University of Padova

Department of Information Engineering

Ph.D. School in Information Engineering

Information Science and Technology

XXIX Class

Stochastic Optimization and Machine Learning Modeling for Wireless Networking

Ph.D. candidate

Davide Del Testa

Supervisor:
Prof. Michele Zorzi

Ph.D. school director:
Prof. Matteo Bertocco

Academic Year 2016/2017

ABSTRACT

In the last years, the telecommunications industry has seen an increasing interest in the development of advanced solutions that enable communicating nodes to exchange large amounts of data. Indeed, well-known applications such as VoIP, audio streaming, video on demand, real-time surveillance systems, safety vehicular requirements, and remote computing have increased the demand for the efficient generation, utilization, management and communication of larger and larger data quantities. New transmission technologies have been developed to permit more efficient and faster data exchanges, including multiple input multiple output architectures or software defined networking: as an example, the next generation of mobile communication, known as 5G, is expected to provide data rates of tens of megabits per second for tens of thousands of users and only 1 ms latency. In order to achieve such demanding performance, these systems need to effectively model the considerable level of uncertainty related to fading transmission channels, interference, or the presence of noise in the data.

In this thesis, we will present how different approaches can be adopted to model these kinds of scenarios, focusing on wireless networking applications. In particular, the first part of this work will show how stochastic optimization models can be exploited to design energy management policies for wireless sensor networks. Traditionally, transmission policies are designed to reduce the total amount of energy drawn from the batteries of the devices; here, we consider energy harvesting wireless sensor networks, in which each device is able to scavenge energy from the environment and charge its battery with it. In this case, the goal of the optimal transmission policies is to efficiently manage the energy harvested from the environment, avoiding both energy outage (*i.e.*, no residual energy in a battery) and energy overflow (*i.e.*, the impossibility to store scavenged energy when the battery is already full).

In the second part of this work, we will explore the adoption of machine learning techniques to tackle a number of common wireless networking problems. These al-

gorithms are able to learn from and make predictions on data, avoiding the need to follow limited static program instructions: models are built from sample inputs, thus allowing for data-driven predictions and decisions. In particular, we will first design an on-the-fly prediction algorithm for the expected time of arrival related to WiFi transmissions. This predictor only exploits those network parameters available at each receiving node and does not require additional knowledge from the transmitter, hence it can be deployed without modifying existing standard transmission protocols. Secondly, we will investigate the usage of particular neural network instances known as autoencoders for the compression of biosignals, such as electrocardiography and photo plethysmographic sequences. A lightweight lossy compressor will be designed, able to be deployed in wearable battery-equipped devices with limited computational power. Thirdly, we will propose a predictor for the long-term channel gain in a wireless network. Differently from other works in the literature, such predictor will only exploit past channel samples, without resorting to additional information such as GPS data. An accurate estimation of this gain would enable to, *e.g.*, efficiently allocate resources and foretell future handover procedures. Finally, although not strictly related to wireless networking scenarios, we will show how deep learning techniques can be applied to the field of autonomous driving. This final section will deal with state-of-the-art machine learning solutions, proving how these techniques are able to considerably overcome the performance given by traditional approaches.

CONTENTS

CHAPTER 1

INTRODUCTION

Recent advancements in telecommunication physical and MAC/DLL layers have made it possible to approach Shannon's channel capacity bound. As shown in Figure 1.1, the performance of technologies such as HSDPA or IEEE 802.16 (WiMax) are close to their theoretical limits, and, regarding the link layer, much research has already been conducted.

However, in these years, technological progresses have stimulated new research efforts focused on the network layer: examples include management of different transmitting units for massive Multiple Input Multiple Output (MIMO) architectures [1], collaboration among transmitting nodes belonging to the same network to prevent losses due to mutual interference [2,3], clustering techniques for Machine-to-Machine (M2M) communication [4], Software Defined Networking (SDN) technologies applied to reduce communication latency for the paradigm of Tactile Networking [5], efficient Quality of Service (QoS) driven management of mobile devices connected to a group of Base Stations (BSs) [6–8], as required by recent wireless communication protocols such as LTE. All the aforementioned scenarios require the ability to properly model the considerable level of uncertainty inherent in many telecommunications problems, *e.g.*, interference among devices, noisy communication channels, random data generation and dynamic cellular environments.

A traditional approach adopted to study this kind of scenarios is based on stochastic optimization, which allows a rigorous characterization of the mathematical details needed to take uncertainty into account. In recent years, stochastic optimization has been the methodology of choice for optimal decision support under uncertainty: the authors in [9] divide telecommunications problems into three classes, namely technological, network and enterprise, based on the scale of the decisions relative to the whole telecommunication environment. Indeed, different types of uncertainty come into play at different levels. Traditionally, in performance anal-

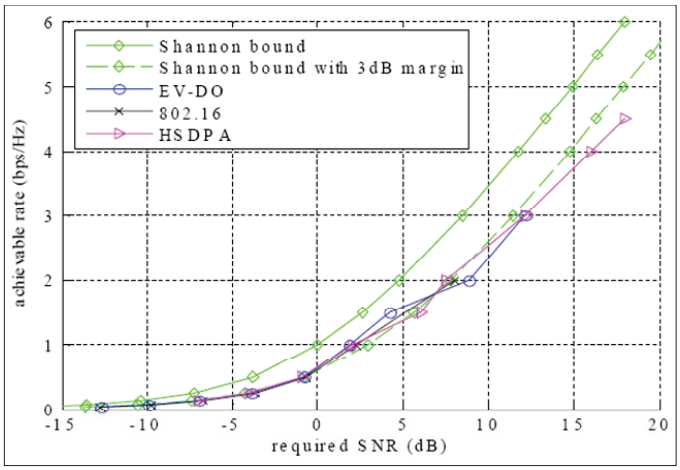


Figure 1.1: Performance relative to theoretical limits.

ysis two complementary approaches were developed to model randomness, namely analytical characterization and simulation. The former approach generally relies on Markov chains, *i.e.*, stochastic processes in which future and past states of the considered system are independent given the present state (Markov memoryless property). This approach allows for a mathematical characterization of the considered environment, but usually does not scale well when dealing with complex scenarios. In this case, a proper analysis of the environment requires a Markov chain with a huge amount of states, whose stationary distribution computation can be, for instance, too time consuming or computationally intensive. A further possibility is to complement the analytic approach by simulation techniques. These consist of direct simulations of the interaction between the considered variables, allowing a realistic representation of the whole system. However, the simulation times necessary to obtain reliable estimates of the stationary performance can be prohibitively long as well. This is especially true, for instance, when design requirements are expressed in terms of low probability of packet loss, which is a popular performance measure for modern data networks.

Machine learning techniques are now facing increasing popularity, being able to build models from input examples in order to make data-driven predictions, without the need of pre-programmed and rule-based sets of precise instructions. As such, compared to stochastic models, they are able to manage big amounts of data more easily, by learning to perceive and comprehend the significance of the data with which they are trained. Differently from rule-based techniques (*e.g.*, stochastic models), where it is usually possible to both gain an understanding and check how a system achieves its solution (thus verifying that this system will operate within certain reference parameters), it is typically hard to extract meaning from the variables of machine learning algorithms, *e.g.*, the weights of an Artificial Neural

Network (ANN), also because, during training, such systems can even give highly different outputs in consecutive iterations of the learning process. Nevertheless, machine learning methods have already been applied to many wireless communication fields, including Wireless Sensor Networks (WSNs): routing [10, 11], data aggregation [12, 13], event detection [14, 15] and medium access control [16, 17] are, among others, a few applications where learning algorithms have proved to be competitive with traditional approaches.

This thesis deals with the optimization of wireless networking protocols by means of both the aforementioned approaches. In particular, the first part describes stochastic models, based on Markov decision processes, able to compute efficient transmission policies for a particular category of wireless sensor networks. After that, the second part of this work applies machine learning techniques to a number of networking problems, namely network parameters estimation, radio channel gain prediction and signal compression, showing how such methods provide interesting performance improvements with respect to traditional approaches.

In detail, this work is subdivided into two parts. In the first part we use stochastic models to design energy management polices for wireless sensor networks equipped with energy harvesting devices. In particular, in Chapter 2, we focus on networks composed of two receivers and a central controller, computing optimal transmission policies dealing with finite battery capacities, correlations in the harvesting processes, and non-deterministic packet arrivals at the receivers. In addition, we take into account non-idealities, such as imperfect knowledge of the states of charge of the devices, or secondary costs. This work has been reported in the journal paper [J1] and in the conference paper [C2]. Although related to these research topics, conference papers [C1] and [C3] have not been included in this thesis, as they involve a slightly different mathematical model with respect to the one presented in [J1, C2], while studying a very similar problem.

In the second part, we address a number of typical wireless networking research topics by means of machine learning techniques. Specifically, this part is subdivided into four chapters, each addressing a specific topic and the corresponding results: each chapter can thus be read separately. Chapter 3, based on journal paper [J4] and conference papers [C4, C5], shows how receiving nodes in a WiFi network can perform on-the-fly prediction of an important quality of service measure, namely the estimated time of arrival, only based on a subset of networking parameters (transmission power, distance, ..) available at each node without modifying standard transmission protocols. Chapter 4 presents a particular ML-based compression algorithm for quasi-periodic biosignals such as ECG, plethysmographic and respiratory. The proposed technique is compared to traditional compression approaches, showing a thorough performance comparison in terms of reconstruction

error, compression efficiency and energy consumption for both coding and decoding. This work is based on journal papers [J2, J3]. In Chapter 5, we investigate how to predict the long-term channel gain of a radio link in an urban scenario, simulating both a pedestrian and a vehicular environment, only exploiting a limited amount of information and with no geographic knowledge about the user. This work is based on the conference paper [C6]. Finally, Chapter 6 presents how advanced machine learning algorithms are being applied to the field of autonomous driving. The material presented in this chapter comes from the conference paper [C7] and describes the research I took part in during my internship with the autonomous driving team at Nvidia in Holmdel, NJ, USA. Due to a non disclosure agreement, this chapter will intentionally give only a general overview of the project.

Part I

Stochastic Models for Energy Harvesting Sensor Networks

CHAPTER 2

TRANSMISSION POLICIES FOR TWO-USER ENERGY HARVESTING WIRELESS NETWORKS

This chapter considers a wireless network composed by a pair of sensors powered by Energy Harvesting Devices (EHDs), which transmit data to a receiver over a shared wireless channel. At any given time, based on the energy levels of the two rechargeable batteries of the sensors, a central controller (CC) decides on the amount of energy to be drawn from the two batteries and used for transmission. The problem considered is the maximization of the long-term average reward associated with data transmission, by optimizing the transmission strategy of the two nodes, in the case of a collision channel model and both i.i.d. and correlated energy arrivals. In addition, contrary to the traditional assumption that the amount of energy available to the sensors can be easily estimated, we derive the optimal policy in the cases where the State Of Charge (SOC) may not be perfectly known by the central controller, analyzing the performance degradation caused by this imperfect knowledge of the SOC. For this second scenario, supposing that the CC is only aware that each SOC is "LOW" or "HIGH," we show that the impact of imperfect knowledge decreases with the two battery capacities and is negligible in most cases of practical interest.

2.1 Introduction to the chapter

Recent advancements in the areas of micro-electro-mechanical systems technology, digital electronics and wireless communications have made it possible to develop multifunctional sensor nodes, capable of communicating over short distances. The possibility of producing such sensors in a low-power, low-cost and small-size way, leverages the idea of sensor networks based on collaborative efforts of a large number of nodes. These small devices, also named motes, are able to sense physical quantities

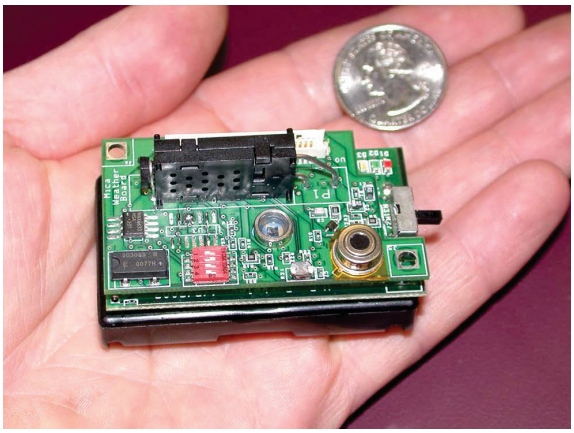


Figure 2.1: A practical sensor node

(position, temperature, humidity, etc), to process data and to communicate to each other (see Figure 2.1). The sensor nodes constituting a network can be deployed very close to the phenomenon and their position does not need to be predetermined, thus allowing random deployment in inaccessible or disaster areas [18].

The communication is usually performed in an asymmetric way: nodes send the data to one or more special nodes, called *sink* (or base station), whose aim is to collect data. The base station is a component of the WSN with much more computational power, memory, energy and communication resources, which acts as a gateway between sensor nodes and the end user, typically forwarding data from the WSN on to a server. A transmission can be initialized autonomously by the sensor (if a certain event occurs) or by the sink (by sending a *query* towards a specified node). On the other hand, it is possible to exploit the great density of nodes performing multihop communication, which is able to consume less power than the traditional single hop strategy. In addition, this technique can lower the transmission power levels (highly desirable in most scenarios) and overcome some of the signal propagation effects experienced in long-distance wireless communications. One of the most important constraints on sensor nodes is the low-energy consumption requirement: a node generally carries a limited and irreplaceable power source. As a result, while traditional wired networks aim to meet high quality of service constraints, sensor network protocols have to focus primarily on power conservation: a permanent trade-off between network lifetime and transmission throughput has to be taken into account.

2.1.1 Applications

A sensor network can be seen as a set of different types of sensors able to sense a wide variety of ambient conditions like pressure, humidity, temperature, lighting, noise

but also vehicular movement, soil makeup, mechanical stress on attached objects and also the presence or absence of certain kinds of objects [19]. Moreover, nodes can be used for event detection, continuous sensing or local control of actuators useful in many application areas.

1. **Environmental**
   The term environmental sensor network has evolved to cover many applications of WSNs to earth science research [20]. Some of the major fields are listed below.

   - Air quality monitoring: in urban areas the degree of pollution of the air needs to be frequently measured in order to protect people from any kind of damage due to air pollution.
   - Natural disaster prevention: wireless sensor networks can prevent the consequences of natural disasters like earthquakes or floods. For instance, wireless nodes have successfully been installed in rivers to monitor the water levels in real time.
   - Forest fire detection: a network of sensor nodes can be deployed in a forest to detect when a fire has started. Nodes can be capable of measuring temperature, gases and humidity produced by fires among the vegetation. Thanks to WSNs, fire fighters would be able to early detect a fire and track its spreading.
   - Water quality monitoring: water properties in rivers, lakes, dams and oceans, as well as underground reserves [21], can be monitored avoiding manual data retrieval in difficult-access locations.

2. **Military**
   The possibility to easily and rapidly spreading as well as self-organization and tolerance to damages make a sensor network a promising technique towards military application. Since sensor networks are based on dense deployment of low cost and disposable nodes, destruction of some nodes by enemy actions does not damage military operations as much as the destruction of a traditional sensor. A possible application is the monitoring of friendly forces, equipment and ammunition: every troop, vehicle and critical device could be attached with a small sensor to report its status. These reports could be gathered in sink nodes and sent to the troop leader or directly forwarded to the upper level of the hierarchy together with the data from the other units. Different applications are battlefield surveillance, battle damage assessment as well as nuclear, biological and chemical attack detection.

3. **Health**
   Some applications in this field are addressed to provide an interface for disabled people, to monitor human physiological data, to administer drugs in hospitals (avoiding the chance to prescribe the wrong medications to patients) and also to track doctors and patients inside hospitals [18, 22].

4. **Home**
   Sensor nodes inside the domestic appliances can interact with each other and with the external network by an Internet point. This could allow users to easily manage home appliances locally or remotely.

5. **Others**
   Some other commercial applications are virtual keyboards, interactive toys and museums, factory process control and automation, robot control and guidance in automatic manufacturing environments. Finally, also local control of actuators and vehicle tracking and detection are today available.

2.1.2 Harvesting capabilities

Energy management is one of the main issues in WSN because it critically threats their sustainability. Since the nodes may be distributed in extensively wide and complex environments, it becomes very difficult to replace the battery: various studies have been performed to increase the lifetime of the battery of a node by choosing the best modulation strategy [23], by exploiting power saving modes (sleep/listen) periodically [24], by reducing the number of bits to transmit [25,26], by using energy efficient routing [27,28] and MAC [29] and by using efficient transmission scheduling to take advantage of charge recovery phenomenon [30].

In addition to these, another very interesting strategy is that of exploiting Energy Harvesting techniques [31], by which ambient energy is captured and, if necessary, stored to provide electricity for small autonomous devices, such as satellites, mobile phones or nodes in sensor networks. EH is used for many reasons, from providing long life and no maintenance to saving costs. A list of some scavenging sources is presented below.

Types of scavenging

There are many free energy sources in nature [32, 33]: how to harvest and store this energy efficiently in small devices is still an open topic for research.

- **Solar** The basic principle of optical collection is to absorb a large number of photons by the use of photovoltaic materials. The main disadvantage of this energy source is the great dependence on time and on solar environment

exposure. Indeed during night and cloudy days sufficient incoming energy cannot be guaranteed.

- **Thermal** Thermoelectric scavenging exploiting the differences of temperature is nowadays a very well known technology. Devices of this type can be small, light and are able to work in harsh environments.

- **Motion** If nodes are subject to movements, oscillations and vibrations, energy could be scavenged according to Faraday's law of electromagnetic induction. The main advantage of this source is that, in some particular scenarios, it could provide constant energy.

- **Electromagnetic** When a node is exposed to an electromagnetic field, energy can be drawn with the use of an inductor. Manos Tentzeris, a professor in the Georgia Tech School of Electrical and Computer Engineering, and his team state: "There is a large amount of electromagnetic energy all around us, but nobody has been able to tap into it. We are using an ultra-wideband antenna that lets us exploit a variety of signals in different frequency ranges, giving us greatly increased power-gathering capability" [34]. It is believed that the technique could provide a promising new way to power wireless sensors networks.

Thanks to EH, energy conservation does not need to be the most important concern any more, and energy efficient policies and harvesting techniques can be jointly developed. However, some issues related to scavenging are still present: harvested energy could not always be available (for example with solar cells), despite sensor nodes' constant needs, or energy generation rates could be limited and hence the energy generation profile of the harvesting source should be matched with the energy consumption profile of the sensor node, in a way avoiding energy starvation from being the main reason for the node to die (*energy neutral operation* [31]). Furthermore, it must be noted that, treating energy arrivals as a random process, the way in which harvesting occurs is random and barely predictable.

Paradigms for energy efficient operations

In conclusion, differently from traditional sensors, where the objective is to minimize energy consumption under a performance constraint (for example the delay [35]), with Energy Harvesting Devices, which are utilized in this thesis, the objective is the "management" of the harvested energy. Intuitively, when a finite battery is available, an EHD should judiciously perform its assigned task based on its available energy, becoming more "conservative" as its energy supply runs low to ensure uninterrupted

operation, and more "aggressive" when energy is abundant, to avoid that harvested energy is wasted due to lack of storage space.

2.1.3 Node architecture

The overall architecture of a sensor node consists in five basic components as shown in Figure 2.2:

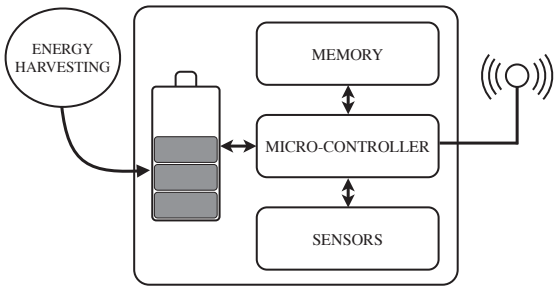


Figure 2.2: Architecture of a wireless sensor node

- The **power supply** unit of the sensor node provides power to all its components. In the majority of cases it consists of a rechargeable DC battery and can be supported by a power scavenging unit such as solar cells.
- **Micro controller** is responsible for all processing and decision making.
- **Sensing units** monitor the surrounding environment and inform the controller about what is being observed. For example, they can sense light, temperature, humidity, pressure. Sensing units are usually composed by two subunits: a sensor and an analog to digital converter (ADC). The analog signal produced by the sensor is converted to digital by the ADC, and then passed to the processing unit.
- The **Transceiver** deals with transmission and reception of the data to and from the base station. Usually RF based communication is preferred, as Infrared or Laser technologies need a direct propagation path for a correct communication.
- Sensor nodes are equipped with a programmable Flash **memory**. Usually storage capacity is limited, so the protocols that are designed for sensor networks should be simple enough to be loaded into the small available memory.
- There can also be some applications-dependent additional components, such as a **location finding system** (GPS).

2.1.4 Contributions and related work

Recent work related to harvesting capabilities includes [36], which analyzes the packet dropping and packet transmission probabilities for a single node powered by solar cells, and [37], which focuses on the problem of cross-layer resource allocation for wireless networks operating with rechargeable batteries under general arrivals, proposing a policy that achieves asymptotically optimal transmission rate for sufficiently large battery capacity. [31] derived energy management policies that are either throughput optimal, *i.e.*, policies for which the data queue of packets to be transmitted remains stable for the largest possible data rate, or delay minimizing, also identifying a greedy policy able to satisfy both throughput and delay constraints.

An EHD is typically modeled as an energy buffer, which is supplied by an energy arrival process with some statistical distribution. Here, we consider a WSN composed by two EHDs, which report incoming data to a Receiver (RX), also acting as a Central Controller (CC). This case can be described with a relatively simple notation, and at the same time captures all conceptual difficulties of a multi-user system, introducing a number of new observations with respect to a single-user scenario. These include coordination between users, collision avoidance, issues related to symmetry/asymmetry, and scenarios characterized by unbalanced EHD equipments, *e.g.*, different battery sizes, harvesting rates or transmission channel conditions. Also, it shows that even in the simplest multi-user network, the increased dimensionality does not allow to derive analytical results similar to the ones obtainable in the single-user scenario, unless trivial cases are considered. Furthermore, it is worth noting that the extension from a single-user network to the two-user case involves significant conceptual difficulties (*e.g.*, multidimensional model, different policy structure) as well as some completely new issues that do not arise in the single-user case (*e.g.*, need to manage the collisions, effect of asymmetric assumptions). On the other hand, further extension from a two-user network to a bigger multi-user system is conceptually similar, and its difficulty lies primarily in a more involved notation and a higher complexity due to the exponential scaling of the state space. For this reason, and for ease of presentation, we will develop our analysis and results for the two-user case, highlighting its similarities and differences with respect to the single-user case, and then discuss how the framework can be extended to the general multi-user case.

The transmissions of the two devices are managed based on the energy level of the two batteries and the mean energy harvesting rate. Intuitively, an EHD should be more "conservative" when the energy supply is low, and more "aggressive" if energy is abundant. As a consequence, the State-of-Charge (SOC) of a device

is an important piece of information, and is generally assumed to be known in most of the literature. However, as practical EHDs store energy in electrochemical batteries and/or super-capacitors, it is not realistic to assume that the SOC can be characterized with infinite precision and immediate availability at any time. In [38], it is shown that the value of a super-capacitor capacitance can fluctuate up to 30% with respect to the data-sheet value, due to temperature variations or age: an online estimation algorithm, based on controller discharge, was proposed, but was shown to have a significant additional energy cost. [39, 40] propose other estimation algorithms for the open circuit voltage of an electro-chemical battery, which is closely related to the SOC, but only by means of a non-negligible complexity, which may be unsustainable for computationally-limited devices. Hence, the estimation of the SOC related to sensor batteries is complex and highly prone to errors, so that a precise knowledge of this value cannot be easily acquired in general. Moreover, another possible cause of imperfect knowledge of the SOC by the CC is that, in limited energy systems, the information related to the SOC may not be transmitted in real time, but only when substantial variations occur, so as to reduce the transmission overhead of the overall system. Consequently, we also study the case of a scenario in which the energy levels of the two EHDs are not precisely known by the CC, which can only distinguish a region the SOC belongs to at any given time. In this way, it will be possible to compare the performance achieved under perfect and imperfect SOC knowledge, thus quantifying the degradation caused by the latter (more realistic) scenario with respect to the former.

The objective of our study is to determine the optimal amount of energy to be drawn from the batteries, so as to maximize a long-term expected reward, *e.g.*, the throughput of the system. In the special case of a logarithmic reward function, and a CC only knowing if the SOC of each device is LOW or HIGH, it is shown that the performance penalty due to imperfect SOC knowledge strongly decreases with the capacity of the two batteries, and is typically less than 5%. This can be explained by noting that the optimal policy must avoid *energy outage* (*i.e.*, depleting the batteries) when the SOC is LOW, and must be aggressive to prevent *energy overflow* (*i.e.*, wasting harvested energy due to the limited battery capacities) if the SOC is HIGH [41]. With these expedients in place, the exact SOC knowledge may not be as important, at least for batteries of sufficient size.

The analysis of battery imperfections for a single EHD has been carried out in [42, 43], which present optimal transmission policies accounting for battery leakage, and [44] which evaluates the impact on the performance of battery degradation over time, depending on partial discharge depths. Moreover, for a WSN with just a single EHD, [45] derives transmission policies that are independent of the SOC, showing that at most 3% performance degradation with respect to the optimum

is achieved, [46] considers imperfections of the energy buffer and of the EH circuit power consumption, for a framework with wireless power transfer harvesting capabilities, and [41] investigates optimal policies for a scenario similar to the one presented in this work, namely the analysis of optimal policies under imperfect knowledge of the SOC. As to multi-user WSNs equipped with EHDs, [47] investigates the possibility of energy sharing among the nodes: it is proved that jointly managing the energy queues of the sensors provides an increasing overall system throughput. [48] considers a two-user multiple access channel, where each transmitter node has both a data and an energy queue, with the objective of adaptively changing the transmission power. However, [48] derives an off-line transmission policy by assuming that the data packets and energy arrive in a deterministic fashion. Finally, [49] takes into account a WSN of U EHDs where each sensor node, depending only on its own energy level and on an importance value associated to its own data packet, decides whether to transmit the packet or remain idle. Differently from the model used here, [49] designs a distributed transmission scheme, rather than a centralized one.

Finally, we investigate the impact of correlation in the energy arrival mechanism of the two devices. Modeling this phenomenon as a hidden Markov chain, we distinguish a set of possible energy arrival scenarios, characterized by different energy harvesting properties. We thus provide both temporal (between subsequent time instants) and spatial (between the two sensors) correlation, and show how performance varies accordingly. In this work, we investigate the cases of both i.i.d. and correlated EH arrival processes, as the former is suitable for modeling, *e.g.*, radio-frequency/piezo-electric scavenging, whereas the latter can be adopted for solar harvesting.

The remainder of this chapter is organized as follows. Section 2.2 presents the system model and the main assumptions. The optimization problem, for both the perfect and the imperfect knowledge scenario, as well as the formal policy definitions are described in Section 2.3. Section 2.4 contains the numerical results, obtained when the SOC of the two EHDs is known by the CC perfectly or only approximately. Finally, Section 2.5 presents some possible extensions, whereas Section 2.6 draws the conclusions.

2.2 System model

In this section, we discuss the system model as well as the assumptions made to mathematically represent its parameters. Specifically, the following subsections present the framework with respect to battery dynamics, scenario process, packet arrival probability and imperfect SOC knowledge, respectively.

2.2.1 Battery dynamics

We consider a WSN consisting of two EHDs, able to collect energy from the environment (*i.e.*, solar, aeolian, heat). Time is slotted, and slot $k \in \mathbb{N}_0$ corresponds to the time interval $[k, k+1)$. The energy scavenged from the environment is stored in a rechargeable battery, modeled as a buffer with capacity $e_{max,i}$, where $i = 1, 2$ denotes the EHD index. For simplicity, as in previous work [50, 51], each position in the buffer is assumed to hold one energy quantum, whose absolute value depends on the application-specific scenario. The amount of energy quanta available at time k at EHD i, *i.e.*, the SOC, denoted as $E_{i,k}$, takes values in the set $\mathcal{E}_i = \{0, 1, \ldots, e_{max,i}\}$ and, assuming that secondary costs like processing/sensing are negligible, is governed by the following equation

$$E_{i,k+1} = \min\{[E_{i,k} - Q_{i,k}]^+ + B_{i,k}, e_{max,i}\}, \tag{2.1}$$

where $[\cdot]^+ \triangleq \max\{\cdot, 0\}$. In (2.1), $Q_{i,k}$ represents the number of energy quanta to be drawn from i's buffer and devoted to transmission, and is chosen from the action space $\mathcal{Q}_i = \{0\} \cup \{q_{min,i}, q_{min,i} + 1, \ldots, q_{max,i}\}$ related to EHD i, for some $0 < q_{min,i} \leq q_{max,i} \leq e_{max,i}$, so that $Q_{i,k} \in \mathcal{Q}_i$, $\forall k$. $q_{min,i}$ represents the minimum load requirement, which can involve fixed energy costs or power needed to perform data acquisition and transmission [52,53]. At the beginning of the k-th time slot, the CC requests $Q_{1,k}$ and $Q_{2,k}$ quanta for data transmission from EHD 1 and 2, respectively, where $Q_{1,k}$ and $Q_{2,k}$ cannot be simultaneously positive since the centralized controller always avoids collisions. Therefore, the joint energy requested from the CC to the couple (EHD1, EHD2) at time k is $\mathbf{Q}_k = (Q_{1,k}, Q_{2,k})$, with $\mathbf{Q}_k \in \mathcal{Q}$ and:

$$\begin{aligned}\mathcal{Q} =& \ (\{0\} \times \mathcal{Q}_2) \cup (\mathcal{Q}_1 \times \{0\}) \\ =& \ \{(0,0), (0, q_{min,2}), \ldots, (0, q_{max,2}), (q_{min,1}, 0), \ldots, (q_{max,1}, 0)\}.\end{aligned}$$

When $\mathbf{Q}_k = (0, 0)$, both EHDs remain idle in slot k, due to the lack of available energy or a decision by the CC; also, the parameter $q_{max,i}$ reflects a physical constraint on the maximum amount of energy that can be drawn from the battery at any given time. Finally, $B_{i,k}$ models the amount of energy harvested from the environment by EHD i in time slot k (a more detailed description can be found in the next subsection). Note that the discrete variables $E_{i,k}$, $Q_{i,k}$, $B_{i,k}$ and $e_{max,i}$ are expressed as multiples of the energy quantum Δc, such that, *e.g.*, $e_{max,i} = \frac{C_{max,i}}{\Delta c}$, for a given value $C_{max,i}$ of the physical nominal capacity (expressed in Joule) of EHD i's battery. Although these variables are actually continuous (they represent physical energy quantities), this is not a limiting assumption since it is always possible to achieve a more accurate representation by reducing the value of the energy quantum

and, consequently, that of the quantization error, though at the price of a bigger state space and increased complexity.

2.2.2 Scenario process

$B_{i,k}$ represents the *energy arrival process*, which, as already explained, models the energy quanta scavenged from the environment during time slot k. We assume an underlying *scenario process* S_k, taking values in the finite set $\mathcal{S}$, governed by a stationary irreducible Markov chain with transition probability $p_{\mathcal{S}}(s_{k+1}|s_k) \triangleq \mathbb{P}(S_{k+1} = s_{k+1}|S_k = s_k)$. Given the scenario $S_k = s$, each element of the pair of energy harvests $(B_{1,k}, B_{2,k})$ is drawn with probability mass function $p_{B_i}(b|s) \triangleq \mathbb{P}(B_{i,k} = b|S_k = s)$, for all $b \in \mathcal{B}_i = \{0, 1, \ldots, b_{max,i}\}$, $s \in \mathcal{S}$, modeling the randomness in the energy harvesting mechanism, *e.g.*, due to an erratic energy supply. Note that, thanks to the common scenario process S_k, the energy arrival processes $B_{1,k}$ and $B_{2,k}$ of the two devices have both a temporal (between subsequent time slots) and a spatial (one sensor with the other) correlation. The i.i.d. scenario, where $p_{B_i}(b|s) = p_{B_i}(b) = \mathbb{P}(B_{i,k} = b)$, and $B_{i,k}$ takes values in $\mathcal{B}_i$, is a particular case of this generalized framework. We define the *average EH rate* for sensor i as $\bar{b}_i = \mathbb{E}[B_{i,k}] = \sum_{s\in\mathcal{S}} \pi_S(s) \sum_{b\in\mathcal{B}_i} b p_{B_i}(b|s)$ (with $\pi_S(s)$ being the steady-state probabilities of the scenario states), and assume that a new energy quantum harvested in slot k can only be used at a later time instant $> k$. The practicality of this model was studied more generally for the single user case in [54], where the scenario process is described as a first-order Markov chain, and B_k depends on both S_k and $B_{k-L}^{k-1} = (B_{k-L}, B_{k-L+1}, \ldots, B_{k-2}, B_{k-1})$, for some order $L \geq 0$. Assuming a quantization of B_k with 20 states, [54] shows that $L = 0$ models well a piezo-electric energy source, whereas a solar energy source is better modeled with $L = 1$. In this work we assume $L = 0$ to reduce the complexity, and leave the case $L = 1$ as a future research effort.

The joint mechanism of energy harvesting and consumption described in (2.1) entails the following two important phenomena: *energy outage* and *energy overflow*.

Definition 1. *Energy outage occurs, for EHD i, when $Q_{i,k} > E_{i,k}$, since the node runs out of energy before the completion of the task being executed.*

Definition 2. *Energy overflow occurs if $B_{i,k} > e_{max,i} - [E_{i,k} - Q_{i,k}]^+$,* i.e.*, the energy buffer is unable to store all of the harvested energy $B_{i,k}$, and the part exceeding the battery capacity is lost.*

When the former occurs, the requested task fails and the energy available in the battery is depleted, whereas the latter prevents part of the incoming energy form being successfully stored in the battery, due to its finite storage capacity, resulting

in a loss of $B_{i,k} - e_{max,i} + [E_{i,k} - Q_{i,k}]^+$ quanta. As a result, since energy is lost, energy overflow potentially represents a lost transmission opportunity in the future.

2.2.3 Packet arrival probability

We assume that packet arrivals are i.i.d. in time and between sensors: this could be the case for EHDs sensing a pair of distinct physical quantities, whose values are either higher or lower than a given respective threshold. To take into account the randomness of arrivals, a packet arrives in each slot at node i with probability $p_{pck,i}$, $i = 1, 2$. As a result, if at time slot k no packet arrives at both sensors with probability $(1 - p_{pck,1})(1 - p_{pck,2})$, the only possible action is $\mathbf{Q}_k = (0, 0)$, whereas if a packet arrives at node 1 or 2 only, with probability $p_{pck,1}(1-p_{pck,2})$ and $(1-p_{pck,1})p_{pck,2}$, respectively, the optimal action will belong to the set $(\mathcal{Q}_1 \times \{0\})$ or $(\{0\} \times \mathcal{Q}_2)$. Clearly, in the case of a pair of packets arriving to the two devices, with probability $p_{pck,1}p_{pck,2}$, the action space will be $\mathcal{Q}$. Note that this extension of the case $p_{pck,1} = p_{pck,2} = 1$, allows to model a scenario in which the EHDs sense data at different rates, depending on the pair of sources, *i.e.*, simultaneously acquiring data from fast-varying and slow-varying entities. We assume that the CC knows whether a node has a packet to send, and decides the transmitting node accordingly. In addition, considering a delay-sensitive WSN with stringent delay requirements, we assume that no buffering of packets is available, thus packets are either immediately sent or discarded.

2.2.4 Imperfect SOC knowledge

When only partial knowledge of the SOC $E_{i,k}$ is available at the controller, *e.g.*, due to inaccuracy in its estimation, we model this uncertainty by partitioning the state space $\mathcal{E}_i$ in $\{\mathcal{I}_i(n_i),\ n_i = 0, \ldots, \tilde{n}_i - 1\}$, with $\mathcal{I}_i(n_i) = \{\tilde{e}_{n_i}, \ldots, \tilde{e}_{n_i+1} - 1\}, n_i \in \{0, \ldots, \tilde{n}_i - 1\}$, $0 = \tilde{e}_0 < \tilde{e}_1 < \cdots < \tilde{e}_{\tilde{n}_i} = e_{max,i} + 1$. At each time slot k, we have that $E_{i,k} \in \mathcal{I}_i(N_{i,k})$, for some $N_{i,k} \in \{0, \ldots, \tilde{n}_i - 1\}$, *i.e.*, the controller knows only the interval index $N_{i,k}$ rather than the exact SOC $E_{i,k}$. Consequently, $\{N_{i,k},\ k \geq 0\}$ is defined as the *interval index process* for EHD i, taking values in $\{0, \ldots, \tilde{n}_i - 1\}$. In particular, the case with perfect SOC knowledge is obtained when $\tilde{n}_i = e_{max,i}+1$, for which $E_{i,k} = N_{i,k}$.

2.3 Optimization problem

2.3.1 Policy definition and general optimization problem

Generally, a *controller policy* μ is a function deciding, at every time slot k, on the amount of energy quanta $Q_{i,k}$ to be drawn from the batteries of the EHDs and used for transmission. This decision is made according to S_{k-1} and either to the exact value of $E_{i,k}$ or to the interval index $N_{i,k}$ and the past history $\mathcal{H}_{i,k} = \{N_{i,0}, \ldots, N_{i,k-1}, Q_{i,0}, \ldots, Q_{i,k-1}, S_{-1}, \ldots, S_{k-2}, \text{ past outage events}\}$, for $i = 1, 2$, in the cases of perfect and imperfect SOC knowledge, respectively. In particular, μ is a probability measure on the action space $\mathcal{Q}$, parameterized by the state $\mathbf{W}_k = (S_{k-1}, E_{1,k}, E_{2,k})$ or $\mathbf{W}_k = (S_{k-1}, N_{1,k}, N_{2,k}, \mathcal{H}_{1,k}, \mathcal{H}_{2,k})$: given $\mathbf{W}_k$, $\mu(\mathbf{q}; \mathbf{W}_k)$ is the conditional probability of choosing action $\mathbf{Q}_k = \mathbf{q} = (q_1, q_2) \in \mathcal{Q}$ in slot k.

We define the reward function $g : \mathcal{Q} \times \mathcal{E}_1 \times \mathcal{E}_2 \mapsto \mathbb{R}^+$ as

$$g(\mathbf{Q}_k, \mathbf{E}_k) = \begin{cases} 0 & Q_{1,k} > E_{1,k} \text{ or } Q_{2,k} > E_{2,k} \\ \tilde{g}(\mathbf{Q}_k) & Q_{1,k} \leq E_{1,k} \text{ and } Q_{2,k} \leq E_{2,k} \end{cases} \tag{2.2}$$

where $\tilde{g} : \mathcal{Q} \mapsto \mathbb{R}^+$ is a concave increasing function of $Q_{1,k}$ and $Q_{2,k}$, with $\tilde{g}(0,0) = 0$. $g(\mathbf{Q}_k, \mathbf{E}_k)$ is the reward accrued in slot k when the SOC level is $(E_{1,k}, E_{2,k}) \in \mathcal{E}_1 \times \mathcal{E}_2$, and the action $\mathbf{Q}_k$ is chosen. It can be seen that $g(\mathbf{Q}_k, \mathbf{E}_k) = 0$ when $Q_{i,k} > E_{i,k}$, *i.e.*, when energy outage occurs, modeling the inability of the sensor to complete its assigned task. For example, assuming the reward function as the transmission rate, then Shannon's formula gives $\tilde{g}(Q_{1,k}, Q_{2,k}) \propto \ln(1 + \alpha_1 Q_{1,k}) + \ln(1 + \alpha_2 Q_{2,k})$, where $\alpha_i > 0$ is an SNR scaling factor. The controller spreads the energy over the entire codeword and, if the nonzero action is greater than the corresponding value of E_k, that EHD runs out of energy when only a fraction of the codeword has been transmitted, hence the codeword is discarded and no reward is accrued.

The *long-term average reward* per time slot, given the initial state $(S_{-1}, \mathbf{E}_0 = (E_{1,0}, E_{2,0}))$ and under policy μ, is defined as

$$G(\mu, S_{-1}, \mathbf{E}_0) = \lim_{K \to \infty} \inf \frac{1}{K} \mathbb{E}\left[\sum_{k=0}^{K-1} g(\mathbf{Q}_k, \mathbf{E}_k) \middle| S_{-1}, \mathbf{E}_0 \right] \tag{2.3}$$

where the expectation is computed over the random variables $\{\mathbf{B}_k, S_k, \mathbf{Q}_k\}$, for $k = 0, \ldots, K-1$. Hence, the problem considered in this work is that of obtaining a policy μ^* maximizing (2.3), *i.e.*,

$$\mu^* = \arg \max_{\mu} G(\mu, S_{-1}, \mathbf{E}_0). \tag{2.4}$$

2.3.2 Optimization under perfect SOC knowledge

If the state of charge of the two EHDs is perfectly known, the controller will select action $\mathbf{Q}_k = \mathbf{q}$ when the SOC is $(E_{1,k}, E_{2,k})$ and the energy arrival state of the previous time slot is S_{k-1}, with probability $\mu(\mathbf{q}; S_{k-1}, \mathbf{E}_k)$. In this case, the sequence $\{(S_{k-1}, \mathbf{E}_k, \mathbf{Q}_k),\ k \geq 0\}$ can be modeled as a Markov Decision Process, hence (2.4) can be solved using standard stochastic optimization techniques, such as the Policy Iteration Algorithm (PIA) [55]. From [56], the optimal policy in the perfect SOC case is deterministic: consequently, assuming that $(S_{k-1}, \mathbf{q}(\mathbf{E}_k))$ induces an irreducible Markov chain[1], the long-term reward does not depend on the initial state $(S_{-1}, \mathbf{E}_0)$, and the average reward becomes

$$G(\mu) = \sum_{e_1=0}^{e_{max,1}} \sum_{e_2=0}^{e_{max,2}} \sum_{s \in \mathcal{S}} \pi_\mu(s, e_1, e_2) g((q_1, q_2), (e_1, e_2)), \tag{2.5}$$

where $\pi_\mu(s, e_1, e_2)$ is the steady state distribution of the pair $(S_{k-1}, \mathbf{E}_k)$ induced by policy μ, calculated solving the system of equations:

$$\begin{cases} \sum_{\mathbf{e} \in \mathcal{E}_1 \times \mathcal{E}_2} \sum_{s \in \mathcal{S}} \pi_\mu(s, \mathbf{e}) = 1 & \text{(normalization)} \\ \pi_\mu(s, \mathbf{e}) \geq 0,\ \forall s \in \mathcal{S}, \forall \mathbf{e} \in \mathcal{E}_1 \times \mathcal{E}_2, & \text{(non-negativity)} \\ \sum_{\mathbf{u} \in \mathcal{E}_1 \times \mathcal{E}_2} \sum_{s \in \mathcal{S}} \pi_\mu(s, \mathbf{u}) \times \mathbb{P}_\mu(S_0 = s, \mathbf{E}_1 = \mathbf{e} | S_{-1} = s_{-1}, \mathbf{E}_0 = \mathbf{u}) = \pi_\mu(s, \mathbf{e}), \\ \qquad \forall s \in \mathcal{S}, \forall \mathbf{e} \in \mathcal{E}_1 \times \mathcal{E}_2, & \text{(steady-state equations)} \end{cases} \tag{2.6}$$

where $\mathbb{P}_\mu$ denotes the probability of visiting state $(S_0 = s, \mathbf{E}_1 = \mathbf{e})$ coming from state $(S_{-1} = s_{-1}, \mathbf{E}_0 = \mathbf{u})$.

The optimization problem in (2.4) can thus be restated, for the case with perfect SOC knowledge, as

$$\mu^* = \arg \max_{q_1, q_2} G(\mu), \tag{2.7}$$

and has been computed using the PIA [55].

2.3.3 Optimization under imperfect SOC knowledge

When the SOC $E_{i,k}$ of the two sensors is only partially known by the CC, to reduce the computational complexity, we exploit sub-optimal policies that only rely on S_{k-1} and the current interval indices $N_{1,k}$ and $N_{2,k}$, without taking into account the

[1]Please note that this is not a limiting assumption, as the underlying Markov chain will be irreducible (or at least unichain) in most cases of practical interest (*e.g.*, Bernoulli or truncated geometric energy arrivals).

past history $\mathcal{H}_{i,k}$. As a result, when $E_{1,k} \in \mathcal{I}_1(N_{1,k})$ and $E_{2,k} \in \mathcal{I}_2(N_{2,k})$, the CC decides on action $\mathbf{Q}_k = \mathbf{q}$ with probability $\mu(\mathbf{q}, S_{k-1}, \mathbf{N}_k)$, where $\mathbf{N}_k = (N_{1,k}, N_{2,k})$. Due to the imperfect SOC knowledge, policy μ imposes that the action $\mathbf{Q}_k$ be the same for all the energy levels $(E_{1,k}, E_{2,k}) \in \mathcal{I}_1(N_{1,k}) \times \mathcal{I}_2(N_{2,k})$, for a fixed value of S_{k-1}. Differently from the perfect SOC knowledge case, this constraint is not linear with respect to the joint steady-state distribution of the scenario/SOC/action vector $(s, \mathbf{e}, \mathbf{q})$, hence the optimal policy μ^* cannot be found via the PIA. In the following, we analyze policies that to every state $(s, \mathbf{n} = (n_1, n_2))$ associate only a subset $\rho(s, \mathbf{n}) \subseteq \mathcal{Q}$ of all possible actions. In particular, $\rho(s, \mathbf{n})$ can be of unit size (in which case the policy is deterministic for that state) or contain a number of equally likely actions (*e.g.*, in the case of states in which there exist multiple equivalent options).

Therefore, function ρ takes as input the vector $(s, n_1, n_2) \in \mathcal{S} \times \{0, \dots, \tilde{n}_1 - 1\} \times \{0, \dots, \tilde{n}_2 - 1\}$ and provides the action set $\rho(s, \mathbf{n}) \subseteq \mathcal{Q}$: consequently, (2.3) can be reformulated as

$$G(\rho, s_{-1}, \mathbf{e}_0) = \sum_{n_1=0}^{\tilde{n}_1-1} \sum_{n_2=0}^{\tilde{n}_2-1} \sum_{\substack{e_1 \in \mathcal{I}_1(n_1) \\ e_2 \in \mathcal{I}_2(n_2)}} \sum_{s \in \mathcal{S}} \sum_{\mathbf{q} \in \rho(s,\mathbf{n})} \pi_\rho(s, \mathbf{e}; s_{-1}, \mathbf{e}_0) \, \frac{g(\mathbf{q}, \mathbf{e})}{|\rho(s, \mathbf{n})|}, \tag{2.8}$$

where $|\rho(s, \mathbf{n})|$ is the number of available actions associated with state $(s, \mathbf{n})$.

In (2.8), $\pi_\rho(s, \mathbf{e}; s_{-1}, \mathbf{e}_0)$ is the steady-state distribution of the state pair $(s, \mathbf{e}) \in \mathcal{S} \times \mathcal{E}_1 \times \mathcal{E}_2$ induced by policy μ, if the initial state is $(s_{-1}, \mathbf{e}_0)$. Thus, denoting the k-step transition probability of the Markov chain modeling ρ as $\mathbb{P}_\rho(S_{k-1} = s, \mathbf{E}_k = \mathbf{e} | S_{-1} = s_{-1}, \mathbf{E}_0 = \mathbf{e}_0)$, we have

$$\pi_\rho(s, \mathbf{e}; s_{-1}, \mathbf{e}_0) = \lim_{K \to \infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{P}_\rho(S_{k-1} = s, \mathbf{E}_k = \mathbf{e} | S_{-1} = s_{-1}, \mathbf{E}_0 = \mathbf{e}_0). \tag{2.9}$$

Note that, in most practical cases, a policy μ induces an irreducible Markov chain, and the values of (2.9) can be calculated as the unique solution of the system of steady-state equations

$$\begin{cases} \sum_{n_1=0}^{\tilde{n}_1-1} \sum_{e_1\in\mathcal{I}_1(n_1)} \sum_{n_2=0}^{\tilde{n}_2-1} \sum_{e_2\in\mathcal{I}_2(n_2)} \sum_{s\in\mathcal{S}} \pi_\rho(s,\mathbf{e}) = 1, & \text{(normalization)} \\ \pi_\rho(s,\mathbf{e}) \geq 0, \ \forall s \in \mathcal{S}, \forall \mathbf{e} \in \mathcal{E}_1 \times \mathcal{E}_2, & \text{(non-negativity)} \\ \sum_{n_1=0}^{\tilde{n}_1-1} \sum_{u_1\in\mathcal{I}_1(n_1)} \sum_{n_2=0}^{\tilde{n}_2-1} \sum_{u_2\in\mathcal{I}_2(n_2)} \sum_{s\in\mathcal{S}} \pi_\rho(s,\mathbf{u}) \times \\ \qquad \times \mathbb{P}_\rho(S_0 = s, \mathbf{E}_1{=}\mathbf{e} | S_{-1} = s_{-1}, \mathbf{E}_0{=}\mathbf{u}){=}\pi_\rho(s,\mathbf{e}), \\ \qquad \forall s \in \mathcal{S}, \forall \mathbf{e} \in \mathcal{E}_1 \times \mathcal{E}_2, & \text{(steady-state equations)} \end{cases} \tag{2.10}$$

and so π_ρ and G are independent of the initial state $(s_{-1}, \mathbf{e}_0)$.

Finally, with the aforementioned notation, in the case of imperfect SOC knowledge, (2.4) becomes

$$\rho^* = \arg\max_{\rho} G(\rho). \tag{2.11}$$

Note that the long-term reward under imperfect SOC knowledge is upper bounded by the analogous reward computed for the scenario in which the SOC is perfectly known.

2.4 Numerical results

2.4.1 Summary of the results

In this section, we discuss the performance of the system, characterizing the throughput as a function of the battery capacities, the energy harvesting rates and the SNR coefficients at the two EHDs. The function $\tilde{g}$ in (2.2) is the total normalized capacity, calculated as

$$\tilde{g}(q_1, q_2) = \ln(1 + \alpha_1 q_1) + \ln(1 + \alpha_2 q_2), \tag{2.12}$$

which represents the achievable transmission rate under Gaussian signaling over an AWGN channel with gains α_1 and α_2 [57], so that $\alpha_1 q_1$ and $\alpha_2 q_2$ represent the signal-to-noise ratios (SNRs) related to EHD 1 and 2, respectively.

In Section 2.4.2, we first consider the case of perfect SOC knowledge and i.i.d. energy arrivals, which also provides an upper bound for the performance of the case with imperfect knowledge of the SOC. The main results of this subsection consist of a characterization of the energy capacities providing the best performance as a function of the pair of harvesting rates. In a symmetric setting (same values of

$\bar{b}_i$ and α_i, $i = 1, 2$) the EHDs are interchangeable for a given maximum capacity, whereas, if the values of $e_{max,i}$ are different, it is sufficient that only one EHD has high capacity to obtain good performance.

Section 2.4.3 presents a performance comparison between the transmission policies with perfect and partial knowledge of the SOC, under a number of different environmental conditions. In particular, it shows that, for sufficiently high battery capacities, the two policies give comparable results. Similarly to the perfect SOC knowledge case, the pair of EHDs is symmetric for an equal parameter configuration, whereas, if the channel conditions are unbalanced, the performance degradation can be mitigated by providing higher battery capacities. Lastly, this subsection establishes a comparison between the two-user system and an equivalent single-user one, showing that, for high battery capacities, these behave similarly, hence the simpler system can be used as a good approximation for the other.

Finally, Section 2.4.4 deals with the scenario with correlated energy arrivals, which may be a more appropriate model for some harvesting phenomena (*e.g.*, solar). The scavenging mechanism is modeled through a scenario process S_k with three hidden states, representing a "Random," "Good" and "Bad" state of the arrival process, respectively. Assuming that, in every time slot k, the value of S_{k-1} is known, the performance of the system is analyzed as a function of the state transition probabilities. In particular, it is shown that the convergence rate grows as the harvesting rate diminishes and that the reward achieved considering an equiprobable alternation between the three scenario states is higher than the mean of the rewards obtained when the "Good" or "Bad" states prevail.

In the following, we present extensive results to support the general conclusions outlined in this subsection.

2.4.2 Perfect SOC knowledge and i.i.d. energy arrivals

In this section we consider the computation of (2.7), for $q_{min,i} = 1$, $p_{pck,1} = p_{pck,2} = 1$, $q_{max,i} = e_{max,i}$ and geometric energy arrival distributions, i.i.d. among the EHDs, with mean $\bar{b}_i$, truncated at $b_{max,i} = 4\bar{b}_i$, $i = 1, 2$, for a symmetric setting in which $\alpha_1 = \alpha_2 = \alpha$. The aim is to compute the optimal policy, employing the PIA. A performance upper bound $\tilde{g}_{ub}$ for the scenario considered in this subsection can be found in (2.15) in the Appendix: note that, in general, (2.15) is an upper bound to the original maximization problem. However, as shown by the numerical results, this bound is asymptotically achieved and is tight for sufficiently high values of the capacities of the two batteries. Furthermore, it is worth noting that (2.15) represents an upper bound also for the imperfect SOC knowledge scenario, which will be discussed in Section 2.4.3. As a result, the analysis of the performance

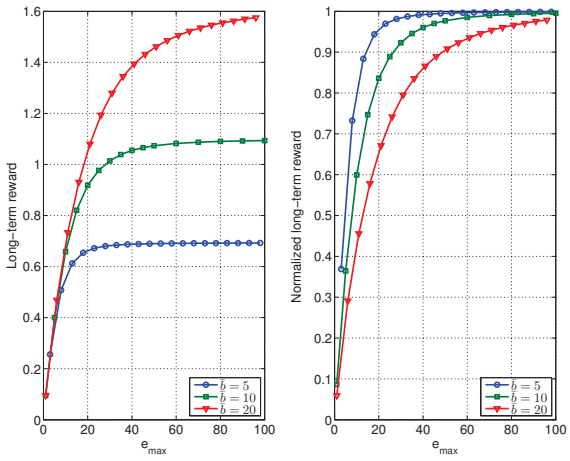


Figure 2.3: Long-term reward versus battery capacity $e_{max,1} = e_{max,2} = e_{max}$, for different values of the mean energy harvesting rate $\bar{b}_1 = \bar{b}_1 = \bar{b}$ and $\alpha = 0.1$.

achieved in the perfect case can give a useful insight into the maximum achievements obtainable when the information about the SOC is not completely available.

In Figure 2.3, we plot the long-term reward (*i.e.*, the throughput), for different values of $\bar{b}_1 = \bar{b}_2 = \bar{b}$, both non-normalized and normalized using the upper bound (2.15) computed in Appendix. We notice that the throughput keeps increasing in the capacity of the two batteries, until $e_{\max,1} = e_{\max,2} \simeq 50$, after which the performance saturates at a constant value. This is because, the larger the battery, the smaller the impact of energy outage and overflow, hence the better the performance. When the battery capacity becomes larger than 50, the improvement due to decreased overflow and outage events becomes negligible, and the performance is very close to the upper bound. An important implication of this result is that EHDs do not need to be equipped with very large energy buffers to achieve maximum performance. Moreover, as expected, the reward increases with the harvesting rate of the EHDs: however, due to the fact that the practical values of $\bar{b}_i$ are not very large, the reward cannot grow too much. Finally, from the normalized curves, it can be seen that, as $\bar{b}$ grows, throughput saturation occurs more slowly. This can be explained by considering that when $e_{max} < \bar{b}$, the number of energy quanta that can be used for transmission will be at most e_{max}, hence much energy will be wasted.

Figure 2.4 shows a three-dimensional plot representing the normalized long-term reward as a function of $e_{\max,1}$ and $e_{\max,2}$, when $\bar{b}_1 = \bar{b}_2 = 10$. The normalization is done with respect to the upper bound $\tilde{g}_{ub}$, given in (2.15). The reward increases with the capacity of the two batteries in a "symmetric" way, in the sense that the two EHDs are interchangeable: the result achieved when $e_{\max,1} = e_1$ and $e_{\max,2} = e_2$ is the same that can be obtained when $e_{\max,1} = e_2$ and $e_{\max,2} = e_1$. This is due to the symmetry of the system, since $\bar{b}_1 = \bar{b}_2$ and $\alpha_1 = \alpha_2$. Furthermore, as in Figure 2.3, it

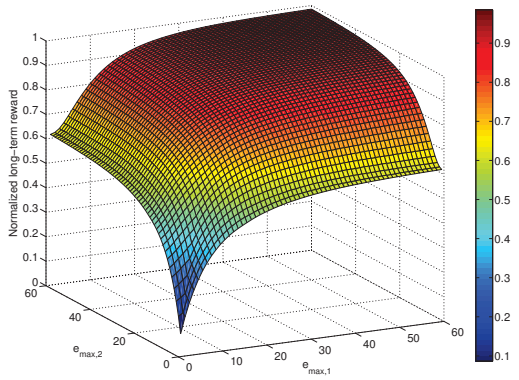


Figure 2.4: Normalized long-term reward versus battery capacities, $b_1 = b_2 = 10$, $\alpha = 0.1$.

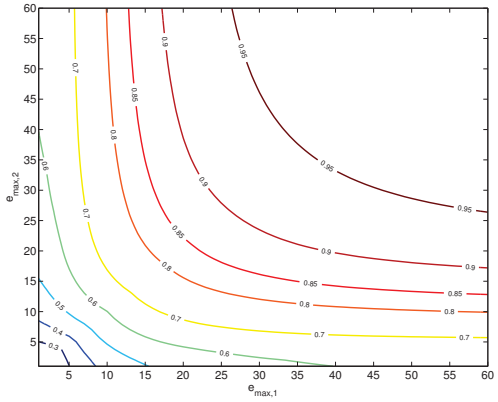


Figure 2.5: Contour plot of Figure 2.4.

can be seen that saturation occurs at $e_{\max,1} = e_{\max,2} \simeq 50$, where the upper bound (2.15) is closely approached. From Figure 2.4 we can derive the combinations of battery sizes that guarantee a given level of performance (this is easier to read from a contour plot obtained from the intersection of Figure 2.4 with horizontal planes, see 2.5). For example, 75% of the maximum performance is achieved even when the EHDs have relatively small capacities ($e_{\max,1} = e_{\max,2} = 15$), whereas 90% can be attained only with larger batteries (in particular, in this case $e_{\max,1} = e_{\max,2} = 26$). In addition, saturation ($> 95\%$) starts when $e_{\max,1}$ and $e_{\max,2}$ are > 36.

A deeper analysis of the long-term reward as a function of $e_{\max,1}$ and $e_{\max,2}$ in the non-symmetric case (obtained by intersecting Figure 2.4 with vertical planes) is shown in Figure 2.6. Good performance is achieved when at least one EHD can afford high energy capacity: the situation where $e_{\max,2} = 40$ and $e_{\max,1}$ is just a fourth of $e_{\max,2}$ provides a throughput that is 81% of that obtained when both sensors have capacity $e_{\max,1} = e_{\max,2} = 40$. While the results for the single-user case could give a valuable insight into the behavior of the two-user system when the batteries of the EHDs are both small or large, predicting the performance of the general unbalanced case based purely on intuitive reasoning is more difficult. However, our two-user analysis suggests that, for instance, a good implementation strategy may be to couple two devices whose battery capacities are such that one is half of the other. The reward achieved when $e_{max,2} = 40$ and $e_{max,1} = 20$ is 94% of that obtained when both $e_{max,i} = 40$, and is 90% of the maximum feasible performance given by (2.15). Consequently, if one of the two devices has a relatively large battery, the requirement on the battery of the other device becomes much looser.

Although it is not very visible from Figure 2.4 and Figure 2.6, a closer examination reveals that there are points at which the slope of the normalized performance

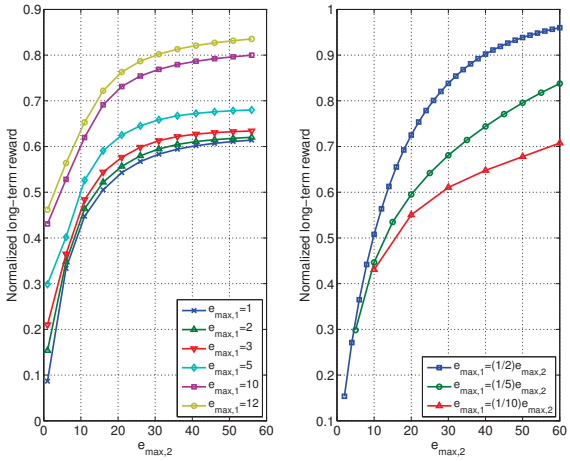


Figure 2.6: Normalized long-term reward versus battery capacity of EHD 2, for different values of $e_{\max,1}$ and energy harvesting rate $\bar{b} = 10$.

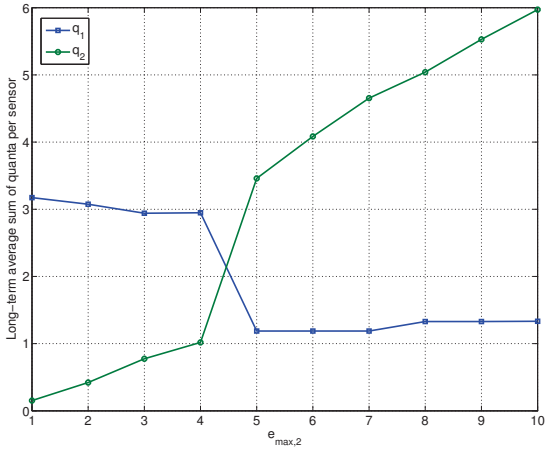


Figure 2.7: Long-term average sum of quanta used for transmission by sensor 1 (q_1) and sensor 2 (q_2), as a function of $e_{max,2}$ and for $e_{max,1} = 4$, related to the framework of Figure 2.4.

changes. For instance, fixing the value of $e_{max,1}$, we have that, as $e_{max,2}$ grows, the reward function is initially convex, but eventually becomes concave for larger values of the capacity, with the inflection point reached for $e_{max,1} = e_{max,2}$. This phenomenon produces an initial lower slope followed by an increased one, until saturation finally occurs. This particular behavior can be explained with the help of Figure 2.7, where we plot, as an example, the long-term average sum of quanta used for transmission by each sensor, as a function of $e_{max,2}$ and for a fixed value of $e_{max,1} = 4$. When $e_{max,1} > e_{max,2}$, the first device employs more energy for transmission than the other device, as $q_{max,i} = e_{max,i}$ for both the EHDs, but the two roles switch when $e_{max,1}$ becomes lower than $e_{max,2}$. In this second region, where the growth of the slope associated to the reward occurs, the amount of energy devoted to transmission by the second user (green line) is greater than the corresponding quantity in the previous

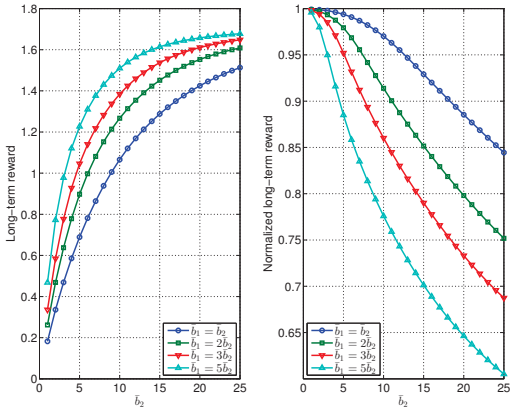


Figure 2.8: Long-term reward versus EH rate of EHD 2, for $e_{\max,1} = e_{\max,2} = 45$ and $\bar{b}_1$ and $\bar{b}_2$ in fixed ratios

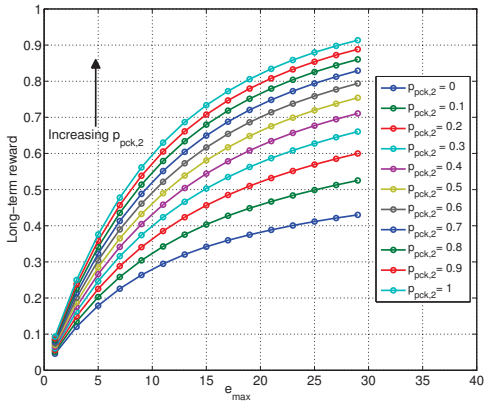


Figure 2.9: Long-term reward as a function of e_{max}, for different values of $p_{pck,2}$ and $p_{pck,1} = 0.5$ ($\alpha = 0.1, \bar{b} = 10$).

region (blue line), resulting in the flex given for $e_{max,1} = e_{max,2}$.

In Figure 2.8, to study the impact of the harvesting rate, we plot the long-term reward versus $\bar{b}_2$, for different values of $\bar{b}_1/\bar{b}_2$ and for $e_{\max,1} = e_{\max,2} = 45$. It can be seen that, for lower values of the mean energy harvesting rates $\bar{b}_1$ and $\bar{b}_2$, the system performance nearly approaches the maximum achievable reward (2.15). This is due to the value of the battery capacities: as the upper bound (2.15) only depends on the mean harvesting rates $\bar{b}_i$, higher performance is achieved, for given values of $\bar{b}_1$ and $\bar{b}_2$, when energy overflows are reduced. Hence, if the capacity of the batteries is larger than the average amount of energy scavenged in a time slot, overflow events become infrequent and the normalized long-term reward is close to 1. On the other hand, when the total amount of harvested energy grows, the two batteries fail to store all the energy coming from the environment, which is sometimes wasted, resulting in the occurrence of overflow. As a result, the normalized reward decreases, as in this case greater battery capacities would be needed to properly exploit all the available energy.

In Figure 2.9, to study the effect of the data packet arrival rate, we plot the long-term reward as a function of e_{max}, for different values of $p_{pck,2}$ and a fixed value of $p_{pck,1}$. This framework with different packet arrival probabilities could model the case of the two devices simultaneously sensing different kinds of physical quantities. For instance, assuming sensors deployed in an urban scenario, the first device could control ambient temperature, while the second could monitor air pollution. The first may be sampled less frequently with respect to the second, as temperature varies more slowly than pollution levels, hence the different values of $p_{pck,1}$ and $p_{pck,2}$. In this case, the long-term reward increases with the battery capacity of the two devices, but the gap between consecutive increasing values of $p_{pck,2}$ diminishes as $p_{pck,2} \to 1$.

2.4.3 Imperfect SOC knowledge and i.i.d. energy arrivals

In this subsection, we study how a partial knowledge of the battery state of charge affects the performance. Towards this goal, we determine the optimal policy ρ^* for the case in which the energy state space of every EHD is partitioned into the same number of regions, *i.e.*, $\tilde{n}_1 = \tilde{n}_2 = \tilde{n}$. In this particular scenario, the two-dimensional state space $\mathcal{E}_1 \times \mathcal{E}_2$, inclusive of both the EHDs, can be divided into $\tilde{n}'$ two-dimensional partitions $\mathcal{I}'(n),\ n \in \{0, \dots, \tilde{n}' - 1\}$. Specifically, we consider a scenario with two equal-interval uncertainty sets for every EHD, *i.e.*, $\tilde{n} = 2$ and $\tilde{n}' = 4$, resulting in the following four partitions:

- $\mathcal{I}'(0) = \{(e_1, e_2),\ 0 \le e_1 \le \tilde{e}_1 - 1,\ 0 \le e_2 \le \tilde{e}_2 - 1\}$ ("LL")
- $\mathcal{I}'(1) = \{(e_1, e_2),\ 0 \le e_1 \le \tilde{e}_1 - 1,\ \tilde{e}_2 \le e_2 \le e_{max,2}\}$ ("LH")
- $\mathcal{I}'(2) = \{(e_1, e_2),\ \tilde{e}_1 \le e_1 \le e_{max,1},\ 0 \le e_2 \le \tilde{e}_2 - 1\}$ ("HL")
- $\mathcal{I}'(3) = \{(e_1, e_2),\ \tilde{e}_1 \le e_1 \le e_{max,1},\ \tilde{e}_2 \le e_2 \le e_{max,2}$ ("HH")

with $\tilde{e}_i = \lceil \frac{e_{max,i}}{2} \rceil$.

The assumption is that, at time k, $(E_{1,k}, E_{2,k}) \in \mathcal{I}'(N_k)$, with $N_k \in \{0, 1, 2, 3\}$, and the central controller knows only that $\mathbf{E}_k = (E_{1,k}, E_{2,k}) \in \mathcal{I}'(N_k)$, *i.e.*, N_k rather than the exact SOC $\mathbf{E}_k$. In addition, we employ a ρ function such that:

- if $e_{max,1} \neq e_{max,2}$, $\rho(n)$ is a singleton:

$$\begin{cases} \mu_\rho(\mathbf{q}; n) = 1 & \mathbf{q} = \rho(n) \\ \mu_\rho(\mathbf{q}; n) = 0 & \mathbf{q} \in \mathcal{Q} \setminus \{\rho(n)\} \end{cases}$$

- if $e_{max,1} = e_{max,2}$, $\rho(n)$ is a singleton if $n = 1$ or $n = 2$ (*i.e.*, when the current energy partition is "LH" or "HL"), or contains two symmetric and equiprobable actions, namely $(0, q)$ and $(q, 0)$ (with $q \ge 0$ a deterministic function of n) when $n = 0$ or $n = 3$ (the two energy levels belong to the same partition "LL" or "HH"), so as to prevent the system from getting unbalanced and to maintain the overall symmetry between the two devices.

We now consider the computation of (2.11), for $q_{max,i} = e_{max,i}$, $\alpha_1 = \alpha_2 = \alpha$ and, unless otherwise stated, the same geometric energy arrival distributions as in the perfect SOC case (i.i.d. among the EHDs, with mean $\bar{b}_i$, truncated at $b_{max,i} = 4\bar{b}_i,\ i = 1, 2$). Also, the same throughput function $\tilde{g}$ as in (2.12) will be employed.

Figure 2.10 shows $G(\mu)$ as a function of e_{max}, for the following two policies: policy with perfect SOC knowledge as in Subsection 2.4.2 (PP-2) and policy with imperfect

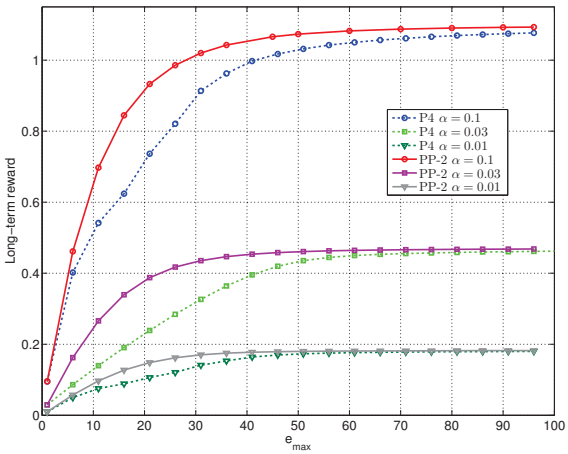


Figure 2.10: Throughput as a function of e_{max}, for different values of α and $\bar{b} = 10$.

SOC knowledge and two-equal-interval uncertainty per EHD (P4), both for $\bar{b} = 10$ and for different values of α. As expected, for a fixed value of α, P4 is inferior with respect to PP-2, but the two policies become comparable at a buffer capacity $e_{max} \simeq 30$, where the reward degradation of P4 with respect to PP-2 is less than 10%. As e_{max} increases, the degradation of P4 decreases further, as the impact of overflow and outage (which occur when the SOC gets close to 0 or e_{max}) diminishes. In fact, for sufficiently high battery capacities ($e_{max} = 40$), PP-2 outperforms P4 by just 5%. A comparison between Figure 2.10 and the analogous results related to the single-user case [41] indicates that a system with two EHDs has lower performance for low battery capacities but, after an initial slow increase, grows more rapidly and saturates earlier. Hence, if both batteries are low, a two-user system performs worse than a single-user one but, for bigger batteries, it is able to achieve higher rewards, exploiting the diversity arising from the availability of a pair of EHDs.

In Figure 2.11 we plot the throughput of the system, both non-normalized and normalized using the upper bound in (2.15), for different values of $\bar{b}$ and $\alpha = 0.1$. As expected, the reward increases with the value of the mean energy harvesting rate $\bar{b}$. In addition, as was the case in Figure 2.3, throughput saturation occurs more slowly as $\bar{b}$ grows, due to the limited maximum amount of energy available for transmission. It can be seen that the upper bound (2.15) is still achievable but, compared to Figure 2.3, this occurs for higher values of the battery capacities due to the imperfect SOC knowledge.

The analysis of the amounts of energy drawn by P4, for given values of α and $\bar{b}$ (see Figure 2.12), shows that the optimal actions for states "LH" and "HL" are the same, *i.e.*, the system is symmetric, as expected. In addition, when at least one of the two devices is in state "H," the amount of energy used for transmission is e_{max}, until $e_{max} \approx \bar{b}$, thus exploiting the abundance of energy harvested from

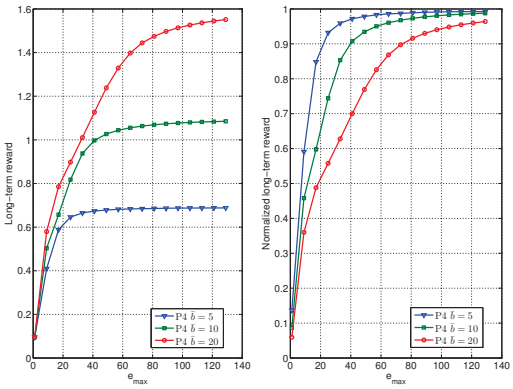


Figure 2.11: Throughput as a function of e_{max}, for different values of $\bar{b}$ and $\alpha = 0.1$.

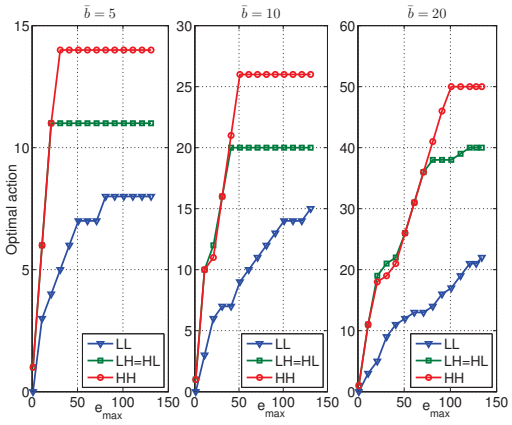


Figure 2.12: Actions $\rho(0)$ (LL), $\rho(1)$ (LH), $\rho(2)$ (HL) and $\rho(3)$ (HH) of P4 corresponding to Figure 2.11, for $\bar{b} = 5,\ 10,\ 20$. The optimal actions for LH and HL are identical, *i.e.*, the system is symmetric.

the environment. Then it becomes $\approx e_{max}/2$, so as to exploit all the energy surely present in the EHD (this value corresponds to $\tilde{e} = \lceil e_{max}/2 \rceil$) and finally, when $e_{max} \approx 4\bar{b}$, the actions for "HL" and "HH" saturate to a constant value, showing that, when e_{max} is sufficiently high, it is more advantageous to devote a smaller amount of energy to transmission (obtaining a lower reward) and remain in state "H," rather than transmitting a higher amount of energy, increasing the reward but also making it more likely to downgrade to state "L," with a negative impact on future performance. Finally, when e_{max} is high, the optimal policy uses more energy for transmission in state "HH" than in "HL/LH." This can be explained by observing that when the system is in state "HL/LH," it is possible that using too much energy will lead both devices to be in a low energy state in the next slot, whereas this situation never occurs if the system is in state "HH," since only one device can transmit at any time.

Figure 2.13 shows the normalized long-term reward, for different values of $e_{max,1}$ and $e_{max,2}$, with $\bar{b}_1 = \bar{b}_2 = 10$ and $\alpha = 0.1$ (this is the imperfect-SOC counterpart of Figure 2.4). It can be seen that the performance increases with the battery capacities of the two devices and, as expected, this occurs in a symmetric way, similarly to the perfect SOC knowledge scenario. In particular, we have that the reward has a piecewise linear behavior, that is due to the truncation in the computation of $\tilde{e}_i = \lceil \frac{e_{max,i}}{2} \rceil$. As stated for the case with perfect knowledge of the SOC, a good implementation strategy could be that of pairing two devices for which the battery capacity of the first is half of that of the second: for instance, when $e_{max,1} = 20$ and $e_{max,2} = 10$ the achieved reward is 85% of that obtained when both the two devices have a large battery ($e_{max,1} = e_{max,2} = 23$).

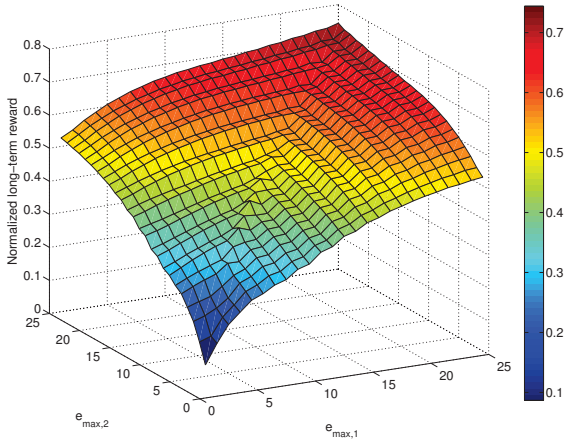


Figure 2.13: Normalized long-term reward as a function of $e_{max,1}$ and $e_{max,2}$, with imperfect SOC knowledge ($\bar{b} = 1, \alpha = 0.1$).

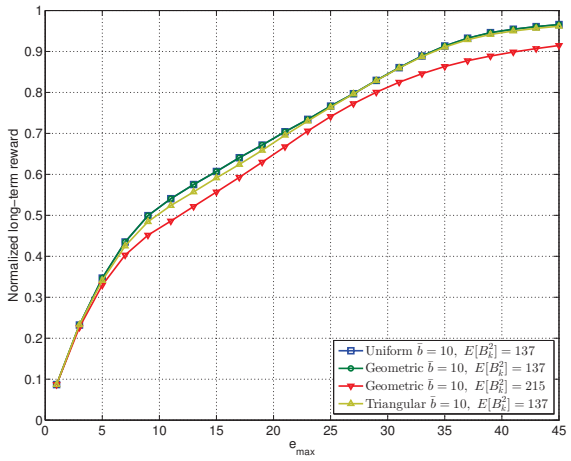


Figure 2.14: Throughput as a function of $e_{max,1} = e_{max,2} = e_{max}$, for different energy arrival statistics ($\alpha = 0.1$).

Figure 2.14 shows the dependence between the long-term reward G and the energy arrival statistics, under imperfect knowledge of the SOC. The higher the second order moment (or, equivalently, the variance, as the value of the mean $\bar{b}$ is fixed), the larger the performance degradation, for both perfect and imperfect SOC knowledge. Here, the distribution marked as "triangular" refers to a particular distribution of the energy arrivals in which a nonzero probability is associated with the event that to no energy is harvested, whereas the probability of harvesting a positive number of quanta follows a triangular distribution [58]. The results show that the three distributions, all with mean $\bar{b} = 10$ and statistical power $\mathbb{E}[B_k^2] = 137$, practically achieve the same performance, whereas the performance changes if a different value of the variance is selected. This indicates that the overall reward mostly depends on the second order statistics of the arrival process. Finally, when the battery capacity of the two EHDs is sufficiently large, the impact of energy

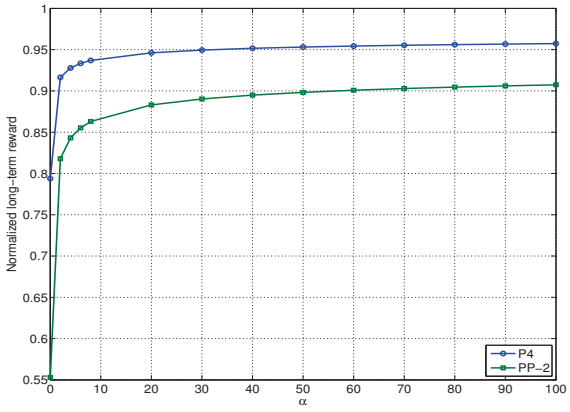


Figure 2.15: Long-term reward as a function of $\alpha_1 = \alpha_2$, for $e_{max,1} = e_{max,2} = 20$ and $\bar{b}_1 = \bar{b}_2 = 20$.

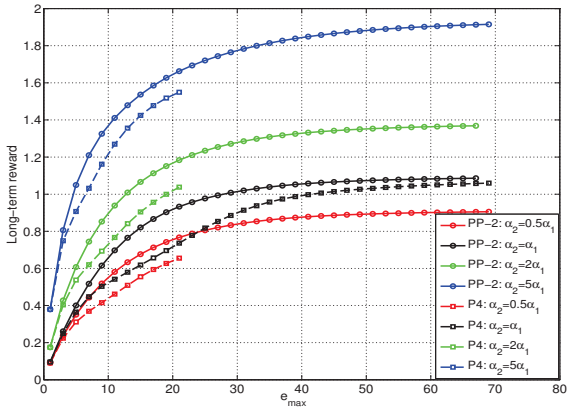


Figure 2.16: Long-term reward as a function of $e_{max,1} = e_{max,2} = e_{max}$, for a fixed value of $\alpha_1 = 0.1$ and different values of α_2 ($\bar{b}_1 = \bar{b}_2 = 10$).

outage and overflow, which are related to the erratic energy source, diminishes and the performance is mainly affected by the energy harvesting rate $\bar{b}$. Due to this phenomenon, all the curves asymptotically reach the same normalized value.

Figure 2.15 shows the behavior of the throughput for a fixed value of the battery capacities $e_{max,1} = e_{max,2} = 20$ and of the mean harvesting rates $\bar{b}_1 = \bar{b}_2 = 20$, as a function of the SNR factors $\alpha_1 = \alpha_2$. Both the perfect and imperfect SOC knowledge policies increase with α, until saturation occurs. However, differently from Figure 2.3 or Figure 2.10, where saturation takes place in conjunction with the maximum achievable rate (*i.e.*, the normalized reward is one), here, even for very high values of α, the performance may be sub-optimal. This is due to the limited amount of energy that can be stored in the batteries of both the EHDs: a higher value of e_{max} is needed to properly exploit the favorable conditions of the shared wireless channel. Hence, Figure 2.15 proves that the bottleneck of the system with respect to the long-term reward is given by the battery capacities $e_{max,i}$, rather than by the channel gains.

Figure 2.16 shows the long-term reward as a function of $e_{max,1} = e_{max,2} = e_{max}$,

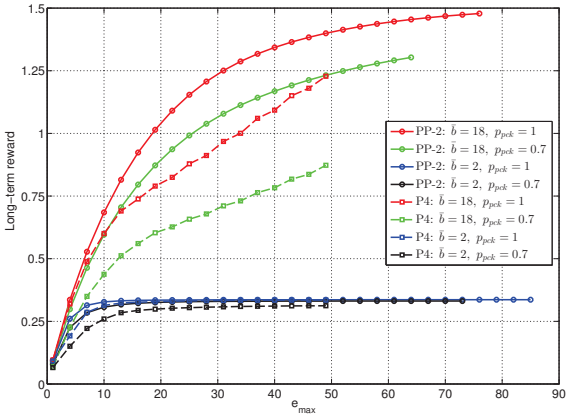


Figure 2.17: Long-term reward as a function of $e_{max,1} = e_{max,2} = e_{max}$, for different values of $\bar{b}_1 = \bar{b}_2 = \bar{b}$ and different packet arrival probabilities ($\alpha = 0.1$).

with $\bar{b}_1 = \bar{b}_2 = \bar{b} = 10$ and $\alpha_1 = 0.1$, for varying values of α_2. This framework represents a possible scenario in which the two devices suffer from different channel capacities, for instance due to an obstacle placed between the RX and one of the two devices. The reward accrued when $e_{max} = 10$ passing from $2\alpha_1$ to $5\alpha_1$ is 35% higher than that obtained for the same upgrade when $e_{max} = 50$. Actually, the advantage taken from higher values of α_2 decreases with increasing battery capacities of the two devices, therefore it is important to have a favorable wireless channel when the system is not provided with large batteries. In addition, it can be seen that throughputs achieved when $\alpha_2 = 5\alpha_1$ for $e_{max} = 5$ and when $\alpha_2 = \alpha_1$ for $e_{max} = 40$ are the same, showing that nodes equipped with larger batteries are able to overcome worse channel conditions.

Figure 2.17 shows policies PP-2 and P4 for two different values of the mean energy harvesting rate $\bar{b}_1 = \bar{b}_2 = \bar{b}$, and two possible values of the packet arrival probability $p_{pck,1} = p_{pck,2} = p_{pck}$. Policies obtained when $p_{pck} = 0.7$ converge more slowly than those with $p_{pck} = 1$. Moreover, when $\bar{b} = 18$, the gap between PP-2 (P4) for $p_{pck} = 1$ and PP-2 (P4) for $p_{pck} = 0.7$ is increased with respect to that incurred for the corresponding policies when $\bar{b} = 2$. This is because the lower the probability of packet arrivals, the higher the probability of no packet transmission, and thus the lower the performance when more energy arrives $\left(\bar{b} = 18\right)$ and is wasted. Finally, it can be noted that the smaller packet arrival probability mainly affects the policies with imperfect knowledge of the SOC: this is clearer for $\bar{b} = 18$, where, for $e_{max} = 40$, P4 with $p_{pck} = 0.7$ is only 70% of that with $p_{pck} = 1$, whereas PP-2 with $p_{pck} = 0.7$ is 87% of that with $p_{pck} = 1$.

Finally, we investigated the possibility of approximating the performance of a two-EHD system with that of an equivalent single-user system. Figure 2.18 shows the throughput as a function of the battery capacity of the entire system, for the perfect and imperfect policies PP-2 and P4, and for $\bar{b} = 5, \alpha = 0.1$. The performance

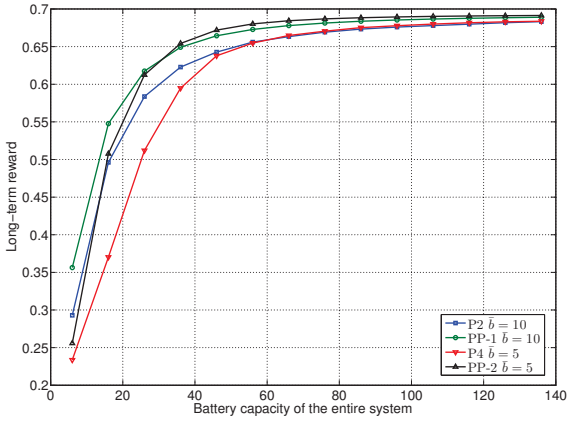


Figure 2.18: Throughput for P4, P2, PP-2 and PP-1 as a function of the total battery capacity of the system and different values of $\bar{b}$.

achieved by the analogous single-user scenario, studied in [41] and added here for comparison, consists of a similar framework, with either perfect (PP-1) or imperfect SOC knowledge (P2, *i.e.*, $\mathcal{I}(0) = \{0, \ldots, \tilde{e}_1 - 1\}$ and $\mathcal{I}(1) = \{\tilde{e}_1, \ldots, e_{max}\}$, with $\tilde{e}_1 = \lceil \frac{e_{max}}{2} \rceil$). Clearly, if in the two-user framework the maximum battery capacity of each EHD is e_{max}, the capacity of the comparable single-user system must be $2e_{max}$, and similarly for the harvesting rate. It can be seen that the performance for the single-user case with imperfect SOC (P2) is better than that for the corresponding two-user case (P4) up to $e_{max} \approx 50 \div 60$, whereas the two cases have essentially the same performance for higher values of e_{max}. A similar comparison holds for the perfect SOC case (PP-1 and PP-2), although the performance gap is much smaller and very similar throughput values are obtained already for $e_{max} \approx 20 \div 25$. These results suggest that, for sufficiently large values of the total battery capacity, the results for the two-user scenario (which would require heavy computations) can be approximated by using those for the single-user case instead. Therefore, the optimization for the two-user case needs to be explicitly carried out only for relatively small values of e_{max}, where the computation is not too demanding.

2.4.4 Correlated energy arrivals

The i.i.d. energy arrival model considered so far, while being representative of some relevant physical phenomena (*e.g.*, vibration), may not capture the behavior of other important harvesting sources (*e.g.*, solar). For this reason, in this subsection we present numerical results for the case of correlated energy arrivals at the two sensors. We assume a pair of energy arrival processes $(B_{1,k}, B_{2,k})$, both with mean $\bar{b} = 10$, and a scenario process S_k with three hidden states ($\mathcal{S} = \{\text{R, G, B}\}$), representing a "Random," "Good" and "Bad" state of the arrival process, respectively. The transmission probabilities between the energy arrival states are defined by the $|\mathcal{S}| \times$

$|\mathcal{S}|$ transition matrix $\mathbf{P}_{\mathcal{S}}$, with entries $[\mathbf{P}_{\mathcal{S}}]_{s_0,s_1} = \mathbb{P}(S_k = s_1|S_{k-1} = s_0) \triangleq p_{\mathcal{S}}(s_1|s_0)$, given by

$$\mathbf{P}_{\mathcal{S}} = \begin{bmatrix} p_{\mathcal{S}}(R|R) & \frac{1-p_{\mathcal{S}}(R|R)}{2} & \frac{1-p_{\mathcal{S}}(R|R)}{2} \\ 1-p_{\mathcal{S}}(G|G) & p_{\mathcal{S}}(G|G) & 0 \\ 1-p_{\mathcal{S}}(B|B) & 0 & p_{\mathcal{S}}(B|B) \end{bmatrix}. \tag{2.13}$$

In all states "R," "G" and "B," both arrival processes follow a geometric distribution, with mean $\bar{b}$, truncated at $b_{max} = 4\bar{b}$. For solar energy harvesting, the "G" state corresponds to optimal sun exposure with $\bar{b} = 18$, whereas the "B" state corresponds to almost complete darkness, in which $\bar{b} = 2$. Finally, state "R" models a transient situation between the other two states, where the arrival processes exhibit a random behavior, thus in this situation $\bar{b} = 10$. The steady-state distribution of the scenario states is:

$$\begin{aligned} \pi_{\mathcal{S}}(G) &= \frac{1-p_{\mathcal{S}}(R|R)}{2(1-p_{\mathcal{S}}(G|G))}\pi_{\mathcal{S}}(R) \\ \pi_{\mathcal{S}}(B) &= \frac{1-p_{\mathcal{S}}(R|R)}{2(1-p_{\mathcal{S}}(B|B))}\pi_{\mathcal{S}}(R) \\ \pi_{\mathcal{S}}(R) &= \left(1 + \frac{1-p_{\mathcal{S}}(R|R)}{2(1-p_{\mathcal{S}}(B|B))} + \frac{1-p_{\mathcal{S}}(R|R)}{2(1-p_{\mathcal{S}}(G|G))}\right)^{-1} \end{aligned}$$

Note that the i.i.d. energy arrival scenario, discussed in the previous sections, is obtained if $p_{\mathcal{S}}(R|R) = 1$, $\forall p_{\mathcal{S}}(G|G)$, $p_{\mathcal{S}}(B|B) \in [0, 1)$. In this case, the random state "R" absorbs the Markov chain modeling the scenario process S_k, and the energy arrival processes $B_{1,k}$ and $B_{2,k}$ become i.i.d. with probability mass function $p_{B_i}(b)$, with $b \in \mathcal{B}_i$. In the following, we assume that, in every time slot k, the CC knows the exact value of the scenario process in slot $k-1$, S_{k-1}. Moreover, we employ the same reward function defined in (2.12). Due to the large dimensionality of the state space, the algorithms needed to compute the optimal policies have high complexity. As a consequence, we reduce this optimization complexity by decoupling the different time scales relevant to the dynamics of the system: the short-term average performance is first optimized with respect to the realization of each scenario state variables, varying over short time scales, by imposing only a mean energy consumption constraint. Secondly, the optimization involves policies establishing the average energy consumption as a function of the state variables evolving over longer time scales, *i.e.*, the complete scenario process. As an example of the results obtained in this case, Figure 2.19 shows the long-term reward as a function of $e_{max,1} = e_{max,2} = e_{max}$, for different realizations of the scenario process S_k, under perfect (PP-2) and imperfect (P4) knowledge of the SOC. In particular, we analyze three different instances:

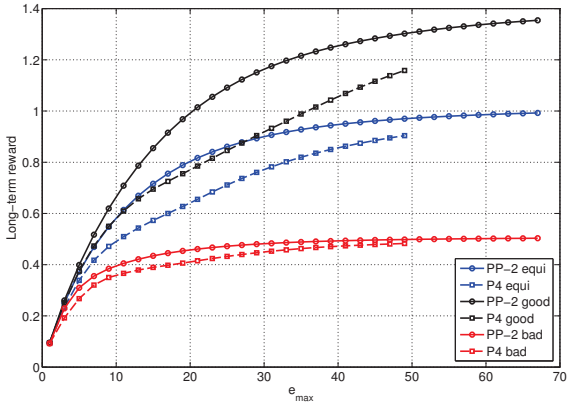


Figure 2.19: Long-term reward, as a function of e_{max}, for different instances of the scenario process S_k ($\alpha = 0.1$).

- the "Equi" curves, in which $p_S(R|R) = p_S(G|G) = p_S(B|B) = 0.5$, representing an equiprobable alternation between all the states "R," "G" and "B" of the scenario, resulting in the following steady-state distribution: $\pi_S(R) = 0.5, \pi_S(G) = \pi_S(B) = 0.25$.
- the "Good" curves, for which $p_S(R|R) = 0.5$, $p_S(G|G) = 0.95$ and $p_S(B|B) = 0.05$, modeling the case of the sensors being persistently exposed to direct sunlight: $\pi_S(R) \simeq 0.16, \pi_S(G) \simeq 0.8, \pi_S(B) \simeq 0.04$.
- the "Bad" curves, defined similarly to the previous case, but with state "B" prevailing; $p_S(R|R) = 0.5$, $p_S(G|G) = 0.05$ and $p_S(B|B) = 0.95$ ($\pi_S(R) \simeq 0.16, \pi_S(G) \simeq 0.04$, $\pi_S(B) \simeq 0.8$).

It can be seen that the bad scenario presents faster convergence, and, in general, the more favorable the harvesting rate, the slower the convergence rate, as already noted in Figure 2.11. Secondly, the reward achieved in the equiprobable case is higher than the mean of the reward for the good and bad curves. Note that, if $p_S(R|R) = 1$ and $p_S(G|G), p_S(B|B) \in [0, 1)$, we would find the i.i.d scenario for the energy arrivals, whose rewards are depicted in Figure 2.10 (case $\alpha = 0.1$).

2.5 Extensions

In this section, we discuss some possible extensions to the model adopted so far, where some assumptions are relaxed.

2.5.1 Secondary costs and non-idealities

We now present how model (2.1) could be extended to take into account other secondary costs like processing, sensing and activation or non-idealities, such as

battery degradation. The temporal evolution of the energy stored in battery i becomes

$$E_{i,k+1} = \min\{[E_{i,k} - Q_{i,k} - L_{i,k}]^+ + B_{i,k}, e_{max,i}\},$$

with $L_{i,k}$ representing the overall energy cost in slot k (neglecting the cost for action $Q_{i,k}$), accounting for battery leakage, processing, sensing and activation of the circuitry after the node switches off (if $Q_{i,k-1} = 0$). We define the *activity state* $A_{i,k} = \chi(Q_{i,k-1} > 0)$, tracking the activity of sensor node i, which is 0 if the node was idle in the previous time slot $k-1$ ($Q_{i,k-1} = 0$) or 1 if the node was active ($Q_{i,k-1} > 0$). $L_{i,k}$ can be modeled as a random variable with probability distribution $p_{L_i}(L_{i,k}|Q_{i,k}, A_{i,k})$, taking values in the set $\mathcal{L}_i = \{0, 1, \ldots, L_{max,i}\}$. The dependence of p_{L_i} on $Q_{i,k}$ could model different energy costs related to different actions, *i.e.*, higher battery leakages when the node transmits at full power: $\mathbb{P}(L_{i,k} \geq l|Q_{i,k} \simeq q_{max,i}, A_{i,k} = a_{i,k}) \geq \mathbb{P}(L_{i,k} \geq l|Q_{i,k} \simeq 0, A_{i,k} = a_{i,k})$, $\forall l$ and a fixed value of $a_{i,k}$. Instead, the dependence on $A_{i,k}$ may be exploited to model activation costs of the sensor circuitry, such as the case of a higher energy cost incurred switching from idle to active mode ($A_{i,k} = 0,\ Q_{i,k} > 0$) with respect to staying active ($A_{i,k} = 1,\ Q_{i,k} > 0$), *i.e.*, $\mathbb{P}(L_{i,k} \geq l|Q_{i,k}, A_{i,k} = 0) \geq \mathbb{P}(L_{i,k} \geq l|Q_{i,k}, A_{i,k} = 1), \forall l,\ \forall Q_{i,k} > 0$.

With this extended model, policy μ discussed in Section 2.3 must take $A_{i,k} \in \{0, 1\}$ into account, and the reward function $g(\mathbf{Q}_k, \mathbf{E}_k, \mathbf{L}_k)$ will be zero if $Q_{i,k} > [E_{i,k} - L_{i,k}]^+$ for $i = 1$ or $i = 2$.

2.5.2 Memory effects in the transmission channels

Memory effects of the communication channels, *e.g.*, time correlation in the channel gains, can be considered by assuming a Markov behavior. This modeling choice could be treated similarly to the scenario process S_k described in Section 2.2.2, *i.e.*, using an underlying process Z_k, with values in $\mathcal{Z}$, governed by an irreducible Markov chain with transition probability $p_{\mathcal{Z}}(z_{k+1}|z_k)$. Thus, for a given realization of this process $Z_k = z$, each channel state variable $C_{i,k}$ in the pair $(C_{1,k}, C_{2,k})$ is drawn with probability mass function $p_{C_i}(c|z) = \mathbb{P}(C_{i,k} = c|Z_k = z)$, $\forall c > 0,\ z \in \mathcal{Z}$. The reward function becomes $\tilde{g} = \ln(1 + c_1 q_1) + \ln(1 + c_2 q_2)$, where $c_i q_i$ represents now the SNR at node i. Finally, also the expressions of the long-term average reward $G(\mu, S_{-1}, Z_0, \mathbf{E}_0)$ and of the optimal policy $\mu^* = \arg\max_{\mu} G(\mu, S_{-1}, Z_0, \mathbf{E}_0)$ can be updated accordingly.

2.5.3 General number U of EHDs

Assuming the centralized system model presented beforehand, the extension from the two-user case to the general multi-user scenario is conceptually straightforward, although the notation becomes more involved and the complexity grows exponentially with U (the cardinality of the state space can be calculated as $\prod_{i=1}^{U}(e_{max,i}+1)$ and, for the case in which the battery of each EHD has capacity e_{max}, it becomes $(e_{max}+1)^U$). The number of energy quanta to be drawn from node i's buffer and devoted to transmission $Q_{i,k}$ is now valid for $i = 1, \ldots, U$, and is still chosen from the action space $\mathcal{Q}_i = \{0\} \cup \{q_{min,i}, q_{min,i}+1, \ldots q_{max,i}\}$, $\forall$ EHD i (as before, $Q_{1,k}, \ldots, Q_{U,k}$ cannot be simultaneously positive). All the other variables and equations in Section 2.2 can be updated according to the generic number U of users. The optimization problem involves a reward function that is nonzero if $Q_{i,k} \leq E_{i,k} \quad \forall i = 1, \ldots, U$ and, as in the case $U = 2$, can be separately discussed for the case of either perfect or imperfect SOC knowledge, updating (2.5), (2.7) or (2.8), respectively. Finally, for the particular choice of the reward function (2.12) discussed in Section 2.4.2, we have

$$\tilde{g}(\mathbf{q}) = \sum_{i=1}^{U} \ln(1+\alpha_i q_i). \tag{2.14}$$

2.6 Chapter conclusions

In this chapter, we considered a wireless sensor network composed by two Energy Harvesting Devices and a central controller. We investigated optimal transmission policies using as the performance metric the average long-term throughput of the system, and performing an optimization which took into account the maximum capacity of the two batteries, the mean energy harvesting rates and the SNR factors related to the transmission channels. When studying an asymmetric scenario, we showed that a good implementation strategy is to couple two devices whose battery capacities are such that one is half of the other: in this case the degradation incurred with respect to a scenario with both large battery capacities is smaller than 10%. In addition, motivated by the characteristics of practical implementations, we analyzed optimal policies for the case in which the State-of-Charge of the sensors cannot be perfectly known. The performance results for this case were fully analyzed, discussed and compared against the perfect SOC knowledge case, for a large variety of system parameters. We showed that the degradation of the performance caused by this phenomenon is mitigated when the capacity of the devices is large, as in this case the negative effects of energy outage and overflow are diminished: for sufficiently high battery capacities of the two EHDs, the performance degradation with respect to

the scenario with perfect SOC knowledge is smaller than 5%. Several extensions can be taken into account to generalize the adopted model. For instance, an important practical issue related to batteries is the degradation produced by frequent or strong discharges over time, which can gradually reduce the maximum storage capacities. Secondly, it would be possible to design a policy considering the history of both the harvesting and the transmission processes, which would allow a better battery control in the long run. Finally, an important extension could be the analysis of asymmetric decentralized access schemes for multi-user wireless sensor networks, in which each EHD benefiting from a particular energy arrival distribution is able to adapt its transmission policy accordingly.

Appendix to Chapter 2

For the reward function (2.12), an upper bound to the performance achieved when the SOC is perfectly known can be calculated solving the optimization problem

$$\begin{aligned} &\max_{q_1,q_2} && \mathbb{E}\left[\ln(1+\alpha_1 q_1)+\ln(1+\alpha_2 q_2)\right] \\ &\text{subject to} && \mathbb{E}[q_1]\leq \bar{b}_1,\ \mathbb{E}[q_2]\leq \bar{b}_2,\ q_1 q_2 = 0 \end{aligned}$$

as (2.1) imposes the constraints

$$\lim_{K\to+\infty} \sup \frac{1}{K}\mathbb{E}\left[\sum_{k=0}^{K-1} Q_{i,k}\right] \leq \bar{b}_i, \quad i=1,2$$

i.e., the mean steady-state energy used for transmission cannot exceed the mean energy harvesting rate, and q_1 and q_2 cannot be simultaneously positive. Consequently, we have that, for our particular scenario,

$$\ln(1+\alpha_1 q_1)+\ln(1+\alpha_2 q_2) = \ln(1+\alpha_1 q_1+\alpha_2 q_2)$$

and exploiting the concavity of this function,

$$\mathbb{E}\left[\ln(1+\alpha_1 q_1)+\ln(1+\alpha_2 q_2)\right] \leq \ln(1+\alpha_1\mathbb{E}[q_1]+\alpha_2\mathbb{E}[q_2])$$

which is upper-bounded by

$$\tilde{g}_{ub} = \ln\left(1+\sum_{i=1}^{2}\alpha_i \bar{b}_i\right). \tag{2.15}$$

Part II

Application of Machine Learning Techniques to Wireless Networking

CHAPTER 3

A MACHINE LEARNING BASED ETA ESTIMATOR FOR WIFI TRANSMISSIONS

Recent advancements related to Device to Device (D2D) communication make it possible for a transmitting node to dynamically select the interface to be used for data transfers locally, without traversing any network infrastructure. In this scenario, a controller has to be identified, whose goal is to manage the D2D connection after its establishment. The paradigm of Software Defined Networking (SDN) makes it possible to select this controller node via software: a device becomes the master node of a WiFi-Direct network, whereas the remaining units, i.e.*, the clients, have the possibility to exchange data with other devices through the master. This chapter develops a machine learning based prediction algorithm for the aforementioned scenario, in which multiple elements, while receiving data from the controller, require an accurate on-the-fly estimation of the remaining transmission time,* i.e.*, the Expected Time of Arrival (ETA). Different machine learning approaches are considered for this task, with the goal of exploiting only the information available at each client, without modifying any standard communication protocol. This information is critical when, for instance, a mobile user needs to decide whether or not to delay a data transfer, based on the load of the network and on the residual time under radio coverage from an access point.*

3.1 Introduction to the chapter

Nowadays, more than two billions of smartphones are active in the global market [59]. One of the most important features of a smartphone is to provide an Internet connection, giving each user the ability to access/share their data, *e.g.*, e-mails, photos, videos, *etc.*

Two main factors usually affect a smartphone's ability to stay connected and exchange data: the battery life and the amount of data provided by the cellular carrier subscription. These can be addressed by exploiting heterogeneous wireless interface cards (WNICs) such as WiFi and Bluetooth, in order to simultaneously minimize the battery consumption and reduce the amount of data transferred through the cellular network [60]. However, this ability to automatically exploit heterogeneous WNICs is not yet supported by any smartphone service; indeed, if a user wants to connect her/his device to a WiFi connection, she/he has to manually set it up, provided that she/he either knows the SSID and the WiFi encryption parameters, or decides that an open WiFi connection is worth trusting.

In the last few years, researchers in the Software Defined Networking (SDN) area have been focusing on creating a software protocol suite that does not need to be linked with the underlying hardware. This entails the possibility to, *e.g.*, prototype new protocols [61] and create network virtualization and traffic isolation (the reader can find a complete overview in [62]). The SDN paradigm goes in this direction by decoupling the network traffic into two planes: the control plane and the data plane. The control plane aims at monitoring routers, switches and Access Points (APs) to handle the network behavior and report any kind of useful information to a centralized unit, called controller. The controller can establish one or more data traffic flows among the network hosts by using the information gathered from the network status. These can be configured through some predefined Quality of Service network policies, such as offloading, number of hops between two users or energy consumption for the User Equipment (UE). Therefore, a controller can exchange information and manipulate the switch/router behavior by employing an ad hoc protocol like OpenSwitch [63]. On the other hand, the data plane handles all the data traffic generated by the host, which is completely separated from the control plane.

A further possibility is to integrate the device to device (D2D) communication into the SDN concept. D2D communication arises when two mobile nodes are able to communicate directly without traversing any infrastructure, *i.e.*, a base station (more details regarding the D2D concept can be found in [64]). D2D can generally exploit cellular spectrum (*i.e.*, inband) or unlicensed spectrum (*i.e.*, outband). In a D2D architecture the device network interfaces are expected to be overseen/controlled by a central entity *e.g.*, the SDN controller. Outband allows to eliminate the interference issues between D2D and cellular links, at the cost of an extra interface, as it usually adopts other wireless technologies, such as WiFi, Bluetooth or IEEE 802.15.4. Therefore, the SDN paradigm applied to D2D makes it possible to dynamically switch between different interfaces during transmission, based on predefined policies, *e.g.*, data offloading, traffic balance and battery energy

consumption. Hence, it is possible to design and implement a heterogeneous network environment where the device can exploit both network interfaces through the D2D concept in an unlicensed band. The idea to couple SDN and D2D has already been studied in [65–67], as well as the exploitation of the WiFi-Direct paradigm to set up a D2D network [68, 69] with several devices.

A natural way to apply the SDN concept is to exploit the Cognitive Networking (CN) [70–72] paradigm. CN deals with how wireless systems learn the relationships between the network status and behavior on one side, and its performance on the other, and plan and make decisions to achieve local, end-to-end, and network-wide goals. Differently from cognitive radio (CR) networking, CN tries to jointly optimize the different layers of the protocol stack. Novel MAC, routing, transport and application layer techniques for wireless networks have been proposed and analyzed by the networking community before, and can be adopted in a cognitive network. Mainly, the CN paradigm consists of two elements: the former is the ability to retrieve the network parameters from all layers of the protocol stack (sense phase), the latter is to exploit the knowledge retrieved from the sense phase to achieve two major goals, *i.e.*, increasing the network performance and predicting some network status (Cognitive Engine (CE)).

Finally, exploiting CN, it is possible to infer new network-related data by means of Machine Learning (ML) techniques. In [73], the authors survey a number of learning strategies able to classify IP packets and identify the applications that generate network traffic. A generic architectural model is formalized in [74], which focuses on the learning engine of CNs and adopts this model to describe two common problems in cognitive radio, namely capacity maximization and dynamic spectrum access. [75] presents an overview of potential ML implementation approaches, both related to learning and managing knowledge in CR applications. For instance, [75] suggests the usage of Bayesian techniques in order to dynamically retrieve user preferences regarding the perceived QoS level of an application, and estimate the future network behavior in terms of QoS capabilities.

In this work, we exploit CARMEN [76] to generate an SDN scenario, where a D2D network is configured with WiFi Direct. The SDN controller, installed into the centralized radio access network (C-RAN), exploits an LTE connection firstly to create a D2D network and secondly to select its master node. The aim is to predict client-side information related to the nodes generating data traffic in the same D2D network. This procedure should: i) not interact with any SDN controller; ii) be completely transparent for the user; and iii) avoid to alter any network or SDN protocol (*e.g.*, OpenSwitch/OpenFlow). This architecture could describe a scenario in which a large number of users is connected to an AP. Each of them is assumed to have high mobility, a short period of time available for data download, and a

transmission size greater than 50 MB. The data size could be realistically represented either by the usual average amount of data downloaded from an app running on a smartphone, or by the transfer of a digital map used in an offline navigation system, or even by a music video to be watched/listened to during commute. In any case, the user needs to know with high precision the correct amount of time required to perform the download; otherwise, he/she will not be able to take advantage of the services provided by the apps. In order to do that, all user nodes collect several network parameters in order to accurately predict the Estimated Time of Arrivals (ETAs) related to multiple transmissions from the AP to the clients. In this chapter, we will first show that using the number of simultaneously receiving nodes as an input datum, *i.e.*, assuming that this information is available at each device, and other network parameters already at the client side, a machine learning based algorithm is able to predict the total ETA with low error. However, this assumption is unrealistic in practice, as the number of receiving nodes is only known to the transmitter (the AP) which, in general, does not share this value unless a modification of the transmission protocol is introduced. Therefore, in the second part of this chapter, we will show how the number of active UEs can be accurately estimated by the receiving nodes via additional ML techniques, by taking as input only those network parameters available at the client side, in order to respect the requirement of not modifying any protocol. Finally, we will show how to combine the previous results to design an on-the-fly ETA prediction algorithm, which is able to accurately predict the ETA during the transmission. This algorithm will only exploit those network parameters available at each user's side, and its performance will be compared with the ETA predictions given by the unix program *scp*.

The rest of this chapter is organized as follows. In Section 3.2 we describe the testbed and the dataset employed in our experiments. In Section 3.3 and Section 3.4 we offer an overview regarding the ML techniques considered to predict the ETA and estimate the number of active nodes. Section 3.5 describes how those techniques can be combined in order to obtain an on-the-fly estimate of the ETA while transmissions take place. Finally, Section 3.6 validates our work, showing the prediction results and comparing them to the performance achieved by an analogous tool, commonly available in Unix-based operating systems. Section 3.7 concludes the chapter and proposes some future work.

3.2 Testbed overview

We conducted a thorough experimental data gathering to measure different network realizations and their related outputs. The data was collected using CARMEN [76], in a deployment composed of 50 Alix miniPCs model 3d2 with WiFi driver $ath9k$

and five Nexus 7 devices with WiFi driver *ath9k_htc*. In order to minimize the setup time of a specific WiFi network and quickly collect its performance, this testbed was associated with a software able to automatically gather and change the network configurations. This software can collect almost all TCP/IP stack parameters, starting from the MAC sublayer up to the Application layer, and to remotely set the WiFi transmission channel and power, as well as the distance among the Alix nodes and the transmitter (more details can be found in [76]). In addition, we exploited the GNU program *cron* to schedule all experiments at different parts of the day (PoD), in order to automate the entire data collection phase. Being the data transmissions dependent on the channel status, several WiFi network setups were considered, by changing all possible parameters in order to modify the scenario and consequently the ETA measurements. Since the testbed is located inside our department, other WiFi nodes can transmit data in the same WiFi transmission channel, generating interference during the experiments. Our idea to mitigate this issue was to measure transmission data multiple times over different WiFi frequencies (channels) and parts of the day (*i.e.*, morning, afternoon and night). For each time, we create a dataset varying the following network and topology parameters: transmit power {0 dB, 5 dB, 10 dB, 20 dB}, WiFi transmission channel {1, 6, 11}, number of nodes simultaneously receiving data from the transmitter (from 1 to 4) and distance between the transmitter and the receivers {1 m, 5 m, 10 m}. Finally, for each configuration, we measure the transmission duration of a 100 MB file[1]. The experimental campaign lasted more than 45 days, for a total of 540 different configurations: each of them has been tested 10 times, collecting 5400 sets of measurements in total. The network is deployed as a star single-hop topology, with a transmitting central node, called Access Point (AP), and one or more nodes waiting for data reception.

3.2.1 Dataset features

The features describing each example experiment have been expressed differently, depending on the type of information carried. WiFi channel and daytime are *categorical* data: the values taken by these variables belong to a number of classes, whose ordering is not relevant for our purposes. Therefore, we represent both these elements using binary vectors (*1-of-m* encoding) *e.g.*, morning is mapped as $(1, 0, 0)$, afternoon is $(0, 1, 0)$ and night is $(0, 0, 1)$ (the same coding holds for the transmission channel). The number of nodes simultaneously receiving data from the AP has also been expressed with binary vectors. However, to preserve dimensionality relations between the values taken by this variable, we have encoded it as a 4-dimensional

[1]The average size of a music video in full HD (1080p) resolution.

binary vector with as many 1s as the number of receiving nodes. In particular, one node is represented as $(1,0,0,0)$, two nodes correspond to $(1,1,0,0)$, *etc.* Finally, the transmission power and the distance from the AP to the receivers have been encoded as numerical variables. Consequently, each example in our input matrix $\mathbf{X}$ is expressed as a 12-dimensional vector, *i.e.*, the number of features is $n = 12$. Note that this particular data representation scheme, though clearly suboptimal from a compression point of view, has been designed in order to achieve better learning performance. Some ML techniques, including SVR/SVC machines and RBMs, have been observed to benefit from a pre-standardization of the input data provided to the learning algorithms. This is because, if the feature scales were highly different, those features with higher magnitude could overwhelm the others, leading to a weight adaptation sensitive only to a limited subset of features.

3.2.2 General data structure

ML algorithms are able to describe structural patterns in data by building mathematical models starting from a provided dataset. In this work, we will make use of *supervised* learning techniques, in which the dataset consists of a data matrix of inputs $\mathbf{X} \in \mathbb{R}^{M\times n}$ and a data matrix of outputs $\mathbf{Y} \in \mathbb{R}^{M\times l}$. The i-th rows of $\mathbf{X}$ and $\mathbf{Y}$ define a pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, for $i = 1, \ldots, M$, where $\mathbf{x}^{(i)} \in \mathbb{R}^n$ and $\mathbf{y}^{(i)} \in \mathbb{R}^l$ are the input feature vector (characterized by n features assuming different values) and the output label vector associated to it, respectively. Supervised learning consists in identifying the model which best predicts the target values corresponding to a given subset of the examples in $\mathbf{X}$. ML algorithms commonly divide the dataset into two separate sets, namely a *training set* $\left(\mathbf{X}_{tr} \in \mathbb{R}^{m\times n}, \mathbf{Y}_{tr} \in \mathbb{R}^{m\times l}\right)$ and a *test set* $\left(\mathbf{X}_{test} \in \mathbb{R}^{(M-m)\times n}, \mathbf{Y}_{test} \in \mathbb{R}^{(M-m)\times l}\right)$: the examples belonging to the former are exploited to train the learning algorithm, whereas the latter ones are used to evaluate its performance. This strategy has the goal of preventing *overfitting*, which occurs when the model starts to "memorize" training data rather than "learning" to generalize from trends. In this work, 80% of the data was assigned to the former, leaving the remaining 20% for performance testing.

The next two sections describe the ML techniques that were considered toward the full on-the-fly ETA prediction. In Section 3.3, we first show regression techniques, able to predict only a continuous initial ETA estimate for a transmission, assuming full knowledge about a given set of network parameters. Later, in Section 3.4, we present how one of these parameters (practically not available at the receiving nodes) can be detected from the others via suitable classification algorithms. Finally, Section 3.5 will explain how to combine the previous techniques in order to predict the ETA values during a transmission, exploiting only those network

parameters available at the receiving nodes.

3.3 Predicting the total ETA

In the first part of this work, we exploit all the $n = 12$ above-mentioned dataset features to predict the initial ETA of a number of wireless transmissions. To reduce the variability effects of the transmission channels, we computed the median over all the 10 experiments for each configuration, and use the resulting $M = 540$ examples as dataset. Hence, since here $l = 1$ (we only predict the initial ETA for the entire transmission), $\mathbf{X}_{tr} \in \mathbb{R}^{432\times12}$, $\mathbf{Y}_{tr} = \mathbf{y}_{tr} \in \mathbb{R}^{432}$, $\mathbf{X}_{test} \in \mathbb{R}^{108\times12}$, $\mathbf{Y}_{test} = \mathbf{y}_{test} \in \mathbb{R}^{108}$. In the following, we describe the ML techniques that have been used to predict the initial ETA values. Their performance comparison can be found in Section 3.6.1.

3.3.1 Multivariable linear regression

Linear regression is a technique that tries to model the data using linear predictor functions of the form

$$h_{\mathbf{w}}(\mathbf{x}^{(i)}) = w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)}, \tag{3.1}$$

with $i = 1, \ldots, m$ being the examples in the training set $\mathbf{X}_{tr}$. Here, the j-th feature of the i-th input example is denoted as $x_j^{(i)}$, for $j = 1, \ldots, n$. To simplify the notation, (3.1) can be rewritten including the bias term w_0 by letting $x_0^{(i)} = 1\ \forall i$ (this is the intercept term):

$$h_{\mathbf{w}}(\mathbf{x}^{(i)}) = \sum_{j=0}^{n} w_j x_j^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)}, \tag{3.2}$$

from which it is clear that the predicted values are linear functions of the weight parameters $\mathbf{w}$. The objective is to minimize the cost function

$$J(\mathbf{w}) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 + \lambda||\mathbf{w}||^2\right] \tag{3.3}$$

with respect to the weight vector $\mathbf{w}$. Note that (3.3) consists of two terms, the first accounting for the prediction error and the latter taking into account overfitting (λ is the regularization parameter). This optimization problem can be solved using the common technique of normal equations [77]: defining $\mathbf{X}$ to be the $m \times n$ matrix ($m \times (n + 1)$ including the intercept term) containing in its rows all the training examples, and $\mathbf{y}$ the $m \times 1$ vector consisting of the target values from the training

set, it can be shown that the optimal weight vector is:

$$\mathbf{w} = \left(\mathbf{X}^T \mathbf{X} + \lambda \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_n \end{bmatrix} \right)^{-1} \mathbf{X}^T \mathbf{y}, \tag{3.4}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

Despite the advantage of solving the optimization problem with the closed-form expression (3.4), the previous technique can suffer from limitations in the generalization of the model, due to its linearity in the input variables $x_1^{(i)}, \ldots, x_n^{(i)}$. A possible extension is to "expand" the n-dimensional input features' space into an l-dimensional space with $l > n$, in order to take into account possible non-linear dependencies between output and input variables. Thus, we build a new input data matrix $\mathbf{X}' \in \mathbb{R}^{M \times l}$ in which the first n columns are the same as in the original input matrix $\mathbf{X}$, whereas the other $l - n$ columns contain polynomial combinations of the original n input variables, up to a given maximum degree d. The fundamental idea behind this approach is that the expanded input matrix $\mathbf{X}'$ can be used with the aforementioned regression technique described earlier in this section. Therefore, it would be possible to perform a multivariable regression which is polynomial in the input variables and, as before, linear in the weights, so that (3.4) can still be used by substituting $\mathbf{X}'$ for $\mathbf{X}$. In the rest of the chapter, this extension of Linear regression will be denoted as "Extended Linear regression".

3.3.2 Support Vector Regression

In Support Vector Regression (SVR) machines [78, 79], the objective is to find a function $f_{\mathbf{w}}$ that has maximum absolute prediction error lower than a given constant ϵ for all the training data. This can be accomplished by defining the ϵ-insensitive error function

$$E_\epsilon(z) = \begin{cases} |z| - \epsilon & \text{if} \quad |z| > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

and minimizing the cost function given by

$$C \sum_{i=1}^{m} E_\epsilon(f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) + \frac{1}{2}||\mathbf{w}||^2, \tag{3.6}$$

where the second term, as in (3.3), accounts for regularization (by convention, differently from (3.3), the regularization parameter C is inverted and appears in front of the error function). The optimization problem in (3.6) assumes that all the training

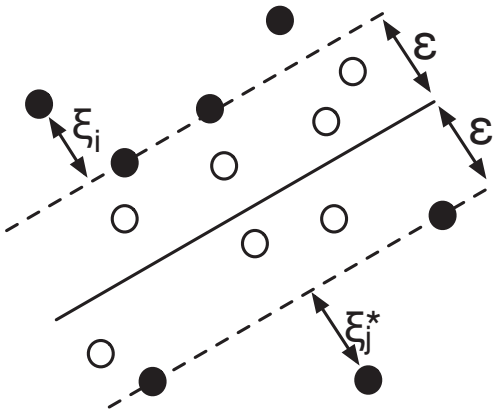


Figure 3.1: Graphical representation of an ϵ-tube with slack variables.

examples can be contained in an "ϵ-tube" (see Figure 3.1). However, this is not verified in general, and (3.6) can be modified so as to allow for some tolerance in the prediction errors. Therefore, for each training example $\mathbf{x}^{(i)}$, it is possible to introduce non-negative *slack variables* ξ_i and ξ_i^*, $i = 1, \ldots, m$, where $\xi_i > 0$ is related to a point for which $(y^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)})) > \epsilon$, and $\xi_i^* > 0$ is related to a point for which $(f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) < -\epsilon$.

Training examples are thus allowed to lie outside the ϵ-tube, as in Figure 3.1, provided that the corresponding slack variables are positive: this conditions can be formulated as

$$y^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)}) \leq +\epsilon + \xi_i \tag{3.7}$$

$$y^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)}) \geq -\epsilon - \xi_i^*. \tag{3.8}$$

The optimization problem becomes

$$\min C \sum_{i=1}^{m} (\xi_i + \xi_i^*) + \frac{1}{2}||\mathbf{w}||^2, \tag{3.9}$$

subject to the constraints $\xi_i, \xi_i^* \geq 0$, (3.7) and (3.8). It turns out that only the examples outside the ϵ-tube contribute to the cost, with deviations being linearly penalized. If, for the moment, we consider the case of $f_{\mathbf{w}}(\mathbf{x}^{(i)}) =< \mathbf{w}, \mathbf{x}^{(i)} > +b$ (where $< \cdot, \cdot >$ denotes the inner product), *i.e.*, $f_{\mathbf{w}}$ is simply a linear function of the inputs (b is a bias parameter), passing to its dual formulation and exploiting

Karush-Kuhn-Tucker conditions [80], (3.9) can be formulated as

$$\begin{aligned} \max_{\alpha_i,\alpha_i^*} \quad & -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*) < \mathbf{x}^{(i)}, \mathbf{x}^{(j)} > \\ & -\epsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i-\alpha_i^*) \\ \text{s. t.} \quad & \sum_{i=1}^{m}(\alpha_i+\alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \tag{3.10}$$

In (3.10), we maximize over α_i and α_i^*, which are the Lagrangian multipliers: it can be shown that

$$\mathbf{w} = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*)\mathbf{x}^{(i)}, \tag{3.11}$$

and the prediction function becomes

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) < \mathbf{x}^{(i)}, \mathbf{x} > +b. \tag{3.12}$$

Note that, in (3.11), the weight vector $\mathbf{w}$ is a function of only the training examples $\mathbf{x}^{(i)}$. Therefore, the complexity of the prediction function, differently from the methods described in Section 3.3.1, depends on the number of training examples m, and is independent of the number of input features n. Actually, it turns out that not all the m training vectors need to be considered in (3.11) and (3.12), as $\alpha_i - \alpha_i^* \neq 0$ only for a subset of elements, whose predicted values lie on or outside the ϵ-tube. As a result, the actual complexity of the predictive process is a function of the cardinality of this subset, whose elements are called *Support Vectors* (SVs). The formulation of (3.12) in terms of the SVs is known as the *sparse support vector expansion*. The reader could refer to [81, 82] for more details about learning in SVR machines and for the derivation of important additional parameters of the algorithm, such as the bias parameter b.

In (3.10), we assumed that the prediction function $f_{\mathbf{w}}$ was linear in each training example $\mathbf{x}^{(i)}$. Actually, this constraint can be relaxed so as to allow better generalization over non linear target functions. This can be done by noting that in (3.12) SVs only appear inside scalar products [83], and $\mathbf{w}$ need not be calculated explicitly. Therefore, it can be proved that $< \mathbf{x}^{(i)}, \mathbf{x} >$ in (3.12) can be replaced by particular non linear functions $k(\mathbf{x}^{(i)}, \mathbf{x})$, known as *kernels*, which correspond to scalar products between non linear transformations of their inputs. Substituting $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ in (3.10) and $k(\mathbf{x}^{(i)}, \mathbf{x})$ in (3.12), we thus obtain the optimal prediction function in a

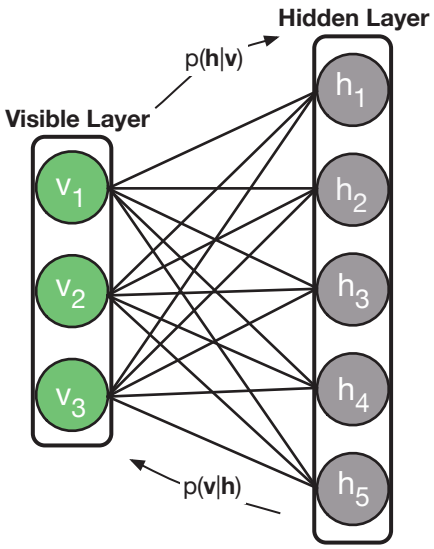


Figure 3.2: Graphical representation of an RBM.

non-linear *feature space*, rather than in input space:

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) k(\mathbf{x}^{(i)}, \mathbf{x}) + b. \tag{3.13}$$

3.3.3 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) [84] belong to the family of generative models known as Boltzmann machines, that are probabilistic graphical models in which a number of visible and hidden units are connected by means of symmetric weights (*i.e.*, weights are equal in both directions). Input data are given to the network through a layer of visible units, which are connected to another layer of so-called hidden units, whose task is to model the latent features inside the data, see Figure 3.2. In RBMs each configuration of the variables of interest is associated to an energy function E, and the goal of the learning algorithm is to shape this function according to a specific objective. Probabilistic models define a probability distribution through the energy function

$$p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}, \tag{3.14}$$

where $\mathbf{v}$ and $\mathbf{h}$ are the column vectors containing the values of the visible and hidden units, respectively, and Z, called partition function, is a normalizing factor[2]. The common choice is that desirable configurations of $\mathbf{v}$ and $\mathbf{h}$ correspond to high values of $p(\mathbf{v}, \mathbf{h})$, and therefore must have low energy. Differently from standard Boltzmann

[2]In the previous ML algorithms we denoted inputs as $\mathbf{x}$: RBM inputs are instead labeled as $\mathbf{v}$ for historical reasons.

Machines, in which both visible and hidden nodes are fully connected, RBMs are assumed to have no connections between nodes belonging to the same layer. Thus, the hidden (visible) variables are independent given the state of the visible (hidden) variables and the energy function can be formulated as

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v}. \tag{3.15}$$

In (3.15), $\mathbf{b}$ and $\mathbf{c}$, called unit biases, are parameter vectors associated to visible and hidden units, respectively, whereas $\mathbf{W}$ is the weight matrix modeling connections between nodes belonging to different layers.

As the input data provided to the learning algorithm consists of the set of training examples, the goal of the learning process is to maximize the probability of these examples $p(\mathbf{v})$ with respect to the weights and biases in (3.15). This can be accomplished by performing gradient descent over the likelihood function of the training data, whose derivative with respect to each weight can be calculated as [84]:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial w_{ij}} + \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v}, \mathbf{h}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial w_{ij}}. \tag{3.16}$$

Note that the first term in (3.16) represents the empirical expectation over the training set, while the second term is the expectation under the actual model distribution. Finally, once (3.16) is evaluated, each weight is updated, at each step of the learning process, using gradient descent methods. For the second element in (3.16), it can be shown that

$$\sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v}, \mathbf{h}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial w_{ij}} = -\sum_{\mathbf{v}} p(\mathbf{v}) \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) h_i v_j \tag{3.17}$$

exploiting the fact that $\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}} = -h_i v_j$. This leads to the final expression for (3.16):

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = p(h_i = 1|\mathbf{v}) v_j - \sum_{\mathbf{v}} p(\mathbf{v}) p(h_i = 1|\mathbf{v}) v_j, \tag{3.18}$$

where, exploiting the linear relation between variables and weights in (3.15),

$$p(h_i = 1|\mathbf{v}) = \sigma\left(c_i + \sum_j w_{ij} v_j\right) \tag{3.19}$$

and σ is the sigmoid logistic function: $\sigma(z) = (1 + \exp(-z))^{-1}$.

As summing over all values of the visible variables in (3.18) would have exponential complexity, a common approach is to approximate the expectation by samples from the model distribution using Gibbs sampling. This technique requires running

a Markov chain until it converges to stationarity, but still has large computational costs due to the large number of sampling steps needed. As a result, the network has been trained using the *constrastive divergence* algorithm [85], which is now the standard way to train RBMs, consisting in the alteration of a positive and a negative phase. Initially, after visible units are clamped to the values of the training data, nodes' activations are propagated to hidden units, according to the weights of the connections, using (3.19). The vector of hidden unit activations constitutes an internal representation of the pattern observed in the visible units. In the second phase, the activations are propagated backwards to the visible units with an equation similar to (3.19), fixing the hidden units, with the goal of reconstructing the original input vector as accurately as possible. The reader could refer to [86,87] for more details about learning in RBMs and for the explanation of important additional parameters of the algorithm. At the end of this unsupervised learning phase, the values taken by the units in the hidden layer provide an alternative and more expressive representation of the input vector: the idea behind this approach is that of learning a generative model (the RBM) able to capture an abstract representation of the input data and to better describe non linearities between raw input and output variables. Finally, these representations are treated as inputs for a subsequent supervised learning algorithm such as the linear regression method analyzed in Section 3.3.1. In other words, the hidden units of a properly trained RBM are expected to learn more descriptive internal representations of the input data, which should be more easily separable by a linear regression method. Therefore, although the RBM alone is an unsupervised learning algorithm, the pair composed of RBM and linear regressor practically defines a supervised learning technique.

3.4 Estimating the number of receiving nodes

In Section 3.3, we used the number of active nodes N as an input feature to predict the ETA. However, this information is usually available at the transmitting device only, which, in general, does not share this value unless a modification of the transmitter protocol is introduced. Therefore, in this section, we investigate how to derive the correct value of N at the receiving nodes, only exploiting those features actually available at the receivers' side: we use different subsets of the network parameters as input features, with the goal of understanding how the knowledge of an additional feature affects the final classification performance. We tested four parameters subsets: the first subset contains only the total ETA ($n = 1$), *i.e.*, the total transmission duration, whereas the others include either the total ETA and the transmission power (ETA, P_{Tx}) or the total ETA and the distance (ETA, d) ($n = 2$), or all three features (ETA, P_{Tx}, d) combined ($n = 3$). Other parameter selections were

also tested, including adding the information about the PoD and the transmission channel, but the improvement over the results showed here was found to be negligible. For this task, we exploit all the 5400 experiments collected from our testbed, as described in Section 3.2, equally distributed among the 4 classes ($N = 1$, $N = 2$, $N = 3$, $N = 4$). Since the goal is to correctly identify the value of N, here $l = 1$, hence $\mathbf{X}_{tr} \in \mathbb{R}^{4320 \times n}$, $\mathbf{Y}_{tr} = \mathbf{y}_{tr} \in \mathbb{R}^{4320}$ and $\mathbf{X}_{test} \in \mathbb{R}^{1080 \times n}$, $\mathbf{Y}_{test} = \mathbf{y}_{test} \in \mathbb{R}^{1080}$. Analogously to Section 3.3, a comparison of the results derived from the different techniques will be shown in Section 3.6.2.

3.4.1 Naive Bayes classifier

The Bayesian classifier is a probability model that computes the posterior probability $p(y_k|\mathbf{x})$ of a class given an input example $\mathbf{x}$, for each of the k possible classes. Using Bayes' rule, this probability is computed as

$$p(y_k|\mathbf{x}) \propto p(\mathbf{x}|y_k)p(y_k) \ , \tag{3.20}$$

where the posterior probability is proportional to the likelihood $p(\mathbf{x}|y_k)$ and the prior $p(y_k)$. The object $\mathbf{x}$ is classified under the class that holds the highest posterior probability:

$$y^*_{\text{MAP}} = \arg \max_k p(\mathbf{x}|y_k)p(y_k), \tag{3.21}$$

known as Maximum A Posteriori (MAP) formulation. The likelihood expresses how probable the input data $\mathbf{x}$ is for a given class y_k, whereas the prior captures the assumptions about the class, before observing the data, in the form of a probability distribution $p(y_k)$. If no prior knowledge about the class distribution is available, a uniform prior $p(y_k) = c$ is assumed and (3.21) turns into the Maximum Likelihood (ML) solution $y^*_{\text{ML}} = \arg \max_k p(\mathbf{x}|y_k)$. The likelihood function can be modeled using different probability distributions: in this work we used both a Gaussian and a Poisson *pdf*, setting their parameters (mean and variance for the former, mean for the latter) according to the examples in the training set. In particular, we ran the classification algorithm with input examples $\mathbf{x}$ of different dimensions, based on the number of features n involved: the idea behind this approach is to analyze how additional knowledge on an experiment impacts the performance of the classifier. Therefore, the input vector contains either the ETA alone, or the pair (ETA, P_{Tx}) or (ETA, distance), or the triple (ETA, P_{Tx}, distance), and different *pdf*s are generated accordingly. For instance, for the second case ($n = 2$) with ETA and distance (d)

as input features, (3.21) is reformulated as

$$y^*_{\text{MAP}} = \arg\max_k p(ETA|y_k, d)p(d)p(y_k), \tag{3.22}$$

where $p(d|y_k) = p(d)$ for a homogeneous dataset.

3.4.2 Support Vector Classification

Support Vector Classification (SVC) machines [82] identify the optimal hyperplane maximizing the margin which separates different classes in a multi-dimensional feature space. A generic hyperplane can be written as the set of points $\mathbf{x}$ satisfying $\mathbf{w}^T\mathbf{x} - b = 0$, where $\mathbf{w}$ is the normal vector to the hyperplane and b represents the offset from the origin along $\mathbf{w}$. When the data points are linearly separable, it is possible to identify a pair of hyperplanes able to perfectly separate all the objects. By defining these hyperplanes as $\mathbf{w}^T\mathbf{x} - b = \pm 1$, the margin, *i.e.*, the region in-between, has length $\frac{2}{||\mathbf{w}||}$, and the objective is to maximize $\frac{1}{||\mathbf{w}||}$. If we consider a binary classification problem, $y^{(i)} \in \{-1, 1\}$, the maximization of $\frac{1}{||\mathbf{w}||}$ is equivalent to the minimization of $\frac{1}{2}||\mathbf{w}||^2$; therefore the problem can be formulated as:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \ i = 1, \ldots, m, \end{aligned} \tag{3.23}$$

where the constraint $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$ has been added to guarantee that all $\mathbf{x}^{(i)}$s lie outside the margin. In (3.23), we assumed that all training examples can be separated, which is not verified in general. As a result, (3.23) can be modified to allow for some tolerance in the classification error. Therefore, for each training example $\mathbf{x}^{(i)}$, there is an associated non-negative slack-variable $\xi_i \geq 0$, $i = 1, \ldots, m$, allowing each example to lie on the other side of the hyperplane separating its class. The optimization problem then becomes:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ i = 1, \ldots, m, \end{aligned} \tag{3.24}$$

where the parameter C balances the trade-off between having a large margin and ensuring that most examples lie in the region associated to their class. The "dual"

formulation of Eq. (3.24) is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m, \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned} \tag{3.25}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ is the vector of Lagrange multipliers. It can be shown that

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)}, \tag{3.26}$$

by which it is possible to calculate the optimal weights in terms of the optimal values of $\boldsymbol{\alpha}$. Now, a prediction for a new example input $\mathbf{x}$ can be performed calculating $\mathbf{w}^T \mathbf{x} + b$, and predicting $y = 1$ (or $y = -1$) if this quantity is bigger (or smaller) than zero. Hence, the prediction function can be written as

$$f_{\mathbf{w}}(\mathbf{x}) = sgn \left(\sum_{i=1}^{m} \alpha_i y^{(i)} (\mathbf{x}^{(i)})^T \mathbf{x} + b \right), \tag{3.27}$$

which only depends on the inner product between the input vector $\mathbf{x}$ and the subset of training vectors $\mathbf{x}^{(i)}$ for which $\alpha_i \neq 0$. Moreover, from (3.25) it turns out that this subset only contains those training examples, known as Support Vectors (SVs), lying within the margin or in the region of the hyperspace belonging to the other class: as a consequence, the actual complexity of the prediction process only depends on the number of support vectors. The inner product $(\mathbf{x}^{(i)})^T \mathbf{x} + b$ in (3.27) can be replaced by particular non linear functions $k(\mathbf{x}^{(i)}, \mathbf{x})$, known as *kernels*, which correspond to scalar products between either linear or non linear transformations of their inputs. Substituting $k(\mathbf{x}^{(i)}, \mathbf{x})$ in (3.27), we thus obtain the optimal prediction function in a non-linear feature space, rather than in input space:

$$f_{\mathbf{w}}(\mathbf{x}) = sgn \left(\sum_{i=1}^{m} \alpha_i y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x}) \right). \tag{3.28}$$

Finally, as the classification problem studied here involves $|C| = 4$ classes, we adopted a generalization of the standard SVM known as multiclass SVM [77], which works by reducing the multiclass problem into a number of binary problems. This generalization, known as one-vs-one classification, works by building a set of $|C|(|C|-1)/2$ binary classifiers, and then selecting the class that is assigned by the majority of the classifiers.

3.4.3 k-Nearest Neighbor

In k-Nearest Neighbor (k-NN), an example $\mathbf{x}$ is classified under the most common class among its k nearest neighbors: k-NN is a type of lazy learning, since a learning phase is not needed at all. The neighbors are taken from a set of examples for which the class is known: this set can be thought of as the training set for the algorithm. In other words, the training phase of the algorithm simply consists in storing the training examples: once a test example is provided, the algorithm classifies it by assigning the most frequent class among its k nearest training examples. The parameter k is chosen so as to balance the trade-off between reducing the effect of noise (large k) and avoiding the creation of indistinct boundaries between the classes (low k).

3.5 On-the-fly ETA prediction

In this section, we show how to join the network parameters available at each receiving node, *i.e.*, WiFi channel, PoD, transmit power and distance between the transmitter and the receiver, together with the number of receiving nodes N detected using the ML techniques in Section 3.4. The goal is to provide an on-the-fly prediction of the ETA signal during the transmission.

For this final task, we use our complete dataset of $M = 5400$ examples. Since each experiment consists in the transmission of a 100 MB file, after an experiment has been launched, we use the time needed for the transmission of the first 5 MB (5% of the total size) to detect the value of N, following the techniques in Section 3.4. Then, we dynamically predict the remaining ETA values, using all the network parameters and the size of the fraction of the file not yet transmitted (decreasing over time). ETA estimates are computed every time a MB has been successfully transmitted, for a total of 95 ETA predictions *i.e.*, from 95 MB to 1 MB.

We chose to detect the value of N after 5 MB in order to properly take into account the trade off between the accuracy in the estimation, the variability in the transmission channel, and the time the user has to wait before receiving the first ETA prediction.

3.6 Results

In this section, we first describe our intermediate results, *i.e.*, predicting the initial ETA (Section 3.6.1) and detecting the number of active receivers N in the network after a transmission has been completed (Section 3.6.2). Finally, in Section 3.6.3, we show how we predict the ETA on-the-fly during the entire transmission, comparing

our results with *scp*. *scp* is a means of securely transferring computer files between a local host and a remote host or between two remote hosts, and is based on the secure shell (SSH) protocol. The source code is fully available online: we partially modified it to easily gather the ETA predictions as well as the actual time to transmit a file.

The hyperparameters of each ML mathematical model (which include the values assigned to λ in (3.3), C and ϵ in (3.6), the kernel function for the SVR/SVC, the number of the nearest neighbors k, *etc*) have been optimized using a *k-fold Cross Validation* (CV) technique [88] with $k = 15$, according to which the training set is split in k subsets and $k-1$ of these are used to train, leaving the k-th for testing. This procedure is iterated k times, so that, at the end, each subset has been used to assess the goodness of the model. The performance measure resulting from k-fold CV is then the average of the error values computed in the k loops, and the hyperparameters obtaining the lowest prediction errors are then chosen to train the final model.

3.6.1 Initial ETA prediction

In order to evaluate the performance of our regression algorithms, we computed, for each model, the root mean square error (RMSE) between the predicted value $\hat{\theta}$ of the ETA and the measured one θ from the test set. The RMSE is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M-m} \left(\hat{\theta}^{(i)} - \theta^{(i)}\right)^2}{M-m}}, \tag{3.29}$$

where $M-m$ is the size of the test set. We adopted this metric because it allows to aggregate the magnitudes of the prediction errors, obtained for various examples, into a single measure of predictive power.

In order to obtain a performance measure as accurate as possible, each technique was trained and tested on 10 different pairs $\left((\mathbf{X}_{tr}^{(i)}, \mathbf{y}_{tr}^{(i)}), (\mathbf{X}_{test}^{(i)}, \mathbf{y}_{test}^{(i)})\right)$, $i = 1, \ldots, 10$. Each pair was randomly defined using different seed values. The total number of considered mathematical models was 332850. In Table 3.1, we report the normalized RMSE performance values obtained by each technique as a function of the seed values. In particular, the normalization was performed by dividing the RMSE values by the actual measured total ETA. Finally, the last two rows report the average and the standard deviation of the values, in order to favor a direct comparison among the algorithms.

It can be noticed that both Extended Linear Regression and SVR methods perform well and achieve an average normalized RMSE less than 9%. In addition, for each ML method, we computed the standard error (SE) of the mean equal to $\sigma/\sqrt{n}$, where $n = 10$ is the number of seeds. The upper and lower 95% confidence limits,

Table 3.1: Normalized RMSE predicted values using the considered ML algorithms.

Seed	Linear Regression	Ext. Linear Regression	SVR	RBM
1	0.1107	0.0867	0.0841	0.1122
2	0.1229	0.0981	0.0944	0.1236
3	0.1006	0.0805	0.0815	0.1038
4	0.1189	0.0820	0.0773	0.1215
5	0.1408	0.1039	0.0985	0.1454
6	0.1072	0.0817	0.0872	0.1122
7	0.1161	0.0967	0.0969	0.1140
8	0.1058	0.0731	0.0763	0.0995
9	0.1003	0.0902	0.0850	0.1038
10	0.1462	0.0992	0.0952	0.1431
μ	0.1170	0.0892	0.0876	0.1179
σ	0.0159	0.0100	0.0082	0.0158

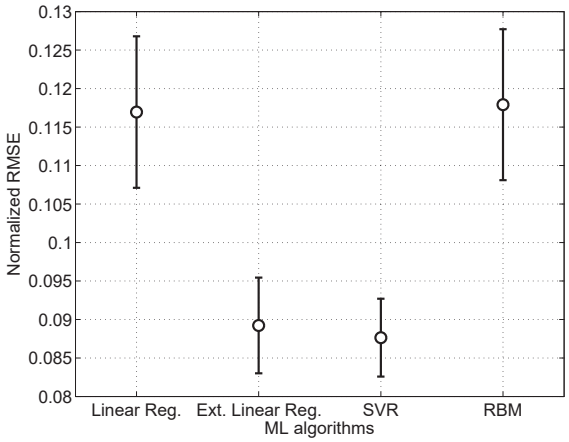


Figure 3.3: Confidence intervals for the considered ML methods.

depicted in Figure 3.3, allow to state that SVR and Ext. Linear Regression achieve better accuracy among the considered techniques and, in particular, SVR seems to perform best.

We compared the results obtained against the predicted ETAs using the *scp* program. As already discussed, *scp* does not take into account any information about the network to compute the ETA prediction, because it only considers the first transferred file chunk to compute the ETA predictions. Figure 3.4 reports a typical ETA prediction trend produced by *scp*, for an example of the dataset. It can be seen that the initial *scp* prediction (circle marker) is far from the actual measured transmission time (cross marker), resulting in a prediction error of 76.3%. Instead, it is possible to see that all the ML techniques achieve better results, giving an average prediction error of 13.7%. In addition, Figure 3.4 shows the lines corresponding to

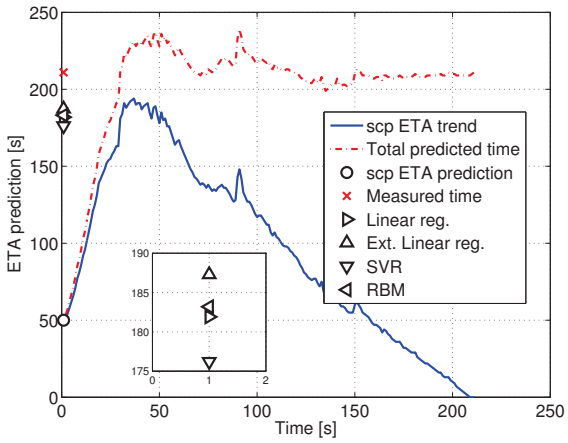


Figure 3.4: ETA estimation time using *scp* and the ML techniques with a four-node network topology.

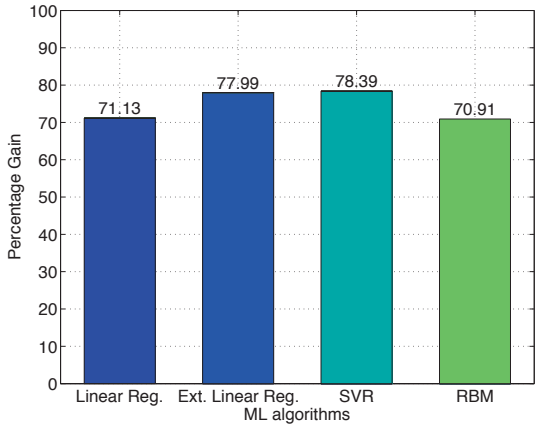


Figure 3.5: Comparison of the performance gain of the considered ML algorithms, in terms of reduction of the prediction RMSE with respect of the *scp* baseline.

scp ETA trend and total predicted time. The former represents the residual ETA estimate while the transmission occurs, whereas the latter reports the prediction of the total transmission time. For example, at $t = 100$ s, the solid line predicts $t = 117$ s to complete the transmission, and the dotted one correspondingly reports a total transmission time of $t = 217$ s. It can be seen (solid line) that the ETA estimate at $t = 1$ s is far from the measured time necessary to complete the data transfer: actually, an accurate estimate of the effective total transmission time occurs around $t = 100$ s (dotted line).

In Figure 3.5 we show the performance gain between the two approaches, computed as $(\text{RMSE}_{scp} - \text{RMSE}_{ML})/\text{RMSE}_{scp}$, experienced by choosing the ML techniques rather than the *scp* ETA predictions. The *scp* prediction errors were computed over the same test sets created to compare the ML algorithms in Table 3.1. Note that all the *scp* data transfers belonging to the same network configuration

Table 3.2: Percentage ETA prediction errors (mean ± standard deviation).

	$N_{pred} = 1$	$N_{pred} = 2$	$N_{pred} = 3$	$N_{pred} = 4$
$N_{real} = 1$	9.57 ± 4.68	89.90 ± 16.55	179.19 ± 26.68	282.57 ± 42.20
$N_{real} = 2$	46.76 ± 5.79	6.38 ± 5.26	47.90 ± 16.19	103.92 ± 29.58
$N_{real} = 3$	63.96 ± 4.63	32.63 ± 7.86	6.22 ± 4.42	38.23 ± 12.83
$N_{real} = 4$	73.42 ± 2.56	50.54 ± 5.09	26.55 ± 7.27	3.85 ± 2.82

were forced to start at the same time instant. The reason was to avoid that, in network configurations with multiple receivers, the first prediction benefited from temporary advantageous conditions where both the transmitter node and the WiFi channel were unloaded. The average normalized RMSE for *scp* ETA prediction was 0.4053: as the corresponding result for the SVR method was 0.0876, the maximum percentage gain of the ML techniques over *scp* is 78.39%.

3.6.2 Number of nodes detection

In the previous section, we showed how an accurate initial estimate of the ETA for a wireless transmission can be computed exploiting a number of network parameters. However, as already noted, not all of these parameters are available at each network receiving node, if current standard transmission protocols are used. In this section, we show the performance results related to the detection of the number of nodes N simultaneously receiving data from the AP. We claim that the exact number of users in the network is an important piece of information, as the predicted ETA is highly dependent on the value of N. To prove this, in Table 3.2 we show the percentage mean prediction errors on the ETA, as a function of the number of users given as input, computed using the best-performing ML technique in Section 3.6.1. In the table, we indicate with N_{real} the actual number of active nodes during a given experiment, and with N_{pred} the value of N given as input to the ETA predicted by the algorithm. The goal is to compare the ETA prediction errors when an erroneous number of nodes is used instead of the right one. For each experiment in the test set, the ETA has been predicted with a given value of N_{pred}, and a percentage error has been computed with respect to the measured ETA value. The average of all these errors, together with their standard deviation, is reported in Table 3.2 for each (N_{real}, N_{pred}) pair. Note that the errors on the diagonal are related to the predictions performed using the "right" value of N ($N_{pred} = N_{real}$); as expected, these values are the lowest for each class of experiments. The worst case is when the algorithm predicts a number of nodes greater than 1, $N_{pred} > 1$, when $N_{real} = 1$. In addition, when $N_{real} = 2$ (or $N_{real} = 3$), using the nearest value of N for the prediction, *i.e.*, either $N_{pred} = 1$ or $N_{pred} = 3$ (or $N_{pred} = 2$ or $N_{pred} = 4$, respectively), gives

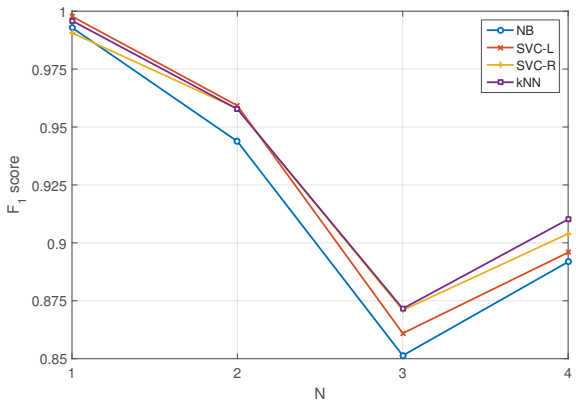


Figure 3.6: Classification performance with $n = 3$ (ETA, P_{Tx}, d).

approximately symmetric mean percentage errors.

These results show how the value of N used as feature in the previous techniques has a fundamental role in the accuracy of the ETA prediction. Therefore, we now present the results related to classification algorithms presented in Section 3.4, *i.e.*, Naive Bayes (NB), Support Vector Classification with linear (SVC-L) and radial (SVC-R) kernels, and k-Nearest Neighbor (kNN), with the goal of identifying the techniques able to detect the value of N as accurately as possible.

Since we deal with a multi-class classification problem, we adopt the F_1 score as a measure of performance for each class [89]:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{3.30}$$

where P is the precision, *i.e.*, the class agreement of the data labels with the positive labels given by the classifier, and R is the recall, that is the effectiveness of a classifier in identifying positive labels [90]. In addition, we analyze the performance in terms of Receiver Operating Characteristic (ROC) curve which shows the fraction of true positive decisions of a classifier for a given rate of false positive decisions [91].

The classification performance is shown in Figure 3.6. For each class (number of nodes $N \in \{1, \ldots, 4\}$), we plot the F_1 score of each classifier when all the features are used (ETA, P_{Tx}, and distance). In general, the performance is very high (F_1 score above 0.85) and it is possible to identify a common trend for all the classifiers: the 3-node class is harder to classify, mainly because the "tails" of this class overlap with the 2-node and 4-node classes. Table 3.3 summarizes the performance for each class and the total F_1 score given a particular set of features. Increasing the number of exploited features generally gives better performance for each classifier.

In Figure 3.7, we show the analysis of the classification performance in terms of the ROC curve. This curve can be used to select the suitable operating point of the classifier; in particular, we see that at 20% rate of false positives, all the classifiers

Table 3.3: F_1 score performance comparison among the classification techniques.

Features	Classifier	$N = 1$	$N = 2$	$N = 3$	$N = 4$	Average
ETA	NB	0.986	0.929	0.818	0.863	0.899
	SVC-L	0.996	0.952	0.845	0.884	0.919
	SVC-R	0.997	0.950	0.841	0.888	0.919
	kNN	0.997	0.948	0.825	0.875	0.911
ETA, d	NB	0.990	0.917	0.813	0.877	0.899
	SVC-L	0.997	0.953	0.840	0.877	0.917
	SVC-R	0.996	0.960	0.861	0.899	0.929
	kNN	0.997	0.960	0.854	0.892	0.926
ETA, P_{Tx}	NB	0.989	0.939	0.84	0.879	0.912
	SVC-L	0.997	0.958	0.851	0.887	0.923
	SVC-R	0.992	0.957	0.842	0.876	0.916
	kNN	0.996	0.958	0.842	0.879	0.919
ETA, P_{Tx}, d	NB	0.993	0.944	0.851	0.892	0.920
	SVC-L	0.998	0.959	0.861	0.896	0.928
	SVC-R	0.991	0.958	0.871	0.904	0.931
	kNN	0.996	0.958	0.872	0.910	0.934

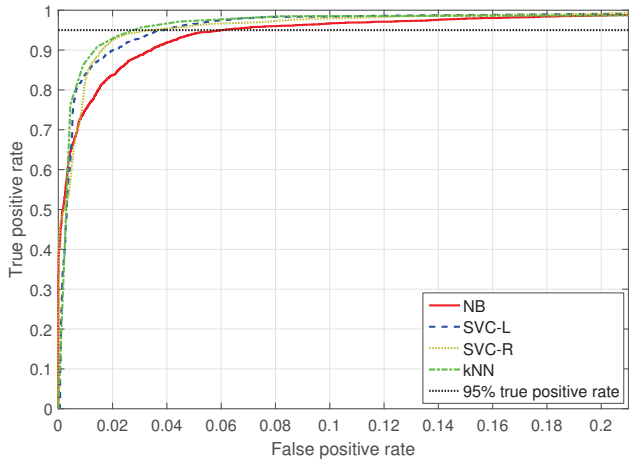


Figure 3.7: ROC curves for the considered classification techniques.

achieve an almost perfect performance. It is also worth noting that between 3% and 6% of false positive rate all the classifiers achieve the 95% true positive rate (dotted line). Table 3.4 shows the cutoff points for the false positive rate that is required to achieve at least 95% of true positive rate. Figure 3.7 and Table 3.4 confirm that the NB approach is the slowest to reach the best performance.

3.6.3 On-the-fly ETA prediction

We finally present the results related to the on-the-fly ETA prediction. As already mentioned in Section 3.5, we exploited all the 5400 experiments collected with our

Table 3.4: False positive rate @ true positive rate = 0.95.

Classifier	ETA, P_{Tx}, d
NB (prior)	0.060
SVC-L (no weight)	0.046
SVC-R (no weight)	0.035
kNN	0.030

Table 3.5: F_1 scores after 5 MB.

Features	$N = 1$	$N = 2$	$N = 3$	$N = 4$	Average
ETA	1.0000	0.9615	0.8519	0.8571	0.9176
ETA, d	1.0000	0.9615	0.9091	0.9091	0.9449
ETA, P_{Tx}	1.0000	0.9615	0.8889	0.8929	0.9358
ETA, P_{Tx}, d	1.0000	0.9615	0.9091	0.9091	0.9449

testbed: since these correspond to 540 different network configurations, we used 4320 experiments for training (maintaining each of the 10 different realizations per experiment), and 108 for performance testing (as the considered ML algorithms have deterministic outputs, we randomly selected a single realization per testing configuration). In the following, we decided to use the best-performing ML techniques from Section 3.3 and Section 3.4, leaving a full comparison among all the previously considered techniques as a future research effort. Nevertheless, the hyperparameters of the selected algorithms were still optimized through k-fold cross validation.

We first trained an SVC with the aforementioned training set, with the goal to identify the best classification model able to detect the number of active users N given the amount of time needed to transmit the first 5 MB of a 100 MB file, the transmission power P_{Tx} and the distance d. For the sake of clarity, we highlight that, differently from the dataset described in Section 3.4, here we use the transmission time for the first 5% of the total file size, whereas, in Section 3.4, we waited until the transmission was completed (100%).

Table 3.5 shows the classification accuracy as a function of the features exploited to detect the number of active users N after the first 5 MB have been successfully transmitted. As expected, increasing the number of features gives better results: using the ETA, the transmission power P_{Tx} and the distance d together, the classification model has an F_1 score of 0.9449 (averaged over all the classes), and 101 out of the 108 experiments in the testing set are correctly classified. Interestingly, adding extra features to the dataset only improves the classification performance of the higher classes ($N \geq 3$), and, for this specific task, knowing the distance d from

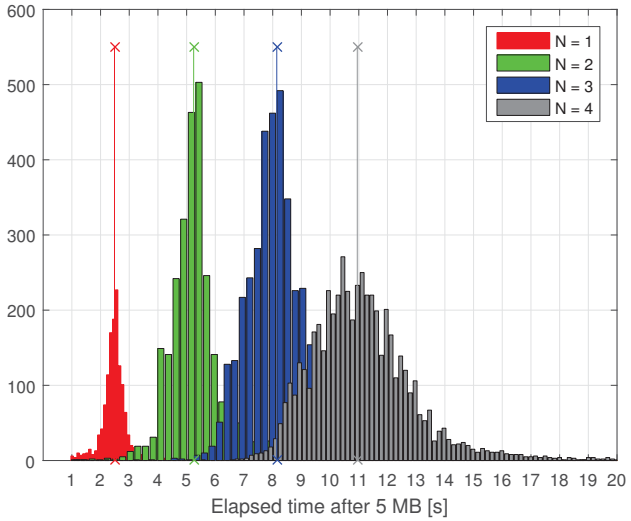


Figure 3.8: Histogram of time needed to transmit the first 5 MB.

the AP to the receivers turns out to be more beneficial than the knowledge about the WiFi transmission power P_{Tx}. This can be explained noting that, when distances are not substantial, even low values of P_{Tx} are able to achieve high bitrates, hence the information about the exact value of the transmission power is not as beneficial as the knowledge of the distance from the AP to the receiving node.

Figure 3.8 presents a histogram of the amounts of time needed for the transmission of the first 5 MB, as a function of the number of active users N receiving data in the network (the figure is related to the entire dataset). Median values for each value of N have been overlapped as vertical lines. It is worth noting that, although the relation among the median values seems to be linear (*e.g.*, the median values for $N = 2$ and $N = 4$ are, respectively, almost two and four times the median value for $N = 1$), and classes $N = 1$ and $N = 2$ are mostly separable, both the presence of a non-negligible number of outliers (not shown in the Figure for the sake of readability), and the considerable overlapping between classes $N = 4$ and $N = 3$, suggest that a linear separator (*i.e.*, a linear kernel) would not be powerful enough to distinguish among the examples belonging to each class. Indeed, we found the non-linear RBF kernel to work best for this dataset.

Secondly, after the actual value of N has been detected, we use an SVR to predict the remaining ETA during the transmission of the last 95 MB: to do this, we added a feature to the $\mathbf{X}$ matrix first described in Section 3.2.1, consisting of the residual amount of data to be transmitted.

Table 3.6 shows the normalized RMSE values for the on-the-fly ETA predictions. The first row is related to the case described above, where we first detect the number N of active users using the SVC ($N = N_{pred}$), and then we predict the ETA values on-the-fly during the transmission. In the second row, as a comparison, we add

Table 3.6: Normalized RMSE for the on-the-fly ETA predictions.

	$N = 1$	$N = 2$	$N = 3$	$N = 4$	Average
$N = N_{pred}$	0.0160	0.0215	0.0400	0.1171	0.0486
$N = N_{real}$	0.0160	0.0268	0.0439	0.0855	0.0430
scp	0.0293	0.0366	0.0950	0.1636	0.0811

the normalized RMSE values related to the case in which we always use the correct value of N ($N = N_{real}$), practically bypassing the initial classification task. In the first case, we found that the average normalized error is 0.0486 and the RMSE is increasing in the value of N. This can be explained by noting that examples belonging to class $N = 1$ have a lower variability than the ones from class $N = 4$ (see Figure 3.8). This is due to the randomness of the backoff algorithm implemented in the WiFi transmission protocol: a more populated scenario ($N = 4$) will present a more disturbed transmission channel with respect to a situation in which only one device ($N = 1$) is receiving data and is less prone to interference. As expected, we found that the overall RMSE is lower in the second case ($N = N_{real}$), since no additional errors originated by the SVC are introduced. Due to the incremental overlapping of the curves in Figure 3.8, it can be noted that the RMSE is similar among the two cases for $N = 1$ and $N = 2$, whereas worse performance arises for $N \geq 3$, due to mutual misclassifications among these two classes (a non negligible number of $N = 3$ examples is wrongly associated with class $N = 4$, which hence presents a higher error). Note that these RMSE values are lower than those obtained in Table 3.1: this is because the former (the initial ETA estimates) are prone to higher estimation errors, whereas the latter are averaged over a prediction error signal which is decreasing as long as the residual ETA gets lower. Comparing these results with those obtained by *scp* (third row of Table 3.6), it can be seen that the error associated with the latter is higher for every value of N: the RMSE associated with our proposed ML-based technique is, on average, 40% lower.

Figure 3.9 shows a comparison between the ETA predictions obtained from *scp* and from the ML-based technique, analogously to Figure 3.4 in Section 3.6.1. It can be seen that *scp* tends to give mostly inaccurate predictions during the first part of the transmission, and finally corrects its trend only in the second half of the file transfer. On the other hand, the ML-based technique gives a fairly accurate estimate of the time needed to complete the file transfer right after the beginning of the transmission, following the correct trend already after a few seconds. For this particular 4-node configuration, the normalized RMSE given by *scp* was 0.1693, whereas the corresponding value from the on-the-fly algorithm was 0.1125, yielding a relative gain of $\sim 34\%$. Note that, in Figure 3.9, the curves describing both the

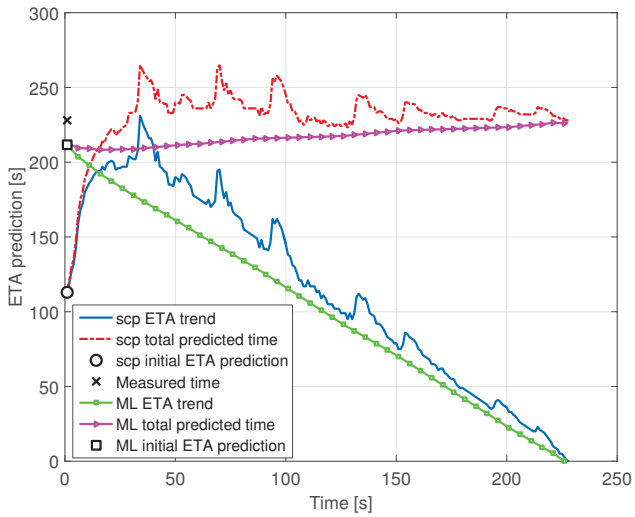


Figure 3.9: Comparison of the ETA estimations given by *scp* and the on-the-fly ML-based technique.

Table 3.7: Normalized RMSE for the ETA estimates at 10% of the transmissions.

	$N = 1$	$N = 2$	$N = 3$	$N = 4$	Average
ML	0.0196	0.0331	0.0577	0.1762	0.0717
scp	0.0861	0.1074	0.3146	0.5170	0.2562

ML and the *scp* trends should be compared to the line connecting the total measured time on the upper-left part of the figure (represented by a black cross on the vertical axis) to the point corresponding to the same amount of time on the horizontal axis. From the figure, it can be seen that the difference between this "real trend" line and the *scp* trend curve, especially in the first part of the transmission (first 50 s), would be higher (*scp* gives a first ETA prediction of $\sim$ 105 s) than what obtained comparing the "real trend" line to the ML-based trend curve.

A further comparison between the ML-based and the *scp* ETA estimators is related to the accuracy of the ETA estimation in the initial part of the transmission, when only the first fraction of a file has been successfully received. Table 3.7 presents the normalized RMSE values related to the ETA estimates when only 10% of each transmission has been completed. In other words, for each transmission, we wait for the reception of the first 10% of the total file size, and then compute the error between the actual time needed to complete the transmission and the corresponding ETA estimate given by the ML-based technique and *scp*. It can be seen that the ETA estimate from *scp* is considerably higher (roughly 72% worse than the ML estimate on average): this is especially important as, usually, a user should be able to decide whether to interrupt or maintain a transmission based on an early ETA estimate, without waiting for it to stabilize.

3.7 Chapter conclusions

In this chapter, we proposed an on-the-fly machine learning-based approach for the prediction of the Estimated Time of Arrival in wireless transmissions. This technique was designed with the constraint of exploiting only the amount of information available at each receiving device, without modifying any standard communication protocol. In order to do that, the algorithm works by first detecting the number of users simultaneously receiving data from the same access point, and then exploiting this knowledge to perform a prediction of the ETA during the transmission. As a correct detection of the number of users proved to have a high impact on the final ETA prediction performance, a further step, starting from this work, could be to periodically perform this detection during the file transfers, in order to either correct a wrong estimate, or to dynamically update this parameter if initial conditions have changed.

CHAPTER 4

LIGHTWEIGHT LOSSY COMPRESSION OF BIOSIGNALS VIA DENOISING AUTOENCODERS

Modern wearable IoT devices enable the monitoring of vital parameters such as heart or respiratory rates (RESP), electrocardiography (ECG), photo plethysmographic (PPG) signals within e-health applications. However, a common issue of wearable technology is that signal transmission is power-demanding and, as such, devices require frequent battery charges and this poses serious limitations to the continuous monitoring of vitals. To ameliorate this, this chapter advocates the use of lossy signal compression as a means to decrease the data size of the gathered biosignals and, in turn, boost the battery life of wearables and allow for fine-grained and long-term monitoring. Considering one dimensional biosignals such as ECG, RESP and PPG, which are often available from commercial wearable IoT devices, this chapter provides a thorough review of existing algorithms and introduces a novel approach based on autoencoders, elucidating their operating principles and providing a quantitative assessment of their compression, reconstruction and energy consumption performance. As we quantify, the most efficient schemes allow reductions in the signal size of up to 100 *times, which entail similar reductions in the energy demand, by still keeping the reconstruction error within* 4% *of the peak-to-peak signal amplitude. Avenues for future research are finally discussed.*

4.1 Introduction to the chapter

Internet of Things (IoT) technology enables objects to sense the physical environment and to seamlessly integrate the gathered data into sophisticated Internet applications that allow for substantial improvements of human activities at large. The focus of this work is on human sensing [92] through wearable IoT devices, such as

smart watches, chest straps or wristbands, which permit the collection of biosignals such as heart-rate, oxygen level, respiration and blood pressure and can be used to help address the individual health and fitness needs of the users [93].

For instance, wearables can be utilized to gather and share information about the status of outpatients, making it possible to collect, record and analyze new data streams faster and more accurately. This allows for an improved access to health-care, an increase of its quality and ultimately, a reduction in its cost. Tele-health systems could deliver care to people in remote locations and provide streams of accurate data for making better care decisions (*e.g.*, in terms of therapy adjustments or prompt interventions). In addition, these systems are expected to have a big impact on the field of rehabilitation where, for example, users may wear e-textile systems for remote, continuous monitoring of physiological and movement data [94]. Through IoT technology, a large number of physiological signals can be monitored including oxygen saturation, blood pressure, heart rate, respiration rate, glucose level [93, 95] and user activities such as walking, standing, sleeping, *etc.*, can be inferred [96]. A recent survey of wearable devices and their use is offered in [93], whereas rehabilitation systems are discussed in [94].

We look at an IoT scenario for e-health, where wearables are utilized to collect physiological signals, preprocess and transmit them over their wireless interface for their final storage and manipulation via backend server infrastructures. Within this context, we are concerned with the design of online signal processing algorithms, so that the gathered signals can be effectively stored in the limited memory space of wearables and conveniently transmitted over their radio interface. Ideally, we would like this software to adapt to the signals being sampled, by being prompt when required by the application and gently go into some power saving mode when the signals exhibit regular patterns. This means that high resolution should be provided when the user wants to track some dynamic activity or when a critical behavior is detected. Toward this end, we advocate the use of lossy compression as a means to reduce the space taken by the collected biosignals and, at the same time, to save battery power through a reduced transmission time. This amounts to compressing the physiological data directly at its source.

As for the physiological signals of interest, we consider one dimensional and quasi-periodic biomedical signals as those provided by typical sensors in chest straps or wristbands, *i.e.*, electrocardiography (ECG), photo-plethysmographic (PPG) and respiratory (RESP) signals. ECG is probably the most important among them for the diagnosis of heart malfunctions and IoT technologies are expected to be very useful to assess cardiac conditions within patient-monitoring applications. Commercial devices such as the Bioharness 3 from Zephyr Technology Corporation [97] can be utilized to measure this type of signal. RESP signals are also very relevant

and can be obtained from chest straps [98] or rubber straps [99] placed around the abdomen to, e.g., assess the status of outpatients affected by chronic respiratory failure and allow monitoring them from home. PPG is often available in low-cost IoT devices for the consumer market (such as smart watches or wristbands designed for fitness applications), see the Angel sensor wristband [100]. PPG can be used to estimate heart-rate [101] and recent studies indicate that blood pressure can also be inferred [102].

We believe that, despite the focus and hype on wearable technology, research on data processing algorithms for wearable IoT devices is still in its infancy and most still has to be done to take full advantage of this portable technology, especially in the medical field. In past research, a large number of compression algorithms were proposed for ECG, but signal compression has never been applied to RESP or PPG. Moreover, performance assessments were only carried out for quality of compression and reconstruction, whereas the energy consumption aspect has often been neglected. Instead, we stress that energy should be sparingly used by the software running on wearables, as these devices are often battery operated and, in turn, their energy consumption is a key consideration. Also, to the best of the authors' knowledge, no quantitative comparison among existing solutions can be found in the literature and, due to this, it is unclear which algorithms are best suited for use in wearable devices.

The aim of this chapter is to fill these gaps. First, in Section 4.2 we present a taxonomy of popular signal compression schemes from the literature, touching upon linear approximations [103, 104], Fourier [105], Wavelet [106] transforms and novel compression techniques based on *compressive sensing* [107, 108] and vector quantization and pattern recognition [109].

In Section 4.3.1 we take a novel approach based on autoencoders, as these are known for their ability to capture the essential relationships in the input data and compactly represent them through non-linear maps. After a learning phase, which is is executed offline, the proposed compression architecture employs a matrix multiplication which can be conveniently implemented in constrained hardware. Hence, our present work extends that in the related literature as our compressor is extremely cheap but at the same time very efficient in terms of compression ratios and accurate in the signal reconstruction. Last but not least, the compression methodology that we propose here can be applied to any quasi-periodic biomedical signal: ECG, PPG and RESP traces being some relevant examples.

The other selected algorithms from the literature are detailed in Section 4.3 and a comparative performance evaluation of all the considered compression approaches is carried out in Section 4.4, where we quantify their compression efficiency, signal reconstruction fidelity and, most importantly, their energy consumption. Also,

we estimate the battery time improvement due to the adoption of the discussed compression technology for continuous monitoring applications.

Finally, our conclusions are presented in Section 4.5, along with a discussion of open research issues.

To summarize, the main contributions of this chapter are:

- A taxonomy of existing signal compression schemes that are amenable to implementation in wireless wearable IoT devices.
- An autoencoder-based original lossy compression architecture for the biosignals acquired by constrained devices, along with its validation.
- A detailed performance evaluation of the considered compression schemes in terms of reconstruction error, energy consumption (isolating the energy required for compression and transmission) and compression efficiency when applied to ECG, RESP and PPG signals.
- A discussion of open areas for improvement and new research avenues.

4.2 Taxonomy of Lossy Compression Schemes

In the last few years, a great deal of work has been carried out on tools for the efficient ECG signal analysis, facial image recognition or the identification of fingerprints acquired by a cell phone, see [110]. PPG is being intensively investigated for the estimation of the heart rate [101] and motion data is being used for activity detection [111]. Nevertheless, apart from ECG, little has been done regarding the compression of other signals, such as PPG, RESP, etc. In this taxonomy, we first focus on ECG and then elaborate on the use of compression for other signal types.

The two most important tasks to be accomplished in the ECG domain are 1) QRS[1] complex detection and 2) signal compression. As per QRS detection, it is crucial to split the ECG time series into heart beat segments (one segment per beat) as this allows the fine-grained assessment of inter-beat signal features, which are useful to detect certain pathologies. Note that ECG can be efficiently split into beat segments as it is a quasi-periodic time series exhibiting recurrent patterns. As to signal compression, we emphasize that wearable devices are energy and memory constrained and, as such, minimizing the amount of data to store and send is an important consideration. As an example, a typical sampling rate of 250 samples per second with 12 bits per sample (e.g., from a Zephyr's Bioharness device) leads to 32.4 Mbytes of data for a full day. As we will see below, compression algorithms can

[1]The QRS complex is a name for the combination of three of the graphical deflections seen on a typical electrocardiogram.

easily reduce this number by 70 times to about 463 kbytes, leading to much higher efficiencies in terms of memory and transmission.

1) QRS complex detection has been extensively studied in the literature. Several methods were proposed to detect QRS complexes and to enhance their features. The importance of QRS enhancement has been demonstrated to detect the QRS complex [112]. In particular, amplitude thresholding [113], first and second derivative methods [114], mathematical morphology [115, 116], filter banks [117], and wavelet transform techniques [118] are among the methods used for the enhancement of the QRS complex. The QRS detection is instead usually performed with a combination of techniques such as thresholding [113, 115], neural networks [119], wavelet transform [120], matched filters [121]. These techniques are of foremost importance as they split the ECG time series into segments (*i.e.*, the data points between subsequent heartbeats), which are then utilized for the subsequent estimation of the pulse, and for the compression of the ECG trace.

2) Signal compression. Quite a few lossy and lossless compression algorithms for ECG signals have been proposed in the literature in the last decades. Typically, they can be classified into three main categories:

- **Time domain processing**: within this class we have AZTEC [122], CORTES [123] and Lightweight Temporal Compression (LTC) [103]. Both AZTEC and CORTES achieve compression by discarding some of the signal samples and applying a linear approximation, whereas LTC approximates the original time series through piecewise linear segments, where the two end points of a segment are sent in place of the points in between. As we show in Section 4.4, in spite of its simplicity, LTC closely matches the performance of Principal Component Analysis (PCA) [104, 124].

- **Transform based coding**: these exploit transformations such as Fast Fourier Transform (FFT) [105], Discrete Cosine Transform (DCT) [125] and Discrete Wavelet Transform (DWT) [106]. The rationale behind them is to represent the signal in a suitable transform domain and select a number of transform coefficients to be sent in place of the original samples. The amount of compression depends on the number of coefficients that are selected, the representation accuracy depends on how many and which coefficients are retained. Although the schemes belonging to this class have good compression capabilities, their computational complexity is often too high for wearable devices [126]. Lightweight implementations are possible and are considered in the present

work.

- **Parametric techniques**: these schemes use neural networks [127], vector quantization [128], Compressed Sensing (CS) [107] and pattern matching [129]. Their rationale is to process the temporal series to obtain some kind of knowledge and use it to predict the signal behavior.

Despite these developments, we recall that no systematic comparison was carried out in the existing literature and, moreover, the proposed algorithms were not evaluated in terms of their energy expenditure. This is of course very important for wearables, which are battery operated and thus call for algorithms that are at the same time extremely effective and computationally cheap.

In addition, besides ECG, recent advances in technology for wearable devices have made it possible to efficiently collect and analyze other signals such as PPG, motion and respiration through body worn sensor technologies [130]. The PPG signal can be a powerful diagnostic tool due to simple, portable, and low-cost technology available for its fast, easy, and reliable acquisition and can be non-intrusively measured using wristbands or smart-watches. An increasing number of works in the literature deal with the extraction of physiological parameters from the PPG signal such as heart rate, blood pressure, blood oxigen saturation, and respiration [102, 131, 132]. Nevertheless, to the best of our knowledge, no algorithms have been proposed so far for the compression of these signals. Note that with future application developments, besides the calculation of selected features or health indicators right on the mobile devices, users or doctors may want to fully monitor the vitals, which could be sent to smartphones or control centers for further processing so as to provide a fine-grained assessment of the patient's condition, e.g., to assess the evolution or occurrence of a certain pathology. In this case, compressed but accurate representations of vital signals from heterogeneous sensor technology are expected to be very useful.

4.3 Signal Compression Algorithms

Next, we detail the selected signal compression algorithms for quasi-periodic biosignals, by initially presenting a novel technique based on autoencoders, first introducing the concept of neural networks, and then detailing their use within the proposed lossy compression architecture. The compression methods that we describe below are based on differing paradigms. In fact, some use the degree of similarity (correlation) across subsequent patterns (or segments), whereas others consider the correlation within the same segment. We refer to the former approach as "inter-segment correlation" based compression, whereas for the latter we use the term "intra-segment

correlation". The algorithms belonging to the inter-segment class are vector quantization and autoencoders, whereas algorithms based on principal component analysis, LTC, discrete cosine and wavelet transforms exploit intra-segment correlation properties. The implementation of compressive sensing that is considered here belongs to both classes.

4.3.1 Autoencoders (AE)

There is a number of reasons to consider Neural Networks (NNs) as a means to compress biometric signals. First, it has been proved that they can effectively approximate any function with arbitrary accuracy, acting as *universal approximators* [133–135]. This is because they exploit self-adaptive methods to map the complex relationships that underpin real world data, without assuming any prior knowledge about their underlying statistical model. Also, NNs rely on non-linear functions, and this makes them more appealing than traditional linear dimensionality reduction techniques such as PCA [124]. An additional limitation of PCA is that, differently from NNs, it ignores correlation properties between the provided input and the target output. In fact, [136, 137] showed that NNs perform a sort of *non-linear* PCA.

NNs are composed of a number of connected neurons. Each neuron takes a vector $\boldsymbol{x}$ of $k+1$ input variables $\boldsymbol{x} = (x_0, \ldots, x_k)^T$ (among which $x_0 = 1$ is an intercept term) and uses a vector of weights $\boldsymbol{w} = (w_0, w_1, \ldots, w_k)^T$ to output $h_{\boldsymbol{w}}(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x}) = f(\sum_{j=0}^{k} w_j x_j)$, where $f : \mathbb{R} \to \mathbb{R}$ is an *activation function*, often assumed to be the *logistic sigmoid* function $f(z) = (1 + \exp(-z))^{-1}$. A neural network is organized into k_ℓ *layers* of neurons, so that the output of a neuron is input into the following ones: since no loops are present in the connectivity graph, this type of network is known as *feed-forward* NN. The first and last layers are called *input* and *output* layers, respectively, whereas the layers in between are called *hidden*. Network parameters are contained in the weight matrices $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(k_\ell - 1)}$ (whose size depends on the specific number of input and output neurons in each layer). In the following, with $\mathcal{W}$ we denote the set $\mathcal{W} = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(k_\ell - 1)}\}$, whereas $w_{ij}^{(\ell)} \in \boldsymbol{W}^{(\ell)}$ is the weight associated with the connection between neuron j in layer ℓ and neuron i in layer $\ell+1$. The outputs of layer ℓ, called *activations*, are stored into the vector $\boldsymbol{a}^{(\ell)} = (a_1^{(\ell)}, \ldots, a_{s_\ell}^{(\ell)})^T$, where s_ℓ is the number of nodes (not including the bias unit) in layer ℓ. If we let $a_i^{(1)} = x_i$ denote the i-th input, the activations for a generic layer ℓ are obtained as

$$a_i^{(\ell)} = f(w_{i0}^{(\ell-1)} a_0^{(\ell-1)} + \cdots + w_{is_{\ell-1}}^{(\ell-1)} a_{s_{\ell-1}}^{(\ell-1)}), \tag{4.1}$$

with $\ell = 2, ..., k_\ell$. If we denote $z_i^{(\ell)}$ the total weighted sum of inputs to unit i in layer

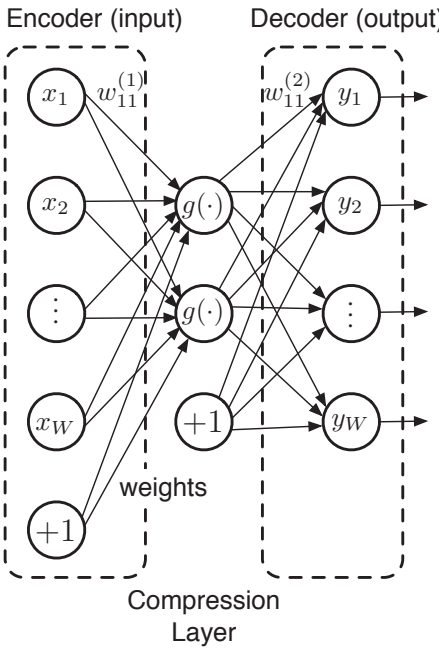


Figure 4.1: Graphical representation of an autoencoder: input and output layers have the same dimension W, whereas the compression layer has $h = 2$ neurons. $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is assumed to be the sigmoid activation function $g(z) = (1 + \exp(-z))^{-1}$.

ℓ, i.e., $z_i^{(\ell)} = \sum_{j=0}^{s_{\ell-1}} w_{ij}^{(\ell-1)} a_j^{(\ell-1)}$, the activations of layer ℓ can be computed, given $\boldsymbol{a}^{(\ell-1)}$, as:

$$\boldsymbol{a}^{(\ell)} = f(\boldsymbol{z}^{(\ell)}) = f\left(\boldsymbol{W}^{(\ell-1)}\boldsymbol{a}^{(\ell-1)}\right). \tag{4.2}$$

From now on, we denote the network output by $h_{\mathcal{W}}(\boldsymbol{x})$, which is calculated sequentially using (4.2) from the second ($\ell = 2$) to the output layer ($\ell = k_\ell$): this procedure represents a forward propagation of information through the network.

An autoencoder [84] is a particular neural network instance where input and output layers have the same dimension W, whereas the deepest hidden layer has a smaller dimension c, with $c < W$, see Figure 4.1. Autoencoders are trained using an unsupervised learning algorithm acting on the input data $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}$, with $\boldsymbol{x}^{(i)} \in \mathbb{R}^W$, referred to as *training examples*. The target output values $h_{\mathcal{W}}(\boldsymbol{x})$ are set equal to the input, so that the AE's output matches as much as possible the provided input sequence *i.e.*, $h_{\mathcal{W}}(\boldsymbol{x}^{(i)}) \simeq \boldsymbol{x}^{(i)}$, $\forall\, i$, according to a selected distance metric, and the objective is to learn the identity function $h_{\mathcal{W}}(\boldsymbol{x}^{(i)}) = \boldsymbol{x}^{(i)}$. For a single example $\boldsymbol{x}$, as the distance metric we use the squared-error cost function $E_{\mathcal{W}}$:

$$E_{\mathcal{W}}(\boldsymbol{x}) = \frac{1}{2}||h_{\mathcal{W}}(\boldsymbol{x}) - \boldsymbol{x}||^2, \tag{4.3}$$

whereas the overall cost function $J_{\mathcal{W}}$ is used to quantify the distance for the m

training examples

$$J_{\mathcal{W}} = \frac{1}{m}\sum_{i=1}^{m} E_{\mathcal{W}}(\boldsymbol{x}^{(i)}) + \frac{\lambda}{2}\sum_{l=1}^{k_l-1}\sum_{j=1}^{s_\ell}\sum_{i=1}^{s_{\ell+1}}\left(w_{ij}^{(\ell)}\right)^2 . \tag{4.4}$$

The first term of (4.4) represents the mean squared error estimate, whereas the second term accounts for *regularization*, whose aim is to prevent overfitting [77]. The trade-off between the optimization of these two terms is governed by the parameter λ, the *weight-decay* parameter.

NM-based AEs are usually trained using *gradient descent* methods [138], which update the weights in the direction of the negative gradient of $J_{\mathcal{W}}$, *i.e.*,

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha\frac{\partial J_{\mathcal{W}}(t)}{\partial w_{ij}^{(\ell)}(t)}\,, \tag{4.5}$$

where t represent the iteration number, $\alpha > 0$ is the learning rate and the partial derivative of $J_{\mathcal{W}}$ is computed via *error backpropagation* [139]. The resulting process can be divided into two phases: 1) first, input $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ is mapped into a hidden representation $\boldsymbol{x'}^{(i)} \in \mathbb{R}^W$ via an *encoder*, obtaining a representation of the input with a reduced dimensionality and 2) this representation is mapped back, via a *decoder*, into a reconstruction $h_{\mathcal{W}}(\boldsymbol{x}) = \hat{\boldsymbol{x}}$, see Figure 4.1. Previous work has shown that if the activation function $f(\cdot)$ is linear, the autoencoder has the same performance of PCA [140] (so the c hidden units effectively project the W-dimensional input space into a c-dimensional one). Instead, if $f(\cdot)$ is non-linear, the autoencoder has the ability to capture multi-modal aspects of the input signal [141]. Once the network is trained, a compressed (and lossy) version of the input signal $\boldsymbol{x} \in \mathbb{R}^W$ can be computed by providing $\boldsymbol{x}$ to the encoding part of the autoencoder (termed "Encoder" in Figure 4.1) and obtaining, as a result, the reduced-dimensionality vector $\boldsymbol{x'} \in \mathbb{R}^c$ (*i.e.*, the values assumed by the c neurons in the hidden layer of Figure 4.1). The decoding process will reconstruct $\boldsymbol{x}$ starting from $\boldsymbol{x'}$, iteratively using (4.2) for each decoding layer, provided that the weight set $\mathcal{W}$ is fully specified. To that end, the decoding part is utilized (termed "Decoder" in Figure 4.1).

In the present work, we use a stochastic variation of traditional autoencoders known as *denoising autoencoders (DAs)* [142, 143]. With this approach, the autoencoder tries to learn more robust features from the data, and this is achieved through the reconstruction of corrupted versions of the input examples. The rationale is that good representations should capture intrinsic regularities and correlations in the input data and these should be as well recovered from partial or corrupted observations. As DAs simultaneously encode the input data and compensate for a stochastic corruption applied to it, they generalize better to previously unseen in-

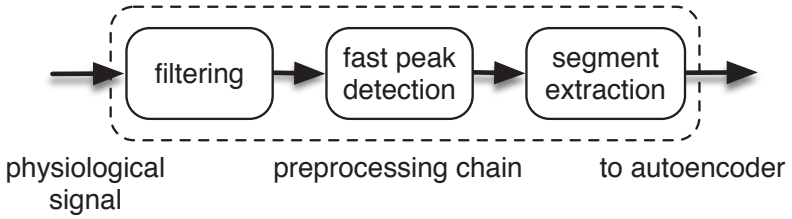


Figure 4.2: High level diagram of the preprocessing phase for the autoencoder-based compression algorithm.

put. Stochastic corruption is performed by randomly zero-masking elements of the input $\boldsymbol{x}$, according to a parameter ν that specifies the "destruction" proportion. As a result, in the matrix $\boldsymbol{X}_{\text{tr}} \in \mathbb{R}^{m \times W}$ denoting the training examples, a fixed number νmW of components are chosen at random, and their value is forced to zero, while the others are left untouched. This modified matrix is fed to the training algorithm. Once the autoencoder is trained to represent the input data, weights $w_{ij}^{(1)}$ fully specify the compressor (encoder), whereas $w_{ij}^{(2)}$ specify the decompressor (decoder), see Figure 4.1.

Signal compression is achieved by applying the preprocessing chain of Figure 4.2, *i.e.*, filtering, peak detection and segment extraction.

1) Passband filtering: as a first step, we use a passband filter to remove artifacts such as high frequency noise and the DC component. For ECG, this filter operates in the band $[8, 20]$ Hz, although these can be changed to best suit other signal types. Here, we implemented the third-order Butterworth filter of [144].

2) Peak detection: with this algorithm we detect the position of the main peaks in the time series. For ECG, these correspond to the heart beats. To this end, we have adopted the technique of [145], which has been conceived for ECG signals but can be easily modified to effectively work with PPG or respiratory traces. This technique is self-tuning and optimizes itself based on the input data sampling rate. We considered this scheme as it is fast and lightweight and thus suitable for use in wearable and energy constrained devices.

3) Segment extractor: once the peaks are detected, we consider the data samples between subsequent peaks. These constitute the input *segments* for our compressor algorithm. Note that, unlike the common practice of positioning the segments so that the peaks (heart beats) are in their center, we define a segment as the data points *between subsequent peaks*. Hence, all segments are normalized according to a predefined length of W samples, which is the same size of the first layer of the

autoencoder. This is accomplished by re-stretching the segment length to W samples through interpolation. While in principle any interpolation technique can be used, such as quadratic or spline based, in our implementation we utilized a simple linear technique as we found it sufficiently accurate while also being computationally inexpensive.

Each segment is inputted to the encoder section of the autoencoder, which returns the c values associated with the neurons in the compression layer. These c values correspond to the compressed representation of the current segment and are sent to the decompressor along with the original segment length. Finally, the decompressor at the receiver uses the values of these c inner neurons, along with weights $w_{ij}^{(2)}$, to obtain the reconstructed W-sample vector $\boldsymbol{y}$ through the decoder of Figure 4.1. $\boldsymbol{y}$ is finally resized to the original segment length.

We remark that all the parameters needed to configure the AE, including λ in (4.4), the learning rate α, the denoising parameter ν and other network specific values have been selected by means of a k-fold cross validation technique, already described earlier in the thesis. Finally, the AE belongs to the inter-segment correlation class of algorithms as it exploits the fact that patterns across different segments have a quasi-periodic behavior.

4.3.2 Gain-Shape Vector Quantization (GSVQ)

In this section we review the Gain-Shape Vector Quantization (GSVQ) method of [128]. The rationale behind this algorithm is to exploit the information redundancy among adjacent heartbeats by segmenting the ECG signal into segments and normalizing the period to a fixed length and amplitude. The normalized heartbeats are then used to build a dictionary having a fixed number of codewords K, through the Linde-Buzo-Gray algorithm [146]. Note that, as our AE-based compressor, GSVQ requires an offline training phase.

Once the dictionary is obtained, the method associates each normalized heartbeat with the closest codeword, and sends the codeword index in place of the original time series. The algorithm also encodes the offset, the gain, and the length of the original segment, see Figure 4.3.

As a last step, the encoder calculates the residual, *i.e.*, the difference between the current heartbeat (*i.e.*, ECG segment) and the selected codeword, and uses the AREA algorithm [147], an adaptive sampling scheme for one dimensional signals, which obtains additional information to increase the quality of reconstruction. The principle behind the residual encoding phase is to encode and send a small number of significant points so as to bound the reconstruction error.

The decoder, upon receiving an encoded packet, retrieves the corresponding code-

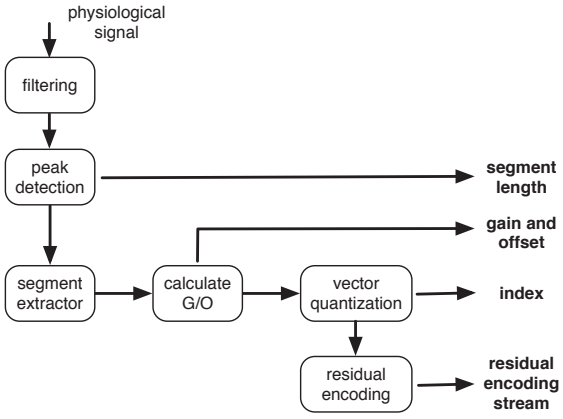


Figure 4.3: Diagram of the GSVQ compression technique.

word from its local copy of the dictionary, performs a denormalization using the gain, the offset, and the length, and adds the residual stream to the reconstructed signal. As we shall see below, GSVQ performance predominantly depends on its residual encoding phase. The threshold used for residual encoding is in fact the main responsible for the amount of data to be transmitted, affecting the performance in terms of compression, reconstruction error, and energy efficiency.

4.3.3 Principal Component Analysis (PCA)

The goal of Principal Component Analysis (PCA) [104] is to shrink the information provided by a large set of correlated variables into a set of principal components with lower dimensionality. Each principal component is computed as a linear combination (linear transform) of the original variables, and the combination weights are chosen so that the components are mutually uncorrelated. This technique has been successfully selected in a multitude of applications, including ECG signal compression [124].

Before applying PCA, the biomedical signal goes through the preprocessing chain of Figure 4.2, *i.e.*, filtering, peak detection and segment extraction, where at time $t = 0, 1, 2, \ldots$ the last block normalizes each input segment $\boldsymbol{z}_t$ to a common length of W samples. The new segment is then stored into a vector $\boldsymbol{x}_t \in \mathbb{R}^W$ and is fed to the PCA encoder. Specifically, let $\boldsymbol{\mu_x} = E[\boldsymbol{x}_t]$ and $\boldsymbol{R_x} = E[\widetilde{\boldsymbol{x}}_t \widetilde{\boldsymbol{x}}_t^T]$ respectively be the mean of $\boldsymbol{x}_t$ and its covariance matrix, with $\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\mu_x}$. PCA amounts to applying an orthonormal linear transformation $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_W]$ to $\widetilde{\boldsymbol{x}}_t$, so that the elements $w_1, \ldots, w_W$ of the principal component vector $\boldsymbol{w} = \boldsymbol{\Psi}^T \widetilde{\boldsymbol{x}}_t = \boldsymbol{\Psi}^T (\boldsymbol{x}_t - \boldsymbol{\mu_x})$ are mutually uncorrelated. It can be shown that the i-th principal component is obtained as $w_i = \boldsymbol{\psi}_i \widetilde{\boldsymbol{x}}_t$, where $\boldsymbol{\psi}_i$ is the eigenvector corresponding to the i-th largest eigenvalue of $\boldsymbol{R_x}$, for $i = 1, \ldots, W$. The set of eigenvectors corresponding to the W

principal components is obtained solving $\boldsymbol{R_x\Psi} = \boldsymbol{\Psi\lambda}$ for $\boldsymbol{\Psi}$, where $\boldsymbol{\lambda}$ is a diagonal matrix containing the eigenvalues $\lambda_1, \ldots, \lambda_W$, placed in decreasing order. As the theoretical covariance matrix $\boldsymbol{R_x}$ is difficult to compute, a matrix $\boldsymbol{X} \in \mathbb{R}^{W\times m}$ is built by stacking m successive ECG segments: their sample mean $\hat{\boldsymbol{\mu}}_{\boldsymbol{x}}$ and their sample covariance matrix $\hat{\boldsymbol{R}}_{\boldsymbol{x}} = (\boldsymbol{XX}^T)/m \in \mathbb{R}^{W\times W}$ respectively replace $\boldsymbol{\mu_x}$ and $\boldsymbol{R_x}$ for the calculation of the eigenvectors.

According to the above discussion, we can write $\boldsymbol{x}_t = \boldsymbol{\mu_x} + \boldsymbol{\Psi w}$ and, if the signal is sufficiently correlated, only a fraction of the weights in $\boldsymbol{w}$ suffices to accurately describe the input vector $\boldsymbol{x}_t$. Compression is thus achieved by applying the PCA transform and sending the desired number c of principal components, *i.e.*, the first c weights in $\boldsymbol{w}$, with $c \leq W$. This rationale is similar to the autoencoder approach, described in Section 4.3.1: indeed, [136] proved that autoencoders practically act as a non-linear PCA.

4.3.4 Compressive Sensing (CS)

Compressive sensing (CS) is a recently proposed theory [148] [149] to efficiently acquire and reconstruct a signal, by solving ill-posed linear systems of equations. This technique is based upon the premise that the signal of interest is sparse in some transform domain. This means that the original signal can be represented in a domain where only a few transform coefficients are required for its full description. To be more specific, let $\boldsymbol{x} \in \mathbb{R}^W$ be an W-sized vector and assume that this vector can be represented in a K-sparse domain through the sparse vector $\boldsymbol{s}$, where only $K \ll W$ elements of $\boldsymbol{s}$ are non-zero, *i.e.*, vector $\boldsymbol{s}$ is K-sparse in this domain. If we refer to the sparsification basis as $\boldsymbol{\Psi} \in \mathbb{R}^{W\times W}$, we have that $\boldsymbol{x} = \boldsymbol{\Psi s}$. Now, let $\boldsymbol{\Phi} \in \mathbb{R}^{m\times W}$ be a sampling matrix. Note that, using this matrix to sense the full signal $\boldsymbol{x}$, we have $\boldsymbol{y} = \boldsymbol{\Phi x}$, with $\boldsymbol{y} \in \mathbb{R}^m$ and $m < W$, which means that $\boldsymbol{x}$ is being subsampled.

CS tools allow the recovery of $\boldsymbol{x}$ from its subsampled version $\boldsymbol{y}$, where: $\boldsymbol{y} = \boldsymbol{\Phi x} = \boldsymbol{\Phi\Psi s}$. This is achieved solving for $\boldsymbol{s}$ the following equation:

$$\min \|\boldsymbol{s}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{\Phi\Psi s}\|_2 \leq \epsilon\,, \tag{4.6}$$

where ϵ represents a bound on the measurement noise. Numerically, a high number of techniques are available to solve (4.6); among them we cite ℓ_1-magic [150] subspace pursuit [151] and NESTA [152].

In this work, we consider two recent ECG compression algorithms from [107] and [153], which are based on CS. At the encoder, they exploit a standard CS matrix multiplication (sampling and sparsification), whereas at the decompressor the former exploits a technique called Simultaneous Orthogonal Matching Pursuit

(SOMP) [107], whereas the latter uses Block Sparse Bayesian Learning (BSBL) [154]. The algorithms are discussed next.

Simultaneous Orthogonal Matching Pursuit (SOMP-CS)

This technique first splits the ECG signal into a number of segments and then applies standard CS-sampling to R consecutive segments at a time. Recovery is based on Orthogonal Matching Pursuit.

SOMP-CS encoder:

- **Peak detection:** similarly to autoencoder-based schemes, a peak detection method is applied to the input signal to decompose it into segments $\boldsymbol{x}_t$, $t = 0, 1, 2, \ldots$.
- **Period normalization:** each segment $\boldsymbol{x}_t$ is normalized to a common length (W samples) using cubic-spline interpolation.
- **Sampling and quantization:** each R consecutive ECG segments are stored into a $W \times R$ matrix $\boldsymbol{X}$. A CS sampled matrix $\boldsymbol{Y}$ is then obtained as $\boldsymbol{Y} = \boldsymbol{\Phi X}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{m \times W}$ is a suitable sampling matrix, with $m \ll W$. As assumed in [107], for matrix $\boldsymbol{\Phi}$ we use a dense Gaussian matrix (each element is independently sampled from a Gaussian pdf with zero mean and variance $1/m$, *i.e.*, $\mathcal{N}(0, 1/m)$). $\boldsymbol{Y}$ and the corresponding original lengths are quantized and sent to the decoder. Note that this implementation of CS belongs to both the inter- and the intra-segment class as matrix $\boldsymbol{Y}$ spans across different adjoining segments.

Note that the CS encoder is extremely lightweight as it just implies the multiplication of the input time series by the sampling matrix $\boldsymbol{\Phi}$. The most computationally intensive tasks are period normalization and peak detection, which are needed in all segment-based approaches. Under the assumption that the source data $\boldsymbol{X}$ can be rewritten as: $\boldsymbol{X} = \boldsymbol{\Psi S}$, where the matrix $\boldsymbol{S} \in \mathbb{R}^{W \times R}$ is sparse and the sparsification transform $\boldsymbol{\Psi}$ is the Daubechies wavelet db4 [155], the original data $\boldsymbol{X}$ can be retrieved solving problem (4.6) using Simultaneous Orthogonal Matching Pursuit. In our implementation, we have exploited the method in [156] and the Matlab Uvi Wave tool [157] to represent the wavelet transform into an equivalent matrix form.

SOMP-CS decoder:

- **Simultaneous Orthogonal matching Pursuit:** each segment is recovered from $\boldsymbol{Y}$ using the modified Simultaneous Orthogonal Matching Pursuit (with

partially known support) of [158], which exploits the structure of the wavelet coefficients (through the knowledge of the support). This method solves for $\boldsymbol{S}$ the ill-posed system $\boldsymbol{Y} = \boldsymbol{\Phi\Psi S}$, through (4.6). Upon recovering $\boldsymbol{S}$, the original data is approximated through $\hat{\boldsymbol{X}} = \boldsymbol{\Psi S}$.

- **Period Recovery:** the reconstructed segments are re-interpolated according to their original lengths.

Note that SOMP-CS considers a number R of subsequent segments ($R = 6$ in our implementation) and, in turn, also accounts for the "inter-segment" correlation structure of the ECG signal.

Block Sparse Bayesian Learning (BSBL)

BSBL exploits the fact that the ECG signal $\boldsymbol{x}$ is already sparse in the temporal domain, being composed of peaks spaced apart by an almost-flat signal. Hence, the input ECG signal is written as $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{n}$, where $\boldsymbol{y} \in \mathbb{R}^m$ is the compressed vector, $\boldsymbol{\Phi} \in \mathbb{R}^{m \times W}$ is a suitable sampling matrix ($m \ll W$), $\boldsymbol{x} \in \mathbb{R}^W$ is a sparse vector and $\boldsymbol{n} \in \mathbb{R}^m$ is the noise vector. Generally, vector $\boldsymbol{x}$ has additional structure and can be further represented as a concatenation of a certain number g of blocks x_i, possibly having different length d_i so that $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_g)^T$. Each block $\boldsymbol{x}_i \in \mathbb{R}^{d_i}$, $i = 1, \ldots, g$, is assumed to satisfy a parameterized multivariate Gaussian distribution $p(\boldsymbol{x}_i, \gamma_i, \boldsymbol{B}_i) \sim \mathcal{N}(0, \gamma_i \boldsymbol{B}_i)$ with unknown parameters γ_i and $\boldsymbol{B}_i$. $\gamma_i \geq 0$ controls the block-sparsity of $\boldsymbol{x}_i$ and when $\gamma_i = 0$ the i-th block becomes the all zero vector. Matrix $\boldsymbol{B}_i \in \mathbb{R}^{d_i \times d_i}$ is a positive definite matrix which captures the correlation structure within the i-th block. Assuming that the sub-blocks $\boldsymbol{x}_i$ are uncorrelated the prior of $\boldsymbol{x}$ is $p(\boldsymbol{x}, \{\gamma_i, \boldsymbol{B}_i\}) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\Sigma}_0 = \text{diag}\{\gamma_1 \boldsymbol{B}_1, \ldots, \gamma_g \boldsymbol{B}_g\}$. For the noise, it is assumed that $p(\boldsymbol{n}, \lambda) \sim \mathcal{N}(0, \lambda \boldsymbol{I})$, where $\lambda \in \mathbb{R}^+$ and $\boldsymbol{I} \in \mathbb{R}^{m \times m}$ is the identity matrix. The posterior of $\boldsymbol{x}$ (given the measured vector $\boldsymbol{y}$) is thus obtained as

$$p(\boldsymbol{x}|\boldsymbol{y}; \{\gamma_i, \boldsymbol{B}_i\}_{i=1}^g) \sim \mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\Sigma_x}) \tag{4.7}$$

where $\boldsymbol{\mu_x}$ and $\boldsymbol{\Sigma_x}$ can be readily derived from λ, $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Phi}$. Finally, the Maximum-A-Posteriori (MAP) estimate of $\boldsymbol{x}$, denoted by $\hat{\boldsymbol{x}}$, is given by [154]:

$$\hat{\boldsymbol{x}} = \boldsymbol{\Sigma}_0 (\boldsymbol{\Phi})^T \left[\lambda \boldsymbol{I} + \boldsymbol{\Phi} \boldsymbol{\Sigma}_0 (\boldsymbol{\Phi})^T \right]^{-1} . \tag{4.8}$$

Thus, the problem boils down to the estimation of the parameters λ and $\{\gamma_i, \boldsymbol{B}_i\}_{i=1}^g$. This is achieved using a Type II maximum likelihood procedure. Moreover, different techniques have been developed according to whether the block partition is known

or not [154].

BSBL encoder: the ECG signal is split into a number of segments $\boldsymbol{x}$, each of which consists of W samples, where W is a tunable parameter not necessarily representing the number of samples in a segment. What the encoder does is to compute $\boldsymbol{y} = \boldsymbol{\Phi x}$, which only entails a matrix multiplication. In our results, $\boldsymbol{\Phi}$ is the 0/1 matrix that was used in [153, 159].

We remark that the encoder is extremely lightweight as it does not have to split the ECG trace into segments, so peak detection and period normalization are not executed.

BSBL decoder: the decoder operates according to the above Bayesian estimation / maximum likelihood approach, see (4.8). Typical values for m and W are $m = 256$ and $W = 512$ [153] and, in turn, the maximum compression efficiency is given by $W/m = 2$ (in Section 4.4, we experiment with different (m, W) pairs).

We observe that BSBL accounts for the intra-block correlation without considering the correlation structure among subsequent ECG segments. We thus classify BSBL as an "intra-segment" compression scheme. For SOMP-CS, we have written our own encoder/decoder pair, whereas for BSBL-CS we used the code provided by the authors of [153]. The numerical results are discussed in Section 4.4.

4.3.5 Discrete Cosine Transform (DCT)

In the signal compression field, Discrete Cosine Transform (DCT) is often preferred to the Fourier Transform due to its superior energy compaction capabilities and the fact that it entails the use of real coefficients. Several ECG compression methods exploiting DCT have been proposed in the literature [160–165]. Basically, in all of the proposed algorithms DCT is used to reduce the amount of data to be sent through the transmission of a subset of transform coefficients, *i.e.*, those which carry more information. Some solutions employ advanced techniques for the pre/post processing of the DCT coefficients that, however, for wearable devices are expected to be energetically prohibitive.

Hence, we consider two DCT based compression methods that differ in the adopted coefficient selection approach:

- **DCT-Cardinality Thresholding:** with this selection method the number of coefficients to be retained is given as input, and the coefficients are added starting from the lowest frequencies, *i.e.*, the leftmost coefficient. Through this strategy the compression ratio can be finely tuned, but there are no guarantees

on the reconstruction error at the decompressor.

- **DCT-Energy Thresholding:** with this method the coefficients are selected so as to meet an energy threshold constraint. The total energy of the DCT spectrum, E, is calculated and the coefficients that contain a predetermined fraction E_{th} of this energy are kept. The coefficients are selected again from the lowest to the highest frequencies, exploiting the energy compaction property of the DCT, so that their frequency position does not have to be encoded.

4.3.6 Discrete Wavelet Transform (DWT)

Wavelet compression schemes are based upon the transmission of a subset of the transform coefficients. In fact, in the Wavelet domain most of the signal information is often concentrated in just a few of them. Letting $z[n]$ be the discrete input temporal signal, defined for $n = 0, 1, \ldots, M-1$, in this work we consider the Discrete Wavelet Transform (DWT) [155], which is defined through the following equations

$$\begin{aligned} \gamma[j,k] &= \frac{1}{\sqrt{M}}\hat{\mathrm{E}}\sum_n z[n]\varphi_{j,k}[n]\,, \\ \text{where} \quad \varphi_{j,k}[n] &= \frac{1}{\sqrt{s_0^j}}\varphi\left[\frac{n-k\tau_0 s_0^j}{s_0^j}\right], \quad j,k \in \mathbb{N}\,. \end{aligned} \tag{4.9}$$

$\gamma[j,k]$ is the DWT coefficient matrix and $z[n]$ can be expressed as

$$z[n] = (1/\sqrt{M})\sum_{j,k}\gamma[j,k]\varphi_{j,k}[n]. \tag{4.10}$$

The parameter s_0, called *scale step*, has a fixed value (greater than 1), and τ_0 is the *translation step*. Dyadic sample is commonly used, which amounts to setting $s_0 = 2$ and $\tau_0 = 1$. $\varphi(\cdot)$ is a function, referred to as *mother wavelet*, that is translated and scaled to represent the original signal $z[n]$. An efficient and widely adopted implementation of DWT, for both decomposition and reconstruction, uses Quadrature Mirror Filters (QMF). As shown in (4.9), $z[n]$ can be decomposed into an infinite number of wavelets, but in practice a few levels of decomposition (*i.e.*, a finite number of Wavelet coefficients) already account for most of the signal energy. In the case of dyadic sampling, at each decomposition step the frequency band of the output signal is halved – this allows downsampling the output of each processing step by a factor of 2, without any loss of information.

Wavelet based compression uses the most significant coefficients for reconstruction and the coefficient selection strategy is the main discriminating factor among the existing algorithms. The most widely adopted selection strategies are:

- **DWT-Level Thresholding:** all the coefficients falling below a certain threshold are discarded. Usually, each level of decomposition has a different threshold. This method is commonly used for denoising.

- **DWT-Cardinality Thresholding:** this is the counterpart of "DCT - Cardinality Thresholding". Here, a fixed number of coefficients is retained, discarding those with the lowest absolute values. As for DCT, this selection strategy allows fine tuning the compression ratio, which directly depends on the number of coefficients retained. With this approach, it is however difficult to precisely control the resulting reconstruction quality [166].

- **DWT-Energy Thresholding:** this is the counterpart of "DCT - Energy Thresholding". An energy threshold $E_{\text{th}} < 1$ is set. At each step of the Wavelet transform, the coefficient with the highest value is retained (*i.e.*, it is included in the compressed vector $\boldsymbol{y}$). The energy of the selected coefficients is defined as $E = \boldsymbol{y}^T\boldsymbol{y}$. This operation is repeated until $E/E_0 > E_{\text{th}}$, where E_0 is the energy of the input signal $z[n]$.

In this work, we implemented the algorithm of [106], which is based on energy thresholding. There, in addition to the vector containing the retained coefficients, a coefficient map is also sent so as to track their position within the considered transform levels. The compression ratio is then tuned via E_{th}. Five levels of dyadic decomposition were considered, and bi-orthogonal was used as the mother wavelet, as it provided the best results among other choices.

4.3.7 Lightweight Temporal Compression (LTC)

LTC [103] is a fast linear approximation technique working as follows. Let $z[n]$, $n = 0, 1, 2, \ldots$ be the input time series. The algorithm starts selecting $z[0]$ as the left endpoint of the current approximating segment. The following points $z[n]$ with $n > 0$ are transformed into vertical intervals $[z[n]-\varepsilon, z[n]+\varepsilon]$ where $\varepsilon > 0$ is an error tolerance on the reconstructed signal. When point $n > 1$ is considered, LTC evaluates the segment with extremes $(z[0], z[n])$ and checks whether this segments falls within *each* of the previously obtained vertical intervals around $z[1], z[2], \ldots, z[n-1]$. If this is the case, the algorithm obtains the vertical interval for the current point n and performs the check for the next point $n + 1$. Otherwise, the algorithm stops, taking $z[n-1]$ as the right endpoint of the current segment. Thus, 1) $z[0]$ and $z[n-1]$ are sent as the left and right endpoints of the current segment as an approximation to values $\{z[0], z[1], \ldots, z[n-2], z[n-1]\}$ and 2) the algorithm reiterates with a new approximating segment, taking $z[n-1]$ as its left endpoint.

4.4 Numerical Results

In this section, we show quantitative results for the considered signal compression algorithms, detailing their energy consumption, compression efficiency and reconstruction fidelity. For the **energy consumption**, following the approach of [167,168] we compute three metrics: 1) the energy consumption for the execution of the compression algorithms in the node (termed *compression energy*), 2) the energy drained by the transmission of the (either compressed or original) signal over a wireless channel (*transmission energy*) and 3) the total energy, which is given by the sum of the previous two metrics. The compression energy has been evaluated by taking into account the number of operations performed by the Micro-Controller Unit (MCU), *i.e.*, the number of additions, multiplications, divisions and comparisons. These were then translated into the corresponding number of MCU cycles and, in turn, into the energy consumption in Joule per bit considering a Cortex M4 [169] processor, see also [167]. For the transmission energy, we took a Texas Instruments CC2541 low-energy Bluetooth system-on-chip [170], which is widely adopted for IoT devices. The **Compression Efficiency (CE)** has been computed as the ratio between the total number of bits that would be required to transmit the full signal divided by those required for the transmission of the compressed bitstream. For the reconstruction fidelity, we computed the Root Mean Square Error (RMSE) between the original and the compressed signals, normalizing it with respect to the signal's peak-to-peak amplitude, that is:

$$\text{RMSE} = \frac{100}{\text{p2p}} \sqrt{\frac{\sum_{i=1}^{L}(x_i - \hat{x}_i)^2}{L}}, \tag{4.11}$$

where L corresponds to the total number of samples in the trace, x_i and $\hat{x}_i$ are the original sample and the one reconstructed after the decompressor in position i, respectively. p2p is the average peak-to-peak signal's amplitude. We observe that other metrics such as the Percentage Root mean square Difference (PRD) are also possible. As pointed out in [171], PRD can mask the real performance of compression algorithms since it depends on the mean value of the original signal, whilst our RMSE metric allows to immediately gauge the error against the signal's range. For this reason, we use (4.11) as our preferred metric throughout the chapter.

In Section 4.4.1 we first assess the performance of the considered compression algorithms for the standard test ECG traces from the PhysioNet MIT-BIH arrhythmia database [172]. In Section 4.4.2, we extend our analysis to ECG traces that we collected from a Zephyr BioHarness 3 wearable chest strap. Finally, in Section 4.4.3, we consider PPG and RESP signals.

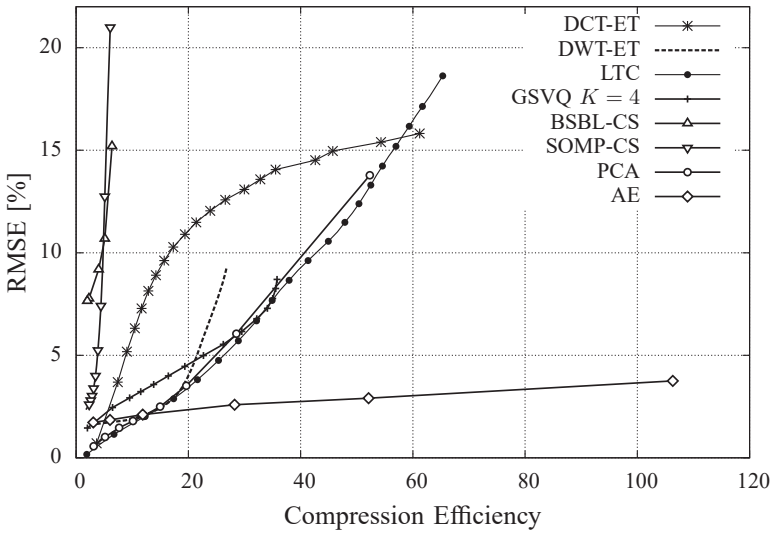


Figure 4.4: RMSE *vs* compression efficiency for ECG signals.

4.4.1 PhysioNet ECG traces

In the first set of graphs, we show results for ECG signals. To this end, we considered the following traces from the MIT-BIH arrhythmia database [172]: 101, 112, 115, 117, 118, 201, 209, 212, 213, 219, 228, 231 and 232, which were sampled at a rate of 360 samples/s with 11−bit resolution. Note that not all the traces of the database are usable (some are very noisy due to heavy artifacts probably due to the disconnection of the sensing devices) and an educated selection has to be carried out for a meaningful performance analysis, as done in previous work [172,173]. The above performance metrics were obtained for these traces and their average values are shown in the following plots.

In Figure 4.4 we show the RMSE *vs* CE performance for all compression algorithms. Here, we consider the energy thresholding version of DCT (DCT-ET): we also experimented with its cardinality thresholding (CT) variant and we found its performance to be very similar to that of DCT-ET in every respect (RMSE, compression efficiency and energy). Thus, in the following, implementation convenience will dictate which of the two variants is to be preferred. From the figure, we can see that LTC outperforms DCT-ET in terms of RMSE and CE up to high compression efficiencies ($CE < 57$), and is definitely better than DWT, which does a much better job than DCT in terms of RMSE, especially at relatively small compression efficiencies ($CE < 30$), but is unable to reach higher compression values. As regards GSVQ, we move along the RMSE *vs* CE curves by changing the threshold that governs the number of bits that are encoded into the residual stream. As discussed in Section 4.3.2, residual encoding is the operation that affects the most the performance of GSVQ. Clearly, the dictionary size K affects the maximum achievable

compression: although not shown in the plot, we tested different values of K, and found $K = 4$ to be a good trade-off between reconstruction error and compression efficiency. However, GSVQ happens to score worse than nearly all the aforementioned approaches: it could be possible to avoid sending the residual encoding stream, so as to reach higher compression efficiencies, but due to the use of a precomputed and fixed dictionary, this would lead to a very high RMSE and is not recommended. Interestingly, the performance of PCA closely matches that of LTC: although both algorithms rely on linear approximations, PCA is rather involved, whereas LTC has a much lighter computational cost, as we show shortly. Also, the trade-off curve for PCA is obtained by varying the number of principal components h from 100 (leftmost point in the figure) down to 5 (rightmost point) in steps of 5, whereas the performance of LTC is plotted varying ε within a continuous interval. Overall, LTC permits a fine-grained control of the RMSE *vs* compression trade-off, whereas this is not possible with PCA, especially at high compression efficiencies (small h). Finally, LTC provides a means to precisely control the maximum reconstruction error, through the parameter ε, whereas the number of retained principal components h does not provide any guarantee in terms of reconstruction accuracy[2]. As for the CS-based algorithms, neither SOMP-CS nor BSBL-CS provides satisfactory performance. The compression efficiency of SOMP-CS is rather small and the corresponding RMSE tends to diverge for, *e.g.*, CE larger than 5. As we shall see shortly below, the overall energy performance of SOMP-CS is unsatisfactory when compared to that of other algorithms and the compression strategy of BSBL-CS has the lowest energy consumption, but its intra-segment approach is less effective in terms of CE than that of other inter-segment schemes such as GSVQ and AE. Although the results that we show here for SOMP-CS and BSBL-CS were respectively obtained using the theory from [107] and [153], we found similar CE figures in other papers [108]. In these studies, the compression efficiency is defined as $\text{CE}' = ((W - m)/W) \times 100$, with W being the number of original samples and m the number of compressed samples that are transmitted to the receiver. With this definition, CS schemes achieve maximum efficiencies of 80-90%. We observe that these figures correspond to a CE ranging from 5 to 10 according to the definition that we use in the present work, *i.e.*, $\text{CE} = W/m$. Finally, for the AE, the number of inner neurons h is varied as a free parameter in $\{100, 50, 25, 10, 5, 2\}$: $h = 100$ is represented by the leftmost point in the graph, whereas the rightmost corresponds to $h = 2$. AE obtains the best performance in terms of both RMSE and CE, surpassing all the other schemes and providing relative errors of about 3%. The compression achieved through AE can be higher than 100 when $h = 2$, still maintaining a low

[2]With PCA, an inverse transform at the compressor is required to assess the reconstruction error provided by a certain value of h.

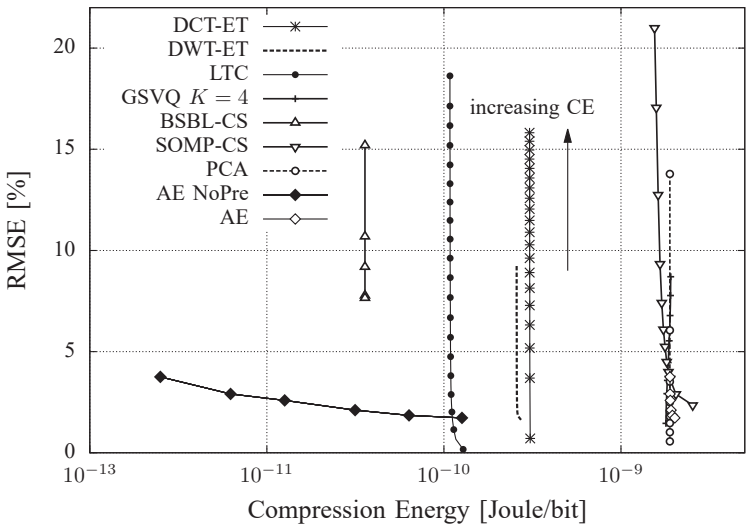


Figure 4.5: RMSE as a function of the energy consumption associated with the compression of ECG signals.

RMSE with respect to the other compression techniques. Also, going from $h = 2$ to $h = 5$ provides only a small benefit in terms of RMSE, while entailing a big leap in compression efficiency. Increasing h beyond 10 does not provide any considerable advantage in terms of representation accuracy, and has a negative impact on the compression performance. At small compression efficiencies, adaptive algorithms may be a valuable option: for instance, one may switch between LTC and AE as a function of the required compression level. At the highest values of CE, the gain of AE against LTC and DCT-ET (the second best algorithms) is 67% as regards compression efficiency, still achieving an RMSE almost 80% lower. The maximum RMSE is kept smaller than 4% in all cases.

In Figure 4.5, we show the RMSE and the energy drained for compression at the transmitter, expressed in Joule per bit of the original ECG sequence. These trade-off curves are obtained by varying the compression efficiency of each algorithm from 1 to the maximum achievable, which is scheme-specific. The RMSE increases with an increasing compression efficiency, whereas the compression energy depends weakly on CE. As expected, BSBL-CS has the smallest energy consumption. LTC is the second best, whereas SOMP-CS, GSVQ, PCA and AE perform very close to one another and have the worst energy consumption for compression. The good performance of BSBL-CS is due to its lightweight compression algorithm, which just multiplies the input signal by sparse binary matrices, with entries in $\{0, 1\}$. We underline that the energy consumption of the techniques on the right-hand side of the figure, including AE, is dominated by the preprocessing chain of Figure 4.2. In this plot, we also show the performance of AE by removing the contribution of the pre-processing blocks: the corresponding curves are referred to in the plot as "AE

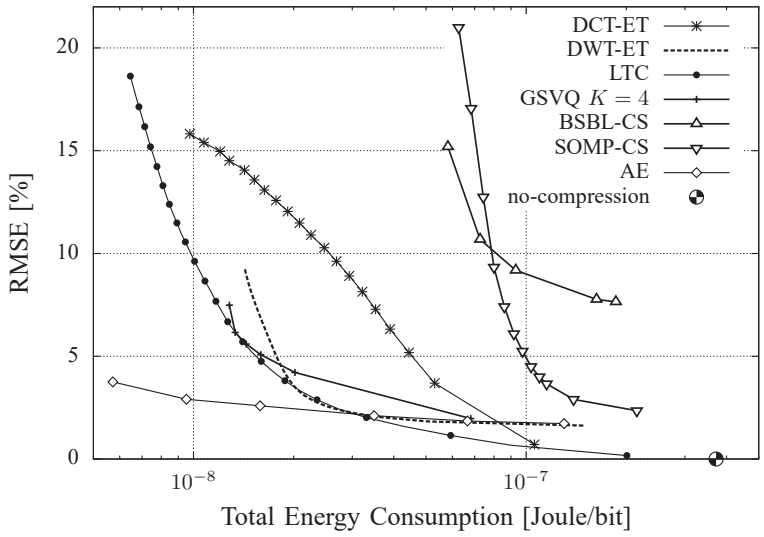


Figure 4.6: RMSE as a function of the total energy consumption (compression plus transmission) of ECG signals.

NoPre". Note that filtering is always performed to remove measurement artifacts and peak detection is also very often utilized to extract relevant signal features. Given this, the energy consumption associated with the required pre-processing functions may not be a problem, especially if these functions are to be executed anyway.

In Figure 4.6, we show the RMSE as a function of the *total energy consumption*, which is obtained summing the energy required for compression to that for the subsequent transmission of the compressed bitstream over a CC2541 Bluetooth low-energy wireless interface. This total energy is then normalized with respect to the number of bits in the original ECG signal. From this plot, we see that the total energy consumption is dominated by the transmission energy, which depends on the compression efficiency. In this respect, the best algorithms are LTC and AE, and the algorithm of choice depends on the target RMSE that, in turn, directly descends from the selected CE. As discussed above, an adaptive algorithm may be a good option, where for each value of CE the scheme providing the smallest RMSE is used. In Figure 4.6, the energy consumption when no compression is applied is also shown for comparison. We see that signal compression, and the subsequent reduction in size of the data to be transmitted, allows a considerable decrease in the total energy consumption. Specifically, LTC and AE enable energy savings of about one order of magnitude while providing RMSEs smaller than approximately 2%. The performance of AE is particularly striking as it allows saving up to two orders of magnitude in terms of energy consumption, by still keeping the RMSE around 4%. This motivates the use of signal compression techniques for continuous monitoring applications for IoT devices. Although not shown for the sake of readability of the plot, PCA nearly achieves the same performance of GSVQ, in the region of practical

Table 4.1: Average complexity [no. operations] and energy consumption [μJ] per ECG segment. RMSE is 7.5% for all algorithms except AE for which the average RMSE is 3.75% (the highest with AE, obtained with $h = 2$ neurons in the inner compression layer).

	DCT-ET	**DWT-ET**	**GSVQ**	**BSBL-CS**	**SOMP-CS**	**LTC**	**AE**
Additions	13463	7809.9	98114	3747	97302	1258.8	82439
Multiplications	9089.8	6883.5	82788	0	95805	0	81535.7
Divisions	0	323.2	1137.2	0	1045	634.1	938.6
Comparisons	30.4	7723.1	475.6	0	522	1231.1	462.7
Compression energy	0.74	0.88	6.47	**0.12**	6.84	0.37	5.83
TX energy	124.22	45.85	35.82	639.03	272.36	37.90	**13.45**
Total energy	124.96	46.73	42.29	639.15	279.20	38.27	**19.28**

interest. Note that the actual RMSE can be dynamically tuned at runtime, by allowing slightly less accurate representations (and thus much higher compressions) when no critical patterns are detected. Also, for AE, a visual inspection reveals that a RMSE smaller than 4% entails excellent approximations of the original biosignals, and that the error is mainly due to smoothing out spurious oscillations that are introduced and that are not filtered by the preprocessing chain of Figure 4.2. A breakdown of the complexity and energy consumption figures for the considered algorithms is provided in Table 4.1, for the same RMSE of 7.5%). These metrics were obtained for the PhysioNet ECG signals and represent the average complexity (expressed in terms of number of operations) and energy consumption (Joules) for the compression and transmission of a single ECG segment of data. From Table 4.1, we see that BSBL-CS is the most energy efficient in terms of compression, LTC is the second best, whereas SOMP-CS utilizes more energy as the ECG signal has to be segmented prior to performing the CS sampling, see Section 4.3.4. AE is more computationally demanding, but its maximum compression efficiency is much higher. Since the transmission energy dominates that needed for data processing, AE represents the best alternatives when all the sources of energy consumptions are added up.

4.4.2 Wearable ECG Signals

We now present some results for ECG signals that we acquired from a Zephyr BioHarness 3 wearable device [97]. To this end, we collected ECG traces from eleven healthy individuals, which were continuously recorded during working hours, *i.e.*, from 8 *a.m.* to 6 *p.m.*. These were sampled at a rate of 250 samples/s with each sample taking 12 bits.

The RMSE *vs* CE trade-off for these signals is shown in Figure 4.7 for the best performing compression algorithms. The results are similar to those of Figure 4.4, with the main difference that in this case the ECG signals have more artifacts and

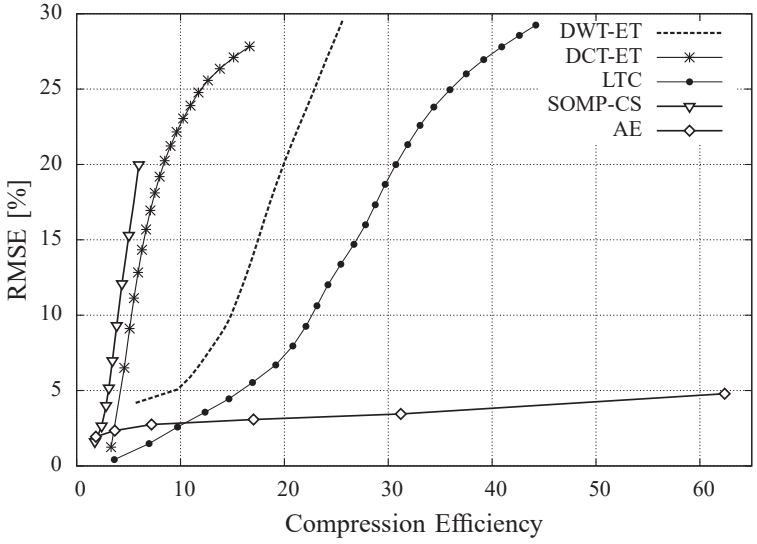


Figure 4.7: RMSE *vs* compression efficiency for ECG signals.

a higher variability. As such, the resulting RMSE is also higher for all schemes and the compression performance is degraded. The general trends and recommendations remain unchanged, *i.e.*, SOMP-CS and LTC are good choices at low up to intermediate compression efficiencies, whereas AE performs better at higher CEs. For this figure, BSBL-CS achieved high RMSE values maintaining low CEs, thus performing worse than its SOMP counterpart, and has thus been omitted. Similarly, GSVQ performed similar to AE in terms of RMSE, but only when $CE \leq 5$.

The energy consumption figures of all schemes, although slightly rescaled, have a totally similar behavior as those obtained with the PhysioNet MIT-BIH traces (see Figure 4.5 and Figure 4.6) and are thus not shown in the interest of space. In fact, the energy consumption is marginally affected by the non-steady behavior of wearable signals, which impacts more on the RMSE and compression performance.

As an illustrative example, in Figure 4.8 and Figure 4.9 we respectively look at the per segment RMSE and CE performance of LTC and AE. In Figure 4.8 we fix the compression efficiency to CE = 28 for all schemes and we show the RMSE for each segment considering 12 minutes of ECG readings from one of the subjects. Overall, LTC settles around an RMSE of 11% and AE achieves the best accuracy, *i.e.*, RMSE = 2.6%. Figure 4.9 shows the per segment compression efficiency for the same ECG trace by operating LTC and AE, so that their average RMSE is 3%. From this plot, we see that AE reaches much higher CEs, delivering strikingly good performance.

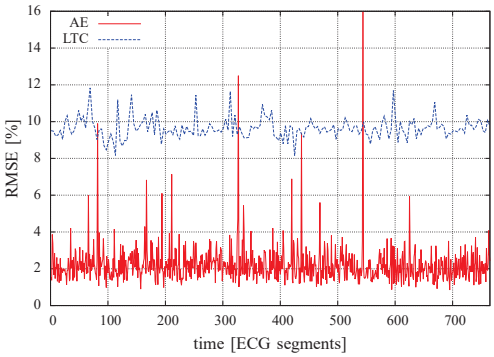


Figure 4.8: RMSE as a function of time. CE = 28 for both schemes, RMSE(LTC) = 11%, RMSE(OD) = 4% and RMSE(AE) = 2.6%.

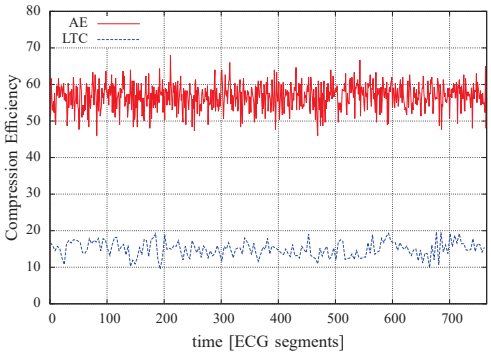


Figure 4.9: CE as a function of time. RMSE = 3% for all schemes, CE(LTC) = 15, CE(OD) = 19 and CE(AE) = 56.

4.4.3 PPG and RESP Signals

As a final result, in Figure 4.10 and 4.11 we respectively show the RMSE *vs* CE performance for PPG and respiratory (RESP) signals from the PhysioNet MIMIC-II waveform database [172]. In these graphs, we only show the performance of the two best algorithms, namely, LTC and AE. AE is plotted considering the number of inner neurons $h \in \{100, 50, 25, 10, 5, 2\}$, where $h = 100$ is represented by the leftmost point, whereas $h = 2$ by the rightmost, and outperforms all the remaining schemes for $h \leq 5$. Clearly, this technique is still effective for these signal types. For respiratory signals, LTC performs best for compression efficiencies up to 40; the compression efficiency obtained for PPG signals is smaller than that achieved for ECG and RESP, but this is due to the lower sampling rate in the PPG traces. For all signals, the RMSE of AE never exceeds 4%, while its CE respectively reaches 56 and 156 for PPG ad RESP when just two inner neurons ($h = 2$) are utilized. These results are impressive and motivate further research, especially to make the AE learning phase online and subject-adaptive.

4.5 Chapter conclusions

In this chapter, we advocated the use of lossy compression as a means to boost energy efficiency in wearable wireless devices. As a first contribution, we presented an original autoencoder based technique as an efficient and lightweight lossy compressor for biosignals. These neural network architectures are found to be extremely effective, leading to the highest compression efficiencies at a reasonable computational cost. Their performance is striking especially at very high compression rates, where just two inner neurons are utilized to represent input patterns comprising several hundreds of points, still providing very small approximation errors (usually

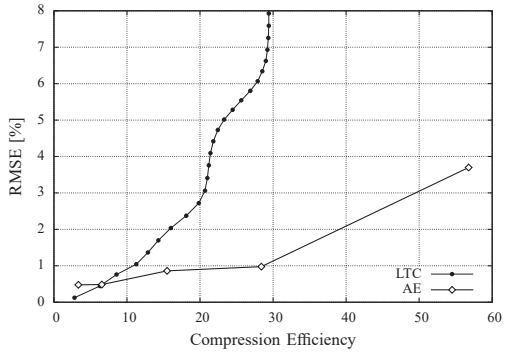


Figure 4.10: RMSE *vs* compression efficiency for PPG signals.

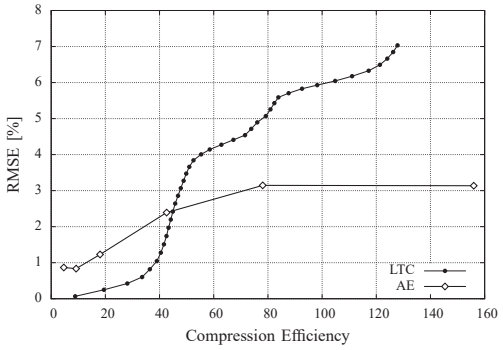


Figure 4.11: RMSE *vs* compression efficiency for respiratory signals.

the RMSE remains bounded within 4%). We then considered compression algorithms based on linear approximations. Despite their inherent simplicity, we found them to be quite effective and, when the required compression efficiency is not too high, they represent a good option among competing solutions. We also found that a recent scheme belonging to this class, called lightweight temporal compression, very closely matches the performance of principal component analysis, at a much smaller computational cost and additionally providing inbuilt guarantees on the maximum approximation error at the decompressor. The performance of these algorithms was numerically evaluated against that of the most prominent schemes from the literature, *i.e.*, Fourier and Wavelet transforms, compressive sensing and vector quantization techniques.

From the numerical analysis that we have carried out in this work, we have identified several avenues for future research. We have seen that the most promising means to reach high compression efficiencies is to exploit inter-segment correlation. Autoencoder-based algorithms belong to this category and do a very good job in all respects. Nevertheless, they suffer from a main drawback: these networks need to go through an offline training phase, during which their weights are shaped utilizing a representative dataset. Although they have excellent generalization capabilities, they will perform worse in representing input patterns that sharply differ from those in the dataset used for training. Hence, a desirable contribution would be to concoct a new neural network based algorithm which, after an initial offline training phase, would be able to adapt its weights to new input patterns in an online fashion, so as to properly react to changing input conditions. Another interesting subject for future investigations is the joint compression of multiple vitals, including respiratory rate, electrocardiogram, plethysmograph, and data from motion sensors.

CHAPTER 5

LEARNING METHODS FOR LONG-TERM CHANNEL GAIN PREDICTION IN WIRELESS NETWORKS

Efficiently allocating resources and predicting cell handovers is essential in modern wireless systems, but is only possible by means of an efficient way to estimate the future state of the network. In order to accomplish this, this chapter investigates two learning techniques to predict the long-term channel gains in a wireless network. Previous works in the literature found efficient methods to perform this prediction with the aid of a GPS signal: here, we predict the future channel gains using only past channel samples, without any additional geographical information.

5.1 Introduction to the chapter

Global mobile data traffic has grown 4,000-fold over the past 10 years [174], and its growth is expected to continue, fueled by the high bandwidth demands of multimedia applications (video alone already accounted for a majority of the total mobile data traffic in 2015). As mobile multimedia applications have strict Quality of Service (QoS) requirements, the development of optimal resource allocation techniques is a priority for both industry and academia. Applications such as video streaming would greatly benefit from an accurate channel prediction, and prediction-based adaptive streaming systems have already been proposed [175]. Most of the efforts in channel prediction have focused on Multiple Input Multiple Output (MIMO) techniques on short time horizons, but several works [176] [177] have expressed the need for long-term accurate channel gain predictions.

In this chapter, we investigate learning methods to predict the wireless channel gain on a long-term scale, without any inputs other than the time-averaged Received Signal Strength Indicator (RSSI). This prediction is performed by the Base Station

(BS), which is the only fixed element in the network: the BS can indirectly learn the mobility patterns of the mobile users and the fading characteristics of the channel by observing patterns in their RSSI. The two machine learning techniques we use are Graphical Bayesian (GB) models and Support Vector Regression machines (SVRs). The GB model can be used as a baseline, as it does not try to generalize its experience, but simply considers each class as a separate classification problem. SVRs are instead able to find and generalize patterns in the data, making better predictions with fewer data.

In the following, Section 5.2 presents the state of the art in channel prediction, and Section 5.3 illustrates the learning techniques we used. Section 5.4 presents the model we used to generate the data and the simulation results we obtained, and Section 5.5 concludes the paper.

5.2 State of the art

Wireless links are often modeled as Rayleigh fading channels. Most model-based prediction systems concentrate on short-term predictions of the fading envelope for wideband channels [178], and cannot be directly used for optimization at the higher layers. As mobility was not a central issue, the relatively predictable nature of path loss made long-range channel prediction an easy problem.

In [179], the authors predict future channel quality from receiver-side Channel State Information (CSI), but the Autoregressive (AR) filters they use are only accurate on a timescale of a few milliseconds. The work in [180] proposes an OFDM-specific prediction method based on time-domain statistics with a slightly longer range, but the timescales for accurate predictions are still far below 100 ms. Another AR model is proposed in [181], but its predictions are extremely short-range and how to choose the order of the filter is an issue. Several works in the literature have attempted to solve the long-term channel prediction problem with machine learning techniques. A typical example can be found in [182], where a distributed channel prediction algorithm for sensor networks is proposed, based on message passing techniques to minimize the Kullback-Leibler divergence between the expected prior distribution and the actual posterior. The problem of channel prediction is central in Cognitive Radio (CR) systems, and the authors in [183] introduce a Bayesian Network (BN) to solve this issue. Their BN predicts the future capacity for each possible CR configuration, and adapts to channel conditions online, but it is meant for Modulation and Coding System (MCS) selection rather than for optimization at higher layers. An empirical approach is followed in [184], combining a probing mechanism with Support Vector Regression (SVR) to predict link quality in dense wireless networks. Thanks to mobility, the probing system can learn about several

different topologies and extend this knowledge to larger, denser networks. Finally, the authors in [177] perform long-term channel predictions using both spatial and temporal information. A Gaussian Process (GP) regression technique is proposed, training the system through a series of routes on a city map. Their prediction method is robust against spatial errors, with better performance than both Support Vector Machines (SVMs) and AR filters. Although their method is sound for large-scale scenarios, it does not deal with smaller cells and needs Global Positioning System (GPS) information from clients, which might not be available.

5.3 Learning techniques

The two learning techniques we use are extremely versatile: they do not assume a specific channel model, so they can be trained and deployed on any wireless channel with only minor adjustments. In order to make the necessary data easy to obtain in a practical scenario, we decided to use the Received Signal Strength Indicator (RSSI) [185] as training data, averaging the data over a long enough window to reduce measurement errors and avoid sampling holes even in a congested network.

5.3.1 Graphical Bayesian Model

The GB model can be represented by the graph shown in Figure 5.1. As the Bayesian model only works for discrete attributes, the dynamic interval of the channel gain needs to be discretized into M classes. The memory-n Bayesian model uses the past n samples as features, resulting in M^n possible combinations. The predictor is essentially a classifier, in which the future channel sample is the correct class.

The multimodal classifier is implemented by a Dirichlet distribution [186] over the M-dimensional simplex, which is parameterized by a real non-negative vector α:

$$p(X = (x_1, \ldots, x_M)|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{M} x_i^{\alpha_i - 1}, \tag{5.1}$$

where the normalizing constant $B(\alpha)$ is the multivariate Beta function [187]. The random vector $X = (X_1, \ldots, X_M)$ is a probability distribution over the M classes, representing the probability that the given sample is in each class (*i.e.*, the probability distribution of the next channel sample). The expected value of X is given

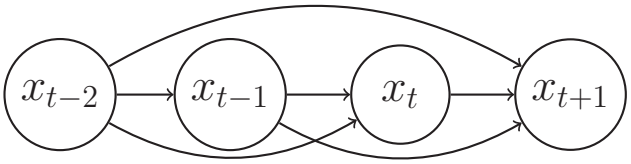


Figure 5.1: Representation of the graphical Bayesian model with 3-state memory

by:

$$\mathrm{E}[X_i] = \frac{\alpha_i}{\sum_{j=1}^{M} \alpha_j} \tag{5.2}$$

$$\mathrm{Var}[X_i] = \frac{\alpha_i \sum_{j \neq i} \alpha_j}{(\sum_{j=1}^{M} \alpha_j)^2 (\sum_{j=1}^{M} \alpha_j + 1)} \tag{5.3}$$

Intuitively, the value of α_i relative to the sum of the α vector is a measure of how probable a class is, and the value of the sum measures the uncertainty on that probability. As the conjugate distribution of the Dirichlet distribution is the Dirichlet-multinomial distribution, Bayesian inference can be performed by generating a new parameter vector α', defined as

$$\alpha'_i = \alpha_i + n_i, \tag{5.4}$$

where n_i is the number of observed transitions to class i. Then, the expected probability distribution of the next sample is given by

$$p(i) = \frac{\alpha'_i}{\sum_{j=1}^{M} \alpha'_j}, \tag{5.5}$$

and the predicted class then corresponds to the maximum probability value.

5.3.2 Support Vector Regression Machine

Support Vector Regression (SVR) machines [78, 79], minimize the following cost function:

$$C \sum_i E_\epsilon(f_{\mathbf{w}}(\mathbf{x}^{(i)}) - x_{t+1}^{(i)}) + \frac{1}{2}||\mathbf{w}||^2. \tag{5.6}$$

In equation (5.6), $f_{\mathbf{w}}$ is a function taking as input a memory-n feature vector $\mathbf{x}^{(i)} = (x_{t-n+1}, \ldots, x_t)$ and predicting a future sample $\hat{x}_{t+1}$, for a given training example i. The error between this predicted sample and the actual sample at time $t+1$, x_{t+1},

is then fed to an ϵ-insensitive error function

$$E_\epsilon(z) = \begin{cases} |z| - \epsilon & \text{if} \quad |z| > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

so that $f_{\mathbf{w}}$ is constrained to have maximum absolute prediction error lower than a given constant ϵ for all the training data. The second term in (5.6) accounts for regularization: the trade-off between the minimization of the two terms is governed by the constant C (the reader can refer to [81, 82] for more details). In (5.6), all

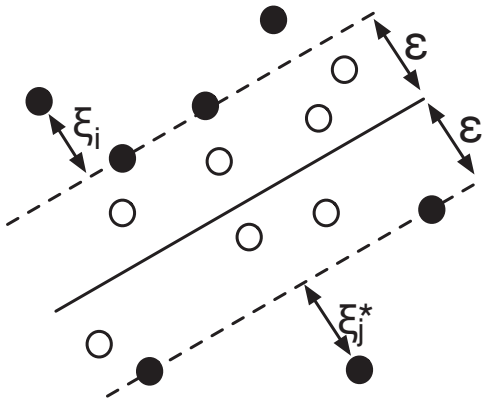


Figure 5.2: Graphical representation of an ϵ-tube with slack variables.

the training examples are assumed to lie in an "ϵ-tube" (see Figure 5.2). However, this is not verified in general, and (5.6) can be modified so as to allow for some tolerance in the prediction errors. Therefore, for each training example $\mathbf{x}^{(i)}$, it is possible to introduce *slack variables* ξ_i and ξ_i^*, where $\xi_i > 0$ is related to a point for which $(x_{t+1}^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)})) > \epsilon$, and $\xi_i^* > 0$ is related to a point for which $(f_{\mathbf{w}}(\mathbf{x}^{(i)}) - x_{t+1}^{(i)}) < -\epsilon$. Training examples are thus allowed to lie outside the ϵ-tube, as in Figure 5.2, provided that the corresponding slack variables are positive: this condition can be formulated as

$$-\epsilon - \xi_i^* \leq x_{t+1}^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)}) \leq +\epsilon + \xi_i \tag{5.8}$$

The optimization problem becomes

$$\min C \sum_i (\xi_i + \xi_i^*) + \frac{1}{2}||\mathbf{w}||^2, \tag{5.9}$$

subject to the constraints $\xi_i, \xi_i^* \geq 0$, and (5.8). It can be seen that only the examples outside the ϵ-tube contribute to the cost, with deviations being linearly penalized.

Computing the dual formulation of (5.9), exploiting the Karush-Kuhn-Tucker conditions [80, 188], and assuming that $f_{\mathbf{w}}$ is simply a linear function of the inputs, i.e., $f_{\mathbf{w}}(\mathbf{x}^{(i)}) =< \mathbf{w}, \mathbf{x}^{(i)} > +b$, it can be found that

$$\mathbf{w} = \sum_i (\lambda_i - \lambda_i^*)\mathbf{x}^{(i)}, \tag{5.10}$$

were λ_i and λ_i^* are the Lagrange multipliers. The prediction function becomes

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_i (\lambda_i - \lambda_i^*) < \mathbf{x}^{(i)}, \mathbf{x} > +b. \tag{5.11}$$

In (5.10), the weight vector $\mathbf{w}$ is a function of the training examples $\mathbf{x}^{(i)}$; however, only those examples such that $\lambda_i - \lambda_i^* \neq 0$, called *Support Vectors* (SVs), have to be evaluated in (5.10) and (5.11). Finally, it is possible to allow the prediction function $f_{\mathbf{w}}$ to be non linear in each training example $\mathbf{x}^{(i)}$, so as to allow better generalization over non linear target functions. In fact, in (5.11), the SVs only appear inside scalar products, and (5.10) does not need to be calculated explicitly. Therefore, it can be proved that $< \mathbf{x}^{(i)}, \mathbf{x} >$ in (5.11) can be replaced by particular non linear functions $k(\mathbf{x}^{(i)}, \mathbf{x})$, known as *kernels*, which correspond to scalar products between non linear transformations of $\mathbf{x}^{(i)}$ and $\mathbf{x}$. Substituting $k(\mathbf{x}^{(i)}, \mathbf{x})$ in (5.11), we thus obtain the optimal prediction function in a non-linear *feature space*, rather than in input space:

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{m} (\lambda_i - \lambda_i^*) k(\mathbf{x}^{(i)}, \mathbf{x}) + b. \tag{5.12}$$

5.4 Simulation settings and results

The two learning methods were trained on the same RSSI data, generated by simulating a realistic urban scenario. The wireless channel we considered used a 945 MHz downlink carrier frequency (one of the commercial bands used in LTE), and the users moved in a Manhattan grid of 100 buildings. The grid we used is composed of 20 m wide square buildings, with 10 m wide one-way streets at each corner. The BS is placed at coordinates $(140, 140)$, on top of a building close to the center of the simulation area.

5.4.1 Propagation loss and fading

The propagation loss was computed with the open-source system-level network simulator NS–3 [189]. In particular, we used the LTE module [190] and a radio propagation model called Hybrid Buildings Propagation Loss Model, which chooses the correct propagation model based on the reciprocal position of transmitter and re-

ceiver (both outdoors, both indoors, only one indoors). This model also takes into account the external wall penetration loss (for different types of buildings, *i.e.*, concrete with windows, concrete without windows, stone blocks, wood), and the internal wall penetration loss.

We used the NS–3 simulation to create a square grid of path loss measures in our urban scenario, with a sampling distance of 33 cm. The path loss was then approximated as a linear combination of the 4 closest points in the grid, weighted by the relative distance. The main parameters of the ns–3 simulation are listed in Table 5.1.

The fading and shadowing processes were both generated using MATLAB, implementing well-known models. We used the log-normal model for shadowing, with a standard deviation of 4 dB and a correlation distance of 8 m. Finally, Doppler fading was modeled with a Rayleigh distribution, using the parameters listed in Annex B.2 of [191] and the MATLAB Welch periodogram method. In the fading calculation, the node speed was assumed to be constant, simplifying the computation significantly with negligible error.

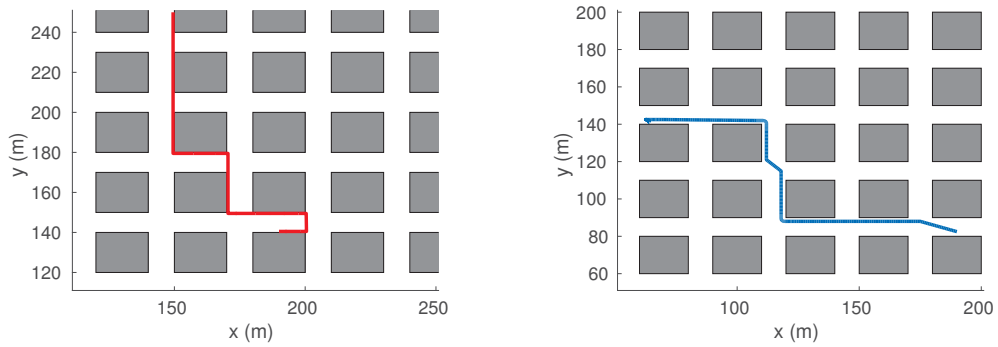
5.4.2 Mobility model

We used two mobility models: *pedestrian* and *vehicular*. In both models, the user goes from point A to point B by choosing the direction that takes him/her closer to point B at each intersection.

In the *pedestrian* model, a person walks at a constant speed of 1.5 m/s along the side of the nearest building at a distance of 0.5 m. Road crossings are placed at each intersection, and the pedestrian waits for a random time between 0 and 5 seconds before crossing to wait for cars.

Table 5.1: Path loss computation parameters

Parameter	Value
Downlink carrier frequency	945 MHz
Uplink carrier frequency	900 MHz
RB bandwidth	180 kHz
Available bandwidth	25 RB
Number of eNBs	1
eNBs beamwidth	360° (isotropic)
TX power used by eNBs	43 dBm
eNB noise figure	3 dB
Number of buildings	100
Floors for each building	5
Radio Environment Map resolution	9 samples/m^2



(a) Pedestrian (b) Vehicular

Figure 5.3: Examples of trajectories for the two mobility models.

In the *vehicular* model, the driver keeps a constant speed of 15 m/s while driving straight, switching between the 3 lanes by moving at a 45 degree angle. Before a turn, the driver switches to the correct lane (*e.g.*, switching to the right lane before turning right), then slows down to 5 m/s with a constant deceleration in the 5 meters before the curve and finally makes a circular turn. After reaching the destination, the driver stops and reverses to slowly park on the curb, with a semi-circular trajectory.

The channel data was generated by running the urban scenario 5000 times for the pedestrian model and 10000 for the vehicular model, obtaining 3-4 days of data for the vehicular model and 20 hours for the pedestrian model (the car reaches its destination faster, so the traces are shorter). Two example trajectories for both models are shown in Figure 5.3.

5.4.3 Learning parameters and results

Both prediction methods were trained on the full dataset, with two different sampling rates: the channel was averaged over a window of 1 s and 0.5 s. The Bayesian model used a Gaussian prior, centered on the last known channel sample; the probability vector for all classes was multiplied by a factor k to obtain the Dirichlet parameter vector α. Both the prior weight factor k and the variance σ of the Gaussian distribution were optimized as hyperparameters by cross-validation. As regards the SVR learning algorithm, we found the Radial Basis Function (RBF) kernel: $k(\mathbf{z}_i, \mathbf{z}_j) = e^{-\gamma||\mathbf{z}_i - \mathbf{z}_j||^2}$ to perform best with respect to other possible kernel choices. In this case, the hyperparameters of the model are γ and C in (5.9): a grid search on (γ, C) pairs was thus performed, and the one with the best cross-validation RMSE was selected. After cross-validation, the performance of both prediction methods was measured on a previously unknown test set.

Figure 5.4 shows the prediction RMSE for the pedestrian mobility model: the quality of the prediction is very good even with the simpler model, as pedestrians

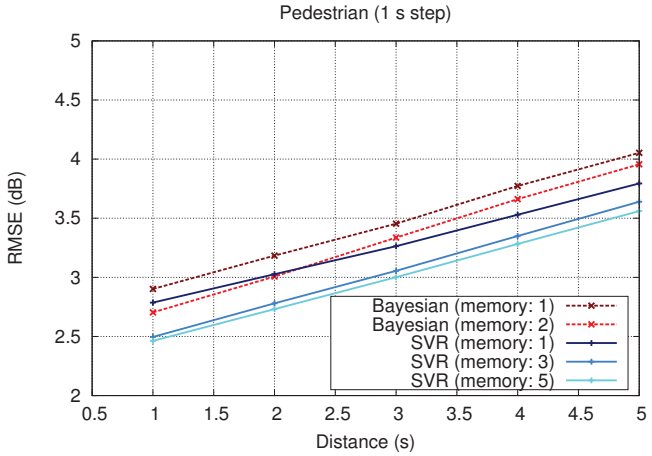


Figure 5.4: Prediction error for the pedestrian scenario (1 s window)

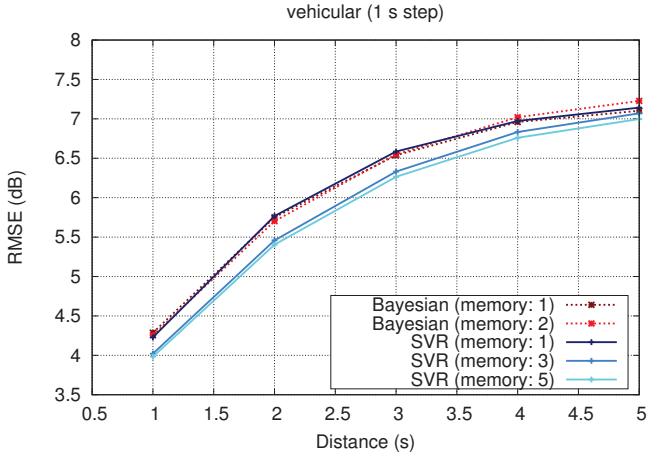


Figure 5.5: Prediction error for the vehicular scenario (1 s window)

are slow and generally highly predictable. As expected, SVR clearly outperforms the naive Bayesian model, exploiting its generalization capabilities and to better capture the features of the model. The gain of the longer memory is less pronounced for the Bayesian model, as it is overshadowed by the small size of the dataset (a longer memory means that a bigger dataset is necessary, and the memory-3 Bayesian model is not plotted, as its performance is not better than that with memory 2).

In the vehicular scenario, the RMSE is higher and the performance gap between the two methods is smaller (see Figure 5.5). The Bayesian model even outperforms the SVR if the prediction is more than 3 seconds ahead, but a prediction error of more than 7 dB is only slightly better than no prediction at all (the prediction RMSE when using a memoryless channel model is about 8 dB). This may be due to the high speed of the vehicles ($\sim$ 10 times the speed of the pedestrians), which makes accurate generalizations about the evolution of the channel hard.

Figure 5.6 and Figure 5.7 show the performance of the Bayesian method with a channel sampling window of 0.5 s; due to the computational cost of the SVR

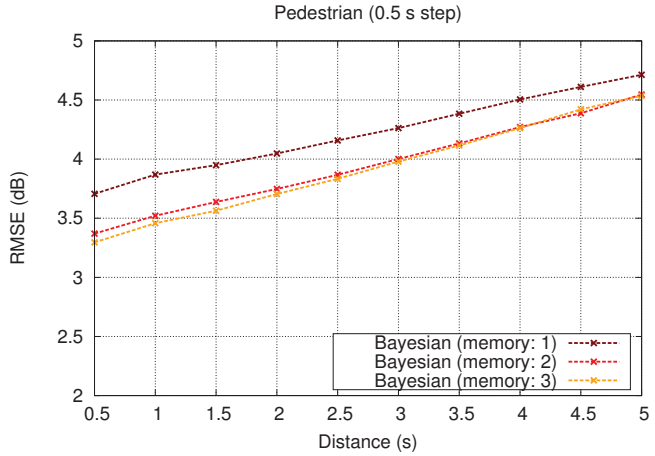


Figure 5.6: Prediction error for the pedestrian scenario (0.5 s window)

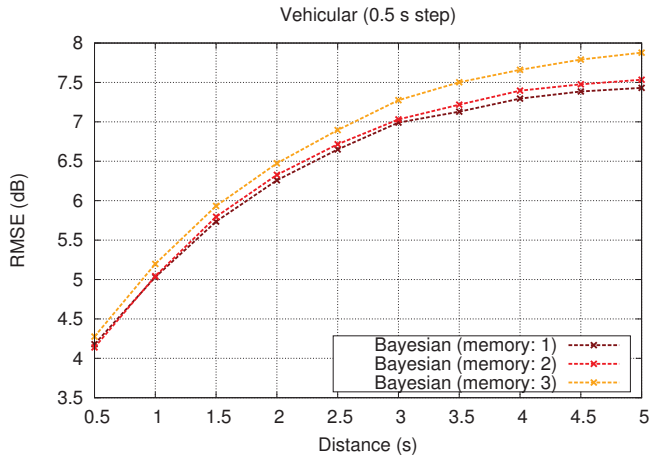


Figure 5.7: Prediction error for the vehicular scenario (0.5 s window)

training, its performance in this case has not been evaluated. The figures show that the trend in the performance of the Bayesian method is essentially the same, although the error is higher; the performance of the memory-3 Bayesian model hints that a longer memory is beneficial for the pedestrian model, but it loses most of its benefits in the more chaotic vehicular scenario unless a bigger dataset is used.

Finally, Figure 5.8 shows the performance of the two predictors when they are trained with a reduced dataset: the two predictors were trained on 20% of the available data in the vehicular scenario with a 1 second step. The plot shows how the performance of the SVR degrades far less than that with the Bayesian model, thanks to its better ability to generalize from experience. Indeed, the reduced-dataset SVR performs better than the full-dataset Bayesian model when the prediction distance is less than 5 seconds.

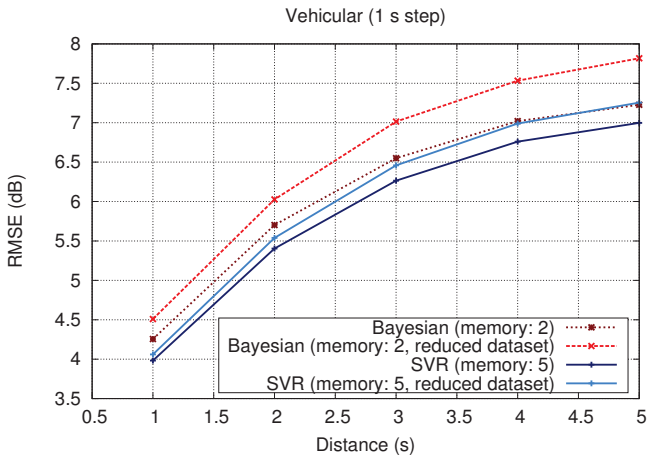


Figure 5.8: Prediction error for the vehicular scenario (1 s window) including results from a partial training set

5.5 Chapter conclusions

In this chapter, we described and tested two learning-based methods to use past wireless channel information to predict the future channel gain. We compared the performance of the two methods over a synthetically generated dataset with random mobility for both pedestrian and vehicular scenarios.

The training process was performed with just a few hours of RSSI data, so a BS with multiple connected users might be able to quickly gather the necessary training data and achieve a high-quality prediction in a very short time. However, the computational cost of the training itself is not negligible; while SVRs show a clear performance gain in both scenarios, the Bayesian model might be enough for applications that need a lower precision. It is worth noting that the SVR can have a satisfactory performance even when trained using a reduced dataset: this makes it ideal if the limiting factor is not the computational capability, but the size of the available dataset (*e.g.*, in adaptive systems that are trained online to follow a time-varying scenario). The quality of the predictions is generally high, and the RMSE is almost as low as the results shown in [177], but without the use of GPS data, which greatly impacts battery lifetime.

Future work may include a refinement of the prediction methods and the training of the predictors on data from real cellular systems. Finally, a promising development might include the creation of a prediction-based resource optimization system like the one presented in [175].

CHAPTER 6

END TO END LEARNING FOR SELF-DRIVING CARS

From January, 2016 to July, 2016 I was on a leave at the autonomous driving team of Nvidia in Holmdel, NJ, USA, as a deep learning research intern. There, I participated in the effort of developing Nvidia's first self-driving car, known as "BB8", by applying state-of-the-art deep learning techniques to teach a car how to drive, only exploiting road images from a front-facing camera. Due to a non disclosure agreement signed with the company, the aforementioned results can not be described in detail. Hence, this chapter gives a general overview of the results obtained by the entire research group, during the period I spent as an intern.

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads. The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. We never explicitly trained it to detect, for example, the outline of roads. Compared to explicit decomposition of the problem, such as lane marking detection, path planning, and control, our end-to-end system optimizes all processing steps simultaneously. We argue that this will eventually lead to better performance and smaller systems. Better performance will result because the internal components self-optimize to maximize the overall system performance, instead of optimizing human-selected intermediate criteria, e.g.*, lane detection. Such criteria understandably are selected for ease of human interpretation which doesn't automatically guarantee maximum system performance. Smaller networks are possible because the system learns to solve the problem with the minimal*

number of processing steps. We used an NVIDIA DevBox and Torch 7 for training and an NVIDIA DRIVE PX self-driving car computer also running Torch 7 for determining where to drive. The system operates at 30 frames per second (FPS).

6.1 Introduction to the chapter

CNNs [192] have revolutionized pattern recognition [193]. Prior to the widespread adoption of CNNs, most pattern recognition tasks were performed using an initial stage of hand-crafted feature extraction followed by a classifier. The breakthrough of CNNs is that features are learned automatically from training examples. The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D nature of images. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations. While CNNs with learned features have been in commercial use for over twenty years [194], their adoption has exploded in the last few years because of two recent developments. First, large, labeled data sets such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [195] have become available for training and validation. Second, CNN learning algorithms have been implemented on the massively parallel graphics processing units (GPUs) which tremendously accelerate learning and inference. In this paper, we describe a CNN that goes beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile. The groundwork for this project was done over 10 years ago in a Defense Advanced Research Projects Agency (DARPA) seedling project known as DARPA Autonomous Vehicle (DAVE) [196] in which a sub-scale radio control (RC) car drove through a junk-filled alley way. DAVE was trained on hours of human driving in similar, but not identical environments. The training data included video from two cameras coupled with left and right steering commands from a human operator. In many ways, DAVE-2 was inspired by the pioneering work of Pomerleau [197] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. It demonstrated that an end-to-end trained neural network can indeed steer a car on public roads. Our work differs in that 25 years of advances let us apply far more data and computational power to the task. In addition, our experience with CNNs lets us make use of this powerful technology. (ALVINN used a fully-connected network which is tiny by today's standard.) While DAVE demonstrated the potential of end-to-end learning, and indeed was used to justify starting the DARPA Learning Applied to Ground Robots (LAGR) program [198], DAVE's performance was not sufficiently reliable to provide a full alternative to more modular approaches to off-road driving. DAVE's mean distance between crashes was about 20 meters in complex environments. Nine months ago, a new effort

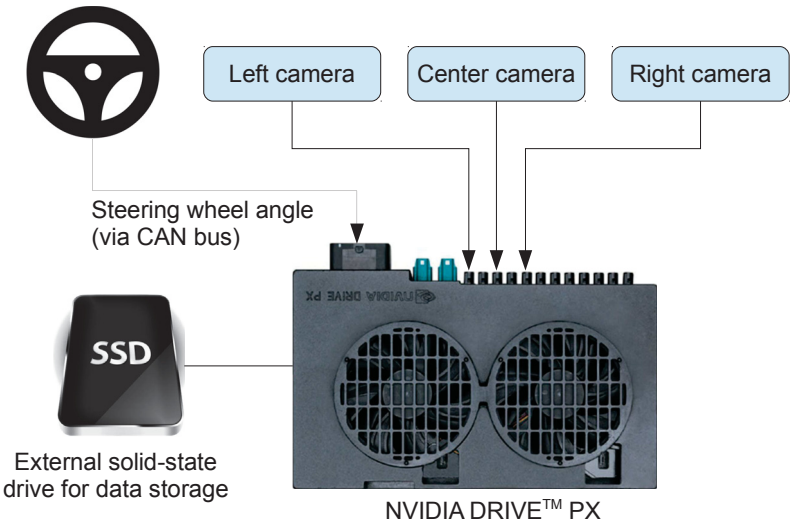


Figure 6.1: High-level view of the data collection system.

was started at NVIDIA that sought to build on DAVE and create a robust system for driving on public roads. The primary motivation for this work is to avoid the need to recognize specific human-designated features, such as lane markings, guard rails, or other cars, and to avoid having to create a collection of "if, then, else" rules, based on observation of these features. This section describes preliminary results of this new effort.

6.2 Overview of the DAVE-2 System

Figure 6.1 shows a simplified block diagram of the collection system for training data for DAVE-2. Three cameras are mounted behind the windshield of the data-acquisition car. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver. This steering command is obtained by tapping into the vehicle's Controller Area Network (CAN) bus. In order to make our system independent of the car geometry, we represent the steering command as $^1/_r$ where r is the turning radius in meters. We use $^1/_r$ instead of r to prevent a singularity when driving straight (the turning radius for driving straight is infinity). $^1/_r$ smoothly transitions through zero from left turns (negative values) to right turns (positive values).

Training data contains single images sampled from the video, paired with the corresponding steering command ($^1/_r$). Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. Otherwise the car will slowly drift off the road. The training data is therefore augmented with additional images that show the car in different shifts from the center of the lane and rotations from the direction of the road.

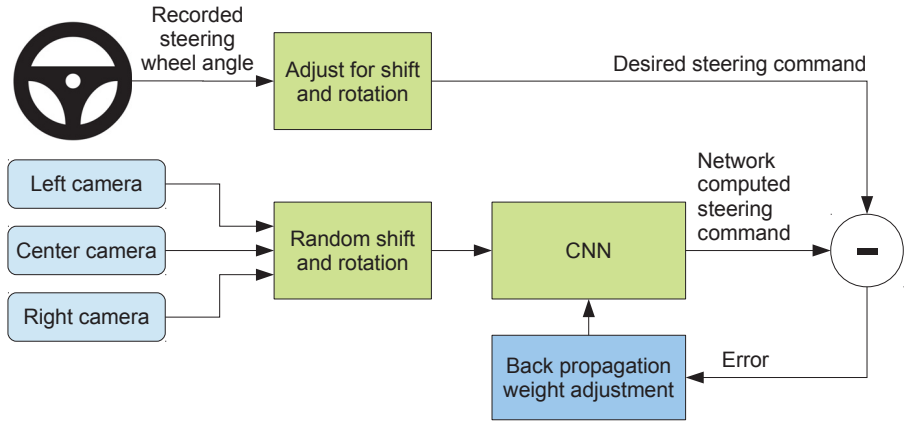


Figure 6.2: Training the neural network.

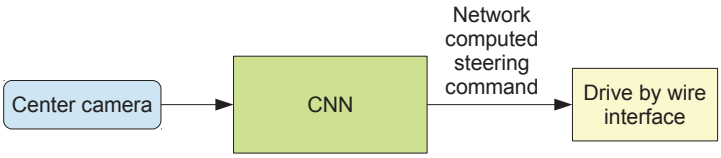


Figure 6.3: The trained network is used to generate steering commands from a single front-facing center camera.

Images for two specific off-center shifts can be obtained from the left and the right camera. Additional shifts between the cameras and all rotations are simulated by viewpoint transformation of the image from the nearest camera. Precise viewpoint transformation requires 3D scene knowledge which we don't have. We therefore approximate the transformation by assuming all points below the horizon are on flat ground and all points above the horizon are infinitely far away. This works fine for flat terrain but it introduces distortions for objects that stick above the ground, such as cars, poles, trees, and buildings. Fortunately these distortions don't pose a big problem for network training. The steering label for transformed images is adjusted to one that would steer the vehicle back to the desired location and orientation in two seconds. A block diagram of our training system is shown in Figure 6.2. Images are fed into a CNN which then computes a proposed steering command. The proposed command is compared to the desired command for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. The weight adjustment is accomplished using back propagation as implemented in the Torch 7 machine learning package.

Once trained, the network can generate steering from the video images of a single center camera. This configuration is shown in Figure 6.3.

6.3 Data collection

Training data was collected by driving on a wide variety of roads and in a diverse set of lighting and weather conditions. Most road data was collected in central New Jersey, although highway data was also collected from Illinois, Michigan, Pennsylvania, and New York. Other road types include two-lane roads (with and without lane markings), residential roads with parked cars, tunnels, and unpaved roads. Data was collected in clear, cloudy, foggy, snowy, and rainy weather, both day and night. In some instances, the sun was low in the sky, resulting in glare reflecting from the road surface and scattering from the windshield.

Data was acquired using either our drive-by-wire test vehicle, which is a 2016 Lincoln MKZ, or using a 2013 Ford Focus with cameras placed in similar positions to those in the Lincoln. The system has no dependencies on any particular vehicle make or model. Drivers were encouraged to maintain full attentiveness, but otherwise drive as they usually do. As of March 28, 2016, about 72 hours of driving data was collected.

6.4 Network architecture

We train the weights of our network to minimize the mean squared error between the steering command output by the network and the command of either the human driver, or the adjusted steering command for off-center and rotated images (see Section 6.5.2). Our network architecture is shown in Figure 6.4. The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The input image is split into YUV planes and passed to the network. The first layer of the network performs image normalization. The normalizer is hard-coded and is not adjusted in the learning process. Performing normalization in the network allows the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing. The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers. We follow the five convolutional layers with three fully connected layers leading to an output control value which is the inverse turning radius. The fully connected layers are designed to function as a controller for steering, but we note that by training the system end-to-end, it is not possible to make a clean break between which parts of the network function primarily as feature extractor and which serve as controller.

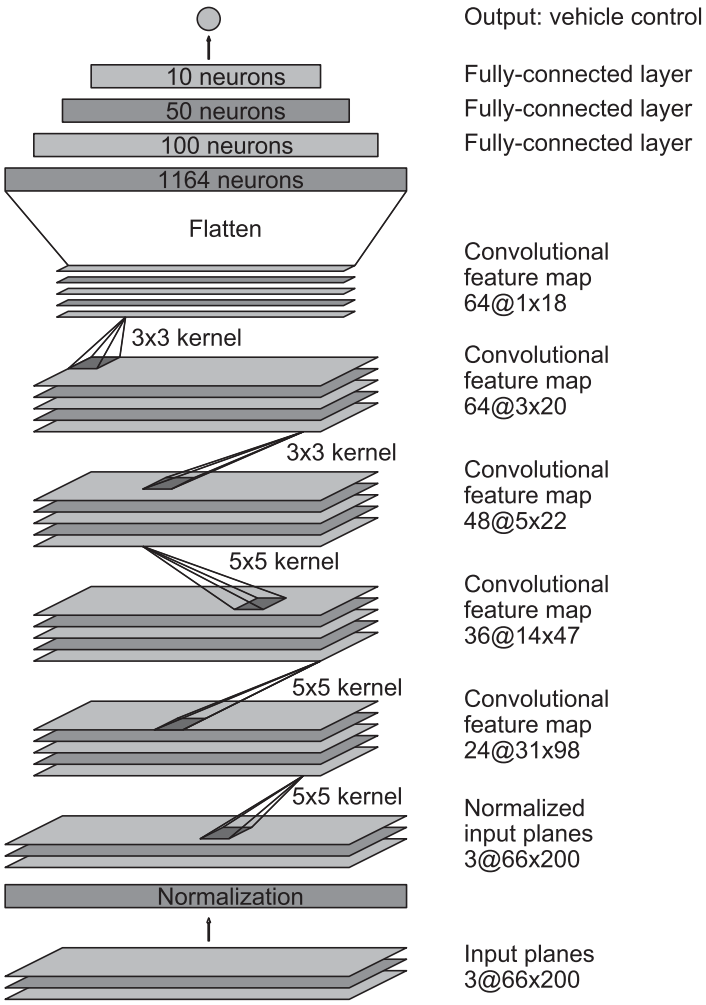


Figure 6.4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

6.5 Training details

6.5.1 Data selection

The first step to training a neural network is selecting the frames to use. Our collected data is labeled with road type, weather condition, and the driver's activity (staying in a lane, switching lanes, turning, and so forth). To train a CNN to do lane following we only select data where the driver was staying in a lane and discard the rest. We then sample that video at 10 FPS. A higher sampling rate would result in including images that are highly similar and thus not provide much useful information. To remove a bias towards driving straight the training data includes a higher proportion of frames that represent road curves.

6.5.2 Augmentation

After selecting the final set of frames we augment the data by adding artificial shifts and rotations to teach the network how to recover from a poor position or orientation. The magnitude of these perturbations is chosen randomly from a normal distribution. The distribution has zero mean, and the standard deviation is twice the standard deviation that we measured with human drivers. Artificially augmenting the data does add undesirable artifacts as the magnitude increases (see Section 6.2).

6.6 Simulation

Before road-testing a trained CNN, we first evaluate the network performance in simulation. A simplified block diagram of the simulation system is shown in Figure 6.5. The simulator takes pre-recorded videos from a forward-facing on-board camera on a human-driven data-collection vehicle and generates images that approximate what would appear if the CNN were, instead, steering the vehicle. These test videos are time-synchronized with recorded steering commands generated by the human driver. Since human drivers might not be driving in the center of the lane all the time, we manually calibrate the lane center associated with each frame in the video used by the simulator. We call this position the "ground truth". The simulator transforms the original images to account for departures from the ground truth. Note that this transformation also includes any discrepancy between the human driven path and the ground truth. The transformation is accomplished by the same methods described in Section 6.2. The simulator accesses the recorded test video along with the synchronized steering commands that occurred when the video was captured. The simulator sends the first frame of the chosen test video, adjusted for any departures from the ground truth, to the input of the trained CNN. The CNN then returns a steering command for that frame. The CNN steering commands as well as the recorded human-driver commands are fed into the dynamic model [199] of the vehicle to update the position and orientation of the simulated vehicle. The simulator then modifies the next frame in the test video so that the image appears as if the vehicle were at the position that resulted by following steering commands from the CNN. This new image is then fed to the CNN and the process repeats. The simulator records the off-center distance (distance from the car to the lane center), the yaw, and the distance traveled by the virtual car. When the off-center distance exceeds one meter, a virtual human intervention is triggered, and the virtual vehicle position and orientation is reset to match the ground truth of the corresponding frame of the original test video.

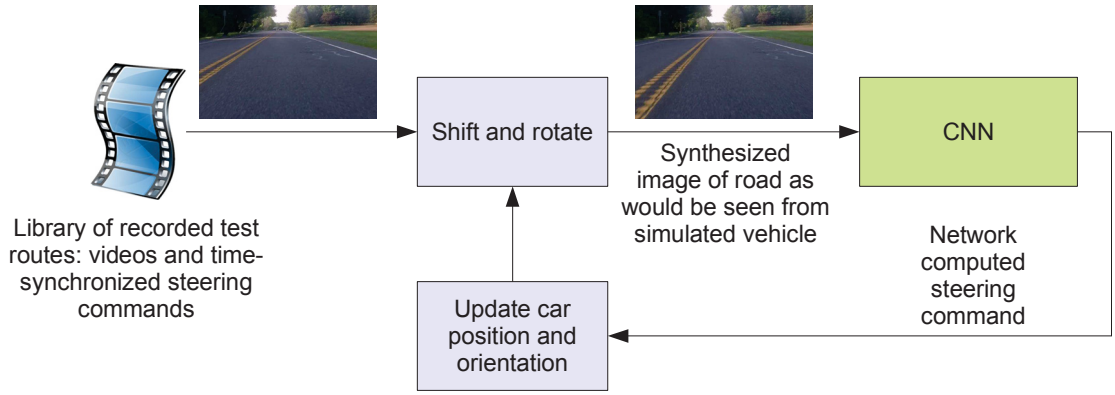


Figure 6.5: Block-diagram of the drive simulator.

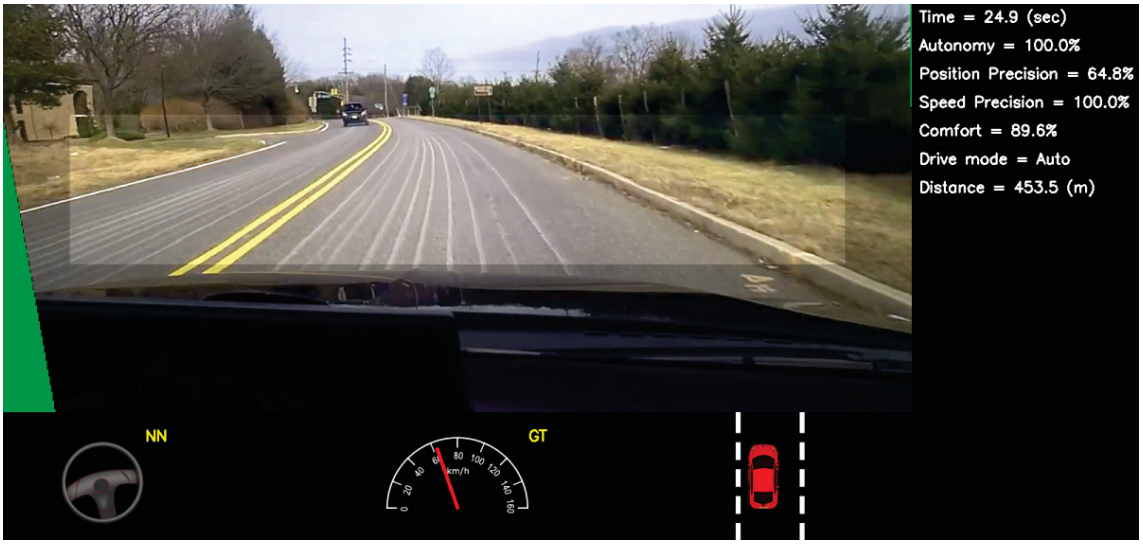


Figure 6.6: Screen shot of the simulator in interactive mode. See Section 6.7.1 for explanation of the performance metrics. The green area on the left is unknown because of the viewpoint transformation. The highlighted wide rectangle below the horizon is the area which is sent to the CNN.

6.7 Evaluation

Evaluating our networks is done in two steps, first in simulation (see Figure 6.6), and then in on-road tests. In simulation we have the networks provide steering commands in our simulator to an ensemble of prerecorded test routes that correspond to about a total of three hours and 100 miles of driving in Monmouth County, NJ. The test data was taken in diverse lighting and weather conditions and includes highways, local roads, and residential streets.

6.7.1 Simulation tests

We estimate what percentage of the time the network could drive the car (autonomy). The metric is determined by counting simulated "human interventions" (see Section 6.6). These interventions occur when the simulated vehicle departs from the center line by more than one meter. We assume that in real life an actual intervention would require a total of six seconds: this is the time required for a human to retake control of the vehicle, re-center it, and then restart the self-steering mode. We calculate the percentage autonomy by counting the number of interventions, multiplying by 6 seconds, dividing by the elapsed time of the simulated test, and then subtracting the result from 1:

$$autonomy = \left(1 - \frac{\#interventions \times 6\ [seconds]}{elapsed\ time\ [seconds]}\right) \times 100 \tag{6.1}$$

Thus, if we had 10 interventions in 600 seconds, we would have an autonomy value of

$$\left(1 - \frac{10 \times 6}{600}\right) \times 100 = 90\% \tag{6.2}$$

6.7.2 On-road tests

After a trained network has demonstrated good performance in the simulator, the network is loaded on the DRIVETM PX in our test car and taken out for a road test. For these tests we measure performance as the fraction of time during which the car performs autonomous steering. This time excludes lane changes and turns from one road to another. For a typical drive in Monmouth County NJ from our office in Holmdel to Atlantic Highlands, we are autonomous approximately 98% of the time. We also drove 10 miles on the Garden State Parkway (a multi-lane divided highway with on and off ramps) with zero intercepts. A video of our test car driving in diverse conditions can be seen in [200].

6.7.3 Visualization of internal CNN state

Figure 6.7 and Figure 6.8 show the activations of the first two feature map layers for two different example inputs, an unpaved road and a forest. In case of the unpaved road, the feature map activations clearly show the outline of the road while in case of the forest the feature maps contain mostly noise, *i.e.*, the CNN finds no useful information in this image. This demonstrates that the CNN learned to detect useful road features on its own, *i.e.*, with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads, for example.

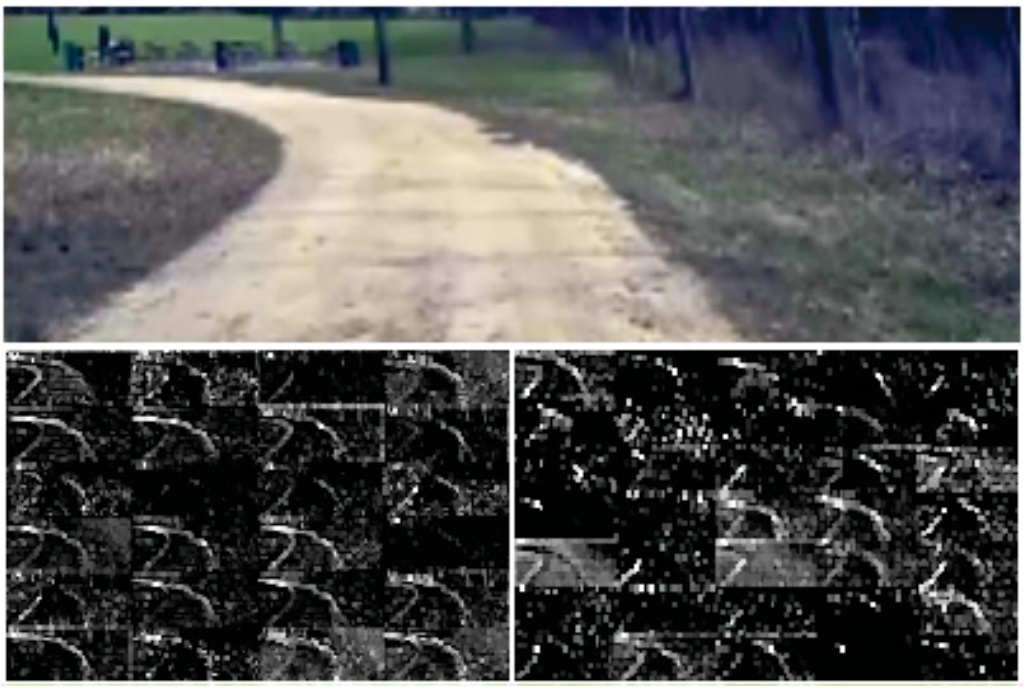


Figure 6.7: How the CNN "sees" an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, *i.e.*, with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

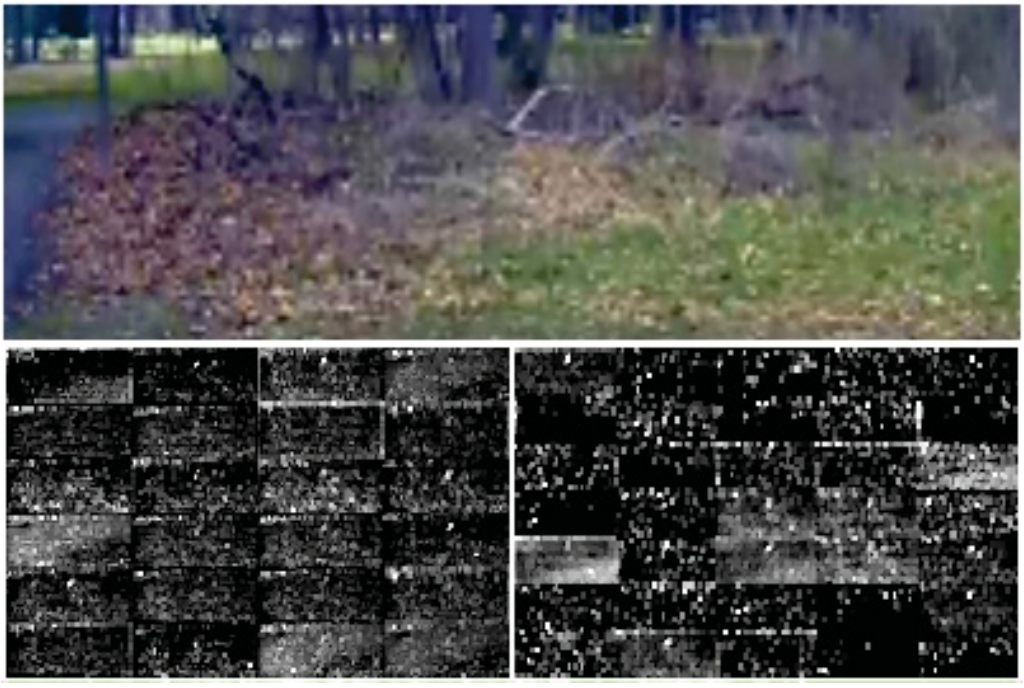


Figure 6.8: Example image with no road. The activations of the first two feature maps appear to contain mostly noise, *i.e.*, the CNN does not recognize any useful features in this image.

6.8 Chapter conclusions

We have empirically demonstrated that CNNs are able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction, path planning, and control. A small amount of training data from less than a hundred hours of driving was sufficient to train the car to operate in diverse conditions, on highways, local and residential roads in sunny, cloudy, and rainy conditions. The CNN is able to learn meaningful road features from a very sparse training signal (steering alone). The system learns for example to detect the outline of a road without the need of explicit labels during training. More work is needed to improve the robustness of the network, to find methods to verify the robustness, and to improve visualization of the network-internal processing steps.

CHAPTER 7

CONCLUSIONS

In this thesis, we analyzed how recent advancements in Stochastic Optimization and Machine Learning techniques have made it possible to effectively address common wireless networking problems. In particular:

Chapter 2 dealt with a scenario in which a pair of battery-equipped sensors has to report to a Central Controller (CC). Differently from typical wireless networks, in which the goal of the transmission policy is to reduce the average amount of energy utilization, so as to preserve battery charges, here we considered sensors with Energy Harvesting capabilities. As a result, the optimal policy is related to an efficient management of the energy scavenged from the environment, trying to avoid both energy outage (*i.e.*, no energy available at the device) and energy overflow (*i.e.*, harvested energy is "wasted" due to an already full battery). To prevent mutual interference between the devices, a centralized transmission policy was designed, in which the CC allows, at any time, the transmission of only one device, based on the residual energy available at its battery. Modeling the randomness in the energy arrivals as a Markov Decision Process, we first computed the performance of an optimal policy, *i.e.*, a policy with perfect knowledge of the State-of-Charge (SOC) of both devices. Later, we analyzed the impact of a partial knowledge of the SOC at the CC, showing that almost optimal performance can be achieved using sufficiently large batteries, even though the CC knows if, at any given time instant, the level of a device battery is either "LOW" or " HIGH". Different configurations of the system have been considered, paving the way for the analysis of larger multi-device environments.

Chapter 3 presented an original estimator for the Expected Time of Arrival (ETA) related to file transmissions over a WiFi network consisting of a common

transmitter and a number of receivers. Varying multiple networking parameters *e.g.*, distance between the transmitter and the receivers, number of nodes simultaneously receiving data, transmission power, WiFi transmission channel and part of the day in which the transmission occurred, we collected a large dataset specifying, for each configuration and at each time instant, the amount of time needed to complete the transmission, *i.e.*, the ETA. Then, a prediction algorithm for the ETA was designed, comparing multiple machine learning techniques and finally selecting the best performing one. The goal was to design an algorithm ready to be implemented in each receiver, without the need to modify any transmission protocol already existing. In order to do this, the prediction algorithm was split into two parts: the former exploits the knowledge of all the aforementioned parameters except for the number of simultaneous receivers, which is the only parameter not natively available at each receiver itself. Hence, we designed a machine-learning algorithm able to detect the number of simultaneous receivers only based on the available parameters and the time needed to transmit only the first fraction of the total file. Exploiting also this piece of information, the second part of the predictor is able to output an on-the-fly prediction of the ETA for the remaining portion of the file, while transmission occurs. The prediction performance of the proposed algorithm was compared to the ETA estimates given by *scp*, *i.e.*, the software tool available in common Unix systems, showing considerable reductions of the prediction error, especially in the first part of each transmission, when a user has to decide whether to wait for the completion of the transmission or to delay it to some other time in the future.

Chapter 4 dealt with the design of a compression algorithm for biosignals such as heart or respiratory rates (RESP), electrocardiography (ECG) and photo plethysmographic (PPG) signals, within e-health applications. The goal was to realize a lightweight compressor, able to be deployed into wearable battery-equipped devices with limited computational power. Before entering the compression stage, each signal is pre-processed in order to remove noise, detect quasi-periodic peaks, and extract peak-to-peak segments. Then, each segment is windowed in order to contain a fixed number of samples, and input to a particular machine learning architecture called autoencoder. This technique basically maps a segment to the values associated to a number of neurons belonging to a so-called "hidden" layer, exploiting non-linear transfer functions. If this number is lower than the original segment size, then this mapping practically works as a lossy encoder, provided that the network is trained to reconstruct the segment at the receiver with a low error. As a result, the training phase consists of optimizing both the encoding and the decoding part of the autoencoder, with the constraint of reducing the amount of information used to represent each segment during the encoding process. The proposed technique was

compared with recent compression techniques from the literature, including Gain-Shape Vector Quantization (GSVQ), Compressive Sensing (CS), Discrete Cosine Transform (DCT) and Lightweight Temporal Compression (LTC), showing considerable performance gains in terms of compression efficiency, reconstruction fidelity and energy expenditure. For instance, at the highest compression level, the gain of the proposed technique against the second-best algorithm was 67% in terms of compression efficiency, while achieving a reconstruction error almost 80% lower.

Chapter 5 investigated the utilization of machine learning techniques able to predict the long-term channel gains in a wireless network. An accurate estimation of these metrics would enable both to efficiently allocate resources and to predict cell handovers, hence achieving, for instance, higher Quality of Service (QoS) levels for the end users. Differently from the literature, the proposed methods only exploited the knowledge of past channel samples, without resorting to additional information, such as geographical data coming from GPS signals. The predictions are performed by the Base Station (BS), which is able to indirectly learn both the channel fading characteristics and the user mobility pattern analyzing trends in the time-averaged Received Signal Strength Indicator (RSSI). We generated a realistic synthetic dataset for an urban scenario, in which a user exploiting an LTE connection moves in a Manhattan grid of 100 buildings following random mobility patterns for both pedestrian and vehicular scenarios. Propagation loss, wall penetration loss, fading and shadowing were taken into account for the computation of the RSSI values associated to the user. After a proper training phase of the considered machine learning models, namely graphical Bayesian models and support vector regression machines, the quality of the predictions was generally high, and comparable to the results obtained by other works available in the literature, which however exploited additional information such as GPS data, with negative impact on, *e.g.*, user's battery consumption.

Finally, **Chapter 6** showed how recent improvements in advanced deep learning techniques can be applied to the development of a self-driving car. Although this area of research is not strictly related to the field of wireless networking, it deals with modern machine learning based advancements and shows how such techniques are able to achieve surprising performance when compared to traditional approaches. Using a Convolutional Neural Network (CNN), raw pixels from a front-facing camera were directly mapped to car steering commands. The idea was to design an end-to-end approach which does not rely on the output of human-designed intermediate tasks, *e.g.*, lane marking detection or path planning, hence avoiding explicit decompositions of the problem, which are instead optimized simultaneously. Basically,

such criteria are usually selected for ease of human interpretation, which however does not guarantee optimal general performance. The system was successfully implemented in a real car, which was able to drive with high autonomy in local roads with and without lane markings, as well as in highways.

LIST OF PUBLICATIONS

The work presented in this thesis has appeared in the articles reported below.

Journal papers

[J1] **D. Del Testa**, N. Michelusi and M. Zorzi, "Optimal transmission policies for two-user Energy Harvesting Device networks with limited state-of-charge knowledge," *IEEE Transactions on Wireless Communications*, vol. 15(2), Feb. 2016.

[J2] **D. Del Testa** and M. Rossi, "Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders," *IEEE Signal Processing Letters*, vol. 22, Dec. 2015.

[J3] M. Hooshmand, D. Zordan, **D. Del Testa**, E. Grisan and M. Rossi, "Boosting the Efficiency of Wearable Devices through the Compression of Biosignals," submitted to *IEEE IoT Jornal*.

[J4] **D. Del Testa**, M. Danieletto and M. Zorzi, "A machine learning based ETA estimator for WiFi transmissions," *IEEE Transactions on Wireless Communications*, submitted.

Conference papers

[C1] **D. Del Testa**, N. Michelusi and M. Zorzi, "On Optimal Transmission Policies for Energy Harvesting Devices: the case of two users," in *Proceedings of the Tenth International Symposium on Wireless Communication Systems (ISWCS 2013)*, August 2013, Ilmenau, Germany.

[C2] **D. Del Testa** and M. Zorzi, "Optimal policies for two-user Energy Harvesting Device networks with imperfect State-of-Charge knowledge," in *Information Theory and Applications Workshop (ITA)*, February 2014, San Diego, USA.

[C3] A. Biason, **D. Del Testa** and M. Zorzi, "Low-complexity Policies for Wireless Sensor Networks with Two Energy Harvesting Devices," in *13th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2014, Piran, Slovenia.

[C4] **D. Del Testa**, M. Danieletto and M. Zorzi, "Applying Machine Learning Techniques to a Real Cognitive Network: File Transfer ETAs prediction," in *IEEE Global Communications Conference (Globecom)*, December 2015, San Diego, CA, USA.

[C5] **D. Del Testa**, M. Danieletto, G. M. Di Nunzio and M. Zorzi, "Estimating the number of receiving nodes in 802.11 networks via machine learning techniques," in *IEEE Global Communications Conference (Globecom)*, December 2016, Washington, DC, USA.

[C6] F. Chiariotti, **D. Del Testa**, M. Polese, A. Zanella, G. M. Di Nunzio and M. Zorzi, "Learning methods for long-term channel gain prediction in wireless networks," in *International Conference on Computing, Networking and Communications (ICNC)*, January 2017, Santa Clara, CA, USA.

[C7] M. Bojarski, **D. Del Testa**, D. Dworakowski, *et al.*, "End to End Learning for Self-Driving Cars," *arXiv preprint arXiv:1604.07316 (2016)*, April 2016.

BIBLIOGRAPHY

[1] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, Nov. 2010.

[2] T. Zahir, K. Arshad, A. Nakata, and K. Moessner, "Interference management in femtocells," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 293–311, 2013.

[3] E. Hossain, M. Rasti, H. Tabassum, and A. Abdelnasser, "Evolution toward 5g multi-tier cellular wireless networks: An interference management perspective," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 118–127, June 2014.

[4] C. Y. Tu, C. Y. Ho, and C. Y. Huang, "Energy-efficient algorithms and evaluations for massive access management in cellular based machine to machine communications," in *IEEE Vehicular Technology Conference*, Sep. 2011, pp. 1–5.

[5] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The tactile internet: vision, recent progress, and open challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 138–145, May 2016.

[6] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, "Mobility management for femtocells in lte-advanced: Key aspects and survey of handover decision algorithms," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 64–91, 2014.

[7] W. Y. Lee and I. F. Akyildiz, "Spectrum-aware mobility management in cognitive radio cellular networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 529–542, April 2012.

[8] D. Lopez-Perez, I. Guvenc, and X. Chu, "Mobility management challenges in 3gpp heterogeneous networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 70–78, December 2012.

[9] A. A. Gaivoronski, *Stochastic Optimization in Telecommunications*. Springer, Boston, MA, USA, 2006, pp. 761–799.

[10] S. Dong, P. Agrawal, and K. Sivalingam, "Reinforcement learning based geographic routing protocol for UWB wireless sensor network," in *IEEE Global Communications Conference (GLOBECOM)*, Nov. 2007, pp. 652–656.

[11] R. Arroyo-Valles, R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro, "Q-probabilistic routing in wireless sensor networks," in *3rd Interantional Conference on Intelligent Sensors, Sensor Networks and Information I(SSNIP)*, Dec. 2007, pp. 1–6.

[12] G. Ahmed, N. M. Khan, Z. Khalid, and R. Ramer, "Cluster head selection using decision trees for wireless sensor networks," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Dec. 2008, pp. 173–178.

[13] A. Forster and A. L. Murphy, "Clique: Role-free clustering with q-learning for wireless sensor networks," in *29th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2009, pp. 441–449.

[14] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in *International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, Sep. 2005, pp. 1214–1217.

[15] M. Bahrepour, N. Meratnia, M. Poel, Z. Taghikhaki, and P. J. M. Havinga, "Distributed event detection in wireless sensor networks for disaster management," in *2nd International Conference on Intelligent Networking and Collaborative Systems (INCOS)*, Nov. 2010, pp. 507–512.

[16] M. Sha, R. Dor, G. Hackmann, C. Lu, T. S. Kim, and T. Park, "Self-adapting MAC layer for wireless sensor networks," in *IEEE 34th Real-Time Systems Symposium (RTSS)*, Dec. 2013, pp. 192–201.

[17] M. H. Kim and M.-G. Park, "Bayesian statistical modeling of system energy saving effectiveness for mac protocols of wireless sensor networks," *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Jan. 2009.

[18] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[19] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 263–270.

[20] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Science Reviews*, vol. 78, no. 3, pp. 177–191, 2006.

[21] T. Le Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan, "Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network," in *32nd IEEE Conference on Local Computer Networks (LCN)*. IEEE, 2007, pp. 799–806.

[22] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," in *International Symposium on Communication Theory and Applications (ISCTA)*, 2001.

[23] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-constrained modulation optimization," *IEEE Transactions on Wireless Communications.*, vol. 4, no. 5, pp. 2349–2360, 2005.

[24] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.

[25] S. J. Baek, G. De Veciana, and X. Su, "Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation," *IEEE Journal on Selected Areas in Communications.*, vol. 22, no. 6, pp. 1130–1140, 2004.

[26] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

[27] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998, pp. 181–190.

[28] S. Ratnaraj, S. Jagannathan, and V. Rao, "OEDSR: Optimized energy-delay sub-network routing in wireless sensor network," in *IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2006, pp. 330–335.

[29] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *22nd Annual Joint Conference of the IEEE Computer and Communications*, vol. 3. IEEE, 2003, pp. 1713–1723.

[30] P. Nuggehalli, V. Srinivasan, and R. R. Rao, "Delay constrained energy efficient transmission strategies for wireless devices," in *21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2002, pp. 1765–1772.

[31] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *IEEE Transactions on Wireless Communications*, vol. 9, pp. 1326–1336, Apr. 2010.

[32] Z. Wan, Y. Tan, and C. Yuen, "Review on energy harvesting and energy management for sustainable wireless sensor networks," in *IEEE 13th International Conference on Communication Technology (ICCT)*, 2011, pp. 362–367.

[33] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Pervasive Computing*, vol. 4, pp. 18–27, Jan. 2005.

[34] M. Tentzeris, *Ambient Electromagnetic Energy Harnessed for Small Electronic Devices*. [Online]. Available: http://www.news.gatech.edu/2011/07/06/ambient-electromagnetic-energy-harnessed-small-electronic-devices

[35] R. Berry and R. Gallager, "Communication over fading channels with delay constraints," *IEEE Transactions on Information Theory*, vol. 48, pp. 1135–1149, May 2002.

[36] D. Niyato, E. Hossain, and A. Fallahi, "Sleep and wakeup strategies in solar-powered wireless sensor/mesh networks: performance analysis and optimization," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 221–236, Feb. 2007.

[37] M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Control of wireless networks with rechargeable batteries," *IEEE Transactions on Wireless Communications*, vol. 9, pp. 581–593, Feb. 2010.

[38] C. Renner and V. Turau, "CapLibrate: self-calibration of an energy harvesting power supply with supercapacitors," in *International Conference on Architecture of Computing Systems (ARCS)*, Hannover, Germany, February 2010, pp. 1–10.

[39] B. Xiao, Y. Shi, and L. He, "A universal state-of-charge algorithm for batteries," in *ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 687–692.

[40] J. Chiasson and B. Vairamohan, "Estimating the state of charge of a battery," *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 465–470, May 2005.

[41] N. Michelusi, L. Badia, and M. Zorzi, "Optimal transmission policies for energy harvesting devices with limited state-of-charge knowledge," *IEEE Transactions on Communications*, vol. 62, no. 11, pp. 3969–3982, Nov. 2014.

[42] K. Tutuncuoglu and A. Yener, "Optimal power policy for energy harvesting transmitters with inefficient energy storage," in *46th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2012, pp. 1–6.

[43] B. Devillers and D. Gunduz, "A general framework for the optimization of energy harvesting communication systems with battery imperfections," *Journal of Communications and Networks*, vol. 14, no. 2, pp. 130–139, Apr. 2012.

[44] N. Michelusi, L. Badia, R. Carli, L. Corradini, and M. Zorzi, "Energy management policies for harvesting-based wireless sensor devices with battery degradation," *IEEE Transactions on Communications*, vol. 61, no. 12, pp. 4934–4947, Dec. 2013.

[45] N. Michelusi, K. Stamatiou, and M. Zorzi, "Transmission policies for energy harvesting sensors with time-correlated energy supply," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2988–3001, Jul. 2013.

[46] R. Morsi, D. S. Michalopoulos, and R. Schober, "On-off transmission policy for wireless powered communication with energy storage," in *IEEE Asilomar Conference on Signals, Systems, and Computers, CA, USA*, Nov. 2014.

[47] B. Gurakan, O. Ozel, J. Yang, and S. Ulukus, "Energy cooperation in energy harvesting communications," *IEEE Transactions on Communications*, vol. 61, no. 12, pp. 4884–4898, Dec. 2013.

[48] J. Yang and S. Ulukus, "Optimal packet scheduling in a multiple access channel with energy harvesting transmitters," *Journal of Communications and Networks*, vol. 14, no. 2, pp. 140–150, Apr. 2012.

[49] N. Michelusi and M. Zorzi, "Optimal adaptive random multiaccess in energy harvesting wireless sensor networks," *IEEE Transactions on Communications*, vol. 63, no. 4, pp. 1355–1372, April 2015.

[50] N. Jaggi, K. Kar, and A. Krishnamurthy, "Rechargeable sensor activation under temporally correlated events," in *5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007*, Apr. 2007, pp. 1–10.

[51] A. Seyedi and B. Sikdar, "Energy efficient transmission strategies for body sensor networks with energy harvesting," *IEEE Transactions on Communications*, vol. 58, pp. 2116–2126, Jul. 2010.

[52] J. Xu and R. Zhang, "Throughput optimal policies for energy harvesting wireless transmitters with non-ideal circuit power," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 2, pp. 322–332, Feb. 2014.

[53] O. Orhan, D. Gunduz, and E. Erkip, "Throughput maximization for an energy harvesting communication system with processing cost," in *IEEE Information Theory Workshop (ITW)*. IEEE, Aug. 2012, pp. 84–88.

[54] C. K. Ho, P. D. Khoa, and P. C. Ming, "Markovian models for harvested energy in wireless communications," in *IEEE International Conference on Communication Systems (ICCS)*, Nov. 2010, pp. 311–315.

[55] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA, USA, 2005.

[56] K. W. Ross, "Randomized and past-dependent policies for Markov decision processes with multiple constraints," *Operations Research*, vol. 37, pp. 474–477, Jun. 1989.

[57] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[58] C. C. Kokonendji and S. S. Zocchi, "Extensions of discrete triangular distributions and boundary bias in kernel estimation for discrete functions," *Statistics & Probability Letters*, vol. 80, no. 21-22, pp. 1655 – 1662, Nov. 2010.

[59] "Statistics and facts about smartphones," http://www.statista.com/topics/840/smartphones/, accessed: 2016-10-21.

[60] S. Hakola, T. Chen, J. Lehtomäki, and T. Koskela, "Device-to-device (D2D) communication in cellular network - performance analysis of optimum and practical communication mode selection," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2010, pp. 1–6.

[61] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[62] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, Mar. 2015.

[63] OpenFlow Switch Consortium *et al.*, "Openflow switch specification version 1.0.0," Dec. 2009.

[64] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, Nov. 2014.

[65] F. Malandrino, C. Casetti, and C.-F. Chiasserini, "Toward D2D-enhanced heterogeneous networks," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 94–100, Nov. 2014.

[66] S. Andreev, A. Pyattaev, K. Johnsson, O. Galinina, and Y. Koucheryavy, "Cellular traffic offloading onto network-assisted device-to-device connections," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 20–31, Apr. 2014.

[67] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, Dec. 2012, pp. 59–60.

[68] A. Asadi and V. Mancuso, "WiFi direct and LTE D2D in action," in *2013 IFIP Wireless Days (WD)*, Nov. 2013, pp. 1–8.

[69] A. Asadi, V. Sciancalepore, and V. Mancuso, "On the efficient utilization of radio resources in extremely dense wireless networks," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 126–132, 2015.

[70] B. Manoj, R. Rao, and M. Zorzi, "Cognet: a cognitive complete knowledge network system," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 81–88, Dec. 2008.

[71] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive Networks: Adaptation and Learning to Achieve End-to-End Performance Objectives," *IEEE Communications Magazine*, vol. 44, no. 12, Dec. 2006.

[72] M. Zorzi, A. Zanella, A. Testolin, M. De Filippo De Grazia, and M. Zorzi, "Cognition-based networks: a new perspective on network optimization using learning and distributed intelligence," IEEE Access *Special Issue on Artificial Intelligence Enabled Networking*, vol. 3, 2015.

[73] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, Oct. 2008.

[74] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 47–52, 2007.

[75] V. Stavroulaki, A. Bantouna, Y. Kritikou, K. Tsagkaris, P. Demestichas, P. Blasco, F. Bader, M. Dohler, D. Denkovski, V. Atanasovski *et al.*, "Knowledge management toolbox: machine learning for cognitive radio networks," *IEEE Vehicular Technology Magazine*, vol. 7, no. 2, pp. 91–99, 2012.

[76] M. Danieletto, G. Quer, R. Rao, and M. Zorzi, "CARMEN: a cognitive networking testbed on android OS devices," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 98–107, Sep. 2014.

[77] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. Springer New York, 2006, vol. 4, no. 4.

[78] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, NY, USA, 1995.

[79] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.

[80] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming*. University of California Press, 1951.

[81] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[82] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[83] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *ACM Fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[84] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, Jan. 2009.

[85] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[86] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[87] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in *Proceedings of the tenth international workshop on artificial intelligence and statistics*. Citeseer, 2005, pp. 33–40.

[88] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on AI*, vol. 14, no. 2, 1995, pp. 1137–1145.

[89] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, Jul. 2009.

[90] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Elsevier, 2011.

[91] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, 1997.

[92] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric Sensing," *Phylosophical Transactions of Royal Society*, vol. 370, no. 1958, pp. 176–197, Jan. 2012.

[93] S. Riazul Islam, D. Kwak, M. Humaun Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access*, vol. 3, pp. 678–708, Jun. 2015.

[94] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *Journal of Neuroengineering and Rehabilitation*, vol. 9, no. 21, pp. 1–13, Apr. 2012.

[95] R. Istepanian, S. Hu, N. Philip, and A. Sungoor, "The potential of internet of m-health things "m-IoT" for non-invasive glucose level sensing," in *IEEE International Conference of the Engineering in Medicine and Biology Society (EMBC)*, Aug. 2011, pp. 5264–5266.

[96] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *MDPI Sensors*, vol. 14, no. 6, pp. 10 146–10 176, Jun. 2014.

[97] Zephyr Technology Corporation, "Bioharness 3 - Wireless Professional Heart Rate Monitor and Physiological Monitor," 2015. [Online]. Available: http://www.zephyranywhere.com/

[98] M. Hesse, P. Christ, T. Hormann, and U. Ruckert, "A respiration sensor for a chest-strap based wireless body sensor," in *Proceedings of IEEE Sensors*, Valencia, Spain, Nov. 2014, pp. 490–493.

[99] Thought Technology Ltd, "Respiration Sensor: Model SA9311M," 2015. [Online]. Available: http://www.thoughttechnology.com/

[100] Seraphim Sense Ltd., "Angel open sensor wristband," 2015. [Online]. Available: http://www.angelsensor.com/

[101] Z. Zhang, Z. Pi, and B. Liu, "TROIKA: A General Framework for Heart Rate Monitoring Using Wrist-Type Photoplethysmographic Signals During Intensive Physical Exercise," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 2, pp. 522–531, Feb. 2015.

[102] R. Mukkamala, J.-O. Hahn, O. T. Inan, L. K. Mestha, C.-S. Kim, H. Toreyin, and S. Kyal, "Toward Ubiquitous Blood Pressure Monitoring via Pulse Transit Time: Theory and Practice," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 8, pp. 1879–1901, Aug. 2015.

[103] T. Schoellhammer, B. Greenstein, M. W. E. Osterweil, and D. Estrin, "Lightweight temporal compression of microclimate datasets," in *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*, Nov. 2004.

[104] C. R. Rao, "The Use and Interpretation of Principal Component Analysis in Applied Research," *Sankhyā: The Indian Journal of Statistics*, vol. 26, no. 4, pp. 329–358, Dec. 1964.

[105] R. Shankara and S. M. Ivaturi, "ECG Data Compression Using Fourier Descriptors," *IEEE Transactions on Biomedical Engineering*, vol. 33, no. 4, pp. 428–434, Apr. 1986.

[106] B. A. Rajoub, "An Efficient Coding Algorithm for the Compression of ECG Signals Using the Wavelet Transform," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 4, pp. 355–362, Apr. 2002.

[107] L. Polania, R. Carrillo, M. Blanco-Velasco, and K. Barner, "Compressed sensing based method for ECG compression," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 761–764.

[108] G. D. Poian, R. Bernardini, and R. Rinaldo, "Gaussian dictionary for Compressive Sensing of the ECG signal," in *IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BIOMS)*, Oct. 2014.

[109] R. Francescon, M. Hooshmand, M. Gadaleta, E. Grisan, S. K. Yoon, and M. Rossi, "Toward Lightweight Biometric Signal Processing for Wearable Devices," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug. 2015.

[110] A. K. Jain, R. Chellappa, S. C. Draper, N. Memon, P. J. Phillips, and A. Vetro, "Signal Processing for Biometric Systems," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 146–152, Nov. 2007.

[111] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity Recognition using Cell Phone Accelerometers," in *International Workshop on Knowledge Discovery from Sensor Data (SensorKDD)*, Jul. 2010.

[112] M. Elgendi, B. Eskofier, S. Dokos, and D. Abbott, "Revisiting qrs detection methodologies for portable, wearable, battery-operated, and wireless ecg systems," *PLoS ONE*, vol. 9, no. 1, Jan. 2014.

[113] P. Morizet-Mahoudeaux, C. Moreau, D. Moreau, and J. Quarante, "Simple microprocessor-based system for on-line e.c.g. arrhythmia analysis," *Medical and Biological Engineering and Computing*, vol. 19, no. 4, pp. 497–500, Jul. 1981.

[114] M. L. Ahlstrom and W. J. Tompkins, "Automated High-Speed Analysis of Holter Tapes with Microcomputers," *IEEE Transactions on Biomedical Engineering*, vol. 30, no. 10, pp. 651–657, Oct. 1983.

[115] Y. Chen and H. Duan, "A QRS Complex Detection Algorithm Based on Mathematical Morphology and Envelope," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, Shanghai, China, Jan. 2005, pp. 4654–4657.

[116] F. Zhang and Y. Lian, "QRS Detection Based on Multiscale Mathematical Morphology for Wearable ECG Devices in Body Area Networks," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, no. 4, pp. 220–228, Aug. 2009.

[117] V. Afonso, W. J. Tompkins, T. Nguyen, and S. Luo, "ECG beat detection using filter banks," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 192–202, Feb. 1999.

[118] H. Dinh, D. Kumar, N. Pah, and P. Burton, "Wavelets for QRS detection," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, Oct. 2001, pp. 1883–1887.

[119] Q. Xue, Y. Hu, and W. J. Tompkins, "Neural-network-based adaptive matched filtering for QRS detection," *IEEE Transactions on Biomedical Engineering*, vol. 39, no. 4, pp. 317–329, Apr. 1992.

[120] L.-Y. Shyu, Y.-H. Wu, and W. Hu, "Using wavelet transform and fuzzy neural network for VPC detection from the holter ECG," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1269–1273, Jul. 2004.

[121] D. Kaplan, "Simultaneous QRS detection and feature extraction using simple matched filter basis functions," in *IEEE Computers in Cardiology*, Sep. 1990, pp. 503–506.

[122] J. Cox, H. Fozzard, F. M. Nolle, and G. Oliver, "AZTEC: A preprocessing system for real-time ECG rhythm analysis," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 128–129, Apr. 1968.

[123] J. P. Abenstein and W. J. Tompkins, "A New Data-Reduction Algorithm for Real-Time ECG Analysis," *IEEE Transactions on Biomedical Engineering*, vol. 29, no. 1, pp. 43–48, Jan. 1982.

[124] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig, "Principal Component Analysis in ECG Signal Processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, pp. 1–21, Feb. 2007.

[125] H. Lee and K. M. Buckley, "ECG data compression using cut and align beats approach and 2-D transforms," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 5, pp. 556–564, May 1999.

[126] D. Zordan, B. Martinez, I. Villajosana, and M. Rossi, "On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking," *ACM Transactions on Sensor Networks*, vol. 11, no. 1, pp. 15:1–15:34, Nov. 2014.

[127] N. Maglaveras, T. Stampkopoulos, K. Diamantaras, C. Pappas, and M. Strintzis, "ECG pattern recognition and classification using non-linear transformations and neural networks: A review," *International Journal of Medical Informatics*, vol. 52, no. 1–3, pp. 191–208, Oct. 1998.

[128] C. C. Sun and S. C. Tai, "Beat-based ECG compression using gain-shape vector quantization," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 11, pp. 1882–1888, Nov. 2005.

[129] J. Cardenas-Barrera and J. Lorenzo-Ginori, "Mean-shape vector quantizer for ECG signal compression," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 1, pp. 62–70, Jan. 1999.

[130] P. Soh, G. Vandenbosch, M. Mercuri, and D.-P. Schreurs, "Wearable Wireless Health Monitoring: Current Developments, Challenges, and Future Trends," *IEEE Microwave Magazine*, vol. 16, no. 4, pp. 55–70, May 2015.

[131] A. Johansson, P. Öberg, and G. Sedin, "Monitoring of Heart and Respiratory Rates in Newborn Infants Using a New Photoplethysmographic Technique," *Journal of Clinical Monitoring and Computing*, vol. 15, no. 7–8, pp. 461–467, Dec. 1999.

[132] K. Chon, S. Dash, and K. Ju, "Estimation of Respiratory Rate From Photoplethysmogram Data Using Time–Frequency Spectral Estimation," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 8, pp. 2054–2063, Aug. 2009.

[133] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[134] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[135] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[136] J. Karhunen and J. Joutsensalo, "Generalizations of Principal Component Analysis, Optimization Problems, and Neural Networks," *Neural Networks*, vol. 8, no. 4, pp. 549–562, Jan. 1995.

[137] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural networks*, vol. 2, no. 6, pp. 459–473, 1989.

[138] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985.

[139] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, 1974.

[140] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[141] N. Japkowicz, S. Hanson, and M. Gluck, "Nonlinear autoassociation is not equivalent to PCA," *Neural computation*, vol. 12, no. 3, pp. 531–545, 2000.

[142] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," in *International Conference on Machine learning*, Jul. 2008, pp. 1096–1103.

[143] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, Dec. 2010.

[144] N. M. Arzeno, Z.-D. Deng, and C.-S. Poon, "Analysis of First-Derivative Based QRS Detection Algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 2, pp. 478–484, Feb. 2008.

[145] M. Elgendi, "Fast QRS Detection with an Optimized Knowledge-Based Method: Evaluation on 11 Standard ECG Databases," *PLoS ONE*, vol. 8, no. 9, pp. 1–18, Sep. 2013.

[146] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, Jan. 1980.

[147] S. C. Tai, "Adaptive Sampling of One Dimensional Digital Signal," R.O.C. Patent 083 501, Jan., 1997.

[148] E. Candès and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969–985, Nov. 2007.

[149] R. Baraniuk, "Compressive Sensing [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[150] E. Candès and J. Romberg, "ℓ_1-MAGIC," Matlab routines for solving CS problems, 2005. [Online]. Available: http://statweb.stanford.edu/~candes/l1magic/

[151] W. Dai and O. Milenkovic, "Subspace Pursuit for Compressive Sensing Signal Reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.

[152] S. Becker, J. Bobin, and E. J. Candès, "NESTA: A Fast and Accurate First-Order Method for Sparse Recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, Jan. 2011.

[153] Z. Zhang, T.-P. Jung, S. Makeig, and B. D. Rao, "Compressed sensing for energy-efficient wireless telemonitoring of noninvasive fetal ecg via block sparse bayesian learning," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 2, pp. 300–309, Feb. 2013.

[154] Z. Zhang and B. D. Rao, "Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 2009–2015, Apr. 2013.

[155] A. Jense and A. la Cour-Harbo, *Ripples in Mathematics: the Discrete Wavelet Transform*. Springer, 2001.

[156] J. Yan, "Wavelet matrix," Technical Report, University of Victoria, Victoria, BC, Canada, Nov. 2009.

[157] N. G. Prelcic and O. W. M. Flórez, "UVI WAVE: the ultimate toolbox for wavelet transforms and filter banks," in *Workshop on Intelligent Methods for Signal Processing and Communications*, Jun. 1996, pp. 224–227.

[158] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Simultaneous sparse approximation via greedy pursuit," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Mar. 2005.

[159] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, Sep. 2011.

[160] S. Lee, J. Kim, and J.-H. Lee, "A Real-Time ECG Data Compression and Transmission Algorithm for an e-Health Device," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2448–2455, Sep. 2011.

[161] H. Lee and K. Buckley, "Heart beat data compression using temporal beats alignment and 2-D transforms," in *30th IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 2, Nov. 1996, pp. 1224–1228.

[162] V. Lukin, M. Zriakhov, A. Zelensky, K. Egiazarian, and A. Varri, "Lossy compression of multichannel ECG based on 2-D DCT and pre-processing," in *International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*, Feb. 2008, pp. 159–162.

[163] V. Allen and J. Belina, "ECG data compression using the discrete cosine transform (DCT)," in *Computers in Cardiology*, Oct. 1992, pp. 687–690.

[164] M. Alam and N. Rahim, "Compression of ECG signal based on its deviation from a reference signal using discrete cosine transform," in *International Conference on Electrical and Computer Engineering*, Dec. 2008, pp. 53–58.

[165] E. Anas, M. Hossain, M. Afran, and S. Sayed, "Compression of ECG signals exploiting correlation between ECG cycles," in *International Conference on Electrical and Computer Engineering*, Dec. 2010, pp. 622–625.

[166] R. Benzid, F. Marir, A. Boussaad, M. Benyoucef, and D. Arar, "Fixed percentage of wavelet coefficients to be zeroed for ECG compression," *Electronics Letters*, vol. 39, no. 11, pp. 830–831, May 2003.

[167] C. Karakus, A. C. Gurbuz, and B. Tavli, "Analysis of energy efficiency of compressive sensing in wireless sensor networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1999–2008, May 2013.

[168] M. Hooshmand, M. Rossi, D. Zordan, and M. Zorzi, "Covariogram-based compressive sensing for environmental wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 6, pp. 1716–1729, Mar. 2016.

[169] ARM The Architecture for the Digital World, "ARM Cortex-M4 Processor," 2015. [Online]. Available: http://www.arm.com/products/processors/cortex-m/

[170] Texas Instruments, "CC2451: 2.4 GHz Low Energy and Proprietary System-on-Chip," 2015. [Online]. Available: http://www.ti.com/product/cc2541

[171] M. Blanco-Velascoa, F. Cruz-Roldána, J. I. Godino-Llorenteb, J. Blanco-Velascoa, C. Armiens-Aparicioa, and F. López-Ferreras, "On the use of PRD and CR parameters for ECG compression," *Elsevier Medical Engineering & Physics*, vol. 9, no. 27, pp. 798–802, Nov. 2005.

[172] M. Saeed and M. Villarroel and A.T. Reisner and G. Clifford and L. Lehman and G.B. Moody and T. Heldt and T.H. Kyaw and B.E. Moody and R.G. Mark, "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): a public-access intensive care unit database," *Critical Care Medicine*, vol. 39, no. 5, pp. 952–960, May 2011.

[173] Y. Zigel, A. Cohen, and A. Katz, "ECG Signal Compression Using Analysis by Synthesis Coding," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 10, pp. 1308–1316, Oct. 2000.

[174] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2015–2020," *Cisco White Paper*, 2016.

[175] S. Mekki and S. Valentin, "Anticipatory quality adaptation for mobile streaming: Fluent video by channel prediction," in *16th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015, pp. 1–3.

[176] S. Zhou and G. B. Giannakis, "How accurate channel prediction needs to be for transmit-beamforming with adaptive modulation over Rayleigh MIMO channels?" *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1285–1294, 2004.

[177] Q. Liao, S. Valentin, and S. Stanczak, "Channel gain prediction in wireless networks based on spatial-temporal correlation," in *16th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2015, pp. 400–404.

[178] L. Dong, G. Xu, and H. Ling, "Prediction of fast fading mobile radio channels in wide-band communication systems," in *IEEE Global Communications Conference (GLOBECOM)*, vol. 6, 2001, pp. 3287–3291.

[179] Z. Shen, J. G. Andrews, and B. L. Evans, "Short range wireless channel prediction using local information," in *37th IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 1, 2003, pp. 1147–1151.

[180] I. C. Wong, A. Forenza, R. W. Heath, and B. L. Evans, "Long range channel prediction for adaptive OFDM systems," in *38th IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 1, 2004, pp. 732–736.

[181] D. Jarinová, "On autoregressive model order for long-range prediction of fast fading wireless channel," *Telecommunication Systems*, vol. 52, no. 3, pp. 1533–1539, 2013.

[182] S. Ramanan and J. M. Walsh, "Distributed estimation of channel gains in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 3097–3107, 2010.

[183] P. Demestichas, A. Katidiotis, K. A. Tsagkaris, E. F. Adamopoulou, and K. P. Demestichas, "Enhancing channel estimation in cognitive radio systems by means of bayesian networks," *Wireless personal communications*, vol. 49, no. 1, pp. 87–105, 2009.

[184] E. F. Flushing, J. Nagi, and G. A. Di Caro, "A mobility-assisted protocol for supervised learning of link quality estimates in wireless networks," in *International IEEE Conference on Computing, Networking and Communications (ICNC)*, 2012, pp. 137–143.

[185] A. Zanella, "Best practice in RSS measurements and ranging," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.

[186] D. Geiger and D. Heckerman, "A characterization of the Dirichlet distribution with application to Learning Bayesian Networks," in *11th ACM Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 196–207.

[187] F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, and B. V. Saunders, "Beta function."

[188] W. Karush, "Minima of functions of several variables with inequalities as side constraints," Ph.D. dissertation, Master's thesis, Dept. of Mathematics, University of Chicago, 1939.

[189] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

[190] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented LTE network simulator based on ns-3," in *14th ACM international conference on modeling, analysis and simulation of wireless and mobile systems*, 2011, pp. 293–298.

[191] LTE ETSI, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (3GPP TS 36.104 version 8.6. 0 Release 8), July 2009," *ETSI TS*, vol. 136, no. 104, p. V8, 2009.

[192] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[193] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[194] L. D. Jackel, D. Sharman, C. E. Stenard, B. I. Strom, and D. Zuckert, "Optical character recognition for self-service banking," *AT&T Technical Journal*, vol. 74, no. 4, pp. 16–24, Jul. 1995.

[195] Large scale visual recognition challenge (ILSVRC). [Online]. Available: http://www.image-net.org/challenges/LSVRC/

[196] Net-Scale technologies, Inc., "Autonomous off-road vehicle control using end-to-end learning," Tech. Rep., Jul. 2004. [Online]. Available: http://net-scale.com/doc/net-scale-dave-report.pdf

[197] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1989, pp. 305–313.

[198] Wikipedia.org. DARPA LAGR program. [Online]. Available: http://en.wikipedia.org/wiki/DARPA_LAGR_Program

[199] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, 2001, pp. 3320–3325.

[200] DAVE-2 driving a lincoln. [Online]. Available: https://drive.google.com/open?id=0B9raQzOpizn1TkRIa241ZnBEcjQ.