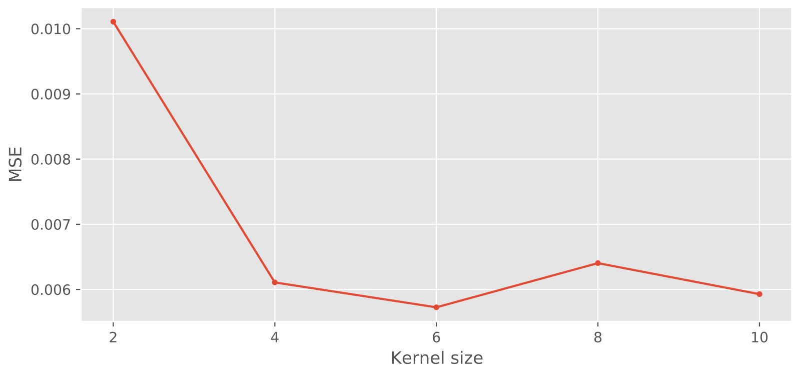


Figure 4.6: The MSE as a function of number of filter size in the CNN network, when using feature set 6. The history size was in set to 100.

cannot convey information about whether the subject is inhaling, exhaling, or any characteristics of the signal. Some of the temporal dependencies in the input signals can be encoded in the engineered features, and thus lessen the need for a large sequence of samples as input. The RIP range, for example, is based on computing the differences between maximum value and the minimum value over a certain amount of time steps. This means that some temporal information is included in a single value of this feature. Even so, our results indicate that a relatively large number of time steps, around 10 seconds of input data, is needed to get accurate power output estimation.

4.5 Neural network hyperparameters

While there are many hyperparameters that control the configuration of neural networks, we have chosen to focus on two core parameters, that seemed to have a significant effect on the model performance. In Figure 4.6 we see the error metrics as a function of filter size in the CNN, when trained on feature set 6. We have in this case used a history size of 100. The results show that a kernel size of 6 proved to be best. The performance of the CNN is quite stable when varying the filter size from 4 to 10, as seen in figure 4.6. The filter size affects the receptive field of the nodes, and a larger filter size means that a node will receive information from a larger number of nodes in the preceding layer. A filter size of 2 is clearly inferior to the larger filter sizes, but above that, the specific value does not seem to matter much.

Figure 4.7 shows an error analysis of the LSTM network as a function of number of hidden units, when trained on feature set 3. It shows that 110 hidden units give the best results, among the tested values.

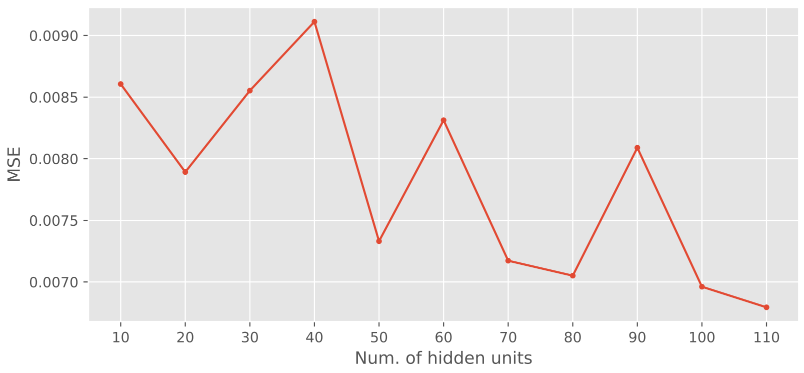


Figure 4.7: The MSE as a function of number of hidden units in the LSTM network, when using feature set 3. The history size was set to 100.

The challenge of tuning hyperparameters for machine learning models, in addition to choosing the optimal feature set, is that the multi-dimensional nature of this optimization makes it exceptionally demanding to find the most appropriate configuration. Even though a history size of 100 proved to work best with the feature set and network configuration we had chosen at that point, that might not be the case if the networks had other architectures, or the various hyperparameters of the model were slightly different. Finding the optimal combination of all factors remains a demanding task with such a large number of parameters to adjust.

4.6 Evaluation on test set

Based on the results in the previous sections, we arrive at the optimal configurations shown in Table 4.4. Using these parameters, we evaluated our models on the test data set with the aim of answering research question **RQ5**: **What is the effectiveness of estimating power output from breathing?** The results of our final model evaluation are shown in Table 4.5. We chose to evaluate the three best performing models for each type of neural network, in addition to the best model that was trained only in RIP signals (CNN on feature set 10). We also evaluated the model trained solely on heart rate, to use as a baseline for the other models. In Table 4.5 we have included the error metric MAPE, defined in Equation 2.17, which indicates the average absolute error expressed as a decimal fraction of the true values. This error metric gives us an indication of how much the estimated values deviate from the true values on average. MAPE differs from MSE in that it is not dependent on the scale of the values, and can therefore give a more intuitive interpretation of the error. If we look at the error metrics in Table 4.5, the CNN model trained on feature set 6 gives the

Table 4.4: A summary of the optimal variables for each of the three network types.

	DNN	CNN	LSTM
Feature set	3	6	3
History size	120	100	100

Table 4.5: Error metrics for final models when evaluated on the test data set. The hyperparameters shown in table 4.4 was used.

Network	Feature set	MSE	R^2	MAPE
DNN	3	0.006	0.36	0.23
CNN	6	0.004	0.56	0.20
LSTM	3	0.006	0.35	0.22
CNN	10	0.005	0.50	0.24
DNN	11	0.006	0.43	0.22

best performance, with a MAPE of 0.20. Feature set 6 contained the gradient and cyclic encoding of the slope for the RIP signals, in addition to the heart rate and a cyclic encoding of the slope of the heart rate. A combination of RIP signals and heart rate features seems be the best approach for obtaining optimal performance of power output estimation, based on our methods. However, the difference in performance of the five models in Table 4.5 is quite small, with the highest MAPE being 0.24 for the CNN trained on feature set 10. Even so, this model has a lower MSE and a higher R^2 than the DNN trained on heart rate, so it is ambiguous which one of these has the best performance. Our assessment of the results is that the differences in performance are not significant enough to judge one of the network architectures or feature set to be clearly superior to the others, at least not exclusively based on error metrics.

We have also produced examples of power output estimation from these final models. All of these examples are produced from **about 20 minutes of data from the test set**, and have a red curve showing the true values, and a blue curve showing the predictions produced by our models. We will discuss the predictions from our various models below.

Figure 4.8 shows the power output estimation produced by the DNN trained on feature set 3. This model shows a less noisy result compared to the results of our CNN and LSTM models. This might indicate that we are able to obtain results with less variability by using DNNs. However, by looking at the first peak in the power output plots, we see that the DNN model in Figure 4.8 has a certain time delay when estimating the peak power output. The estimated peak power output at nearly 300 W is well below the true peak, which is just above 400 W. One explanation of this might be that the data set contains relatively few samples with a power output of 400 and above, as seen in Figure 4.2 and 4.3.

Because of this, we expect the models to be weaker at estimating correct values in that range. It is possible to remedy this by assigning a higher importance to samples in this range during training, which means that they will have a greater impact on adjusting the parameters than other samples. We tested this strategy during the building of our models, but decided to abandon it since the overall performance of the models suffered from it. The DNN model in Figure 4.8 also shows a similar time delay at the peak at minute 6, but otherwise it seems to handle both timing and peak values in a satisfactory manner.

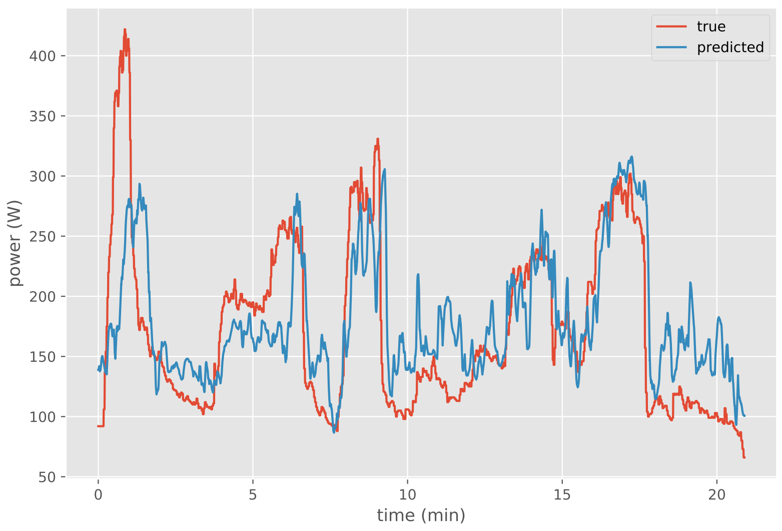


Figure 4.8: Prediction of power output on an interval workout from the test set, using DNN architecture on feature set 3. MAPE: 0.23.

The CNN model in Figure 4.9, which was trained on feature set 6, is our best performing model as concerns the error metrics. It seems to achieve better timing of rapid increases and decreases in power output, when compared to the DNN model in Figure 4.8. An important difference between these two types of network is that the CNN is able to extract information from neighboring values in the input sequence, and might therefore be better at processing the temporal dependent characteristics of the input data. The CNN model also seems better at hitting close to the true values at the maxima and minima of the power output, although the output signal is in general quite noisy.

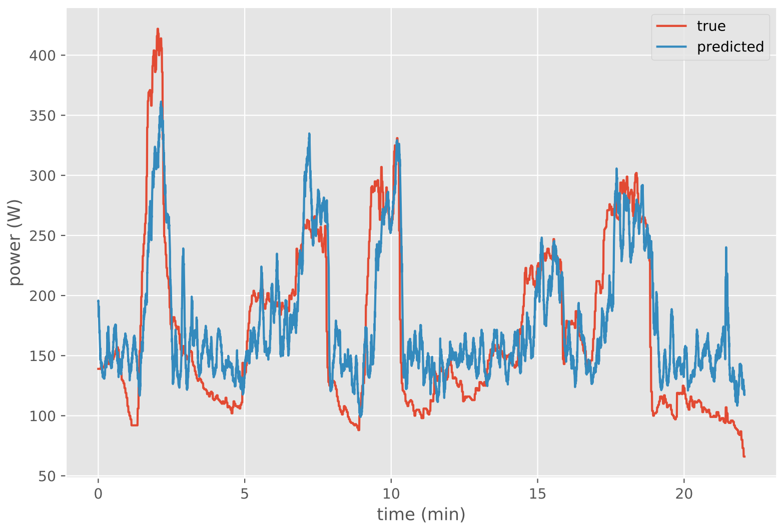


Figure 4.9: Prediction of power output on an interval workout from the test set, using CNN architecture on feature set 6. MAPE: 0.20.

Figure 4.10 contains an example from our LSTM model, which was trained on feature set 3. A visual inspection of the plot shows that the model seems to suffer from a challenge of having a time delay for large increases in power output. This is similar to what we observed for our DNN model in Figure 4.8, but the delay seems to be even more pronounced for our LSTM model. The LSTM model also appears to struggle with hitting accurate values after fast increases in power output. Nonetheless, the output signal from the LSTM model appears more stable than the one we observe in the results from the CNN model in Figure 4.9.

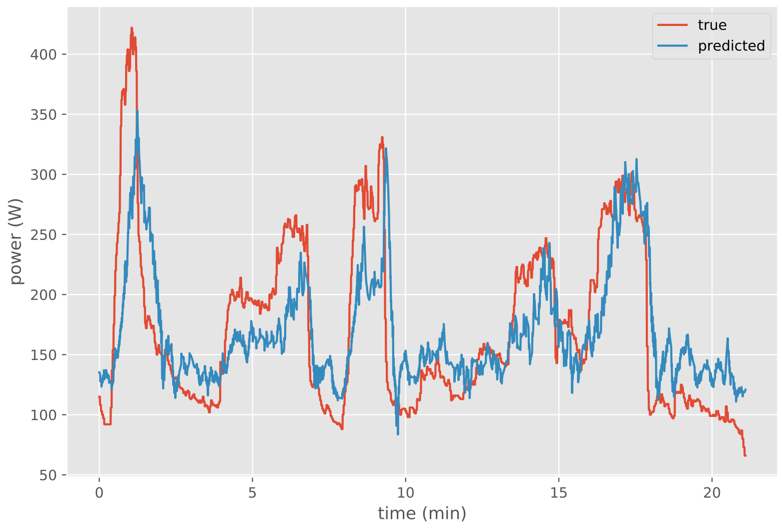


Figure 4.10: Prediction of power output on an interval workout from the test set, using LSTM architecture on feature set 3. MAPE: 0.22.

The CNN model trained on feature set 10, which does not include features derived from heart rate, is shown in Figure 4.11. These results show an output that is quite unstable and has high variability. As we can see from Figure 4.1, the raw RIP signal contains large variations and rapid changes, while the heart rate is much more stable. The RIP signal might be affected by slippage of the belt and muscular movement that is not related to breathing, and can therefore contain noise that leads to less consistency in the results. This might explain why the CNN model in 4.11 produces such noisy output compared to the other models.

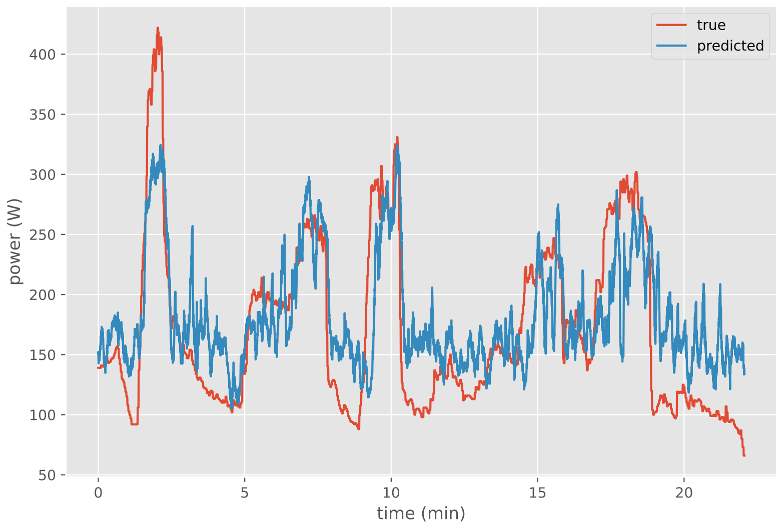


Figure 4.11: Prediction of power output on an interval workout from the test set, using CNN architecture on feature set 10. MAPE: 0.24.

In Figure 4.12 we see a plot of the results from our DNN model trained solely on heart rate. This model produces the least noisy signal of the models we have compared, but it suffers from the same time delay as our other DNN model in Figure 4.8, at least for some of the power output peaks (minute 2 and 6 in Figure 4.12). Even so, the DNN model in Figure 4.12 shows that heart rate as a single predictor can be used to achieve similar performance for power output estimation as RIP signals.

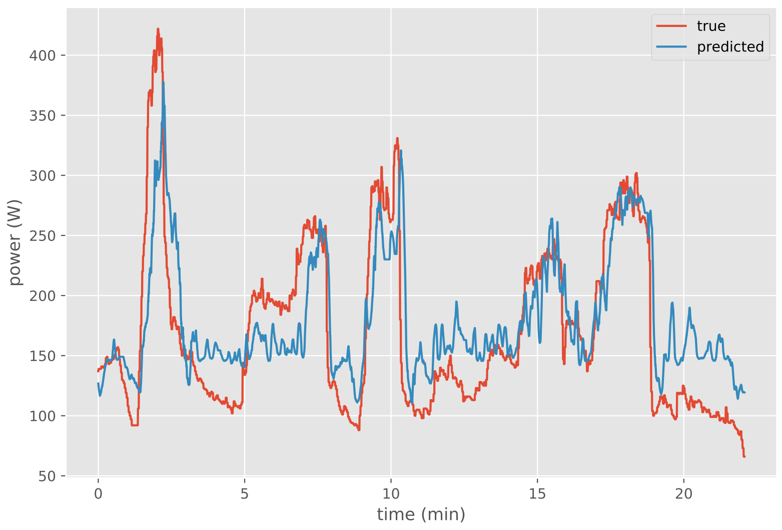


Figure 4.12: Prediction of power output on an interval workout from the test set, using DNN architecture on heart rate. MAPE: 0.22.

4.7 Limitations and threats to validity

Our assessment of the main limitations of our project is summarized below:

- **Limited sample size.** One of the limitations of this project is the small sample size of 21 workouts (totalling approximately 10 hours of recorded data of the data set. In the field of deep learning, large amounts of data are fundamental for the algorithms to generalize well[66]. A doubling or more of the data set size would also have enabled more flexibility in the search for an optimal configuration among the input variables, hyperparameters and network architectures. In our experiments, we first set apart a test set to use for a final model evaluation, which was about 20% of the total data set. This test set was designed to include a workout from the three categories present, interval workout, ramp workout and steady state, in addition to a workout with mixed intensities. 30% of the samples from the remaining training data was set apart as a validation data set, which was used to choose the optimal feature sets and hyperparameters. During the training process, 25% of the remaining data was used to monitor the validation loss, in order to find the optimal number of epochs. This means that about 42% of the total data set was used for actual training, while 38% was used for validation purposes. With more data samples, we could more easily ensure that the distributions in the training, validation and test sets were similar enough to provide us with reliable results. The results found from evaluating validation data are especially important, since we use this as grounds for tuning the network and choosing the best features.

- **Computational power.** In the field of machine learning, computational power often is seen as a limitation on experiments. The processing of large amounts of data takes time, and a considerable number of experiments need to be run to give an extensive analysis. More computational power and faster hardware could have enabled us to run a larger number of experiments, and given a more comprehensive investigation of the optimal configuration of the predictive modelling. Furthermore, the hardware used in the experiments was shared with several other research projects, which placed further constraints on the computational resources available for our project.

The main threat to external validity of our project is that we have used only a single subject for data acquisition. This thesis is an N-of-1 study, where we only have studied data acquired from a single subject. The global pandemic of COVID-19 put limits on our ability to arrange data collection from multiple subjects because of infection control considerations. While the N-of-1 method has reduced the total workload by eliminating the need for administrating experiments with multiple subjects, it has also placed limits on our investigation into RIP signals as predictors for physical effort. With our N-of-1 study, we are unable to discuss whether our chosen methods, especially the engineered features, will give similar results when applied to data from other subjects. This poses a threat to the external validity of our results.

Regarding threats to internal validity, it seems beyond doubt that breathing is related to the exercise intensity during a workout, which is also the case for heart rate. However, a possible problem is the time-related aspect of the cause-and-effect relationship between our input and output variables. Our models estimate the power output of the current time step based on the last few seconds of breathing and heart rate data. The motivation behind this design decision was that we wanted to produce models that can provide real-time power estimation during a workout. In reality, when a person increases the power output during a workout, the change in breathing will in general happen as a response to the increase in power output because of increased need for oxygen. We experimented with using target values of earlier time steps, and also predicting the average target value for the time period corresponding to the input sequence, but since the performance of the models in general did not increase, we chose to stick with our initial idea of real-time estimation. Nevertheless, this problem deserves further investigation, since both breathing and heart rate undoubtedly are variables that respond to exercise intensity, and not the other way around.

An aspect we have not addressed in this thesis is the estimation of uncertainty of the predictions made by our models. The deep learning methods we have used in this thesis does not by themselves provide any measure of the uncertainty, but this could be obtained by using the dropout technique (discussed in Section 2.3.1) to produce a probabilistic distribution of estimated values, which is a technique developed by Gal and Ghahramani[67]. An approach for improving the accuracy of such uncertainty estimations has been proposed by Kuleshov et. al.[68], which involves a recalibration technique. Methods like these could have been applied to our models in order to obtain confidence intervals for our predicted values, but due to time constraints we chose to prioritize the comparison of different types of neural networks.

Furthermore, we have not looked into the reasons behind the predictions of our model, which is an aspect that falls under the field of *explainable artifical intelligence*. Ribeiro et. al. has proposed a method called LIME, which is a technique that aims to explain predictions by making local, interpretable approximations of the model around individual predictions, in order to explain these predictions[69]. A method like this could have been used to gain deeper insight into our models, and contribute to the assessment of both the models and single predictions.

Chapter 5

Conclusion and future work

In this chapter we will conclude our thesis in section 5.1, by giving a summary of the significant elements found in our study. In section 5.2 we give our view on possible further research into the field of deep learning applied to sports and breathing data.

5.1 Conclusion

In this thesis, we studied how we could use deep learning to create personalized predictive models to estimate power output during exercise from breathing data. Through our work we have created a working web application that is able to collect respiratory inductive plethysmography (RIP) signals, heart rate and power output simultaneously through a unified interface, which also visualizes the collected data in real-time. The application connects to peripheral sensors using Web Bluetooth, and can connect to any commercial heart rate monitor and power meter that uses the standardized Bluetooth characteristics for these specific sensor types. To record the RIP signals, two SweetZpot Flow sensors are needed, which are worn around the rib cage and abdomen. The data collected through the web application was used for training neural networks to estimate the power output during cycling workouts. The models were trained using input features based on either RIP signals, heart rate, or a combination of the two.

We investigated the performance of three different types of neural network: Dense neural networks (DNN), convolutional neural networks (CNN) and long short-term memory (LSTM) networks. Our research into the effectiveness of estimating power output from breathing has shown that a CNN trained on features derived from both RIP signals and heart rate gives a mean absolute percentage error (MAPE) of 0.20, when evaluated on our test data set. The CNN used sequences of the last 10 seconds of input features to estimate a single power output value. In comparison, a DNN and an LSTM that were also trained on a combination of RIP and heart rate features resulted in a MAPE of respectively 0.23 and 0.22. Our best performing model that were only trained on RIP features was a CNN model with a MAPE of 0.24 and an R^2 score of 0.50. This is comparable to our best performing model trained only on heart rate, which was a DNN model with a MAPE of 0.22 and an R^2 score of 0.43.

The different types of networks and feature sets show in general quite similar performance, and it is therefore challenging to judge that one of them is clearly preferred over another. However, the results seem to indicate that a combination of RIP features and heart rate features give the best performance, and that CNNs are most suitable to deal with these input sequences. We conclude that using deep learning is a viable method for creating a personalized power output estimation model based on RIP signals and heart rate.

5.2 Future work

We have in this thesis laid a foundation for using deep learning to estimate power output from RIP signals, and there are several possible ways this research could be taken further.

Diverse data sets

A continuation of this research is to gather a more diverse data set. This thesis has been focusing on data from a single subject, but a deeper understanding of the data might be gained by acquiring data from subjects of various ages, genders, heights, weights and physical fitness levels. Applying our methods to a diverse data set might give insights to whether generalized, non-personalized models are able to give accurate power output estimation.

Extending estimation of power and energy expenditure to other sports

The research in this thesis can easily be extended to other activity forms, especially indoor rowing and skiing on exercise ergometers. The web application we made for data acquisition is able to connect to rowing and skiing ergometers from Concept2, the same producer of the bike ergometer used in this thesis. This means that all source code, both for data acquisition and data processing, can be used without modifications on data from these two exercise machines. While power output is an interesting variable in terms of accurate estimations of energy expenditure and for competitive athletes, other people might be more interested in the number of calories burned during a workout. Many ergometers, including the Concept2 BikeErg used in our experiments, provide an estimation of the calories burned in real-time during a workout. It is trivial to change the target variable in our machine learning pipeline from power output to the rate of calories burned. The same procedures that we have demonstrated may then be used to provide another measure of energy expenditure in similar contexts.

Chapter 6

Appendix A: Testing models on data from other subjects

The predictive models in this thesis were trained on data collected from a male subject of age 25, whom we in this appendix will refer to as subject A. Some additional data were collected from two other subjects: One female subject of age 25 (subject B), and one male subject of age 62 (subject C). We did not acquire enough data from subject B and C to train personalized models on their data, but we used the data from those two subjects to evaluate our final models, which were trained on data from subject A. This gives us an indication on how the personalized models of subject A perform on data from other subjects.

The results from evaluating our models on data from subject B and C are presented in Table 6.1. Examples of power output estimation on a workout from the female subject B are presented in Figure 6.1, 6.2, 6.3, 6.4 and 6.5. While the error metrics in Table 6.1 show that we get significantly worse results by applying the personalized models to other subjects, the figures presenting example predictions of power output show that the models are still able to give decent approximations of the ground truth.

Table 6.1: Error metrics for final models when evaluted on a test data set collected from other subjects. The hyperparameters shown in Table 4.4 was used.

Network	Feature set	MSE	R^2	MAPE
DNN	3	0.014	-0.04	0.33
CNN	6	0.013	0.00	0.31
LSTM	3	0.012	0.06	0.32
CNN	10	0.013	0.03	0.30
DNN	11	0.011	0.12	0.32

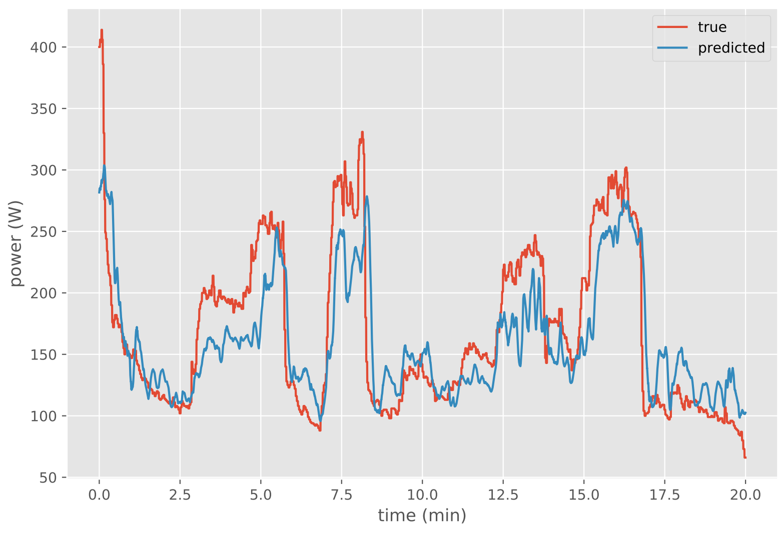


Figure 6.1: Prediction of power output on an interval workout from the test set, using DNN architecture and feature set 3.

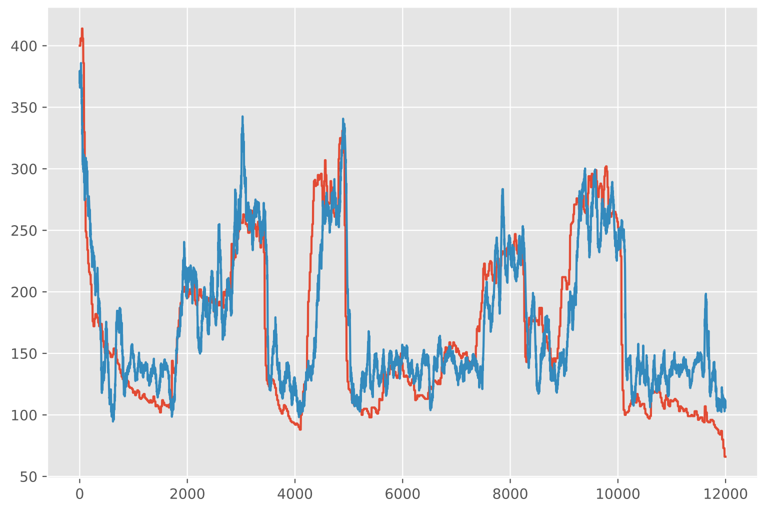


Figure 6.2: Prediction of power output on an interval workout from the test set, using CNN architecture and feature set 6.

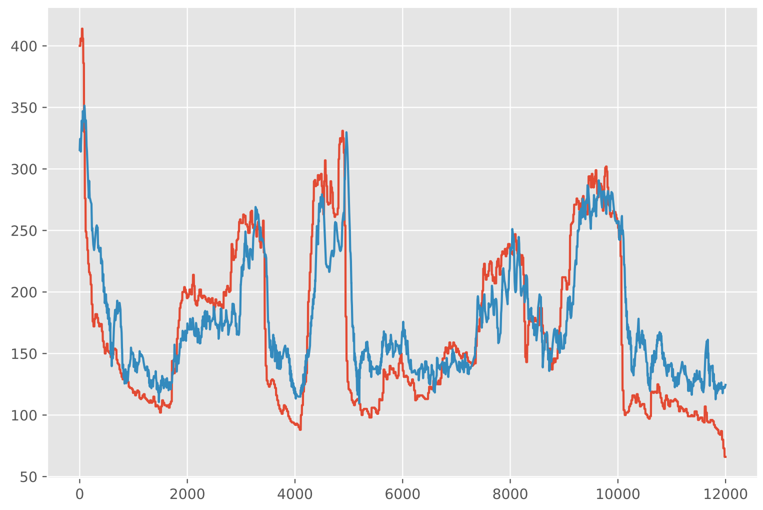


Figure 6.3: Prediction of power output on an interval workout from the test set, using LSTM architecture and feature set 3.

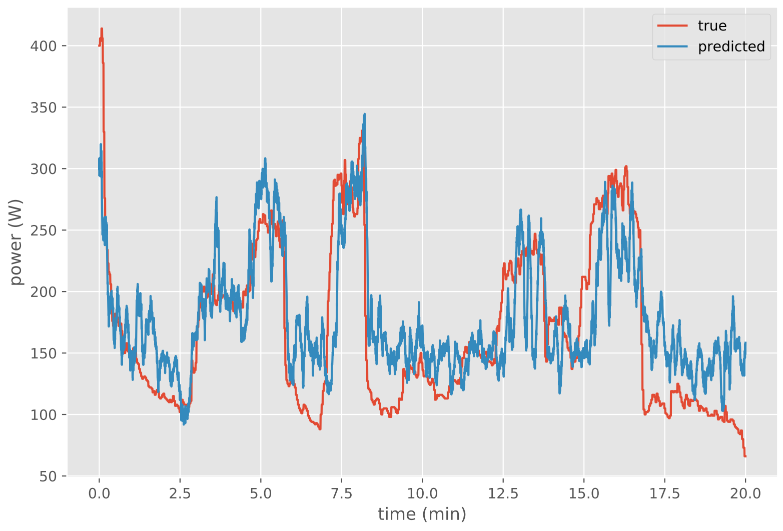


Figure 6.4: Prediction of power output on an interval workout from the test set, using CNN architecture and feature set 10.

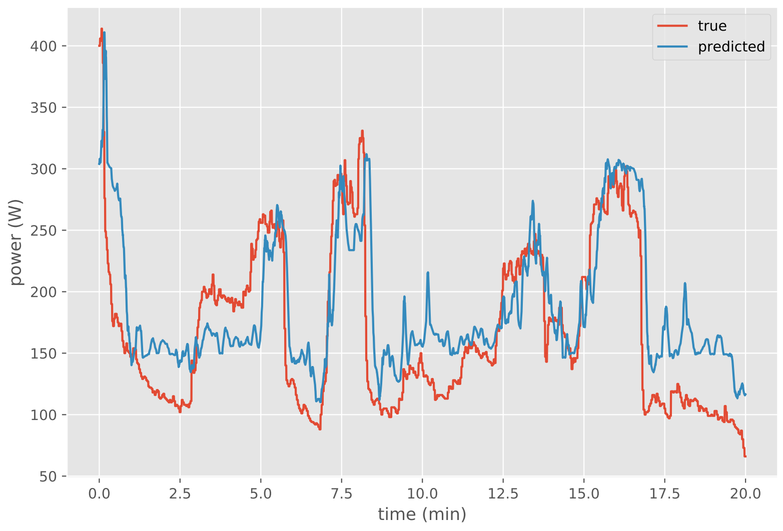


Figure 6.5: Prediction of power output on an interval workout from the test set, using DNN architecture and feature set 11.

Bibliography

[1] Ala Alwan et al. *Global status report on noncommunicable diseases 2010.* World Health Organization, 2011.

[2] World Health Organization et al. *Global health risks: mortality and burden of disease attributable to selected major risks.* World Health Organization, 2009.

[3] Michael Mcgrath and Cliodhna Ni Scanaill. *Sensor Technologies: Healthcare, Wellness, and Environmental Applications.* 2013.

[4] Akram Bayat, Marc Pomplun, and Duc A Tran. A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, 34:450–457, 2014.

[5] Tudor Pascu, Martin White, and Zeeshan Patoli. Motion capture and activity tracking using smartphone-driven body sensor networks. In *Third International Conference on Innovative Computing Technology (INTECH 2013)*, pages 456–462. IEEE, 2013.

[6] Jafet Morales, David Akopian, and Sos Agaian. Human activity recognition by smartphones regardless of device orientation. In *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, volume 9030, page 90300I. International Society for Optics and Photonics, 2014.

[7] Carl Foster, Jose A Rodriguez-Marroyo, and Jos J De Koning. Monitoring training loads: the past, the present, and the future. *International journal of sports physiology and performance*, 12(s2):S2–2, 2017.

[8] Raija M. T. Laukkanen and Paula K. Virtanen. Heart rate monitors: State of the art. *Journal of Sports Sciences*, 16(sup1):3–7, 1998. PMID: 22587712.

[9] Ryan T Li, Scott R Kling, Michael J Salata, Sean A Cupp, Joseph Sheehan, and James E Voos. Wearable performance devices in sports medicine. *Sports health*, 8(1):74–78, 2016.

[10] Edward F Coyle and J Gonzalez-Alonso. Cardiovascular drift during prolonged exercise: New perspectives. *Exercise and sport sciences reviews*, 29(2):88–92, 2001.

[11] Asker Jeukendrup and Adrie Van Diemen. Heart rate monitoring during training and competition in cyclists. *Journal of Sports Sciences*, 16(sup1):91–99, 1998. PMID: 22587722.

[12] Louis Passfield, James G Hopker, Simon Jobson, D Friel, and Mikel Zabala. Knowledge is power: Issues of measuring training and performance in cycling. *Journal of sports sciences*, 35(14):1426–1434, 2017.

[13] HKA Lakomy. Measurement of work and power output using friction-loaded cycle ergometers. *Ergonomics*, 29(4):509–517, 1986.

[14] Sébastien Boyas, Antoine Nordez, Christophe Cornu, and Arnaud Guével. Power responses of a rowing ergometer: Mechanical sensors vs. Concept2® measurement system. *International journal of sports medicine*, 27:830–3, 11 2006.

[15] Martin R Miller, Jats Hankinson, V Brusasco, F Burgos, R Casaburi, A Coates, R Crapo, Pvd Enright, CPM Van Der Grinten, P Gustafsson, et al. Standardisation of spirometry. *European respiratory journal*, 26(2):319–338, 2005.

[16] J Askanazi, PA Silverberg, RJ Foster, AI Hyman, J Milic-Emili, and JM Kinney. Effects of respiratory apparatus on breathing pattern. *Journal of Applied Physiology*, 48(4):577–580, 1980.

[17] Steven Gastinger, Anthony Sorel, Guillaume Nicolas, Arlette Gratas-Delamarche, and Jacques Prioux. A comparison between ventilation and heart rate as indicator of oxygen uptake during different intensities of exercise. *Journal of sports science & medicine*, 9(1):110, 2010.

[18] Steven Gastinger, Alan Donnelly, Rémy Dumond, and Jacques Prioux. A review of the evidence for the use of ventilation as a surrogate measure of energy expenditure. *Journal of Parenteral and Enteral Nutrition*, 38(8):926–938, 2014.

[19] Scott J Strath, Ann M Swartz, David R Bassett Jr, William L O'Brien, George A King, and Barbara E Ainsworth. Evaluation of heart rate as a method for assessing moderate intensity physical activity. *Medicine and science in sports and exercise*, 32(9 Suppl):S465–70, 2000.

[20] Sabino Padilla, Inigo Mujika, Javier Orbananos, and Francisco Angulo. Exercise intensity during competition time trials in professional road cycling. *Medicine and science in sports and exercise*, 32(4):850–856, 2000.

[21] Colm McParland, Joseph Mink, and Charles G Gallagher. Respiratory adaptations to dead space loading during maximal incremental exercise. *Journal of Applied Physiology*, 70(1):55–62, 1991.

[22] Nancy F Butte, Ulf Ekelund, and Klaas R Westerterp. Assessing physical activity using wearable monitors: measures of physical activity. *Medicine & Science in Sports & Exercise*, 44(1S):S5–S12, 2012.

[23] W Larry Kenney, Jack H Wilmore, and David L Costill. *Physiology of sport and exercise*. Human kinetics, 2015.

[24] Carl J Caspersen, Kenneth E Powell, and Gregory M Christenson. Physical activity, exercise, and physical fitness: Definitions and distinctions for health-related research. *Public health reports*, 100(2):126, 1985.

[25] James A Levine. Measurement of energy expenditure. *Public health nutrition*, 8(7a):1123–1132, 2005.

[26] Marie Chan, Daniel Estève, Jean-Yves Fourniols, Christophe Escriba, and Eric Campo. Smart wearable systems: Current status and future challenges. *Artificial intelligence in medicine*, 56(3):137–156, 2012.

[27] Nickfacey. Bottom bracket with connection cables. `https://en.wikipedia.org/wiki/Cycling_power_meter#/media/File:Ergomo_Bottom_Bracket.JPG`, 2008. [Used under Creative Commons Attribution-Share Alike 3.0 Unported; accessed May 5, 2021].

[28] Cosmed. Advanced six minute walk test with ventilation measurement. `https://commons.wikimedia.org/wiki/File:Advanced_Six_Minute_Walk_Test_(6MWT).jpg`, 2010. [Cropped from original; used under Creative Commons Attribution-Share Alike 3.0 Unported; accessed April 27, 2021].

[29] Juha Karvonen and Timo Vuorimaa. Heart rate and exercise intensity during sports activities. *Sports Medicine*, 5(5):303–312, May 1988.

[30] Muriel B Gilman. The use of heart rate to monitor the intensity of endurance training. *Sports Medicine*, 21(2):73–79, 1996.

[31] rrafson. Rowing machine. `https://commons.wikimedia.org/wiki/File:Advanced_Six_Minute_Walk_Test_(6MWT).jpg`, 2014. [Cropped from original; used under Creative Commons Attribution-Share Alike 3.0 Unported; accessed April 27, 2021].

[32] SH Cedar. Every breath you take: The process of breathing explained. *Nursing Times*, 114(1):47–50, 2018.

[33] N Scott Deno, E Kamon, and David M Kiser. Physiological responses to resistance breathing during short and prolonged exercise. *American Industrial Hygiene Association Journal*, 42(8):616–623, 1981.

[34] TS Chadha, H Watson, S Birch, GA Jenouri, AW Schneider, MA Cohn, and MA Sackner. Validation of respiratory inductive plethysmography using different calibration procedures. *American Review of Respiratory Disease*, 125(6):644–649, 1982.

[35] Jonathan D Sackner, Asa J Nixon, Brian Davis, Neal Atkins, and Marvin A Sackner. Non-invasive measurement of ventilation during exercise using a respiratory inductive plethysmograph. *American Review of Respiratory Disease*, 122(6):867–871, 1980.

[36] David M Caretti, Paul V Pullen, Leslie A Premo, and Wade D Kuhlmann. Reliability of respiratory inductive plethysmography for measuring tidal volume during exercise. *American Industrial Hygiene Association Journal*, 55(10):918–923, 1994.

[37] Arne Laugstøl. Ventilation measurement devices, methods and computer program product, March 24 2020. US Patent 10,595,779.

[38] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[40] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[42] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[45] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[46] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993.

[47] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[49] Adelchi Azzalini and Bruno Scarpa. *Data analysis and data mining: An introduction*. OUP USA, 2012.

[50] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[51] FJP Arts and H Kuipers. The relation between power output, oxygen uptake and heart rate in male athletes. *International journal of sports medicine*, 15(05):228–231, 1994.

[52] Vitor P Costa, Luiz GA Guglielmo, and Carl D Paton. Validity and reliability of the powercal device for estimating power output during cycling time trials. *The Journal of Strength & Conditioning Research*, 31(1):227–232, 2017.

[53] Agrin Hilmkil, Oscar Ivarsson, Moa Johansson, Dan Kuylenstierna, and Teun van Erp. Towards machine learning on data from professional cyclists, 2018.

[54] Mark Pfeiffer and Andreas Hohmann. Applications of neural networks in training science. *Human movement science*, 31(2):344–359, 2012.

[55] Sagar Sen, Pierre Bernabé, and Erik Johannes BLG Husom. Deepventilation: Learning to predict physical effort from breathing. 2020.

[56] S Boyas, A Nordez, C Cornu, and A Guével. Power responses of a rowing ergometer: mechanical sensors vs. Concept2® measurement system. *International journal of sports medicine*, 27(10):830–833, 2006.

[57] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng,

Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[58] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[59] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Keras Tuner. `https://github.com/keras-team/keras-tuner`, 2019.

[60] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[61] Web bluetooth specification. `https://webbluetoothcg.github.io/web-bluetooth/`. Accessed: March 30, 2021.

[62] Dean Fogarty. Ergarcade. `https://github.com/ergarcade/pm5-base`, 2019. Accessed: February 5, 2020.

[63] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[64] Ruslan Kuprieiev, Dmitry Petrov, Paweł Redzyński, Saugat Pachhai, Casper da Costa-Luis, Alexander Schepanovski, Peter Rowlands, Ivan Shcheklein, Jorge Orpinel, Fábio Santos, Aman Sharma, Zhanibek, Gao, Batuhan Taskaya, Dani Hodovic, Andrew Grigorev, Earl, Nabanita Dash, nik123, George Vyshnya, maykulkarni, Max Hora, Vera, Sanidhya Mangal, Wojciech Baranowski, Clemens Wolff, Alex Maslakov, Alex Khamutov,

Kurian Benoy, and Ophir Yoktan. Dvc: Data version control - git for data & models, February 2021.

[65] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[66] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[67] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[68] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pages 2796–2804. PMLR, 2018.

[69] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*, pages 1135–1144, 2016.